XATA2006

XML: Aplicações e Tecnologias Associadas

4ª Conferência Nacional

Editores:

José Carlos Ramalho
João Correia Lopes
Alberto Simões

8, 9 e 10 de Fevereiro de 2006

Ficha Técnica:

Design Gráfico: Benedita Contente Henriques
Redacção: J. C. Ramalho e J. C. Lopes
Composição: J. C. Ramalho e J. C. Lopes
Tiragem: 150 exemplares
Edição: Fevereiro de 2006
ISBN: 972-99166-2-4

Prefácio:

Esta é a quarta conferência sobre XML e Tecnologias Associadas. Este evento tem-se tornado um ponto de encontro para quem se interessa pela temática e tem sido engraçado observar que os participantes gostam e tentam voltar nos anos posteriores. O grupo base de trabalho, a comissão científica, também tem vindo a ser alargada e todos os têm colaborado com vontade e com uma qualidade crescente ano após ano.

Pela quarta vez estou a redigir este prefácio e não consigo evitar a redacção de uma descrição da evolução da XATA ao longo destes quatro anos:

2003 Nesta "reunião", houve uma vintena de trabalhos submetidos, maioritariamente da autoria ou da supervisão dos membros que integravam a comissão organizadora o que não invalidou uma grande participação e acesas discussões.

2004 Houve uma participação mais forte da comunidade portuguesa mas ainda com números pouco expressivos. Nesta altura, apostou-se também numa forte participação da indústria, o que se traduziu num conjunto apreciável de apresentações de casos reais. Foi introduzido o processo de revisão formal dos trabalhos submetidos.

2005 Houve uma forte adesão nacional e internacional (Espanha e Brasil, o que para um evento onde se pretende privilegiar a língua portuguesa é ainda mais significativo). A distribuição geográfica em Portugal também aumentou, havendo mais instituições participantes. Automatizaram-se várias tarefas como o processo de submissão e de revisão de artigos.

2006 Nesta edição actual, e contrariamente ao que acontece no plano nacional, houve um crescimento significativo. Em todas as edições, tem sido objectivo da comissão organizadora, privilegiar a produção científica e dar voz ao máximo número de participantes. Nesse sentido, este ano, não haverá oradores convidados, sendo o programa integralmente preenchido com as apresentações dos trabalhos seleccionados. Apesar disso ainda houve uma taxa significativa de rejeições, principalmente devido ao elevado número de submissões. Foi introduzido também, nesta edição, um dia de tutoriais com o objectivo de fornecer competências mínimas a quem quer começar a trabalhar na área e também poder assistir de uma forma mais informada à conferência.

Se analisarmos as temáticas, abordadas nas quatro conferências, percebemos que também aqui há uma evolução no sentido de uma maior maturidade. Enquanto que no primeiro encontro, os trabalhos abordavam problemas emergentes na utilização da tecnologia, no segundo encontro a grande incidência foi nos *Web*

Services, uma nova tecnologia baseada em XML, no terceiro, a maior incidência foi na construção de repositórios, motores de pesquisa e linguagens de interrogação, nesta quarta edição há uma distribuição quase homogénea por todas as áreas temáticas tendo mesmo aparecido trabalhos que abordam aspectos científicos e tecnológicos da base da tecnologia XML. Desta forma, podemos concluir que a tecnologia sob o ponto de vista de utilização e aplicação está dominada e que a comunidade portuguesa começa a fazer contributos para a ciência de base.

À semelhança da edição transacta, nesta edição, houve um elevado número de revisões por revisor. Estes conseguiram desempenhar bem o seu papel e cumprir as datas definidas pelo "chair". Mas o que aconteceu faz-nos pensar que há necessidade de alargar ainda mais a Comissão Científica, o que iremos tentar fazer durante o desenrolar do evento.

XATA 2006

A XATA 2006 surge na sequência das edições de 2003, 2004 e 2005. As áreas que são abordadas nesta edição não são muito diferentes das edições anteriores havendo talvez um maior equilíbrio e melhor distribuição pelas diferentes temáticas.

A maior diferença reside no facto de se ter acrescentado um primeiro dia com tutoriais. Poderá ser algo a repetir em edições futuras, provavelmente com temas mais verticais.

Este ano, vamos também deslocar-nos geograficamente para o Sul, para terras alentejanas. Depois do desafio colocado às várias instituições para que se candidatassem à organização do XATA2006 foi a Escola Superior de Tecnologia e Gestão de Portalegre que fez a proposta mais interessante e dentro do espírito que temos cultivado nas várias edições. Assim, este ano, vamos todos experimentar a hospitalidade alentejana e de certeza que serão uns dias bem passados.

Estrutura da Conferência

À semelhança das edições anteriores, a conferência encontra-se dividida em oito sessões temáticas repartidas pelos dois dias. Há ainda um primeiro dia com quatro sessões tutoriais.

Dia 1: Tutoriais

Esta tutorial sobre XML e Tecnologias Associadas foi pensada para aqueles que gostariam de assistir ao XATA2006 mas que até agora não tiveram qualquer formação nesta área. Assim, a tutorial foi planeada para uma plateia muito heterogénea e não apenas para informáticos.

Os temas e os conteúdos foram pensados de modo a que as sessões tivessem um encadeamento lógico, equilibradas em termos teórico-práticos e variadas em termos pedagógicos de modo a evitar a fadiga acumulada ao longo das 8 horas das várias sessões.

Público Alvo As sessões tutoriais foram pensadas e planeadas para:

- Profissionais de TI que procuram conhecer o potencial da tecnologia XML na resolução de problemas organizacionais e no desenvolvimento de software (e.g. Web Services, geração de páginas Web dinâmicas, interoperabilidade, etc.);
- Profissionais da área da gestão documental (e.g. arquivistas, bibliotecários, etc.) envolvidos em projectos de vertente tecnológica onde a manipulação de documentos XML ou conceitos associados estejam presentes;
- Estudantes de TI que desejem aprofundar os seus conhecimentos na utilização prática de XML e ferramentas associadas;

Estrutura e Conteúdos O tutorial desenrolar-se-á ao longo de um único dia e será repartido por quatro sessões de 2 horas. Cada sessão será constituída por uma introdução teórica de 45 minutos, um exercício prático de 45 minutos seguindo-se um período de 20 minutos para discussão e 10 minutos de intervalo:

Sessão 1 — Introdução ao XML

- Visão histórica do XML
- Introdução às linguagens de anotação
- Apresentação ao XML
- Utilização prática de XML em contextos organizacionais
- Conceitos associados ao XML
- A anatomia de um documento XML
- Exemplos de documentos XML
- Documento válido versus bem-formado
- Validação de XML: DTD/Schema
- Exercício prático: definição de pequenos dialectos XML, criação de instâncias e validação destas instâncias.

Sessão 2 — Modelo de dados e linguagem de interrogação

- Ciclo de vida de um documento XML
- Árvore documental abstracta
- Linguagem de selecção de elementos – XPath
- Exercício prático: realização de vários tipos de interrogações em documentos XML.

Sessão 3 — Processamento de XML

- Arquitectura de um sistema de processamento de XML
- Transformação de XML através de XSLT
- Anatomia de uma folha de estilo
- Exemplos de folhas de estilo
- Ferramentas para processar XML
- Exercício prático: geração de páginas HTML, transformação de XML em XML, geração de ficheiros CSV para o Excel.

Sessão 4 — Aplicações e dialectos XML

- Aplicações XML
  - Web Services
  - Armazenamento de XML
  - Intercâmbio de informação
- Dialectos XML
  - SVG
  - DocBook
  - MathML
  - MusicML
  - JavaML
  - Dublin Core
  - ...
- Discussão

Dias 2 e 3: Conferência

Nesta edição as sessões correspondem aos seguintes temas:

Sessão I: Modelação com XML Esta sessão está dedicada à especificação de novas linguagens XML de domínio específico.

Sessão II: Web Semântica e Ontologias A Web Semântica é um tema recorrente na XATA. Mais uma vez, temos um conjunto de trabalhos que nos demonstram as várias vertentes de investigação na área.

Sessão III: XML e Visualização Esta sessão corresponde à disseminação de informação. Na maior parte dos casos é necessário converter XML num formato mais apropriado para visualização.

Sessão IV: Bibliotecas e Arquivos Digitais – Numa altura em que os repositórios institucionais são cada vez mais importantes, esta sessão inclui alguns trabalhos sobre classificação e metainformação, repositórios digitais e preservação digital.

Sessão V: Web Services e Arquitecturas – Esta sessão apresenta alguns trabalhos onde é utilizada a tecnologia de Web Services para solucionar o problema ou para criar uma solução mais interessante.

Sessão VI: Processamento e Análise de XML – Esta sessão demonstra alguma da maturidade adquirida no panorama nacional. Analisam-se Schemas e calculam-se métricas sobre a sua concepção, discutem-se modelos de processamento e o encadeamento de transformações.

Sessão VII: XML como Representação Intermédia – O XML é cada vez mais usado para representar tudo e mais alguma coisa. Nesta sessão, veremos alguns contextos interessantes onde o XML serve como suporte intermédio de informação.

Sessão VIII: Aplicações XML – Esta é, talvez, a sessão mais heterogénea. Apresenta um conjunto de trabalhos relacionados com problemas reais e específicos de áreas bem diferentes que vão do Workflow ao Documento Único Automóvel.

Sessão IX: Posters – Esta será a nossa sessão "fantasma" pois não tem nenhuma fatia de tempo atribuída. No entanto, os posters serão colocados na sala contígua ao auditório, ou num local mais revolucionário se "as autoridades locais" permitirem, e haverá moderadores que se encarregarão de lançar a discussão sobre cada um dos trabalhos.

Esta edição do XATA, tem mais que as anteriores mas poderá ter ainda muito mais (estamos a pensar numa maneira de utilizar o tempo de almoço – preparem-se para um *brainstorming* em frente a um ensopado). Participe activamente no evento, partilhe as suas ideias e não receie levantar discussões. É para isso que todos lá estaremos.

Uns dias bem "xatos" para todos, que serão *quase* ao ritmo alentejano (quase porque nós somos mesmo "xatos"...)

José Carlos Ramalho

Comissão Organizadora:

Organização Local

Paulo Gomes — Escola Superior de Tecnologia e Gestão de Portalegre
Jorge Machado — Biblioteca Nacional

Gestão de Conteúdos

João Correia Lopes — Universidade do Porto, INESC Porto

Coordenação dos Tutoriais

José Carlos Ramalho — Universidade do Minho

Coordenação de Posters

Pedro Henriques — Universidade do Minho

Secretariado Local

Mónica Martins — Escola Superior de Tecnologia e Gestão de Portalegre
Maria José Corte Real Alegria Martins — Escola Superior de Tecnologia e Gestão de Portalegre
Ana Paula Barradas Vinagre — Escola Superior de Tecnologia e Gestão de Portalegre
Cristina Maria Pereira Pedro — Escola Superior de Tecnologia e Gestão de Portalegre

Sistemas de Informação Local

Valentim Realinho — Escola Superior de Tecnologia e Gestão de Portalegre
David Amador — Escola Superior de Tecnologia e Gestão de Portalegre

Comissão Científica:

José Carlos Ramalho	Universidade do Minho — Chair
Ademar Aguiar	Universidade do Porto, INESC Porto
Adérito Marcos	Universidade do Minho, CCG
Alberto Rodrigues da Silva	Instituto Superior Técnico
Alda Lopes Gançarski	Universidade do Minho
Cristina Ribeiro	Universidade do Porto, INESC Porto
Gabriel David	Universidade do Porto, INESC Porto
Giovani Librelotto	Centro Universitário Franciscano
João Correia Lopes	Universidade do Porto, INESC Porto
João Moura Pires	Universidade Nova de Lisboa
Jorge Cardoso	Universidade da Madeira
José Luís Borbinha	Instituto Superior Técnico
José João Almeida	Universidade do Minho
José Paulo Leal	Universidade do Porto
Luis Carriço	Universidade de Lisboa
Luís Ferreira	Instituto Politécnico do Cávado e do Ave
Luis Moura e Silva	Universidade de Coimbra
Marta Henriques Jacinto	I. das Tecnologias de Informação na Justiça
Nuno Horta	Instituto Superior Técnico
Pedro Antunes	Universidade de Lisboa
Pedro Henriques	Universidade do Minho
Salvador Abreu	Universidade de Évora

Revisores Adicionais:

Alberto Simões	Universidade do Minho
Ana Filipe	Universidade de Évora
Cláudio Fernandes	Universidade de Évora
David Simões	Instituto Politécnico de Setúbal
Miguel Ferreira	Universidade do Minho
Nuno Lopes	Universidade de Évora
Rui Lopes	Universidade de Lisboa
Vitor Beires Nogueira	Universidade de Évora

Agradecimentos:

Os editores deste livro de actas querem expressar o seu agradecimentos a quantos participaram na realização desta conferência.

Em primeiro lugar agradecemos a todos os autores a produção dos trabalhos que aqui ficam publicados, já que sem eles não existiria a XATA2006. Obrigado por terem aceite o desafio.

A todos os revisores, a quem foi solicitado um enorme esforço, um grande obrigado por cumprirem a árdua tarefa que lhes propusemos, nos prazos estabelecidos. A eles se deve a qualidade final desta obra e, não menos importante, o retorno crítico enviado a todos os autores, que cremos ser um contributo para a melhoria da qualidade da investigação e dos trabalhos publicados na área do XML em Portugal.

Fica aqui também um agradecimento especial à organização local, Escola Superior de Tecnologia e Gestão de Portalegre, nomeadamente ao Paulo Gomes, ao Jorge Machado e aos membros do secretariado.

Por fim, a todos os participantes que decidirem comparecer num evento onde se pretende privilegiar a troca e discussão de ideias, agradecemos a sua presença que vem engrandecer o debate.

Vamos "ao ritmo do Alentejo" no XATA2006

José Carlos Ramalho
João Correia Lopes
Alberto Simões

Patrocínios:

A Comissão Organizadora fica muito grata às seguintes organizações pelo seu apoio:

- Microsoft Corporation Portugal
- Departamento de Informática da Universidade do Minho
- Departamento de Engenharia Electrotécnica e de Computadores da Universidade do Porto
- Escola Superior de Tecnologia e Gestão de Portalegre
- FCA — Editora de Informática
- Cafés Delta
- Selenis
- Instituto Português da Juventude
- Região de Turismo de S. Mamede
- Gémeos Bar

Conteúdo

III XML e Visualização

IV Bibliotecas e Arquivos Digitais

V Web Services e Arquitecturas

VI Processamento e Análise de XML

VII XML como Representação Intermédia

VIII Aplicações XML

IX Posters

Algoritmos Morfológicos escritos em XML

Francisco de Assis Zampirolli[1] Roberto de Alencar Lotufo[2]
Rubens Campos Machado[3]

[1] Centro Universitário Senac
Av. Eng. Eusébio Stevaux, 823 – 04696–000 – São Paulo, SP
[2] FEEC – Faculdade de Elétrica e de Computação
6101 – 13083–970 Campinas, SP
[3] CenPRA - Centro de Pesquisa Renato Archer
6162 - 13081-970 - Campinas, SP, Brasil

Resumo Este artigo apresenta a *mmil* (*mathematical morphology intermediary language*), uma forma de representar algoritmos morfológicos usando XML, aplicados em processamento de imagens. Isto foi possível através do uso de um subconjunto da linguagem formal Z. Desta forma, algoritmos morfológicos clássicos foram escritos na *mmil* e, como exemplo, executados em linguagens C e documentados em LaTeX.

1 Introdução

Com o avanço da tecnologia, o software e o hardware estão ficando obsoletos num período de tempo cada vez mais curto. Sistemas operacionais como *MS-Windows* ou *Unix* e programas como *MATLAB* mudam de versão em média a cada três anos, ou menos. Quando isto ocorre, geralmente mudanças nas ferramentas que usam estas plataformas são necessárias. Na prática, quando um software não possui uma boa metodologia, tais mudanças significam reescrita de código. Existe hardware específico para o processamento de imagens e para a morfologia matemática [3], hardware aceleradores nas *CPU's* convencionais e também a possibilidade de se explorar eficientemente o uso do processamento em paralelo. Para um melhor aproveitamento do hardware, as empresas de software criam novos programas, com desempenho bem superior às versões anteriores. Assim sendo, é preciso atualizar freqüentemente os aplicativos e isto evidencia a necessidade de criar ferramentas capazes de minimizar a árdua tarefa de reescrita de código. Quando isto não ocorre, o custo de reescrita dos códigos pode ser alto e muitas vezes inviabilizar um projeto de hardware especial. O ideal seria com pouco esforço gerar novos códigos que executassem, de modo eficiente, nas diversas arquiteturas disponíveis.

Este trabalho contribui nesta questão, criando uma *linguagem intermediária* para escrever algoritmos morfológicos, e como resultado ter geração automática de código em várias linguagens de programação, junto com suas documentações. Exemplos de algoritmos morfológicos podem ser para implementar dilatação, erosão e transformada de distância [23].

Existem propostas de linguagens de programação independente de arquitetura para processamento de imagens [9,22,21], mas nenhuma delas usa XML. Recentemente surgiram normas para diminuir o trabalho de escrita de código, como MDA, OCL e XMI,

definidas pela OMG (*Object Management Group*) [17]. Por exemplo, no MDA (*Model Driven Architecture*) existem modelos independentes de plataforma PIM (*Plataform Indepentent Model*) e modelos específicos de plataforma PSM (*Platform Specific Model*). Podemos dizer que este trabalho define um PIM em XML e as *folhas de estilo* utilizadas poderiam ser consideradas como PSM.

Após esta introdução, a próxima seção descreve sobre os conceitos básicos utilizados. A seção 3 apresenta o ambiente *mmil*. A seção 4 apresenta os nove elementos da *linguagem intermediária*, junto com exemplos em XML. Finalmente, a seção 5 apresentada a conclusão e trabalhos futuros.

2 Conceitos

2.1 Organização da informação

Uma boa metodologia no desenvolvimento de software é definir uma estrutura de armazenamento que possibilite processar os seguintes conceitos [15]:

Conteúdo é a informação em si;
Estrutura defi ne a organização da informação;
Apresentação associa a forma de consumir a informação.

É de consenso que se estas três partes são separadas uma da outra, uma melhor utilização das informações é alcançada. Uma boa ilustração desses conceitos é realizada no sistema de criação de documentos conhecido como LaTeX [13]. O conteúdo é armazenado em um arquivo texto, a estrutura é armazenada em um arquivo de estilo (*book, article, report*, etc.) e a saída é alcançada pelo processador *TEX* [12]. Em contraste, se um autor escrever seu documento usando somente *TEX*, a reutilização do documento é perdida. Por outro lado, é possível converter arquivos LaTeX para outros formatos.

Nos últimos anos, com a proliferação da Internet e a necessidade de ter uma ferramenta eficiente de manipulação da informação, surgiu a linguagem *XML* (*EXtensible Markup Language*), onde o conteúdo é armazenado em uma linguagem de marcação, tal como em *HTML*, mas com a possibilidade de criar novos *tags*. A estrutura é definida através de *esquema* (ou *scheme*), que restringe o conteúdo da informação, como aceitar apenas números inteiros em um certo campo. Finalmente, a apresentação é definida através de *folhas de estilo* (ou *stylesheets*), que governam a tradução da informação para um formato de saída.

2.2 Notação Z

A notação Z é utilizada para especificação formal de problemas e é baseada em teoria dos conjuntos e lógica de primeira ordem. Z foi desenvolvida pelo *Programming Research Group* no *Oxford University Computing Laboratory* na década de 70 e é de domínio público pelo padrão *ISO/IEC 13568:2002* [1].

Como a *notação Z* modela problemas usando notação matemática, é natural usar em paralelo uma linguagem que possibilita editar símbolos matemáticos, como LaTeX. Neste sentido, um trabalho para servir de inspiração para mudança entre formatos é o

conversor *Zed2XML* [8], que transforma especificações Z escritas em LaTeX em documentos correspondentes em HTML [8] e isto também pode ser realizado no ambiente proposto neste documento através das *folhas de estilo*. Existem também editores para a *notação Z*, que podem armazenar seus documentos em LaTeX, como o *ZCREATOR* [7] e visualizadores, como o *Z Browser* [16].

A *notação Z* possui uma quantidade significativa de símbolos próprios, o que dificulta o aprendizado e a utilização. Por este motivo e baseado nos trabalhos existentes da *notação Z*, é possível criar um editor simplificado, que suporta a escrita de algoritmos morfológicos. Neste editor, além de poder visualizar as expressões matemáticas, é possível gerar códigos LaTeX, *HTML*, *C*, *MATLAB*, entre outros. Este editor seria uma proposta de continuação deste trabalho e não será discutido neste texto, porém existem trabalhos recentes definindo editores que poderiam ser usados, como o XESB [18].

2.3 Morfologia matemática

Uma forma elegante de resolver problemas de processamento de imagens é através da utilização de uma base teórica consistente. Uma destas teorias é a *morfologia matemática* criada na década de 60 por Jean Serra e George Matheron na *École Nationale Superiéure des Mines de Paris*, em Fontainebleau, França. Esta teoria diz que é possível fazer transformações entre reticulados completos[4], os quais são chamados de *operadores morfológicos*. Na morfologia matemática existem quatro classes básicas de operadores: dilatação, erosão, anti-dilatação e anti-erosão, chamadas de *operadores elementares*operador! elementar. Banon e Barrera [2] provaram que todos os operadores morfológicos invariantes por translação podem ser obtidos a partir de combinações de operadores elementares juntamente com as operações de união e intersecção. Além disso, quando um reticulado possui uma *família sup-geradora*, estes operadores podem ser caracterizados por *funções estruturantes*. Usando estes operadores elementares é possível construir uma *linguagem formal*, a *linguagem morfológica*, e sua implementação é chamada *máquina morfológica* [5]. Um exemplo de uma máquina morfológica é a *MMach* [4].

Veja na Tabela 1 a gramática da linguagem morfológica definida por Banon e Barrera [3]. Esta linguagem tem como característica representar operadores como dilatação e erosão por funções estruturantes. Porém, a linguagem intermediária definida neste documento, além de possuir esta característica, possui um vocabulário voltado para as diversas possibilidades de implementações destes operadores elementares.

Como exemplo do uso da gramática definida na Tabela 1, veja a seguir a definição da erosão morfológica caracterizada por função estruturante:

Seja $\mathbf{Z}$ o conjunto dos inteiros, $\mathbf{E} \subset \mathbf{Z}^2$ o domínio da imagem e $K = [0, k] \subset \mathbf{Z}$ um intervalo de números inteiros representando os possíveis níveis de cinza da imagem. O operador invariante por translação em níveis de cinza, $\varepsilon_b : K^{\mathbf{E}} \to K^{\mathbf{E}}$ ($K^{\mathbf{E}}$, lê-se conjuntos de funções de $\mathbf{E}$ em K), é definido como [11]:

$$\varepsilon_b(f)(x) = \min\{f(y) \dot{-} b(y - x) : y \in B_x \cap \mathbf{E}\},$$

[4] Um conjunto qualquer com uma relação de ordem é um reticulado completo se todo subconjunto não vazio tem um supremo e um ínfi mo. Para detalhes da teoria dos reticulados veja [6].

<operador>::=<operador elementar>|<limitante>|<composição>
<limitante>::=<argumento><operação de reticulado><argumento>
<argumento>::=<termo>|<composição>
<termo>::=<operador elementar>| (<limitante>)
<composição>::=<termo><termo>|<composição><termo>
<operador elementar>::=<operador morfológico>$_{<\text{função estruturante}>}$
<função estruturante>::=<letra>|<letra><número>
<número>::=<dígito>|<número><dígito>
<operação de reticulado>::= $\vee \mid \wedge$
<operador morfológico>::= $\varepsilon \mid \delta \mid \varepsilon^a \mid \delta^a$
<letra>::= $a \mid b \mid c \mid d$
<dígito>::= $0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

Tabela 1. Gramática da linguagem morfológica [3].

onde $f \in K^{\mathbf{E}}$, $x \in \mathbf{E}$, $B \subseteq \mathbf{E} \oplus \mathbf{E}$ e B é chamado *elemento estruturante*), $B_x = \{y + x,\ y \in B\}$ (translação de B por x) e b é uma *função estruturante* definida em B com $b : B \to \mathbf{Z}$.

A teoria dos reticulados estudada em morfologia matemática é abrangente e o leitor interessado pode consultar, por exemplo, os trabalhos de Serra, Banon e Barrera [19,3].

3 Ambiente *mmil*

O ambiente *mmil* foi o resultado de pesquisas em padrões de algoritmos gerando a tese do autor Zampirolli [23]. Nesta tese as erosões foram implementadas nos padrões de varredura paralelo, seqüencial e por propagação. Este processo também pode ser feito para outros operadores morfológicos, como dilatação, reconstrução e transformada de distância. Além disso, a transformada distância pode ser equivalente a erosão. Assim, os algoritmos clássicos da transformada de distância usando erosões foram reescritos(reimplementados) e classificados através destes padrões, produzindo códigos mais simples e eficientes. Deste estudo surgiram os elementos da linguagem intermediária, definidos na próxima seção.

3.1 Arquitetura

A arquitetura do ambiente *mmil* é ilustrada na Figura 1. Esta figura mostra um editor *GUI* (*Graphic User Interface*) para editar documentos *XML*, como exemplo, XESB [18]. Em nossos estudos, estes documentos são operadores morfológicos implementados através dos elementos definidos na próxima seção. O conteúdo em *XML* é validado pelo *esquema*, que é uma estrutura contendo um conjunto de regras definidas para os atributos e elementos do *XML*. O conteúdo em *XML* é então processado pelas *folhas de estilo* gerando códigos em diversas outras linguagens (*C, MATLAB, PYTHON, TCL/TK*, etc), em diversas plataformas (*UNIX, LINUX, WINDOWS*, etc.), e também para gerar documentação (LaTeX, *HTML*, etc.).

Outra ferramenta usada na máquina de translação da *mmil* é a linguagem *TCL*, que trabalha junto com as *folhas de estilo* no processo.

Atualmente existem *folhas de estilo* para gerar códigos nas linguagens *MATLAB* e *C*. Também existe *folha de estilo* para gerar documentos em LaTeX.

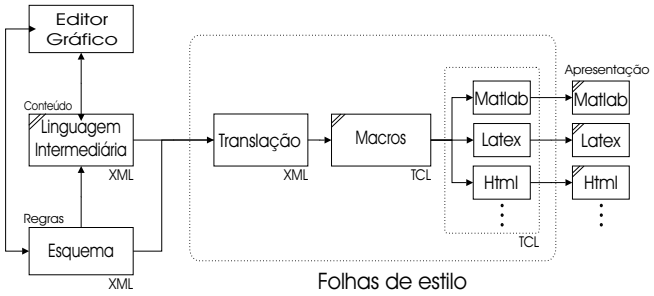


Figura 1. Arquitetura da *mmil*.

A arquitetura da *mmil* foi desenvolvida no *Adesso*, um ambiente computacional de suporte ao desenvolvimento de aplicações científicas [14].

Na Seção 4 será apresentado um breve resumo dos elementos do modelo de informação do *Adesso* necessários para o desenvolvimento da linguagem intermediária.

3.2 Linguagem intermediária

Usualmente os operadores são implementadas em dois passos: No primeiro passo, poderia especificar através da *linguagem morfológica*. No passo seguinte, poderia implementar em uma linguagem de programação de propósito geral, como *C*. Entretanto, é desejável implementar um sistema mais genérico e flexível de forma que os operadores morfológicos serão descritos em uma *linguagem intermediária*, inspirado na *notação Z* [20] e na *linguagem morfológica* [3]. Esta *linguagem intermediária* permite tradução para diversas linguagens e arquiteturas.

Por exemplo, existem várias formas de implementar um operador morfológico. A Figura 2 ilustra três algoritmos para o operador erosão: paralelo, seqüencial e por propagação. Esta figura também ilustra a tradução de um comando da *linguagem intermediária* para comandos da linguagem *C*. A implementação que tiver o melhor desempenho numa dada arquitetura será a escolhida para a tradução.

O principal resultado deste trabalho é a definição da *linguagem intermediária* em XML. Seus elementos devem ser em número reduzido, poderosos na utilidade e que sejam facilmente traduzidos nas diversas arquiteturas disponíveis, mesmo as de alto desempenho. Também se deseja compilar a *linguagem intermediária* para gerar código em diversas linguagens, por exemplo *C*, *MATLAB*, *Python* e *Tcl/Tk*, em várias plataformas, como *MS-Windows* e *Unix*, e em várias arquiteturas, como cartões aceleradores de processamento de imagens.

4 Elementos da linguagem intermediária

A *notação Z* pode ser armazenada numa linguagem intermediária chamada *linguagem intercâmbio* (*interchange language*) [10] usando elementos (*tags*) para armazenar as informações de forma semelhante à linguagem *XML*. Analogamente, a linguagem *XML* é utilizada para armazenar a linguagem intermediária definida neste trabalho através de poucos elementos, como serão apresentados nesta seção.

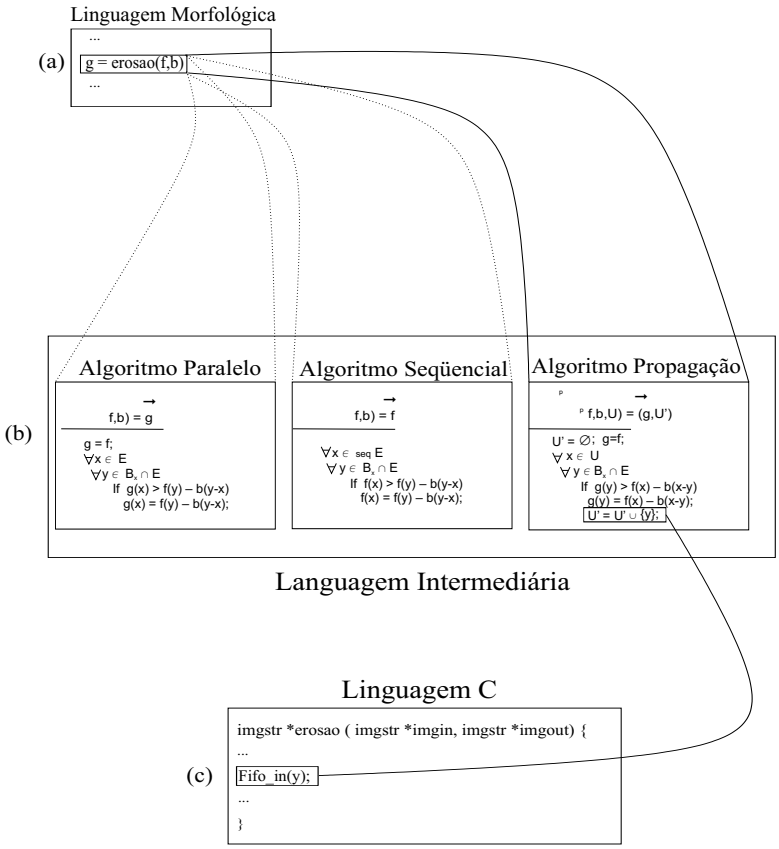


Figura 2. Ilustração da relação entre: (a) *linguagem morfológica,* (b) *linguagem intermediária* e (c) *linguagem C*.

Antes de definir os elementos da linguagem intermediária, será apresentado um breve resumo de onde estes elementos foram incluídos na estrutura do *Adesso*. Para mais detalhes deste ambiente consulte [14].

O elemento *AdFunction* define as transformações implementadas, Figura 3 [5], e tem como filhos: *Platforms* – define as plataformas onde serão gerados os códigos; *Short* – descreve uma descrição da função; *Symbol* – associa um símbolo matemático; *Returns* – define os argumentos de retorno; *Args* – define os argumentos de entrada através dos filhos definidos pelo elemento *Arg* contendo os atributos *name* e *type*; e *Source* – descreve o código da função.

A *linguagem intermediária*, onde são implementados os operadores morfológicos, é definida dentro do elemento *Code*, filho de *Source*, com atributo *lang* recebendo *"mmil"*, veja Figura 3. Será descrito abaixo cada um dos elementos desta linguagem.

[5] Figura gerada pelo software *XMLSpy*.

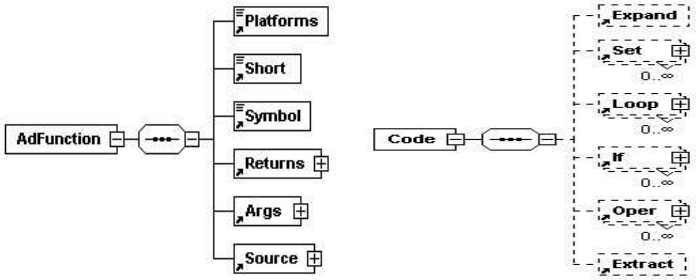


Figura 3. Elementos *AdFunction* e *Code* (fi lho de *Source*).

4.1 *Var*

Var é usado para acessar variável. Os seguintes casos podem ocorrer: o conteúdo é um valor escalar, uma matriz, ou um elemento de uma matriz (para o caso 2D). Neste último caso, um elemento de uma matriz é acessado pelo uso recursivo do elemento *Var* e pelo elemento *Index*.

4.2 *Index*

Index é usado para acessar os elementos de uma matriz e é usado junto com o elemento *Var*.

Example 1. Este exemplo usa os elementos *Var* e *Index*, devolvendo o conteúdo de um elemento de variável $hist$, definida por um elemento de f. Este exemplo pode ser reescrito simplesmente por $hist(f(x))$.

```
<Var>
   hist
   <Index>
     <Var>f<Index>x</Index></Var>
   </Index>
</Var>
```

4.3 *Set*

Set é usado para especificar atribuições, onde existem duas partes, *left* e *right*. O conteúdo da parte *right* é associado a parte *left*. Associado à *left* existe o elemento *Var* e associado a *right* pode existir um dos filhos mostrados na Figura 4. Veja Exemplo 2.

4.4 *Oper*

Oper especifica uma transformação, que é dividida em três padrões: *pontual*, *global* e *geométrico*. O padrão *pontual* tem as operações: *adição* ($+$), *subtração* ($-$), *diferença* ($\neq$), *igualdade* ($==$), *união* ($\vee$), *intersecção* ($\wedge$), *negação* ($\sim$), etc. O padrão *global* tem as operações *máximo* ($\bigvee$), *mínimo* ($\bigwedge$), etc. O padrão *geométrico* tem as transformações *translação* (B_x é a translação do conjunto B pelo vetor x), *find*, *fronteira* (∂), etc. Veja Figura 4.

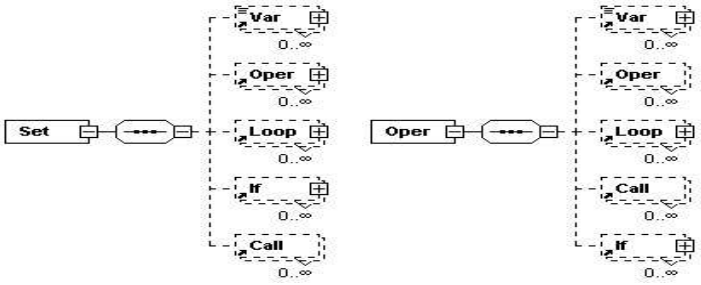


Figura 4. Elementos *Set* e *Oper*.

Example 2. O exemplo abaixo faz a imagem g receber a adição da imagem $f1$ e $f2$. Estas três variáveis possuem o mesmo tamanho **E**.

```
...
<Set>
  <Var>g</Var>
  <Oper name="Add">
    <Var>f1</Var>
    <Var>f2</Var>
  </Oper>
</Set>
```

Este código em *XML* pode ser apresentado através da seguinte expressão matemática (ou em código em *MATLAB*):

$$g = f1 + f2$$

4.5 *Loop*

Este elemento faz referência a repetições *For* ou *While*, denotado por $\forall$. O laço é associado a um índice e a um domínio através de atributos. O laço *While* requer uma expressão lógica. Veja Figura 5.

Example 3. Neste exemplo será mostrada uma outra versão para a adição de duas imagens. As três variáveis $f1$, $f2$ e g possuem o mesmo tamanho **E**. Agora a varredura é explícita para o domínio **E**:

```
...
<Loop name="for" i="x" D="f1">
  <Set>
    <Var>g<Index>x</Index></Var>
    <Oper name="Add">
      <Var>f1<Index>x</Index></Var>
      <Var>f2<Index>x</Index></Var>
    </Oper>
  </Set>
</Loop>
```

Este código pode ser visto como a seguinte expressão:

$\forall x \in \mathbf{E}$
$g(x) = f1(x) + f2(x)$;

ou ainda, pelo seguinte código em *MATLAB*:

```
for x=D(f1)
  g(x) = f1(x) + f2(x);
end
```

onde $\forall x \in \mathbf{E}$ ou $x = D(f1)$ são todos os pixels da imagem $f1$.

4.6 *If*

If é um comando condicional. Se a expressão lógica é *True*, o bloco associado ao elemento *If* é executado. Veja Figura 5.

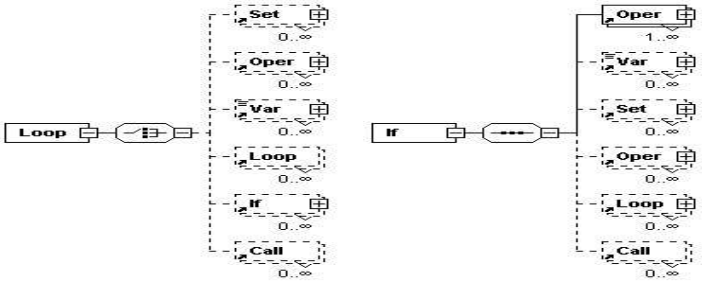


Figura 5. Elementos *Loop* e *If*.

4.7 *Call*

Call é usado para referenciar uma operação implementada pela *mmil*.

Example 4.

```
<Call name="ero" i1="f" i2="b" o1="g"/>
```

Este exemplo é equivalente a $g = \varepsilon_b(f)$, isto é, g recebe a erosão da imagem f pela função estruturante b.

4.8 *Expand*

Expand expande uma imagem pela vizinhança passada como parâmetro. É passado também o valor a ser atribuído na borda expandida. Por exemplo, se for usado vizinhança 3×3, a imagem será expandida de uma linha e uma coluna ao redor da imagem.

4.9 *Extract*

Extract extrai uma imagem pela vizinhança passada como parâmetro. Utilizado pelos operadores morfológicos que usam funções estruturantes.

4.10 Implementação da erosão

Será apresentada uma forma de implementar a erosão usando a linguagem intermediária. A parte central deste exemplo é mostrar a flexibilidade do ambiente desenvolvido, particularmente do elemento *Loop*. Não serão mostrados os códigos *XML* e *MATLAB*, mas apenas as expressões matemáticas equivalentes, que podem ser geradas automaticamente.

A erosão, apresentada na seção 2.3, quando implementada na *mmil* tem a representação apresentada no Algoritmo 1 (veja Figura 6) [6].

Algoritmo 1 Primeiro algoritmo da erosão paralela

$\varepsilon^1 : K^{\mathbf{E}} \times \mathbf{Z}^{\mathbf{B}} \longrightarrow \mathbf{K}^{\mathbf{E}}$
$\varepsilon^1(f, b) = g$

$\forall x \in \mathbf{E}$
$g(x) = \bigwedge_{\forall y \in B_x \cap \mathbf{E}} \{f(y) \dot{-} b(y - x)\};$

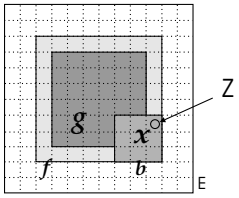


Figura 6. Ilustração do Algoritmo de Erosão.

A expressão $\forall x \in \mathbf{E}$ é especificada pelo elemento *Loop* com o atributo for:

```
<Loop name="for" i="x" D="f">
```

O atributo i especifica um elemento do domínio de f, isto é, um elemento de $\mathbf{E}$. A expressão $\bigwedge_{\forall y \in B_x \cap \mathbf{E}} \{...\}$ é também especificada pelo elemento *Loop*, com um atributo adicional *oper* recebendo *Min* representando a operação de redução $\bigwedge$, que é a operação de mínimo:

```
<Loop name="for" i="y" Dse="b" oper="Min">
```

O atributo *Dse* acima indica que o laço irá varrer o domínio da função estruturante *b*. O parâmetro $b(y - x)$ é obtido pelos elementos *Var*, *Index* e *Oper*. Neste último caso, dois atributos são passados, um indicando a operação de translação e o outro indicando o índice de translação:

```
<Oper name="Transl" i="x">
  <Var>b<Index>y</Index></Var>
</Oper>
```

[6] Para detalhes da notação de algoritmos utilizado neste trabalho consulte Zampirolli [23] ou Seção 2.5 de `www.zamp.pro.br/pub/tese.pdf`.

5 Conclusões e trabalhos futuros

Foi apresentado neste trabalho *mmil* (*mathematical morphology intermediary language*), um ambiente que escreve algoritmos (operadores) morfológicos em *XML* e, ao ser compilado, gera de forma automática código em diversas linguagens de programação e também documentação. Além disso, foi visto que ao gerar documentação em LaTeX, é possível não mais trabalhar com linguagens de programação para descrever um algoritmo morfológico, mas com expressões matemáticas, e a compilação destas expressões (armazenadas em *XML*) são códigos executáveis.

Trabalhos futuros

Para concluir este ambiente está faltando validar os códigos gerados para a linguagem de programação C, corrigir eventuais erros de geração de código, melhorar e completar a documentação do ambiente e a documentação gerada e finalmente testar a eficiência dos códigos gerados de forma automática. Falta também fazer a otimização do código gerado.

Como resultado adicional deste ambiente, fazendo poucas modificações, é possível construir uma estrutura em *XML* contendo todas as informações necessárias de um artigo, a compilação é gerada num formato qualquer (como *pdf*) pronta para a submissão, e é possível também testar o pseudo-código (representados por expressões matemáticas) gerando códigos para a linguagem de programação desejada. Assim, será eliminado o trabalho de implementar um pseudo-código contido em um artigo.

Referências

1. ISO/IEC 13568:2002. Information technology—Z formal specifi cation notation—syntax, type system and semantics. International Standard.
2. G.J.F. Banon and J. Barrera. Decomposition of mappings between complete lattices by mathematical morphology, Part I: general lattices. *Signal Processing*, 30:299–327, 1993.
3. G.J.F. Banon and J. Barrera. *Bases da morfologia matemática para analise de imagens binárias*. IX Escola de Computação, Recife, Brasil, 1994.
4. J. Barrera, G.F. Banon, and R.A. Lotufo. A mathematical morphology toolbox for the KHOROS system. In *Image Algebra and Morphological Image Processing V*, pages 241–252, Bellingham, Julho 1994. SPIE.
5. J. Barrera and G.J.F. Banon. Expressiveness of the morphological language. In *Image Algebra and Morphological Image Processing III*, pages 264–274, San Diego, California, 1992. SPIE.
6. G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, Rhode Island, 1967.
7. J. Bowen and D. Chippington. Z on the web using java. *Proc. 11th Int. Conf. on the Z Formal Method (ZUM)*, 1493:66–80, Setembro 1998.
8. P. Ciancarini, F. Vitali, and C. Mascolo. Managing complex documents over the www: a case study for XML. Technical report UBLCS-99-06, Department of Computer Science, Mura Anteo Zamboni, 7, Italy, 1999.
9. L.G.C. Hamey, J.A. Webb, and Wu I-Chen. An architecture independent programming language for low-level vision. *Computer Vision, Graphics and Image Processing*, 48(2):246–264, Junho 1989.

10. A. Harry. *Formal Methods - Fact File: VDM and Z*. John Wiley and Son Ltd, 1996.
11. H.J.A.M. Heijmans. Theoretical aspects of gray-level morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):568–581, Junho 1991.
12. Donald E. Knuth. *The TEX Book*. Addison-Wesley Publishing Company, Reading, MA, 15th edition, 1989.
13. Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley Publishing Company, Reading, MA, 1986 (also see 2nd edition, 1994).
14. R.C. Machado. Adesso - ambiente para desenvolvimento de software científi co. Dissertação de Mestrado, Universidade Estadual de Campinas - UNICAMP, Campinas, SP, Brasil, 2002.
15. S. McGrath. *XML: Aplicações Práticas*. Campus, Rio de Janeiro, 1999.
16. L. Mikusiak. Z browser: A tool for visualization of z specifi cations. *Proc. 9th Int. Conf. on the Z Formal Specifi cation Notation (ZUM)*, 967:510–525, Setembro 1995.
17. OMG – Object Management Group, Acesso em 2005. www.omg.org.
18. R. Queirós and J. P. Leal. Xesb (xml editor schema based) um editor para as massas. In João Correia Lopes José Carlos Ramalho, Alberto Simões, editor, *XATA2005, XML: Aplicações e Tecnologias Associadas (Vila Verde, Braga, 10 e 11 de Fevereiro de 2005)*, Portugal, Fevereiro 2005. ISMM, Universidade do Minho. http://hdl.handle.net/1822/865.
19. J. Serra, editor. *Image Analysis and Mathematical Morphology - Volume II: Theoretical Advances*. Academic Press, London, 1988.
20. J.M. Spivey. *Understanding Z: A Specifi cation Language and Its Formal Semantics*. Cambridge University Press, 1988.
21. R.S. Walace, J.A. Webb, and Wu I-Chen. Machine-independent image processing: performance of apply on diverse architectures. *Computer Vision, Graphics and Image Processing*, 48(2):265–276, Junho 1989.
22. J.A. Webb. Architecture-independent global image processing. In *10th International Conference on Pattern Recognition*, volume 2, pages 623–628, Atlantic City, NJ, USA, Junho 1990.
23. F.A. Zampirolli. *Transformada de distância por morfologia matemática*. Doutor, Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas, SP, Brasil, 2003.

A XML Data Dictionary and Document Representation Framework for Binary File Structure Description

Nuno Viana[1] and João Moura-Pires[2]

[1] Deimos Engenharia, Avenida D. João II, Lote 1.17, 8ºB, 1998-023 Lisbon, Portugal
nuno.viana@deimos.com.pt
http://www.deimos.com.pt
[2] CENTRIA/FCT, Quinta da Torre, 2829 -516 Caparica, Portugal
jmp@di.fct.unl.pt
http://centria.di.fct.unl.pt/~jmp

Abstract. System interface specification documents are hard to maintain updated, mainly due to the fact that the frequency of new improvements/ additions of new systems is high. To facilitate maintenance of existing interface specifications and enhance its usability, the development of a formal language specification, to address the definition and description of EUMETSAT's (European Organisation for the Exploitation of Meteorological Satellites) ground segment's system interfaces was envisaged. The language specification mechanism enables representation of binary message's structure, exchanged between EUMETSAT's systems. Additionally, the same definitions (derived from the specification language) can be embedded in automatically generated interface description documentation. In this manner, documents, become more than simple crystallized human-readable specifications. Document embedded data becomes "computable" and prone to be re-used by external tools, for generic data volume estimations and binary file structure validations. This paper focuses on the implementation details of a supporting XML Framework and its associated tools.

1 Introduction

The Operations Department at EUMETSAT[1] is currently responsible for the development and maintenance of several software and hardware infrastructures (also commonly known as facilities), which perform generation, archival and dissemination of meteorological products to all regional meteorological European member centres.

These facilities make extensive use of protocols based on binary messages for both control and transfer of meteorological raw/data products between systems (e.g. Atmospheric Motion Vectors, Clear Sky Radiances, Cloud Analysis, Cloud Top Height, Sea Surface Temperature, Total Ozone, Tropospheric Humidity – for a complete list of EUMETSAT available list of products, please refer to [2]). Specification of valid binary message structures and system interface descriptions is currently performed via creation of Interface Control Documents (ICD).

With the constant improvement of EUMETSAT's Ground Segment systems facilities, specifications have been subjected to successive updates. The complexity of the protocols (which rely on binary messages, reaching the Gigabyte order of magnitude) and the improvement of protocols (which are created to support new meteorological data products), generate high network traffic, thus making debugging, maintenance, development and deployment activities extremely difficult.

Nowadays, specifications for the EUMETSAT's system interfaces are maintained as interdependent Word documents (i.e. documents may contain references to specifications present in external documents), on which coherency and consistency is difficult to guarantee. Moreover, document contents are not computable (i.e. document data is generated for human-reading purposes only).

Enhancing the usability of documentation's contents, by transforming them into dynamically generated documents (containing "computable" specifications) represented in a suitable format, thus enabling the development of more generic document-driven tools, is one of the main objectives for this activity. Furthermore, migration of existing specification documents into the new format is also envisaged within the scope of this project.

This paper introduces a formalization approach for definition of a binary data messages' structures and definition of dynamically generated specification documents, which re-use the previously mentioned definitions. The first section (current) introduces the context of the activity. Afterwards, in the second section, the authors unveil the structure of the binary messages. In the third section the specification language used to describe the structure of the binary messages is presented. Further ahead, on section 4, the envisaged XML solution is described, with a special focus on the proposed model mechanism. Afterwards, the authors describe the operational tools which take advantage of this solution (section 5). Finally, on section 6 the authors discuss the conclusions and envisaged future work in the line of the present activity. Finally, the interested reader may find additional information concerning the work being described in this paper in section 7.

2 Binary Message Structure

In order to better understand the binary messages' structure we will follow a top-down descriptive approach. Firstly, the binary File/Packets' structure will be discussed. Afterwards, we will address the definition of Application Data Units (ADU) and finally the binary structure definition specification language will be presented as well as encoding/decoding rules.

2.1 Ground Segment Packets and Files

Binary messages exchanged between the different facilities in the EUMETSAT Ground Segment, use two types of transfer services (i.e. packet transfer and file transfer services) and are codified as binary data streams. Binary messages can therefore be divided into two sub-groups: "Packets" or "Files", which have a pre-defined struc-

ture built on ADU definitions, as described by the following Figs. 1 and 2 (optional fields are identified by dashed-line boxes):

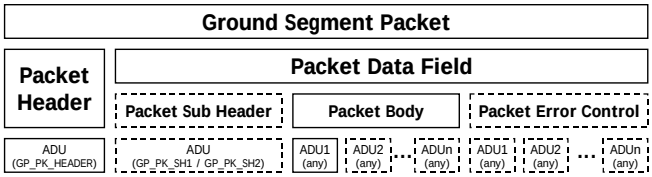


Fig. 1. Structure of a Ground Segment Packet ("Packet Header", "Packet SubHeader", "Packet Body" and "Packet Error Control").

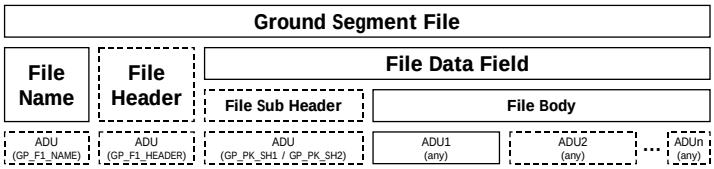


Fig. 2. Structure of a Ground Segment File ("File Name", "File Header", "File SubHeader" and "File Body").

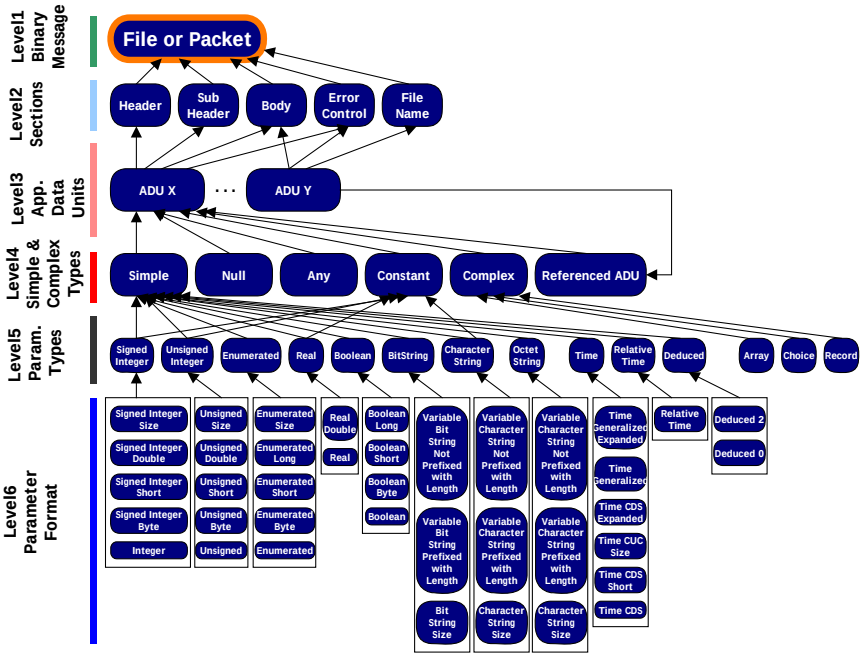


Fig. 3. Hierarchy of concepts showing how a Binary File/Packet can be specified using Application Data Unit which are defined using the formalized specification language using Simple, Complex and additional types (Null, Any, Constant and Referenced ADU).

As illustrated by the previous picture, a Ground Segment Packet is formed by a "Packet Header" (of type "GP_PK_HEADER"). Optionally, it can also be composed by an additional "Packet SubHeader" (of type "GP_PK_SH1" or "GP_PK_SH2"), a "Packet Body" (formed by any sequential collection of one or more ADU definitions). Finally, the "Packet Error Control" field can also be appended to the previous fields. In the same manner as the Ground Segment Packet definition, Ground Segment File definitions are also composed by a "File Header" (of type "GP_F1_HEADER"), an optional "File SubHeader" (of type "GP_F1_SH1") and a "File Body" (sequential composition of one or more ADU definitions). In addition, a Ground Segment File definition also contains the file naming convention to be used when naming files.

2.2 Application Data Units

Ground Segment engineers should define ADU (Level 3) types in order to form collections of sections for Binary File and Packet structure descriptions (Level 2). By its turn, each ADU makes use of an arrangement of simple and complex types (Level 4), including default and optional values.
Here is an example of an Application Data Unit definition ("GP_F1_SH1" – General Purpose File Sub Header type 1):

```
GP_FI_SH1 ::= RECORD
              {SubheaderVersionNo     UNSIGNED BYTE,
              ServiceType             GP_SVCE_TYPE,
              ServiceSubtype          UNSIGNED BYTE,
              FileTime                TIME CDS SHORT,
              SpacecraftId            GP_SC_ID,
              Description             CHARACTERSTRING SIZE (187)}
```

2.3 Encoding Rules

Previously, the hierarchy of concepts, which enable the specification of the binary File's and Packet's structure, has been described. We will now turn our attention to how the field values are in fact read and written from/to binary Files and Packets. In the frame of the EUMETSAT's Meteosat Second Generation Programme's Ground Segment systems, each binary value can be clearly identified by a set of attributes which define:

- The field type is represented under the form of a code, known as **PTC** (Parameter Type Code). E.g. Boolean(1), Integer(4), Unsigned Integer(3), Real(5), etc. This field is of type ENUMERATED_BYTE with a length of 1 byte.
- The representation format for the field is represented under the form of a code, known as **PFC** (Parameter Format Code). E.g. IntegerByte(4), Integer Short(12), Integer(14), Integer Double(16). This field can be of either ENUMERATED_BYTE (for fixed-length fields) or ENUMERATED_SHORT (for variable-length fields) with respective lengths of 1 and 2 bytes, respectively.

- The length of the parameter data field, known as **PDL** (Parameter Data Length), only valid for variable-length data types (variable string derived types such as CHARACTERSTRING, BITSTRING and OCTETSTRING). This value is of type SHORT_UNSIGNED and has a size of 2 bytes.
- The data value itself. Its type depends on the PTC and PFC values (when the field encoding type is "explicit" or if "implicit". In the context of our activity, binary data representation is possible through two methods for encoding simple parameter field values:

  a) "Explicit" method, where each value is preceded by an indication of the field type that follows, as well as the format for the type.
  b) "Implicit" method, where only the parameter value exists. That is the parameter type and format are assumed to be commonly agreed by the applications, which perform the encoding and decoding of the binary values.

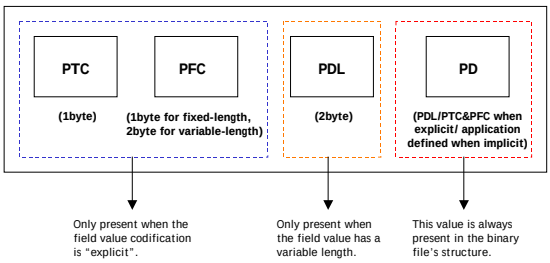


Fig. 4. Codification mechanism used for representation of a field value in the binary messages exchanged between the EUMETSAT's Ground Segment's systems.

On the previous figure, one may find the required and optional fields necessary to encode/decode binary values from the exchanged messages. When a value is "explicitly" encoded, the binary field (stored on the binary file) is prefixed with the PTC and PFC values. If in addition, the field has a variable length, the value is firstly prefixed with its size (PDL), resulting in the structure above (see Fig. 4). There is no explicit method for the complex types (i.e. no PTC and PFC fields). However, when a complex type is declared "explicit", its components (that is to say, all simple types) inherent this attribute's flag value. In short, when "explicit" encoding is used, this means that before reading the actual field value from the binary file, an application should first read its type code (PTC), format code (PFC) and its size fields from the binary file (if the field has a variable length specification). On the opposite side, when declared "implicit", applications assume a commonly agreed type and format for interpreting binary values read/written from/to files.

3 Specification Language

On the previous section (2), we have seen how the binary messages (packets and files) can be decomposed into simple component structures and how their identifications are encoded in the binary file.

In this section, the authors will address the formalization of a specification language, which was envisaged to characterize the binary structure for files and packets. The formalization of the specification language has been described using a BNF (Backus-Naur Form) notation and comprises “Simple Types” (the atomic types) and “Complex Types” (array, choice and record structures). Additionally, the specification language was enriched by the authors with new types such as “Null”, “Any” and “Referenced” types.

3.1 Simple Types

Simple types should be understood as the atomic type definitions, which exist in the specification language. The language comprises “Boolean”, “Enumerated”, “Unsigned Integer”, “Signed Integer”, “Real”, “Bit String”, “Octet String”, “Character String”, “Absolute Time”, “Relative Time” and “Deduced” types (see Level3 in Fig. 3).

3.2 Complex Types

In addition to the simpler atomic type definitions, the language also incorporates the notion of complex structures built on previous defined Simple Types:

1. Array – An ordered set of a fixed or variable number of fields (array elements) of the same simple or complex type (e.g. Array of Array).
2. Record – An ordered set of a fixed number of fields (components) of any simple or complex type (e.g. Record with component1=Array and component2=Integer).
3. Choice – A set of complex or simple fields, from which one can be selected based on the context and determined at application runtime (e.g. Choice with component1=Signed Integer and component2=Unsigned Integer).

3.3 Complementary Types

In addition to the simple and complex types, the authors decided to introduce into the specification language other high-level types such as “Constant”, “Null”, “Any” and “Referenced ADU” (due to representation requirements). “Constant” types (as the name implies) allow definition of environment constants, which can be of type Signed and Unsigned Integer, Real and Characterstring. “Any” allows definition of a field value without specifying its type or internal structure, while “Null” defines an element, which has no assigned type or field value. Finally, yet importantly, there is a special type, which was introduced in this specification language to allow re-usage of existing ADU definitions (“Referenced ADU”) inside other ADU. Below you can find three examples of variable definitions using the specification language:

Day ::= ENUMERATED SIZE (4)
{Mon, Tue, Wed, Thu, Fri, Sat, Sun, NotADay (OTHERS)}

```
NumericValue ::= EXPLICIT CHOICE
                 {UnsignedType      UNSIGNED,
                 IntegerType        INTEGER,
                 RealType  REAL}
Example ::= VARIABLE ARRAY SIZE (1..4) OF RECORD
                 {Date              TIME GENERALIZED,
                 DayOfWeek          Day,
                 Filler             BITSTRING SIZE (4),
                 ParameterName      CHARACTERSTRING SIZE (6),
                 ParameterValue     NumericValue,
                 Unit               CHARACTERSTRING SIZE (4)}
```

4 The XML Framework Solution

Previously (section 2), we have described the internal structure for the binary messages (files and packets) exchanged between the different EUMETAT system facilities. Moreover, a formalization of a specification language capable of describing the complex structure of the binary messages was presented in section 3. We are now going to describe the envisaged solution and its main functionalities.

4.1 Goals

Defining a specification language for description of binary messages, addressed part of the problem. Users have expressed their desire to be able to represent the language in such as way that it would be easily manageable and computable. XML was then selected as a natural candidate for representation of the both the specification language. XML is computable and there is a variety of XML manipulation tools freely available to users.

In addition to the formalization of the binary message specification language using XML, it was envisaged that the definition of files/packets including its internal structure (headers, sub-headers, body and so on), which are composed by ADUs would also be performed via XML. This would allow the full representation of files/packets under XML format. In order to maintain ICD (system interface specification documents) updated, it was also proposed to transform existing documentation into a XML based format. By this manner, documents would be able to embed binary file/packet “computable” definitions based on XML, capable of being used by externally developed tools, while minimizing necessary user intervention during document updates (references to files/packets and ADUs specifications are dynamically updated upon user request). This proposed approach, opens a path for development of more generic tools, in less time with less effort (e.g. computation of data volume estimations for generated network traffic between facilities and estimation of archival allocation requirements for generated data products).

4.2 The Proposed Model

Users wanted to be able to specify the collection of ADUs (Application Data Units) and the binary messages' structure (sequential composition of ADUs). Since ADUs are defined using the formalized specification language components in section 3, the XML-based framework model included the ability to firstly define the specification language. A language vocabulary for specification of the specification language was then designed and implemented (see a in Fig. 5 a) and b))

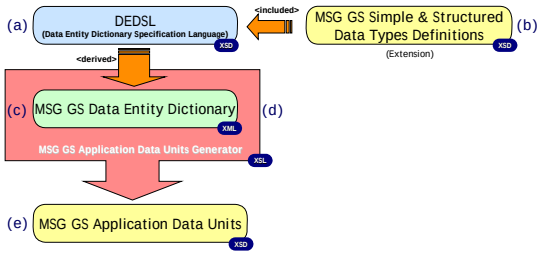


Fig. 5. Engineered solution to address the representation of binary message structures using XML based technology.

By using the rules expressed in this language specification vocabulary a) and b), under the form of a XML schema, users are able to define the full collection of Data Dictionary entities (where each Data Dictionary entity element represents an ADU definition in a data dictionary) which will be made available to users for definition of files/packet structures in interface specification documents. At this point we have a collection of Data Dictionary entities under the form of an XML file. In order to allow definition of binary packets and files (composed by collections of ADUs), we should first generate the library of ADUs. This library can be dynamically generated using a provided MSG GS ADU Generator (implemented as an XML Stylesheet file), which transforms the XML file into an ADU specification language (XML Schema file). As can be seen in Fig. 6, the example ADU is composed by a record with different components of type ("UNSIGNED BYTE", "TIME_CDS_SHORT", "CHARACTERSTRING_SIZE" and even of "REFERENCE_TYPE" – yellow coloured. This last type allows re-usage of previously defined ADU as types themselves).

The proposed model for the implementation of the XML Framework solution relies on "Schema" validation for ensuring that both the Data Dictionary types and the built ADU definitions follow the standardized implemented structure. The use of "Schematron"[3] validation would bring significant advantage for semantic validation (e.g. several field constraints are currently only checked afterwards, and not at XML document definition time – field sizes which depend on variable size definitions of child types).

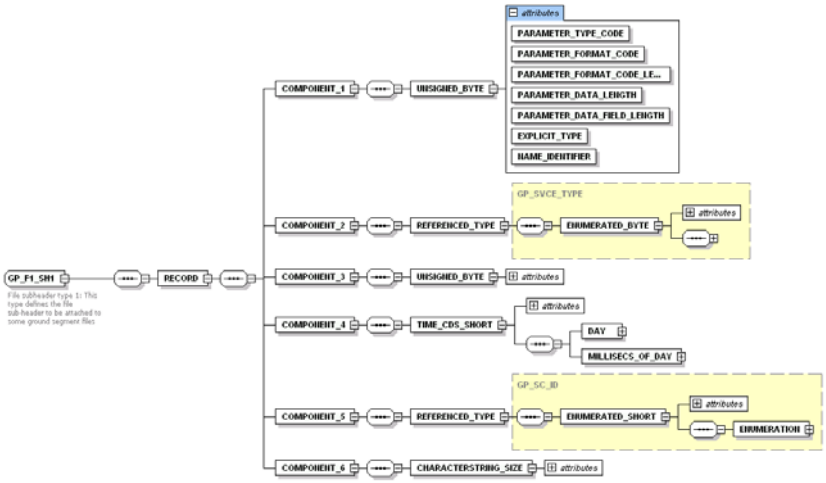


Fig. 6. ADU structure example ("GP_F1_SH1").

4.3 Interface Specification Documents

The paragraphs that follow, describe the mechanism for definition of interface specification documents and how the ADU definition language previously generated will be utilized.

In addition to the standardization of ADU definitions, users wanted to both standardize the contents for interface specification documents and be able to include the previously ADU language vocabulary in each of the generated document. This task was accomplished by defining a XML Schema language for structuring of the documents contents. Moreover, this document structure specification language takes advantage of the ADU library previously generated. This mechanism is described in the following illustration:

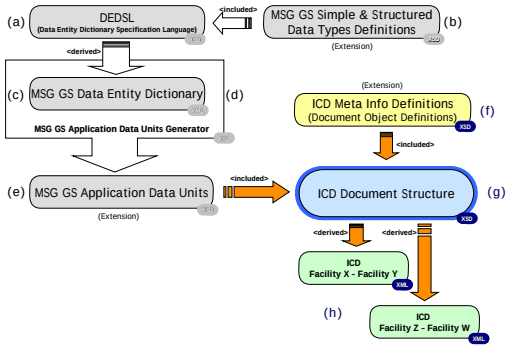


Fig. 7. ICD document structure (g), which includes the dynamically generated ADU specification (e).

4.4 Envisaged Deployment

After defining the collection of ADUs and the structure for interface specification documents, in order to that ensure users use the same language (not customized versions), a centralized repository was designed. This repository stores the vocabulary languages, as XML Schema files (for both the ADUs and interface specification documents). Furthermore, when defining the contents for the interface specification documents, users should insert references to ADUs (which are stored in this centralized repository). When users would like to retrieve any interface specification document (in XML format), the repository automatically replaces the references to ADUs, by its internal structure (used as input data for external tools).

It is also envisaged that for document updating purposes, the central repository will also be able to provide the original document versions in XML (containing references to ADUs), which should be re-uploaded into the central repository once updated.

Given that any major change on ADUs definitions has an impact on all documents which refer them, only administrator users should have access to the XML Framework (located into the central repository) – for updating ADUs definitions for instance.

5 XML Framework's Tools

With the XML Framework deployed in a centralized manner, administrators are able to enforce coherency of ADU definitions in interface specification documents, as well as standardizing of its internal structure across all platforms.

But the real gain comes from developing tools which use the binary files/packets definitions specified in each of the interface specification documents. Validation of binary files and packets (exchanged between the different EUMETSAT system facilities - and specified as collections of ADU definitions in the interface specification document) captured under the form of files can be accomplished via a Binary File Validation Tool, which is currently under development. This tool ingests a binary file or packet, an ICD in XML format and the chosen identification of the file/packet definition. The tool is then able to produce a compliance report containing the results from the field values extracted from the binary file according to its specified structure. Another tool, could take for instance, a specification document, a user specified binary file/packet definition and a given frequency for generation of that particular file/packet, for determination of daily-generated network traffic and/or allocation requirements. Besides the previously described tool, other tools will continue to provide browsable (HTML) and printable (PDF) versions of ICD documents with a format similar to existing documents in Word document format. This process is accomplished by making extensive use of XML Stylesheets for both transformation of specification documents from XML format into XHTML format, or into XSL:FO (Formatting Objects), which can be further transformed into PDF files via a "Formatting Objects Processor" engine such as the "Apache FOP Engine"[4].

Last but not least, a Data Format Extractor Tool is also nearly finished. This tool automates the process of extracting ADU textual definitions from original ICD documents and converting them into their XML representation according to the XML Framework for further integration in ICD documents in XML format. All these command-line tools are being developed using Java for added portability and will also be integrated and transparently made accessible via the centralized Server.

6 Conclusion and Future Work

The paper has focused on the descriptions of an engineering solution devised for standardization of data types representation through the formalization of a specification language, under the form of a Data Dictionary in XML format, re-usage of user-defined data entities specifications (ADUs) inside interface specification documentation (with added validation mechanisms provided by XML Schemas) also represented in XML format and also on tools which are able to ingest the specification language in XML, thus turning previously reading-oriented (document) data contents into "computable" data contents. The proposed model depends exclusively on XML Schemas for validation of XML file structures, nevertheless this solution could be greatly improved by using other validation mechanisms such as Schematron and/or XCSL[5, 6] (XML Constraint Specification Language). It also important to mention that similar work (specification of binary files using XML notation) has also been carried out by edikt since 2003. Their work has culminated in the development of BinX[7, 8], a library and associated editor tool for representation and manipulation of scientific data for grid applications. Unfortunately, the fact that bit types are not supported as well as the limited portability of the solution (library is written using C++, although porting to other languages such as Java is being envisaged), have rendered this approach unusable (for representation of EUMETAST binary data).

Besides the development of the XML Framework, several tools have been developed: the Binary File Validation Tool, an ICD Preview Tool, Data Format Extractor Tool and the server component which wraps all these command-line tools under a multi-user web interface for easy access. The current work has been partially inspired in previous and ongoing research work carried at the Uninova Research Institute[9], namely the development of metadata storage repositories and manipulation tools[10] for SEIS[11] (Space Environment Information Systems) and SESS[12] (Space Environment Support System) projects.

As future work (and as previously discussed), the possible use of Schematron and/or XCSL to improve the XML Framework should be investigated. Additionally, it has been also anticipated that a port of the tools will take place into C/C++ for added speed gains as well the enhancement of the XML Framework with the development of a generic API library capable of providing reading and writing capabilities with little integration effort with external applications.

Furthermore, we hope that it would be possible to broaden the applicability of the XML Framework and tools to other areas of EUMETSAT. This solution is prone to be re-used by any project on which the representation and manipulation of binary data is an issue.

7 References

1. EUMETSAT. *European Organisation for the Exploitation of Meteorological Satellites*. 2005 [cited 31/10/2005]; Available from: http://www.eumetsat.int.
2. EUMETSAT. *Eumetsat Access to Data - Product List*. 2005 [cited 2005/11/06]; Available from: http://www.eumetsat.int/idcplg?IdcService=SS_GET_PAGE&nodeId=522&l=en.
3. *Schematron - A Language for Making Assertions About Patterns Found in XML Documents*. 2006 [cited 2006.01.21]; Available from: http://www.schematron.com/.
4. Apache. *The Apache XML Graphics Project*. 2005 [cited 2005/11/06]; Available from: http://xmlgraphics.apache.org/fop/.
5. José Carlos Ramalho, et al. *XCSL : XML Constraint Specification Language*. 2006 [cited 2006.01.21]; Available from: http://www.di.uminho.pt/~gepl/xcsl/.
6. Ramalho, J.C. *Constraining Content: Specification and Processing*. in *XML Europe'2001*. 2001. Berlin, Germany.
7. *BinX - A Library for Representation of Scientific Data*. 2006 [cited 2006.01.21]; Available from: http://www.edikt.org/binx/index.htm.
8. Rob Baxter, et al. *BinX – A tool for retrieving, searching, and transforming structured binary files*. in *All-Hands Meeting 2003*. 2003. Nottingham.
9. UNINOVA/CA3. *UNINOVA - Centre For the Development of New Technologies/Soft-Computing and Autonomous Agents*. 2005 [cited 2005.11.20]; Available from: http://www.uninova.pt/ca3.
10. R. Ferreira, et al. *XML Based Metadata Repository for Information Systems*. in *EPIA 2005 - 12th Portuguese Conference on Artificial Intelligence*. 2005. Covilhã, Portugal.
11. Pantoquilho, M., et al. *SEIS: A Decision Support System for Optimizing Spacecraft Operations Strategies*. in *IEEE Aerospace Conference*. 2005. Montana, USA.
12. ESA. *Space Environment Support System for Telecom/Navigation Missions*. 2005 [cited 2005/11/06]; Available from: http://telecom.esa.int/telecom/www/object/index.cfm?fobjectid=20470.

Project Markup Language (PML) Schema Proposal

PAULO GOMES[1], JORGE MACHADO[2]

Superior School of Technology and Management,
Polytechnic Institute of Portalegre,
Lugar da Abadessa, Apartado 148,
Portalegre – Portugal

[1]paulogomes@estgp.pt
[2]jmachado@estgp.pt

Abstract. In this paper we present the steps followed to make a proposal for a Project Markup Language (PML). PML is to use in project management solutions, like GPRM (Global Project for Research Management) [1]. PML (Project Markup Language) is a markup language for Project Management Servers (like Microsoft Project Server/EPM Servers [10], Global Project Management/GPM Servers or GPRM Server [1]). PML has the main purpose to establish a standard model to Project information, to use it through the various Project Management Applications and Servers. With that we can use search and retrieval index engines (like SPEAK) to have a free communication between different Project Servers and Applications. This paper focuses on the language features and presentation scheme designed for Project Management.

1 Introduction

This document defines the PML. This document also revises the PML grammar, using the XML concept, defined by W3C Recommendation.

Since March, 2000, some of the industry software leaders join Pacific Edge Software [23] to define the Project Management XML Schema. Creating a flexible Business-to-Business schema, that enables project data exchange between Information Systems. Around this, Pacific Edge Software [23] had worked by developing the concept of project knowledge management (PKM) solutions for project-driven organizations. Companies like eProject.com, Great Plains, Onyx Software, PlanView, Primavera Systems [19] already start to use PMXML model [21].

The coalition of collaboration technology, e-business solutions, customer relationship management (CRM), project management, workforce management and PKM companies will work together to forge an open industry standard by making modifications and extensions to PMXML schema [21].

The schema and its source already can be viewed at BizTalk web site [24].

The goal of the XML schema design is to enable project management tools 'talk' and thus 'understand' each other; as a result (is it maintained) they can exchange in-

formation regarding task and project status, resource assignments, additional project attributes, and the work involved to complete these projects.

In 2002, organizations like NASA, Oracle, PM Boulevard LLC, and PM Solutions have recently joined the PMXML initiative [19]. Also the Yahoo Group (pm_xml - Project Management XML) group are developing is Project Management XML (PMXML) schema [21].

Historically, however, project management has been treated as ancillary to a normal business practice. Successful enterprises need to share project management information such as status, resource allocations, scheduling, and costing. But, for innovative projects, including R&D projects, as more innovative the tasks are, more difficult is to foresee with some severity the time and necessary resources do to it. In the actual processes of project management, such as Enterprise Project Management (EPM) [10] or Global Project Management (GPM), these characteristics, would become (planning and execution) practically impossible to control. In Microsoft Project Server 2003 [9], we solve with easiness the problem of the resources contribution in geographically distant places, but does not solve the management problem when necessary knowledge are need for the solution on emergent problems, because these management systems are based on the local control of the workmanship evolution determined by the empirical knowledge of the Project Manager and not based on a network knowledge management. In these cases, Project Manager cannot enter the time, the resources and costs of tasks with techniques that still are not known. Normally projects with these particularities are developed without an end "on the sight". Project Manager know when project pulls out, but do not make the minim idea at least if it goes to be concluded. This type of situation creates a feeling of lack of control on all the elements, generating inevitably the loss of motivation.

To solve that, Superior School of Technology and Management from Polytechnic Institute of Portalegre create a workgroup to research and develop a solution to implement in projects with this kind of characteristics. To this concept, we name it “GPRM – Global Project for Research Management” [1].

The main goal to GPRM [1] is to provide a sufficient intelligence to help the project manager determining the correct project plan. Instead of Project Manager determine the duration, cost or even how many resources is necessary to each task, GPRM [1] will provide all of this information automatically.

To create an Intelligent Project Framework we need to do much more than PMXML. We already assume PMXML [21] schema as the beginning of our work. And to communicate the project information from a project information system to another, PMXML [21] schema it’s sufficient. But PMXML [21] will never give us the possibility to achieve our goal. That’s why we propose “PML – Project Markup Language” schema, as an extension of PMXML [21] for innovate and research projects. Also, we are including almost the entire PMXML [21] schema inside of PML.

So, the main goal for this paper is to propose an effective schema that provides software solutions like GPRM [1] to use a real Project Knowledge Management (PKM).

2 The PMXML solution

The PMXML [21] standard is a data definition for project management systems. The PMXML Consortium [21] maintains the standard. PMXML [21] was originally created by Pacific Edge Software Inc. in 2000 and published at BizTalk.org [24].

PMXML [21] is an important step for the Project Management industry, following initiatives in other industries to provide integration mechanisms on application-level. Project management systems that are PMXML-enabled allow distributing data easily inside organizations and between different organizations and their respective management systems. The standard can help vendors, who don't have to implement so many different import/export filters, and it can help the users, who can transfer PM data to other systems (e.g., HR, ERP) or share it with partners more easily.

A standard that allows easy sharing and transfer of project management data could help to bring project management out of its relative isolation in comparison to the line management.

3 The PMXML Data Definition

The PMXML data definition is already on the version 2. It contains definitions for four major project management data types and some minor ones. The four major data types are: project, resource, task, and assignment. The definition starts with a ProjectManagementSchema as a root element. It contains InstanceData, a collection of user- and application-specific data; PoolResources, a collection of resource definitions; Projects, a collection of project definitions.

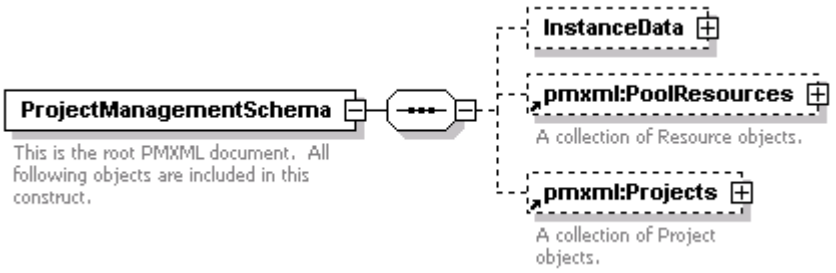


Fig. 1. The ProjectManagementSchema.

The InstanceData element is collection application-specific data that helps to interpret the project management data. It contains fields such as user ID and name, application name and version, and used schema version.

PoolResources is a collection of resource objects. These resource objects define material or working resources. They contain:

- Name and address data
- Cost data (rates, overtime cost, etc)

- Availability data
- Plan and baseline data
- Actuals
- Metrics (ACWP, BCWP, etc) and variances
- Customization and extension data

Projects are a collection of project objects. A project comprises:

- General data (name, company, status date, etc)
- Definitions for calculations (hours per day, days per month, etc)
- Plan and baseline data
- Actuals
- Metrics (ACWP, BCWP, etc) and variances
- Customization and extension data
- Resources (actually two collections, one for all resources, one for al local resources)
- Tasks
- Assignments

Task and assignment are structured similarly. Besides the usual groups for plan, baseline, actual data etc. the task definition contains attributes to define a milestone, a summary task and completion status. The assignment repeats also the plan, baseline etc.

```
<?xml version ="1.0" encoding="UTF-8"?>

<Schema name = "ProjectManagementSchema" xmlns =
"urn:schemas-microsoft-com:xml-data" xmlns:dt =
"urn:schemas-microsoft-com:datatypes">

  <description>The Project Management schema is de-
signed to provide an accurate summary of detailed pro-
ject data such as cost, schedule, and resource informa-
tion for a particular project.  The schema represents
the data most commonly used by project management soft-
ware which is essential for accurate communication of
project status.</description>

  <AttributeType name="ProjectID" dt:type="int" />
  <AttributeType name="Name" dt:type="string" />
  <AttributeType name="Title" dt:type="string" />
  <AttributeType name="Description" dt:type="string" />

<!--  here must include all of project attributes  -->
</Schema>
```

Schema Source 1. Part of a ProjectManagement Schema Source (PMXML)

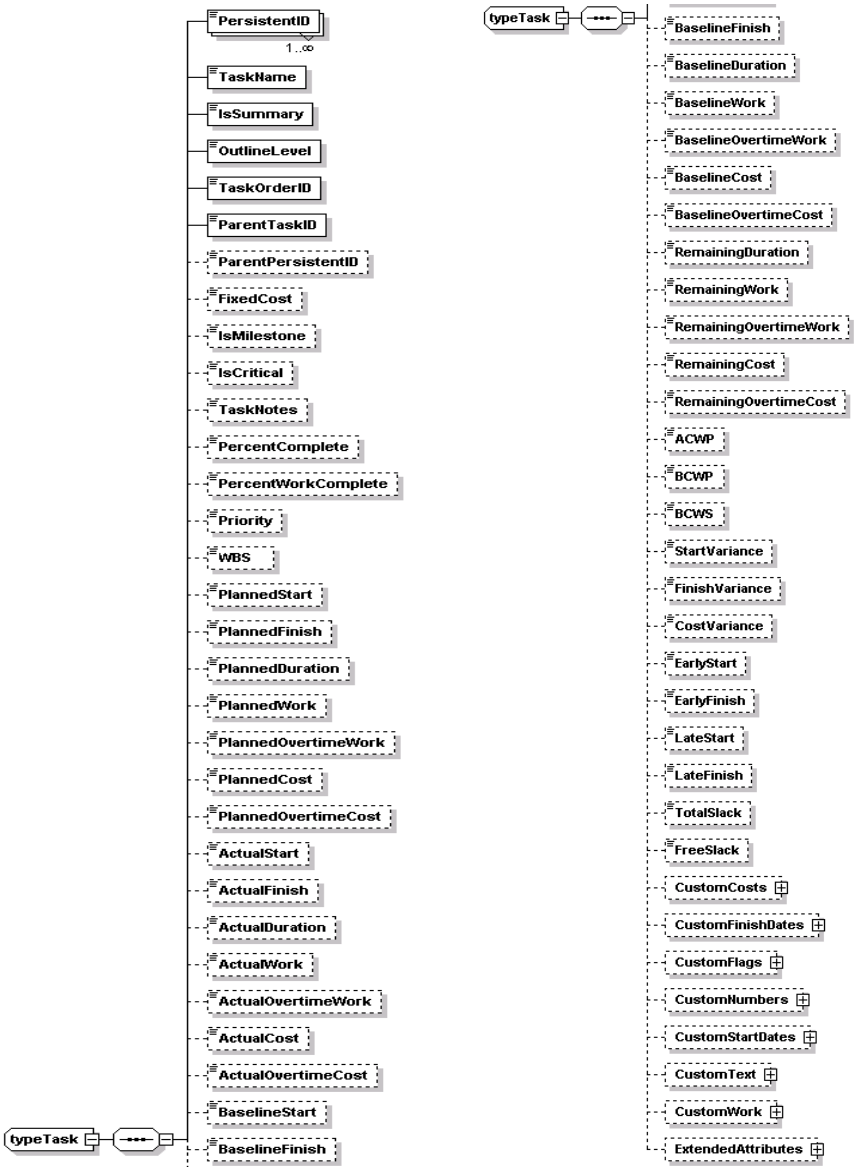


Fig. 2. PMXML version 2, task definition.

4 The PML solution

Despite PMXML [1] solve on a sufficiently and satisfactory way, the allotment of information in projects, tasks and resources between project systems and easy communication from/to any point of the world, continues to represent only a global and open solution.

In relation to the Project Knowledge Management (PKM), we have already a significant evolution with the introduction of a goal-giving system, a research of knowledge engine named as SPEAK (Search Process of Engineering for Assimilation of knowledge) [25].

SPEAK [25] is able to understand where it will find the information that it needs. But unhappily, the SPEAK [25] is not one human being. Therefore, we know that SPEAK [25], to have more success, has to know the semantic and grammar equivalent to the one of the Project Manager. When the Project Manager or the Project Team Leader write task description like:

"*Michael must develop video capture software with Visual Studio.Net 2005, in the University*"

We cannot assure that the same phrase is not written in hundreds of different forms, with the same felt. But most serious problem is that the description of a task could have a completely different interpretation from who it wrote to who reads it. This problem of communication exists between the human beings. Therefore, between one human being and a machine we have millions of more potential errors.

These possibilities inhibit for complete the use this kind of artificial intelligence search engines. In project management, if does not exist reasons to trust on the calculated information, then never will be taken in consideration.

Therefore, as the PMXML [21] do not solve the main problem of the GPRM [1]. For the GPRM [1], it is not important in which provider the specific task information came. What matters is to know, with the biggest severity, which is the esteem average time for the duration of one determined task. And the lesser time? And the greater? Which is the amount and quality of the sampling?

All of this information is important to determine:

- Calculate the relative search
- Rigorous information
- Security and quality of information
- Quality of the supplier of information
- Other pertinent questions...

For such we have to look for correct information. The goal of this project is not to place one human being reading many pages of information, to take later one decision based in what it read. Also we do not intend that the Project Manager be the only detainer of the knowledge, placing in risk the proper project.

What it is really intended is to conceive a proper grammar for information treatment of the project management. This proper grammar gives the possibility to have a correct understanding between machines and humans.

After collecting on a database thousands of names of tasks typed by humans and used in many projects with Microsoft Project, we analyze this data and we verify that, of one everything forms generality has a similar semantic structure.

For example, when we intend to define our task, we use:

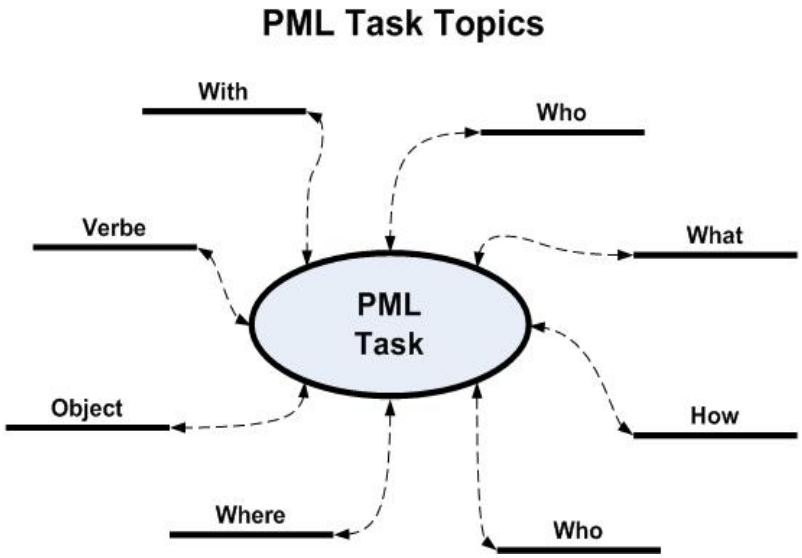


Fig. 3. PML Task Topic Elements

Any meta-information on a task, it will answer to:

- Verb (defines what it is really gone to do)
- Object (what it goes to be made through the verb)
- Where?
- How? In this case we can eventually come back to sub-tasks to explain his composition.
- With who?
- Where?

This defines the “What” to do Task to any task project.

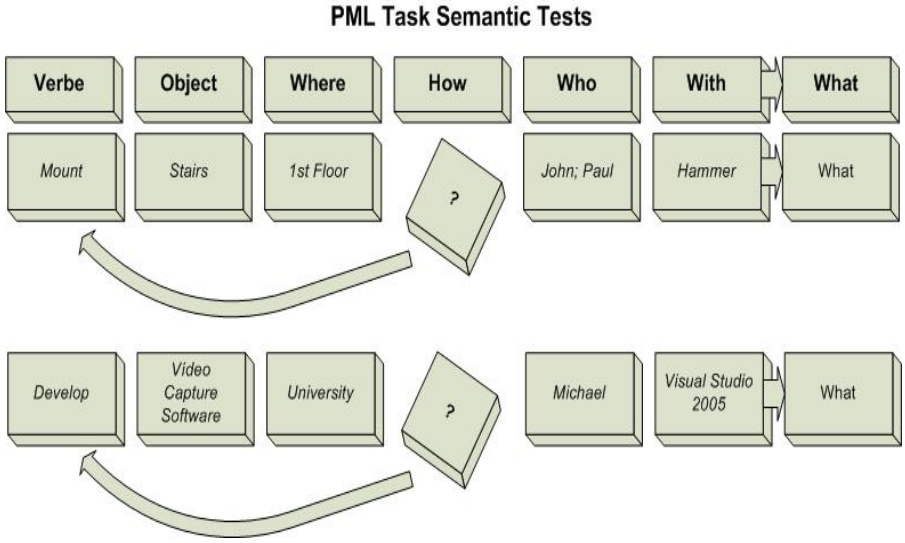


Fig. 4. PML Task algorithmic graphic representation

```
<?xml version ="1.0" encoding="UTF-8"?>

<Schema name = "PML" xmlns =
“http://www.estgp.pt/gprm/pml” xmlns:pmxml =
"urn:schemas-microsoft-com:xml-data" xmlns:dt =
"urn:schemas-microsoft-com:datatypes"> <description>
The PML schema is designed to provide a better IA
search. PML is designed to be used by SPEAK or similar
systems, to provide a realy Project Knowledge Manage-
ment (PKM) </description>

<!-- ////////////////////////////////////////// -->

<element name="Task" dt:type="int">

<sequence>
<element name="How"> <!-- /// … /// --> </element>
<element name="Who"> <!-- /// … /// --> </element>
<element name="With"> <!-- /// … /// --> </element>
</sequence>

<AttributeType name="Verbe" dt:type="string" minOc-
curs=1 maxOccurs=”unbounded” />
<AttributeType name="Object" dt:type="string" minOc-
```

```
curs=1 maxOccurs=”unbounded” />
<AttributeType name="Where" dt:type="string" minOc-
curs=1 maxOccurs=”unbounded” />

<!-- ////////////////////////////////////////// -->

</Schema>
```

Schema Source 2. Part of a Project Markup Language Schema Source (PML)

With our approach we want to make PML an extension to PMXML. This extension will be a way to conquer the WEB distributed projects including search engines. To make something, and to keep it useful and ship, we need to keep it simple. What we need is an engine to give us statistical values about projects. That engine only will be useful with simple data structures. If we want to search in very complicated structures we will get lost. PML is for us like DUBLINCORE [26] is for bibliographic descriptions. We want to find a set of primitives to describe an entire project with simplicity. These six primitives are based on the six primitives of Zachman Framework [27], redimensioned to Project Management reality, being the PM primitives.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <pml:PML xmlns:pml="http://schemas.estgp.pt/xsd/grpm/pml/1.0/"
    xmlns="http://www.pacificedge.com/PMXML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
+ <ProjectManagementSchema>
- <pml:PMLCore TaskID="1">
    <pml:Verb>Develop</pml:Verb>
    <pml:Object>Video Capture Software</pml:Object>
    <pml:Where>University</pml:Where>
  - <pml:How>
    - <pml:PMLCore>
        <pml:Verb>Develop</pml:Verb>
        <pml:Object>Camera Driver</pml:Object>
      </pml:PMLCore>
    - <pml:PMLCore>
        <pml:Verb>Develop</pml:Verb>
        <pml:Object>MPEG Movie Builder From Driver</pml:Object>
      </pml:PMLCore>
    </pml:How>
    <pml:Who>Michael</pml:Who>
    <pml:With>Visual Studio 2005</pml:With>
  </pml:PMLCore>
</pml:PML>
```

Fig. 5. A PML Sample

5 Conclusions and future work

PML it was drawn as an element of a complete framework solution (GPRM) to achieve the real Project Knowledge Management (PKM). To do that we are integrating PMXML inside PML and testing the communication between SPEAK Servers and GPRM Application.

We search for the capability to solve the problem of the knowledge management and new problems evaluation.

Soon we will develop better versions and extensions for PML. This first framework solution (GPRM+SPEAK+SUSI+PML) should be allow documents we will identify individually each one of the problems and also the correspondent solutions.

In this project we will fallow the Tim Berners-Lee vision, working in web semantic solution for PML/SPEAK. We want to test an implementation of the RDF (Resource Description Framework).

References

1. Paulo Gomes, ***"GPRM (Global Project Research Management) Solution Overview",*** ICEEEM - International Congress of Energy and Environment Engineering and Management, ISBN: 84-934089-9-9.
2. Paulo Gomes, Jorge Machado, ***"GPRM: General Platform Presentation"***, Superior School of Technology and Management, Polytechnic Institute of Portalegre.
3. Paulo Gomes, Jorge Machado, ***"Client Interface and connection to other applications"*** - Superior School of Technology and Management – Polytechnic Institute of Portalegre.
4. Paulo Gomes, Jorge Machado, ***"GPRM Dictionary definition and inter-connection of their entities"*** - Superior School of Technology and Management, Polytechnic Institute of Portalegre.
5. Paulo Gomes, Jorge Machado, ***Instantiation, evolution, behavior e learning in GPRM.*** - Superior School of Technology and Management, Polytechnic Institute of Portalegre.
6. Paulo Gomes, Jorge Machado, ***"Normalization and Internationalization of technical terms in GPRM"***. - Superior School of Technology and Management, Polytechnic Institute of Portalegre.
7. José Borbinha, Jorge Machado, ***"Mitra: A semantic web indexer"***, Instituto Superior Técnico, Universidade Técnica de Lisboa, 2005.
8. José Borbinha, Jorge Machado, ***"DEPTAL: A Framework for Institutional Repositories"***, DELLOS workshop, Crete, Greece, 2005 http://deptal.estgp.pt/4
9. Microsoft. ***"Planning, Implementing and Mantaining a Microsoft Windows Server 2003 Active Directory"***. Microsoft MOC 2279, 2003
10. Microsoft. ***"Planning, Deploying and Managing an Enterprise Project Management Soluction"***. Microsoft MOC 2732A, 2003
11. Shtub, A. / Bard, J. / Globerson, S. ***"Project Management Engineering, Technology and Implementation"***, Ed. Prentice Hall, 1994.
12. http://www.gprm.org, Global Project for Research Management official web site
13. http://mitra.bn.pt, Semantic Indexer of described web contents, National Library of Portugal
14. http://www.openarchives.org, OAI-PMH. Open Archives Initiative - Protocol for Metadata Harvesting
15. http://www.adlnet.org/index.cfm?fuseaction=scormabt, SCORM. The Sharable Content Object Reference Model
16. http://www.loc.gov/z3950/agency, Z39.50. Z39.50 Maintenance Agency
17. http://www.loc.gov/z3950/agency/zing/srw, SRU Search and Retrieve URL
18. http://www.udcc.org/mrf.htm, Unitary Decimal Classification

19. http://xml.coverpages.org/MembersJoin-PMXML20020603.html, Announcement 2002-06-03: ***"Pacific Edge Software Adds Industry Leaders to PMXML Consortium. NASA, Oracle among New Members of PMXML Standards Organization."***
20. http://www.pacificedge.com/xml/, PMXML website
21. http://groups.yahoo.com/group/pm_xml, Project Management XML Discussion Group
22. Andrew Goloboy and David A. Shiang**,*"Pacific Edge's Project Office: Improving Enterprise Initiatives."***, IDC White Paper, 2000.
23. http://www.pacificedge.com/, Pacific Edge Software web site
24. http://www.biztalk.org, Microsoft BizTalk web site
25. Paulo Gomes, Jorge Machado, ***"SPEAK (Search Process of Engineering for Assimilation of knowledge), A generic framework to search in Metadata"***, Superior School of Technology and Management, Polytechnic Institute of Portalegre.
26. http://dublincore.org, The Dublin Core Metadata Initiative.
27. Zachman, John, "Enterprise Artifacts vs. Application Artifacts".

Extensão do XQuery com operações de selecção para a construção interactiva das perguntas

Alda Lopes Gançarski[1] and Pedro Rangel Henriques[2]

[1] Universidade do Minho, Braga, Portugal
email: aldalopes@di.uminho.pt
Membro do LIP6, Université Paris 6, Paris, França
[2] University do Minho, Braga, Portugal
email: prh@di.uminho.pt

Abstract. XQuery é a linguagem de interrogação proposta pelo W3C para XML usando restrições estruturais e sobre o conteúdo. XQuery está a ser complementado com a linguagem Full-Text para realizar operações sobre textos, tratando-os como uma sequência de palavras, sinais de pontuação e espaços. Dada a natureza complexa das perguntas estruturadas do XQuery, propõe-se, neste artigo, uma extensão para permitir a *selecção* do subconjunto interessante de elementos de cada resultado intermédio das perguntas. Os resultados intermédios são, portanto, acessíveis ao utilizador durante a construção das perguntas, o que torna mais fácil obter o resultado desejado. As operações de selecção são formalmente definidas, estendendo a gramática do XQuery e definindo novas funções. Estas definições podem ser usadas para construir um sistema de processamento adequado.

1 Introdução

O XQuery [4] é a proposição do W3C como linguagem de interrogação padrão para XML. Esta linguagem utiliza, entre outras funções, a definição de caminhos (*paths*), através da linguagem XPath [3], para aceder a elementos ou atributos dos documentos. Surgiram, entretanto, trabalhos que estendem estas linguagens com operações de similaridade textual da Recuperação de Informação tradicional [2], propondo métodos de cálculo de relevância, como os apresentados em [5]. Uma operação de similaridade textual consiste em verificar se um texto aborda um assunto expresso em linguagem natural. O resultado dessa verificação é a *relevância* do texto, normalmente um valor no intervalo [0, 1].

O W3C propõe a linguagem *Full-Text* como um complemento ao XQuery e ao XPath que inclui a possibilidade de associar um *score* (ou relevância) a uma expressão que verifica se uma dada frase existe no conteúdo de um elemento ou atributo. O cálculo da relevância fica a cargo da aplicação.

Contudo, a dificuldade na construção das perguntas estruturadas levou à definição da linguagem de interrogação interactiva IXDIRQL [6]. Esta linguagem tem por base o XPath, estendendo-o, não só com operações de similaridade

textual, mas também com um paradigma interactivo/iterativo de construção das perguntas. Com este paradigma, cada operação conduz a um resultado intermédio que o utilizador pode consultar para decidir acerca da próxima operação a efectuar, da alteração de alguma operação já introduzida ou da selecção do subconjunto de elementos interessantes dos resultados intermédios, até especificar a pergunta que conduz ao resultado desejado. Foi construído um protótipo para o processamento da IXDIRQL. Este protótipo foi posto ao dispor de utilizadores reais [7] para se verificar o seu funcionamento correcto e a correcta utilização das operações de selecção por parte dos utilizadores face a determinados pedidos de informação. Em [8] é sugerido estender o XQuery com o paradigma interactivo/iterativo de construção das perguntas, incluindo as operações de selecção e propondo um método para o cálculo das relevâncias. O presente artigo propõe uma definição formal das operações de selecção que pode ser usada para construir um sistema de processamento adequado. Esta definição tem por base as definições do XQuery e do Full-Text apresentadas em [4] e [1], respectivamente, seguindo, portanto, o mesmo formalismo.

O artigo é organizado da seguinte forma. A secção 2 introduz as linguagens XQuery e Full-Text. Depois, as secções 3 e 4 definem as operações de selecção *select* e *judgeRel*, respectivamente. A secção 5 propõe o processamento incremental para o XQuery estendido. Finalmente, é feita uma conclusão e são dadas algumas directivas para trabalho futuro.

2 As linguagens *XQuery* e *Full-Text*

O XQuery é formado por vários tipos de expressões, incluindo o XPath e expressões *for..let..where..order_by..return* (FLWOR) baseadas nas linguagens típicas de interrogação das bases de dados, como o SQL. Para passar informação de um operador para outro são usadas variáveis. Por exemplo, seja um documento sobre artigos que inclua o seu título, o seu autor e a sua editora. A pergunta que se segue recupera artigos do autor *Silva* ordenados pelo título respectivo[3]:

```
for $a in doc("http://...") /artigos/artigo
where $a/autor = "Silva"
order by $a/título
return $a
```

O XQuery opera sobre a estrutura lógica e abstracta dos documentos. O modelo de dados correspondente representa os documentos como árvores onde os nós podem corresponder a documentos, elementos, atributos, textos, espaços de nomes, instruções de processamento ou comentários. Cada nó tem um identificador único.

[3] Para simplificar, ao longo do artigo, algumas expressões e símbolos opcionais são subtituídos por "...".

A linguagem Full-Text estende o XQuery com expressões *ftcontains* e com variáveis de tipo *score* dentro das expressões FLWOR. A função *ftcontains* pode ser usada sempre que uma comparação é possível, como a igualdade. Uma expressão *ftcontains* inclui um caminho, para especificar os nós onde a função será aplicada, e a expressão com as frases (expressões em linguagem natural) a procurar no conteúdo desses nós. O valor lógico verdade é retornado se algum nó inclui a frase procurada. Por exemplo, a pergunta seguinte retorna o(s) autor(es) dos artigos cujo título contém *"linguagem XML"*.

```
for $a in doc("...") /artigos/artigo
where $a/título ftcontains "linguagem XML"
return $a/autor
```

Uma variável *score* guarda a relevância associada a uma expressão que consiste numa combinação Booleana de expressões *ftcontains*. Trata-se, portanto, de uma operação de similaridade textual. A variável que guarda a relevância fica associada a um valor de tipo *xs:float* (o espaço de nomes *xs* refere-se ao *XML Schema*) no intervalo [0, 1], um valor maior indicando uma maior relevância. A forma como a relevância é calculada é dependente da implementação. Por exemplo, a pergunta seguinte retorna artigos ordenados pela relevância do seu título em relação à expressão *"XML"*.

```
for $a score $s in doc("...") /artigos/artigo[título ftcontains "XML"]
order by $s
return $a
```

3 A função *Select*

O paradigma interactivo de construção de perguntas é baseado nas operações de selecção. Estas consistem na restrição dos resultados intermédios ao subconjunto de elementos interessantes. A selecção é feita nos caminhos através da função *mf:select* (o espaço de nomes *mf* é associado neste artigo às funções criadas).

A selecção do conjunto de elementos interessantes é feita segundo um determinado critério. Enquanto num filtro do XPath o conjunto de elementos é especificado por intenção, na função *mf:select* é-o por extensão, i.e., referindo explicitamente cada elemento. Isto pode ser interessante quando a especificação do critério é demasiado complicada (o utilizador pode, inclusivé, não saber fazê-lo) ou quando é mais rápido/eficiente referir os elementos desejados directamente.

Suponha-se que cada nó é identificado por uma palavra. A entrada de *mf:select* é um nó e uma lista de identificadores do nós; a saída é o nó dado como entrada se ele corresponde a um dos identificadores da lista de entrada. Por exemplo, suponha-se que o utilizador quer as referências bibliográficas dos artigos interessantes do autor *"Silva"*. O termo *interessante* pode-se referir, entre outras coisas, ao título, aos co-autores, à editora, à data de publicação ou ao tamanho. A per-

gunta pode, então, ser:

for $a in doc("...")/artigos/artigo[autor = "Silva"][mf:select(., ("a4", "a8"))]
return $a//referência

Nesta pergunta, a função *mf:select* selecciona os artigos identificados por *"a4"* e *"a8"*. Dada a natureza interactiva da função *mf:select*, a pergunta é escrita em três passos:

1. O utilizador especifica a cláusula *for* com o caminho que retorna os artigos do autor *"Silva"*: *for $a in doc("...")/artigos/artigo[author = "Silva"]*

2. Ao analisar a lista de artigos dada pelo caminho, o utilizador selecciona os artigos que lhe interessam usando *mf:select*:
for $a in doc("...")/artigos/artigo[autor = "Silva"][mf:select(., ("a4", "a8"))]

3. O utilizador completa a pergunta com a cláusula *return*:
for $a in doc("...")/artigos/artigo[autor = "Silva"][mf:select(., ("a4", "a8"))]
return $a//referência

Apesar de *mf:select* receber uma lista de identificadores de nós, o utilizador não é obrigado a conhecê-los se houver uma interface adequada para visualizar dos resultados intermédios. Esta interface deve permitir a selecção dos elementos usando, por exemplo, um botão associado a cada elemento. O sistema deve, então, escrever automaticamente os identificadores dos nós seleccionados na pergunta em construção no editor das perguntas.

3.1 Definição da sintaxe

O argumento de *mf:select* é um nó e uma lista de identificadores que consistem em nomes, i.e., cadeias de caracteres excluindo espaços. Esta definição está de acordo com a produção 91 da gramática do XQuery apresentada em [1], produção essa que permite derivar chamadas a funções:

[91] FunctionCall ::= <QName "("> (ExprSingle ("," ExprSingle))? ")"*

Nesta produção, o nome da função é derivado por *QName*. Este símbolo deriva nomes possivelmente associados a espaços de nomes. No caso da função *mf:select*, *QName* deriva o nome da função *select* associado ao espaço de nomes *mf*.

Cada argumento de uma função é derivado pelo símbolo *ExprSingle*. Este símbolo deriva numa expressão qualquer do XQuery, incluindo expressões que definem nós ou listas de valores do tipo *xs:string*. Então, os argumentos da função *mf:select* são derivados por *ExprSingle*. Cada identificador de nó é do tipo *xs:string*. Por sua vez, um valor de tipo *xs:string* é derivado por *StringLiteral*, a partir de *ExprSingle* após uma série de passos de derivação.

3.2 Definição da semântica

Seja *IdNodeTab* uma tabela mantida pelo sistema que faz corresponder a cada nó o seu identificador. Sendo *node()* o tipo de um qualquer nó, tem-se:

IdNodeTab : xs:string × *node()*

A função *mf:select* pode, então, ser definida por:

```
declare function mf:select($contextNode as node()?, $selectedIds as xs:string*)
as node( )?
{ for $s in $selectedIds
  let $n := IdNodeTab[$s]
  if ($contextNode=$n) then return $contextNode else return ( )
}
```

Nesta definição, *$selectedIds* é uma variável que contém a lista de identificadores do tipo *xs:string* dos nós explicitados como argumentos da função *mf:select*. A variável *$contextNode* guarda o nó que será retornado se o seu identificador (dado pela tabela *IdNodeTab*) se encontra na lista *$selectedIds*.

4 O operador *judgeRel*

O operador *judgeRel* selecciona o subconjunto de elementos julgados relevantes pelo utilizador na lista ordenada resultante duma expressão *ftcontains* associada a uma variável *score*. Seja a seguinte pergunta:

```
for $a score $s in doc("...")/artigos/artigo[título ftcontains "XML"]
order by $s
return $a//referência
```

Esta pergunta retorna a lista de referências ordenadas pela relevância do título do artigo onde elas são citadas. As referências retornadas correspondem ou não a títulos realmente relevantes, uma vez que a lista é baseada em relevâncias estimadas pelo sistema. Com o operador *judgeRel*, o utilizador pode seleccionar os elementos relevantes durante a construção da pergunta, analisando a lista ordenada dada pela função *ftcontains*. Desta forma, a relevância associada a elementos relevantes fica 1 e não relevantes fica 0. Estes novos valores de relevância são tidos em conta no cálculo final do *score*. Na pergunta precedente, suponha-se que o título identificado por *"t4"* é seleccionado quando o utilizador analisa a lista ordenada dada por *ftcontains*. A pergunta passa, então, a ser:

```
for $a score $s in
        doc("...")/articles/article[título ftcontains "XML" judgeRel ("t4")]
order by $s
```

return $a/referência

Aqui, as referências retornadas pela cláusula *return* pertencem a artigos onde o título é seguramente relevante (o utilizador julgou-o relevante e seleccionou-o) e podem ser usadas para processamento posterior. Como para a função *mf:select*, dada a natureza interactiva do operador *judgeRel*, esta pergunta é escrita em três passos:

1. O utilizador especifica a cláusula *for* e a expressão *ftcontains*:
   for $a score $s in doc("...")/artigos/artigo[título ftcontains "XML"]

2. A lista ordenada resultante de *ftcontains* dá ao utilizador um bom ponto de partida para escolher os títulos relevantes. Ao analisá-la, o utilizador insere o operador *judgeRel* com os títulos relevantes:
   for $a score $s in
   doc("...")/artigos/artigo[título ftcontains "XML" judgeRel ("t4")]

3. O utilizador escreve as cláusulas *order by* e *return* para obter a lista final de referências:
   for $a score $s in
   doc("...")/artigos/artigo[título ftcontains "XML" judgeRel ("t4")]
   order by $s
   return $a/referência

Da mesma forma que para a função *mf:select*, a janela que mostra a lista ordenada deve permitir ao utilizador seleccionar os elementos relevantes, não tendo de conhecer os identificadores dos nós.

4.1 Definição da sintaxe

O operador *judgeRel* tem de ser incluído na gramática da linguagem Full-Text apresentada em [1]. Nesta gramática, as produções 35, 37, 38 e 51 derivam a cláusula *for*, uma variável de tipo *score*, a cláusula *let* e a expressão *ftcontains*, respectivamente:

[35] ForClause ::= "for" "$" VarName ... FTScoreVar? "in" ExprSingle ...
[37] FTScoreVar ::= "score" "$" VarName
[38] LetClause ::= (("let" "$" VarName ... FTScoreVar?) |
("let" "score" "$" VarName)) ":=" ExprSingle ...
[51] FTContainsExpr ::= RangeExpr ("ftcontains" FTSelection
FTIgnoreOption?)?

Na produção 51: *RangeExpr* deriva a expressão que conduz à lista de nós onde *ftcontains* será aplicada; *FTSelection* deriva combinações booleanas de frases a procurar e opções de combinação (*match options*), tais como *case sensitive*; *FTIgnoreOption* permite a exclusão de certos nós dentro do conjunto retornado por

RangeExpr, conduzindo ao conjunto final de nós chamado *contexto de procura*. Nas produções 35 e 38, a variável *score* guarda a relevância associada à expressão derivada por *ExprSingle*. *ExprSingle* permite derivar uma expressão qualquer do XQuery. Contudo, as expressões associadas às variáveis de *score* são restringidas a combinações Booleanas de expressões *ftcontains*, envolvendo apenas "and" e "or". Consequentemente, propõe-se a substituição de *ExprSingle* nas produções 35 e 38 pelo símbolo *ScoreExpr*, obtendo-se:

[35] ForClause ::= "for" "$" VarName ... (FTScoreVar "in" ScoreExpr | "in" ExprSingle) ...
[38] LetClause ::= ("let" "$" VarName ... (FTScoreVar ":=" ScoreExpr | "=:" ExprSingle) | "let" "score" "$" VarName ScoreExpr) ...

O símbolo *ScoreExpr* deriva a combinação Booleana de expressões *ftcontains* através das produções seguintes:

ScoreExpr ::= ScoreOrExpr
ScoreOrExpr ::= ScoreAndExpr | ScoreAndExpr "or" ScoreOrExpr
ScoreAndExpr ::= ScoreExprUnit | ScoreExprUnit "and" ScoreAndExpr
ScoreExprUnit ::= "(" ScoreExpr ")" | RangeExpr ("ftcontains" FTSelection FTIgnoreOption? JudgeRelExpr?) ?

Assim como na gramática do XQuery especificada em [1], estas produções reflectem a precedência dos operadores. Os operadores definidos a mais baixo nível são os de maior precedência. Os símbolos *ScoreOrExpr* e *ScoreAndExpr* derivam um "or" e um "and", respectivamente. O símbolo *ScoreExprUnit* deriva a expressão *ScoreExpr* entre parêntesis ou deriva uma expressão *ftcontains* associada a variáveis *score*. Esta última expressão é similar às derivadas pela produção 51 mas agora aumentadas com o operador opcional *judgeRel*. O símbolo *JudgeRelExpr* deriva o operador *judgeRel* na seguinte produção:

JudgeRelExpr ::= "judgeRel" "(" StringLiteral ")"*

Nesta produção, *StringLiteral* deriva o identificador de nó, como referido na secção 3.1. *judgeRel* é, então, associado ao conjunto de identificadores de nós julgados relevantes pelo utilizador[4].

4.2 Definição da semântica

O operador *judgeRel* está incluído nas expressões associadas a variáveis *score*. Assim, a sua definição semântica é dada em conjunto. Contudo, a definição dessas expressões não pode ser feita em termos de XQuery porque requer a presença de funções de segunda ordem, i.e., funções que não avaliam os seus

[4] Pode acontecer que o utilizador não encontre elementos relevantes, sendo a lista de identificadores vazia.

argumentos como expressões regulares do XQuery, mas usam-nos de forma interpretada. Em [1] assume-se que existe a função de segunda ordem *fts:score* (*fts* refere-se a *Full-Text semantics*) que aceita como argumento uma expressão *ScoreExpr* e retorna a relevância (*score*) desta expressão. Dada esta função, a expressão genérica *score $var as ScoreExpr* é avaliada como se fosse substituída por *$var:=fts:score(ScoreExpr)*. Propõe-se, aqui, a seguinte definição da função *fts:score*:

```
declare function fts:score($e as xs:string) as xs:float
{
1. if (mf:operatorScore($e) = "or") then
2.         mf:scoreOr(fts:score(mf:operandLeftScore($e)),
3.                    fts:score(mf:operandRightScore($e)))
4. else if (mf:operatorScore($e) = "and") then
5.         mf:scoreAnd(fts:score(mf:operandLeftScore($e)),
6.                     fts:score(mf:operandRightScore($e)))
7.       else
8.               let $s := mf:nodeList($e)
9.               return
10.                     if mf:includesJudgeRel($e) then
11.                          let $j := mf:judgeRelIds($e)
12.                          let $i := { for $a in $j return IdNodeTab[$a] }
13.                          return mf:scoreJudgeRel($s, $i)
14.                     else
15.                          let $m := mf:matchExpr($e)
16.                           return mf:scoreFTContains($s, $m)
}
```

O argumento de *fts:score* é uma cadeia de caracteres correspondente à expressão derivada pelo símbolo *ScoreExpr* definido na secção 4.1. A função retorna um valor de tipo *xs:float*.

Devido às chamadas recursivas da função *fts:score* (linhas 2, 3, 5, 6), a relevância é calculada, primeiro, para cada expressão *ftcontains*, e, depois, para cada operador Booleano da expressão *ScoreExpr*, respeitando a precedência dos operadores, até se obter o resultado final.

Na linha 1, a função *mf:operatorScore* aceita uma expressão *ScoreExpr* e retorna o primeiro operador a avaliar: "and", "or" ou nenhum. Dependendo do operador, acções diferentes são tomadas. Se o operador é um "or" (linha 1), a relevância é dada pela função *mf:scoreOr* tendo como argumentos a relevância dos operandos do "or" (linhas 2 e 3). Estes operandos são dados pelas funções *mf:operandLeftScore* e *mf:operandRightScore*. Se o operador é um "and" (linha 4), uma acção similar é efectuada, sendo a relevância desta vez dada pela função *mf:scoreAnd* (linhas 5 e 6).

Se não houver nenhum operador Booleano, a relevância de uma expressão *ftcontains* derivada pelo símbolo *ScoreExprUnit* (definido na secção 4.1) é calcu-

lada pelas acções das linhas 7 a 16. A variável *$s* guarda a lista de nós do contexto de procura (linha 8). Esta lista é retornado pela função *mf:nodeList* que tem em conta, não só a expressão derivada por *RangeExpr* (apresentado na secção 4.1), mas também a expressão derivada por *FTIgnoreOption* (também apresentado na secção 4.1). A existência de um operador *judgeRel* é, então, verificada pela função *mf:includesJudgeRel* analisando a expressão *ScoreExpr*. Se existe o operador (linha 10), as acções seguintes são tomadas. A variável *$j* guarda os identificadores de nós julgados relevantes pelo utilizador (linha 11). Estes são dados pela função *mf:judgeRelIds* que recebe a expressão *ScoreExpr*. Outra variável, *$i*, guarda os nós correspondentes aos identificadores julgados relevantes (linha 12). Estes nós são dados pela tabela *IdNodeTab* (apresentada na secção 3.1). A função *mf:scoreJudgeRel* aceita o contexto de procura (guardado em *$s*) e a lista de nós julgados relevantes (guardada em *$i*) e retorna a relevância da cláusula *score* (linha 13).

Se não houver o operador *judgeRel* na expressão *ScoreExpr* (linha 14), a variável *$m* guarda a combinação Booleana das frases a procurar (e respectivas *match options*) derivadas pelo símbolo *FTSelection* (apresentado na secção 4.1). Esta combinação é dada pela função *mf:matchExpr* (linha 15). Usando a variável *$m*, a função *mf:scoreFTContains* calcula a relevância associada aos nós do contexto de procura (guardado em *$s*) (linha 16).

As novas funções invocadas dentro de *fts:score* não são definidas aqui mais em detalhe. A maior parte (*mf:operatorScore*, *mf:operandLeftScore*, *mf:operandRightScore*, *mf:includesJudgeRel*, *mf:judgeRelIds*, *mf:matchExpr* e *mf:ignoreOption*) baseia-se numa simples análise léxico-sintáctica da expressão *ScoreExpr* para encontrar sub-expressões específicas. A função *mf:nodeList* retorna o contexto de procura. As restantes funções (*mf:scoreOr*, *mf:scoreAnd*, *mf:scoreJudgeRel* e *mf:scoreFTContains*) são dedicadas ao cálculo da relevância e devem ser definidas pela aplicação.

4.3 Exemplo de processamento de *fts:score*

Ao longo das explicações desta e das próximas secções, por simplicidade, as acções da função *fts:score* definida na secção 4.2 são referidas pelo número da linha respectiva. Também, cada nó é referido pelo respectivo identificador.

Seja, por exemplo, a seguinte pergunta:

```
for $a score $s in doc("...")/artigos/artigo[referência ftcontains "XML"
        and secção ftcontains "Aplicação XML" judgeRel ("s1")]
order by $s
return $a/título
```

Esta pergunta retorna títulos de artigos onde as referências abordam *"XML"* e as secções referem *"Aplicação XML"*. Os títulos resultantes são ordenados pela sua relevância. Suponha-se que o artigo *a1* foi encontrado na cláusula *for* e que tem secções *s1* e *s2* e referências *r1* e *r2*. Quando a função *fts:score* é executada, o operador Booleano "and" é detectado pela função *mf:operatorScore* (linha 4). Assim, a função *fts:score* é chamada recursivamente para cada operando do "and"

(linhas 5 e 6). Estes operandos são sub-expressões de *ScoreExpr* dados pelas funções *mf:operandLeftScore* e *mf:operandRightScore*. Os resultados dados por *fts:score* são usados como argumentos da função *mf:scoreAnd* para calcular o resultado final da relevância para o artigo *a1* (linha 5). Se houver mais artigos, uma relevância é calculada para cada um, usando de novo a função *fts:score*.

Para ambos os operandos do "and", a função *fts:score* é executada a partir da linha 7 porque não há mais operadores Booleanos. No que diz respeito ao primeiro operando, a função *mf:nodeList* retorna o contexto de procura *("r1", "r2")* que é guardado na variável *$s*. A função *mf:includesJudgeRel* verifica, então, que não existe o operador *judgeRel* (linha 14) e a execução continua na linha 15. Aqui, a função *mf:matchExpr* retorna a frase *"Aplicação XML"* a procurar guardada na variável *$m* (não há *match options*). Esta frase, juntamente com o contexto de procura (guardado em *$m*), é usada pela função *mf:scoreFTContains* para calcular a relevância final do primeiro operando do "and".

Em relação ao segundo operando, a linha 8 também é executada, conduzindo agora ao contexto de procura *("s1", "s2")*. Tendo este operando um operador *judgeRel*, as linhas 11 a 13 são executadas. O utilizador julgou relevante a secção *s1*. Este identificador é dado pela função *mf:judgeRelIds* (linha 11). O nó correspondente é, então, dado pela tabela *IdNodeTab* (linha 12). A relevância resultante é, finalmente, dada pela função *mf:scoreJudgeRel*, a partir do contexto de procura *("s1", "s2")* e da lista *("s1")* de secções julgadas relevantes dentro desse contexto (linha 13).

5 Processamento incremental

O ambiente de edição do XQuery estendido com operações de selecção deve permitir ao utilizador aceder aos resultados intermédios das perguntas. Além disso, o processador deve ser incremental para que, cada vez que uma operação é inserida ou alterada, apenas os resultados dependentes das alterações feitas são calculados. Um caso particular é o cálculo incremental da função *fts:score* (definida na secção 4.2) porque inclui muitas operações. Suponha-se, por exemplo, que o utilizador introduz a pergunta seguinte:

```
for $a score $s in
        doc("...")/artigos/artigo[título ftcontains "XML"]
```

O resultado da cláusula *score* é dado pela função *fts:score*. Como ainda não é incluído o operador *judgeRel*, a condição *else* da linha 14 é executada. Para um correcto acesso aos resultados intermédios, a lista resultante de títulos guardada na variável *$s* (linha 8) deve ser apresentada ao utilizador, juntamente com as relevâncias respectivas calculadas na linha 16 pela função *mf:scoreFTContains*. Face a esta lista, o utilizador julga relevantes os títulos *"t1"* e *"t4"*, ficando a pergunta da seguinte forma:

for $a score $s in
doc("...")/artigos/artigo[título ftcontains "XML" judgeRel ("t1", "t4")]

A função *fts:score* é executada de novo, passando agora pelas linhas 11 a 13 porque existe o operador *judgeRel*. O processamento incremental deve assegurar que todos os cálculos feitos antes destas linhas não serão efectuados de novo.

A criação de um processador incremental para XQuery estendido pode ser feita recorrendo a um gerador como o LRC [9]. O LRC é um gerador de ambientes incrementais baseados numa definição formal atributiva da linguagem a processar. Este gerador foi usado na criação do sistema de processamento da linguagem IXDIRQL [7].

6 Conclusão e perspectivas

Este artigo define uma extensão ao XQuery com operações de selecção para a contrução interactiva/iterativa das perguntas. Isto ajuda o utilizador, não só a escolher as operações que conduzem ao resultado desejado, mas também a restringir cada resultado intermédio ao subconjunto de nós que interessam o utilizador. A definição formal das operações de selecção aqui proposta pode ser usada para a construção de um sistema de edição e processamento incremental para o XQuery assim estendido. Como trabalho futuro, pretende-se criar um protótipo de um tal sistema usando o LRC. Uma vez criado, este protótipo será alargado de forma a incluir a informação das ontologias associadas à colecção de documentos, como abordado de seguida.

Uso do XQuery na Web semântica

Enquanto as linguagens de interrogação são desenvolvidas, o acesso à informação na Web continua um problema dada a quantidade enorme de recursos disponíveis (XML, HTML e outros). Numa tentativa de atenuar esse problema, os recursos da Web estão a ser associados a descritores semânticos (meta informação) constituindo a chamada Web semântica [10]. A semântica é expressa por ontologias descritas por grafos. Uma ontologia é a representação formal do conhecimento sobre um certo domínio, em que os conceitos são associados entre si por relações binárias (hierárquicas, de sinonímia e outras). Os recursos da Web são ligados às ontologias, constituindo ocorrências de certos conceitos. As ontologias são expressas através de sistemas formais como RDF [11] ou Topic Maps [12].

Pretende-se utilizar o XQuery estendido com a selecção para a interrogação de recursos XML da Web. Para isso, o sistema de processamento deve ser enriquecido de forma a tirar partido da informação dada pelas ontologias. Assim, aquando de uma consulta, a resposta será dada em função dos documentos e das ontologias associadas. Uma possibilidade para o efeito é usar as ontologias no cálculo das relevâncias das operações de similaridade textual: a relevância de um elemento (ou atributo) é calculada em função do conteúdo desse elemento e da descrição semântica fornecida pela ontologia associada.

As ontologias associadas aos elementos podem também ser mostradas juntamente com os resultados intermédios das perguntas. Assim, o utilizador conhecerá melhor o assunto de cada elemento.

Agradecimento: Os autores agradecem à Fundação para a Ciência e a Tecnologia (FCT) o apoio financeiro concedido através da bolsa de pós-doutoramento que suporta o trabalho da Alda Lopes Gançarski.

References

1. S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, D. McBeath, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 Full-Text Working Draft. http://www.w3.org/TR/2004/WD-xquery-full-text-20040709/, September 2005.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
3. A. Berglund, S. Boag, D. Chamberlin, M. Fernandez, M. Kay, J. Robie, and J. Siméon. XML Path Language (XPath) 2.0 W3C Working Draft. http//www.w3c.org/xpath20/, September 2005.
4. S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. W3C Working Draft. http://www.w3.org/TR/xquery/, September 2005.
5. N. Fuhr, M. Lalmas, S. Malik, and Z. Szlávik, editors. *INEX: Initiative for the Evaluation of XML Retrieval Workshop Proceedings*. DELOS Network of Excellence in Digital Libraries, Schloss Dagstuhl, Germany, December 2004.
6. A. Gançarski and P. Henriques. IXDIRQL: an Interactive XML Data and Information Retrieval Query Language. In *Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing*, Guimarães, Portugal, June 2003.
7. A. Gançarski and P. Henriques. Contrução e utilização de um protótipo para o processamento da linguagem de interrogação IXDIRQL. 2005.
8. A. Gançarski and P. Henriques. Extending XQuery with selection operations to allow for interactive construction of queries. In *Proceedings of the 9th ICCC International Conference on Electronic Publishing*, Leuven, Belgium, June 2005.
9. M. Kuiper and J. Saraiva. LRC: A Generator for Incremental Language-Oriented Tools. In K. Koskimies, editor, *7th International Conference on Compiler Construction*, volume 1383, pages 298—301. Lecture Notes in Computer Science, April 1998.
10. E. Miller and S. R. Semantic Web Activity: Advanced Developement. http://www.w3.org/2000/01/sw/, 2003.
11. E. Miller, S. R., and D. Brickley. Resource Description Framework (RDF). http://www.w3.org/RDF, 2005.
12. S. Pepper and G. Moore. XML Topic Maps (XTM) 1.0. http://www.topicmaps.org/xtm/index.html, 2001.

Meta-Objetos SCORM Hierárquicos via Reticulados Conceituais

Luciano Silva, Denise Stringhini e Ismar Frango Silveira
Faculdade de Computação e Informática, Universidade Mackenzie
Rua da Consolação, 930 - 01302-907 São Paulo, Brasil
{lucianosilva,dstring,ismar}@mackenzie.br

Resumo. SCORM é um conjunto de padrões e especificações que disponibilizam facilidades para interoperabilidade, acessibilidade e reuso para ambientes de *e-learning* baseados em Web. Apesar de ser bastante geral, o modelo SCORM ainda carece de mecanismos de acoplamento entre objetos para formação de redes conceituais. Estas redes normalmente são limitadas às seqüências lineares impostas pelas Árvores de Atividades e acessadas pelos Sistemas Gerenciadores de Aprendizado para controlar a navegação. Este trabalho apresenta uma arquitetura alternativa para acoplamento entre objetos SCORM, baseada na teoria dos reticulados conceituais e tecnologia XLink, que permite a construção de meta-objetos SCORM com estrutura conceitual hierárquica e não-linear.

1 Introdução

Objetos de aprendizagem [Wiley 2000] representam uma área de intensa pesquisa em ambientes de ensino-aprendizagem mediado por computador, principalmente naqueles baseados em Web. Existem diversas propostas de padrões para interoperabilidade, acessibilidade e reuso destes objetos entre as várias alternativas de Sistemas Gerenciadores de Aprendizagem (LMS – *Learning Managent Systems*).

Um padrão bastante difundido para atender aos requisitos levantados é denominado SCORM (*Sharable Content Object Reference Model*). Há vários outros padrões de metadados disponíveis para Objetos de Aprendizagem, como o Dublin Core [DCMI 2004], IMS [IMS 2004] e LOM [IEEE 2005]. Contudo, SCORM tende a tornar-se o padrão globalmente aceito, dado o atual suporte recebido por repositórios de objetos de aprendizagem. Tal padrão é construído sob três documentos básicos: Modelo de Agregação de Conteúdo, Ambientes de Execução, Sequenciamento e Navegação. O Modelo de Agregação de Conteúdo permite realizar anotações quantitativas e qualitativas sobre o objeto de aprendizagem, o Ambiente de Execução define o ambiente operacional necessário para a execução do objeto e o Sequenciamento e Navegação define uma ordem linear para exibição de vários objetos de aprendizagem.

O documento de Sequenciamento e Navegação, em particular, é bastante discutido na literatura, principalmente quanto às limitações das Árvores de Atividades que, em linhas gerais, definem a navegação nos objetos. O esquema de navegação definido pelas árvores é interpretado pelo LMS como a ordem de percurso e, limitações apresentadas pelo esquema, também limitam a navegação do aprendiz no LMS.

Este trabalho apresenta a proposta de uma arquitetura de navegação entre objetos SCORM baseado na Teoria dos Reticulados Conceituais, uma maneira bastante natural para especificar objetos de aprendizagem. São propostas introduções de anotações e *links* com alta estrutura semântica, via tecnologia XLink, nos objetos SCORM, o que permite tratamento personalizado de navegação ponto-a-ponto entre os objetos e navegação não-linear. A organização do trabalho é descrita a seguir: a Seção 2 apresenta a Teoria das Hirerarquias e Reticulados Conceituais; a Seção 3 retoma os principais conceitos do padrão SCORM e suas principais limitações de navegação; as Seções 4 e 5 apresentam a propostas de anotações nos objetos e nos links e, finalmente, a Seção 6 apresenta as conclusões e trabalhos futuros.

2 Hierarquias e Reticulados Conceituais

A teoria de conjuntos parcialmente ordenados e reticulados [Davey e Pristley 2002] aplicada ao estudo de hierarquias tem produzido várias contribuições nas mais diversas de computação como Inteligência Artificial, Teoria das Categorias, Semântica de Linguagens de Programação e Teoria de Concorrência.

Esta seção fará uma breve incursão na área denominada Análise de Conceitos Formais, onde grande parte do potencial de Teoria de Reticulados tem sido utilizada para fins de aprendizado hierárquico de conceitos.

2.1 Conceitos e Contextos

A definição de conceito é uma questão filosófica bastante complexa. De uma maneira bastante geral, um *conceito* é determinado pela sua *extensão* e *intensão*. Uma extensão consiste de todos os objetos que pertencem a um determinado conceito e uma intensão é um conjunto de atributos compartilhados por estes objetos. Normalmente, é uma tarefa difícil, senão impossível, enumerar todos os objetos e atributos de um determinado conceito. Assim, muitas vezes, os conjuntos de objetos e atributos são restritos a conjuntos discretos e finitos.

Definição 1 (*Contexto*). *Um **contexto (formal)** é uma tripla* $\mathfrak{k}=(G,M,I)$*, onde G e M são conjuntos e* $I \subseteq G \times M$*. Os elementos de G e M são denominados **objetos** e **atributos**. Quando os conjuntos G e F são finitos, o contexto é dito **finito**.*

A partir da noção de contexto, define-se a noção de conceito (formal):

Definição 2 (*Conceito*). *Seja (G,M,I) um contexto. Para* $A \subseteq G$ *e* $B \subseteq M$*, definam-se os conjuntos:*

$A' = \{ m \in M \mid (\forall g \in A) \Rightarrow (g,m) \in I \}$ $B' = \{ g \in G \mid (\forall m \in B) \Rightarrow (g,m) \in I \}$.

*Um **conceito** num contexto* $\mathfrak{k}=(G,M,I)$ *é um par (A,B),* $A \subseteq G$ *e* $B \subseteq M$*, tal que* $A'=B$ *e* $B'=A$*. Os conjuntos A e B são denominados, respectivamente, **extensão** e **intensão** do conceito. O conjunto de todos os conceitos de um contexto é denotado por* $\mathfrak{b}(G,M,I)$*.*

Para tornar mais claras as definições anteriores, considere-se um contexto para aprendizado de conceitos sobre o sistema solar:

Tabela 1. Contexto adaptado de Davey e Pristley (2002) para aprendizado do sistema solar. Objetos são formados pelos planetas e atributos estão relacionados com observações astronômicas de tamanho, distância ao sol e presença de luas.

Planeta	Tamanho	Distância ao Sol	Possui lua ?
Mercúrio	pequeno	perto	não
Vênus	pequeno	perto	não
Terra	pequeno	perto	sim
Marte	pequeno	perto	sim
Jupiter	grande	longe	sim
Saturno	grande	longe	sim
Urano	médio	longe	sim
Netuno	médio	longe	sim
Plutão	médio	longe	sim

Um conceito bastante simples sobre o contexto anterior é formado pelos conjuntos:

$A = \{ Terra , Marte \}$ e $B = \{ Tamanho\text{-}pequeno, Distância\text{-}perto, Lua\text{-}sim \}$.

Este conceito caracteriza, de maneira bastante elementar, os objetos Terra e Marte através de três atributos bastante simples: tamanho pequeno, ficam perto do Sol e possuem lua. Assim, num sistema de aprendizado dentro do contexto dado, a resposta à questão "*Quais as características dos planetas Terra e Marte?* " poderia ser: "*São pequenos, estão perto do Sol e possuem lua.*". Por outro lado, uma pergunta do tipo "*Quais planetas do Sistema Solar são pequenos, estão perto do Sol e possuem lua?* " poderia ter como resposta "*Terra e Marte*.".

2.2 Reticulados Conceituais

É possível dotar o conjunto $\mathfrak{b}(G,M,I)$ de uma ordem parcial $\leq$ através do seguinte resultado:

Teorema 1. *Sejam* (A_1,B_1) *e* (A_2,B_2) *dois conceitos em* $\mathfrak{b}(G,M,I)$*. Então a relação* $\leq$ definida por

$$(A_1,B_1) \leq (A_2,B_2) \Leftrightarrow A_1 \subseteq A_2$$

é uma relação de ordem parcial em $\mathfrak{b}(G,M,I)$*. Ademais,* $(\mathfrak{b}(G,M,I), \leq)$ *é um reticulado completo.*

A prova deste resultado é bastante simples e será omitida neste trabalho. É importante observar que a definição da relação de ordem parcial ≤ dependeu somente do conjunto de objetos e não dos atributos. Porém, como $A_1 \subseteq A_2$, então tem-se que $A_2' \subseteq A_1'$, ou seja, que $B_2 \subseteq B_1$. Assim, é possível também caracterizar a ordem entre os conceitos através da relação $(A_1,B_1) \leq (A_2,B_2) \Leftrightarrow B_2 \subseteq B_1$. A escolha entre definir a ordem por objetos ou atributos é uma decisão do processo de implementação da estrutura de ordem e será discutida posteriormente.

Definição 3 (*Reticulado Conceitual*). *Seja* $\mathfrak{c}=(G,M,I)$ *um contexto. Denomina-se* ***reticulado conceitual*** *do contexto* $\mathfrak{c}$ *ao reticulado* $(\mathfrak{b}(G,M,I), \leq)$.

Reticulados conceituais são melhor visualizados através dos Diagramas de Hasse, que permitem exibir propriedades de cobertura das ordens através de um diagrama hirerárquico. Seriam necessários, teoricamente, dois Diagramas de Hasse para representar as duas definições de ordem possíveis: via objetos e atribuitos. Porém, é possível combinar estes dois diagramas em um único conforme mostrado em Davey e Priestley (2002).

A Figura 1 exibe os Diagramas de Hasse combinados, objetos e atributos associados aos reticulados conceituais da Tabela 1.

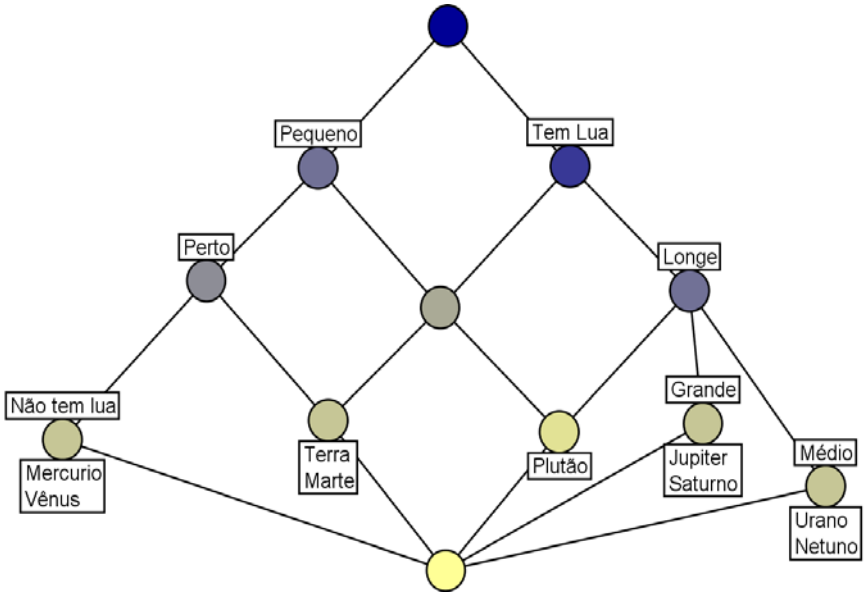


Fig. 1. Diagrama de Hasse combinado dos reticulados conceituais associados ao contexto de planetas do Sistema Solar. Anotações abaixo de cada nó representam conjuntos de objetos, enquanto que as acima representam atributos. A ausência de anotações inferiores e superiores pode significar ou que são facilmente calculadas através de hereditariedade ou que realmente não existem (conjunto vazio)

O diagrama combinado permite a navegação em duas direções verticais: ascendente e descendente. A navegação ascendente parte do ínfimo do reticulado e

permite obter facilmente os objetos. Por exemplo, no diagrama anterior o ínfimo não possui nenhum objeto vinculado (conjunto vazio), enquanto que todos os planetas aparecem no primeiro nível acima. O nó central do reticulado, apesar de estar sem anotação, herda os objetos dos níveis, respeitando-se a ordem de inclusão, imediatamente abaixo (Terra, Marte e Plutão). O supremo do reticulado na navegação ascendente, por hereditariedade, contém todos os objetos do contexto.

A navegação descendente permite obter facilmente os atributos. No ínfimo da navegação descendente, tem-se novamente um conjunto vazio de atributos, pois a Tabela 1 mostra que o conjunto de todos os objetos não possui todos os atributos em comum. Novamente, o nó central do reticulado não possui marcação de atributo, mas pode-se inferir facilmente, via hereditariedade, que ele é pequeno e possui lua. Claramente, o supremo na navegação descendente possui todos os atributos porque se tem um conjunto vazio de objetos.

Os nós do reticulado conceitual serão mapeados, posteriormente, para objetos de aprendizagem SCORM, que serão interligados através de arestas XLINK semanticamente anotadas. Cada nó receberá anotações de objetos e atributos. O reticulado conceitual como um todo representará um grande objeto SCORM.

A flexibilidade da ordem de navegação permitirá duas direções de navegação: uma ascendente (mais operacional), que privilegia os objetos, e outra descendente (mais axiomática), atrelada aos atributos. Em ambas as direções, as características hierárquica e hereditária da navegação estão fortemente presentes.

3 Objetos SCORM

O SCORM é um modelo de referência para o empacotamento e agregação de objetos de aprendizagem que possibilita sua utilização a partir de qualquer LMS (*Learning Management System*) compatível. O modelo SCORM foi apresentado pela ADL (*Advanced Distributed Learning*) e é definido através de três documentos: *Content Aggregation Mode*l (CAM), *Run-Time Environment* (RTE) e *Sequencing and Navigation* (SN) [ADL 2005].

As unidades básicas no modelo SCORM são os SCOs (*Sharable Content Objects*), os quais representam objetos de aprendizagem que fazem parte da estrutura de um curso. A navegação entre estes objetos deve ser definida para que possa ser seguida pelo LMS. Esta etapa da definição de um pacote SCORM é chamada de *agregação de conteúdo* e é realizada a partir da criação de um arquivo XML com as regras de navegação *entre* os objetos (nunca *intra-objetos*). Assim, um pacote SCORM deve conter um *arquivo de manifesto* (que segue as normas do IMS *Global Learning Consortium*) que declara o conteúdo do pacote, descreve a ordem de navegação do conteúdo (formado pelos SCOs) e indica onde os arquivos físicos que representam os SCOs podem ser encontrados.

Além do manifesto e dos arquivos físicos que implementam os SCOs, o pacote SCORM deve incluir arquivos que apresentam a descrição de cada SCO para facilitar sua manipulação. Estes são conhecidos como arquivos de *metadados*, já que contêm dados que descrevem outros dados. Basicamente, eles incluem informações tais como autor, título, versão, data de criação, requisitos técnicos, contexto educacional e objetivo.

O arquivo de manifesto pode referenciar outros, denominados sub-manifestos. O LMS utiliza estes arquivos para estabelecer a ordem de navegação entre os SCOs. Esta ordem é definida através da Árvore de Atividade (*Activity Tree*). Uma atividade de aprendizagem pode ser um recurso (atividade "folha") ou pode ser composta de diferentes sub-atividades. Além disso, as atividades possuem início e fim definidos, assim como testes finais associados. A passagem de uma atividade para outra depende, além da seqüência correta, de tentativas (*attempts*) bem sucedidas nos testes finais.

A seqüência a ser seguida pelo LMS, portanto, é baseada no percurso de uma Árvore de Atividades derivada a partir do conteúdo do(s) arquivos(s) de manifesto. Esta estrutura, no entanto, nem sempre é a mais adequada para determinados cursos. Alguns trabalhos têm sido desenvolvidos com o objetivo de propor estruturas de seqüências alternativas.

Diversos autores vêm apontando limitações no modelo de metadados do SCORM. Abdullah et al. (2004) apontam que, apesar do padrão SCORM versão 2004 [ADL 2005] suportar o modelo de seqüenciamento proposto pela IMS [IMS 2004], tal modelo é demasiado simplista, de modo a não fornecer mecanismos para a implementação efetiva de objetos de aprendizagem adaptativos. Gomes et al. (2005) ressaltam as limitações que os atuais padrões, entre os quais o SCORM, possuem no sentido de representar adequadamente objetos de aprendizagem denominados funcionais, no sentido que sejam "artefatos computacionais cuja funcionalidade deve promover interação entre entidades". Simões *et al.* (2004) propõem uma extensão ao SCORM de modo a suportar informações transversais aos objetos de aprendizagem, como regras de avaliação, currículo ou bibliografia. O artigo de Chang *et al.* (2005) apresenta uma proposta de uso de Redes de Petri para representar a árvore de atividades do modelo SCORM. O objetivo é permitir a visualização linear do fluxo percorrido, assim como possibilitar o "salto" de algumas lições arbitrariamente escolhidas. Os autores defendem que tal esquema pode ser útil em ambientes colaborativos (não cobertos pelo padrão SCORM).

4 Anotações e *Links* em SCOs

Conforme discutido na seção anterior, a flexibilidade de percurso entre SCOs é altamente dependente da estrutura da Árvore de Atividades. Esta seção apresenta uma proposta alternativa para implementação da Árvore de Atividades através da introdução de anotações de navegação XLink diretamente nos SCOs, eliminando a necessidade de uma estrutura maior para a árvore.

A Seção 2 apresentou uma estrutura de navegação hierárquica de objetos e atributos que transcende principalmente a dependência da navegação linear, permitindo a navegação segundo determinadas ordens. Para que os princípios apresentados naquela seção possam ser mapeados com mínimo impacto nas anotações de cada SCO, será adotada a seguinte ordem de construção: anotação de objetos e atributos, introdução de *links* entre objetos e anotações nos *links*.

4.1 Anotações de Objetos e Atributos

A anotação de objetos e atributos é realizada na seção META-DATA do arquivo de manifesto de um SCO, via marcador `keyword`, disponível no espaço de nomes `lom` [IEEE LTSC 2005] e descrito no Modelo de Agregação de Conteúdo SCORM (SCORM CAM).

A sintaxe geral de anotação é dada a seguir:

```
<manifest ... >
        …

   <metadata>

      <lom:lom xmlsns:lom="http://ltsc.ieee.org/xsd/LOM">
         <lom:general>
               <lom:keyword >
                  <lom:string> object | attribute </lom:string>
                  <lom:string> Nome do objeto ou atributo </lom:string>
               </lom:keyword>
                  …
               <lom:keyword >
                  <lom:string> object | attribute </lom:string>
                  <lom:string> Nome do objeto ou atributo </lom:string>
               </lom:keyword>
         </lom:general>
      </lom:lom>
      …

   <metadata>

   ...

</manifest>
```

Cada SCO pode encapsular um conjunto de objetos e atributos. Cada objeto ou atributo utiliza uma *keyword*, descrita através de dois *strings*: tipo (*object* ou *attribute*) e nome do objeto/atributo. A diferenciação entre os dois tipos de anotação é importante para o cálculo de hereditariedade, que deverá sere realizado no processo de navegação.

4.2 *Links* entre SCOs e Anotações

A estrutura de navegação entre SCOs ainda é alvo de intensa discussão, conforme mostra o documento *SCORM Sequencing and Navigation* [ADL 2004]. Para o propósito de navegação no modelo hierárquico mostrado na Seção 2, a tecnologia XLink [Wilde e Lowe 2002] é extremamente adequada. *Links* especificados através da tecnologia XLink permitem navegação bidirecional, associação de regras de processamento do *link* e múltiplas direções.

Links serão adicionados como um novo marcador dentro do namespace `lom`, denominado `navigation`, conforme mostrado na sintaxe abaixo:

```
<manifest ... >

        …

   <metadata>

      <lom:lom xmlsns:lom=”http://ltsc.ieee.org/xsd/LOM”>

         <lom:general>

           <lom:keyword >
                  <lom:string> object | attribute </lom:string>
                  <lom:string> Nome do objeto ou atributo </lom:string>
            </lom:keyword>

           …

           <lom:keyword >
                  <lom:string> object | attribute </lom:string>
                  <lom:string> Nome do objeto ou atributo </lom:string>
            </lom:keyword>

         </lom:general>

         <lom:navigation xmlsns:xlink=”http://www.w3.org/1999/xlink”
               xlink:type     = “extended”
               xlink:to       = Endereço do próximo objeto SCORM
               xlink:from     = Endereço do objeto SCORM anterior
               xlink:arcrole  = Regra de processamento do link
               xlink:show     = “replace”
               xlink:actuate  = “onRequest”
         > Texto de navegação    </lom:navigation>

         <lom:navigation ...>... </lom:navigation>

               ...

          <lom:navigation ...>... </lom:navigation>

      </lom:lom>

      …

   <metadata>

   ...

</manifest>
```

Cada marcador representa uma unidade básica de navegação: sabe-se de onde vem (`from`), para onde se vai (`to`) e qual a regra de processamento nestas navegações. Como sugestão, o formato de atuação no link é sob requisição (`onRequest`) e seu comportamento e deve ocorrer a substituição do conteúdo atual (`replace`).

Um dos detalhes mais importantes do *link* especificado consiste na regra de processamento: se a navegação for ascendente, os objetos válidos até o momento devem ser formados pelos objetos dos ascestrais mais os objetos do nó atual; se a navegação for descendente, os atributos válidos até o momento consistem também dos atributos dos ancestrais mais aqueles do nó atual. Além deste processamento canônico, é possível ainda verificar pré-requisitos na navegação, muito semelhantes aos marcadores do namespace imsss [IMSSS 2005].

A regra de processamento do *link* pode ser especificada indicando-se a referência de um processador (*parser*) XML baseado em SAX ou DOM para tratamento de XLink. Neste contexto, regras de tratamento do *link* são intepretadas como interesses ortogonais às relações de ordem – que podem vir a ser implementadas como aspectos no paradigma de Programação Orientada a Aspectos-AOP [Kiczales 1997] – e são mapeados como objetos externos ao SCO. Assim, pode-se encapsular dentro de cada SCO a referência externa –um ponto de junção, segundo a AOP – para um tratamento personalizado de cada *link*, permitindo alto grau semântico às relações de ordem estabelecidas no reticulados conceitual. Além disso, pode-se alterar a regra de tratamento sem necessidade de se reestruturar o SCO e liberando o LMS da tarefa de controlar a tarefa de navegação nos objetos SCORM.

A introdução do marcador `navigation` dentro do namespace `lom` pode ser feito de maneira bastante simples através da DTD abaixo:

```
<!ELEMENT navigation (#PCDATA)>

<!ATTLIST navigation
 xmlns:xlink   CDATA #FIXED “http://www.w3.org/1999/xlink/namespace/
 xlink:type    CDATA #FIXED "extended"
 xlink:to      CDATA #REQUIRED
 xlink:from    CDATA #REQUIRED
 xlink:arcrole CDATA #REQUIRED
 xlink:show    (new|embed|replace) "replace"
 xlink:actuate (onRequest|auto)    "onRequest"
>
```

Assim, a partir deste momento, cada SCO possui, além das suas características regulares, anotações de objetos e atributos, *links* bidirecionais para navegação ascendente e descendente, tudo encapsulado em uma única unidade representada através de seu arquivo de manifesto.

5 A Estrutura Hierárquica e o Meta-Objeto SCORM

As unidades SCORM construídas na seção anterior serão agora organizadas numa estrutura hierárquica para refletir a estrutura do reticulado conceitual. Nós do reticulado são mapeados para SCOs e relações de ordem através dos links especificado via XLink, cuja unidade total é representada através de um arquivo-manifesto. Cada arquivo-manifesto fará parte de uma rede iniciada por um objeto maior, denominado meta-objeto SCORM ou meta-SCO.

Tome-se como exemplo o Diagrama de Hasse combinado da Figura 1. A figura a seguir exibe os objetos SCORM associados ao nó-base (`manifest0.xml`), ao nó identificado pelos objetos Júpiter-Saturno e atributo Grande (`manifest4.xml`), e o nó que contém somente o atributo Longe (`manifest8.xml`).

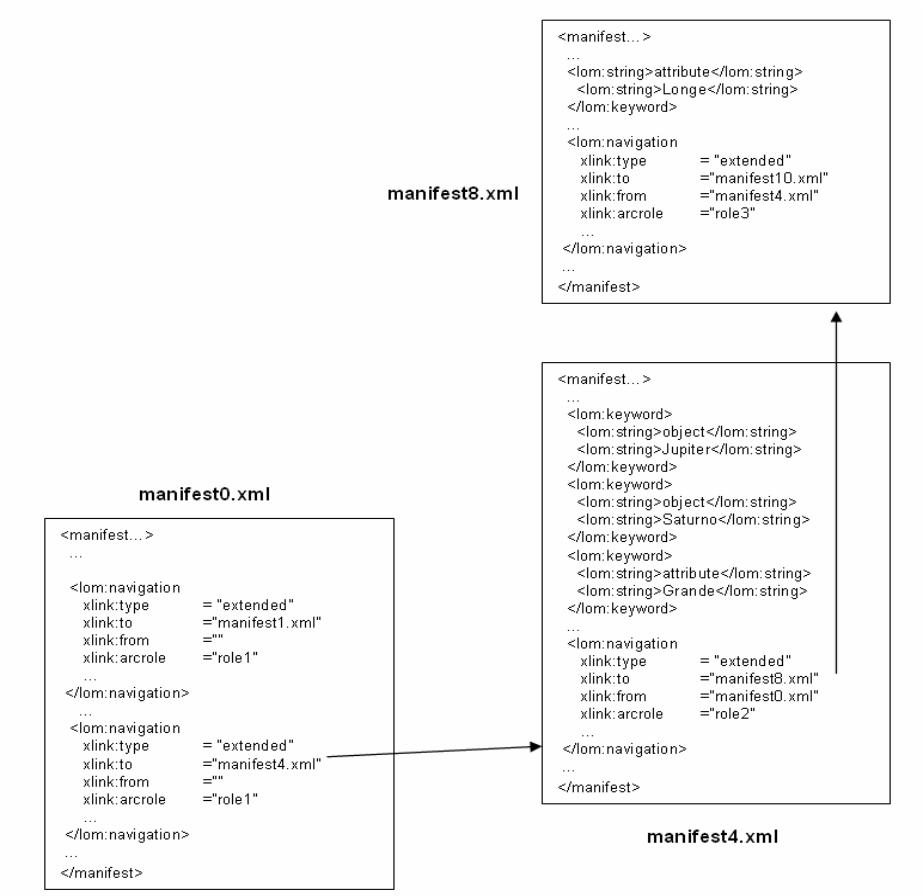


Fig. 2. Diagrama parcial de objetos SCORM (anotações e links), correspondente ao Diagrama de Hasse combinado na Figura 1. Conforme mostrado no Diagrama de Hasse, alguns objetos não possuem nem anotações de objetos nem de atributos (manifest0.xml), alguns somente de atributos (manifest8.xml), alguns só de objetos e outros possuem os dois (manifest4.xml). A omissão destas anotações representa uma grande economia de espaço para grandes reticulados e podem ser obtidas facilmente via regras associadas aos arcos de navegação.

A Figura 2 mostra que a estrutura hierárquica presente no diagrama conceitual pode ser facilmente mapeada para uma estrutura hierárquica de arquivos-manifesto. Numa navegação ascendente, isto é privilegiando os objetos, os atributos `xlink:to` e `xlink:from` são tratados de forma natural. Porém, na navegação descendente (ordem dos atributos), os atributos `xlink:to` e `xlink:from` devem ser processados de forma invertida.

Duas anotações são de vital importância para o meta-SCO: SCO inicial para navegação e sentido na navegação (ascendente ou descendente). Somente o SCO

inicial será mapeado como sub-manifesto do meta-SCO, permitindo reduzir o espaço necessário para armazenar a hierarquia dos outros SCOs.

O SCO inicial para navegação, bem como o sentido de navegação, podem ser facilmente introduzidos num arquivo de manifesto via marcador `<organization>` utilizado conjuntamente com o marcador `<resource>`, conforme mostrado abaixo para o exemplo da Figura 2:

```
<manifest>
   ...
  <organizations>
    <organization>
       <item identifier="Initial"
            Identifierref="RInitial"
            Parameters="?Order=Ascent">
      </item>
     </organization>
  <resources>
    <resource identifier="RInitial"
              href="manifest0.xml">
    </resource>
  </resources>
</manifest>
```

Um objeto inicial na hierarquia de navegação sempre será identificado através do nome `Initial`, colocado no parâmetro `identifier` do marcador `item`. O sentido da navegação é passado através do parâmetro `Order` para o arquivo `manifest0.xml`, especificado em `resource`. Caso a navegação seja ascendente, utiliza-se o valor `Ascent` e, para navegação descendente, `Descent`.

Assim, basta o LMS saber qual o SCO inicial para navegação e transferir-lhe o controle da navegação. Isto permite uma grande flexibilidade na navegação, porque pode-se associar comportamentos semânticos sofisticados na transição de um SCO para outro, além de permitir uma navegação bidirecional.

6 Conclusões e Trabalhos Futuros

A especificação de padrões de interoperabilidade, acessibilidade e reuso para objetos de aprendizagem é de grande interesse para Sistemas Gerenciadores de Aprendizado. Em particular, o padrão SCORM representa uma excelente alternativa para encapsular dados referentes a um objeto de aprendizagem.

Apesar de bastante geral, o padrão SCORM apresenta algumas deficiências, como o fato de carecer de mecanismos de navegação mais sofisticados. Este trabalho apresentou uma proposta de arquitetura de navegação em redes de objetos SCORM via reticulados conceituais. Estes reticulados permitem navegação em objetos e atributos de maneira bidirecional. A arquitetura é baseada na introdução de anotações e *links*, via tecnologia XLink, bastante difundida na integração entre documentos XML.

As anotações e *links* produzem um impacto muito baixo na atual estrutura do padrão SCORM e permitem a construção de redes complexas de objetos SCORM através de construções muito simples. *Links* entre objetos podem ser dotados de

processamento semântico altamente qualificado e permitem abstrair as ligações como aspectos entre arquivos-manifesto associados aos objetos de aprendizagem.

Trabalhos futuros incluem o estudo de modelos conceituais mais elaborados como, por exemplo, modelos híbridos envolvendo Redes de Petri e Reticulados Conceituais. Além disso, a geração automática de objetos SCORM via reticulados conceituais poderia ser uma poderosa ferramenta para auxiliar o desenvolvimento de objetos de aprendizagem com alta coesão, granulosidade fina e altamente adaptáveis a vários estilos de navegação.

Referências

Abdullah, N. A.; Bailey, C. e Davis, H. Synthetic hypertext and hyperfiction: Augmenting SCORM manifests with adaptive links. Em: *Proceedings of the 15th ACM Conference on Hypertext and Hypermedia*. Santa Cruz, EUA, 2004, pp. 183-184.

ADL. Advanced Distributed Learning. Disponível em: http://www.adlnet.org. Acesso em: 10.nov.2005.

Chang, W., Lin, H. W., Shih, T. e Yang, H. *SCORM Learning Sequence Modeling with Petri Nets in Cooperative Learning*. Em: Learning Technology, IEEE Computer Society, Volume 7, Issue 1, Janeiro de 2005. Disponível em: http://lttf.ieee.org/learn_tech/issues/january2005/#_Toc98675005. Acesso em 21.nov.2005.

DCMI, Dublin Core Metadata Initiative. *Dublin Core Metadata Element Set, Version 1.1: Reference Description*, 2004. Disponível em: http://www.dublincore.org/ documents/dces/. Acesso em: 21.set.2005.

Davey, B.A., Priestley, H.A. *Introduction to Lattices and Order*. 2a. ed. Cambridge University Press, Cambridge, 2002.

Gomes, S. R.; Gadelha, B. F.; Mendonça, A. P. e Amortti, M. S. M. Objetos de Aprendizagem Funcionais e as Limitações dos Metadados Atuais. Em: *Anais do XVI Simpósio Brasileiro de Informática na Educação – SBIE 2005*. Juiz de Fora, Brasil, pp. 211-221.

IEEE LTSC. *Learning Object MetaData*. Disponível em: http://ieeeltsc.org/wg12LOM/. Acesso em: 20.set.2005.

IMS Global Learning Consortium. *Learning Resource Meta-data Specification - Version 1.2.4 – XSD Schema*, 2004. Disponível em: http://www.imsglobal.org/metadata/index.html. Acesso em: 21.set.2005.

IMSSS. IMS Global Learning Consortium. Disponível em: http://www.imsglobal.org/xsd/imsss. Acesso em: 25.set.2005.

Kiczales *et. al*. Aspect-Oriented Programming. Em: *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'97)*, Springer-Verlag, 1997..

Simões, D.; Luís, R. e Horta, N. Enhancing the SCORM Metadata Model.. Em: *Proceedings of the 13th international World Wide Web Conference*. New York, EUA, 2004, pp. 238-239.

Wilde, E., Lowe, D. *XPath, XLink, XPointer and XML: A Practical Guide to Web Hyperlinking and Transclusion*. Pearson Education, 2002.

Wiley, D. A. *Learning Object Design and Sequenceing Theory*. Tese de Doutorado, Brigham Young University. Provo, EUA, 2000.

Topic Maps aplicados ao sistema de informação do Museu da Emigração

Giovani Rubert Librelotto[1],
José Carlos Ramalho[2], and Pedro Rangel Henriques[2]

[1] UNIFRA, Centro Universitário Franciscano, Santa Maria - RS, 97010-032, Brasil
giovani@unifra.br
[2] Universidade do Minho, Departamento de Informática
4710-057, Braga, Portugal
{jcr, prh}@di.uminho.pt

Abstract. O presente artigo apresenta uma aplicação do Metamorphosis ao caso do Museu Virtual da Emigração. O Museu da Emigração contém no seu espólio fontes de informação compostas por documentos XML e por uma base de dados relacional. A função do Metamorphosis é propiciar uma visão homogénea destes recursos através da criação de um topic map que represente este universo de discurso. Para isso, usa-se o Oveia para a extracção do topic map e o Ulisses para a navegação sobre o conhecimento do domínio. O resultado é uma visão integrada do Museu da Emigração, de acordo com a ontologia especificada pelo projectista.

1 Introdução

Devido ao facto de as pessoas viverem em constante mudança, o ritmo das suas vidas as faz incorrer em viagens (por puro lazer, por necessidade ou por trabalho). Num passado não muito longínquo, era frequente encontrar em almanaques locais, histórias sobre a vida de determinados indivíduos ou sobre eventos em que os mesmos participavam. Também era usual fazer-se um diário de viagem quando se realizava uma jornada por vários locais.

O *Museu da Emigração*[3] (ME) foi criado para reunir e disponibilizar à comunidade informações diversas acerca dos emigrantes em geral e, em particular, daqueles que emigraram para o Brasil.

Particularmente importante nesse sentido é que o ME é um museu virtual. Para tal, a informação a ser exibida no ME é recolhida a partir de diversas fontes de dados, tais como registos de passaportes, almanaques, jornais, diários de viagem e certidões de nascimento. Esta lista de possíveis fontes de dados não está limitada, pois novas fontes (recheadas de informação relevante para o ME) podem sempre aparecer.

Combinando várias fontes – como por exemplo, as fontes que contém dados sobre os almanaques, diários de viagem e passaportes – pode-se disponibilizar uma quantidade significativa de informação sobre um determinado indivíduo,

[3] Museu da Emigração – http://www.museu-emigrantes.webside.pt

reconstruindo-se assim o seu percurso de vida que irá contextualizar e enquadrar o seu processo de emigração.

Pensando agora em termos do Sistema de Informação digital (SIME) que irá suportar o Museu Virtual da Emigração, é importante observar que, pela natureza dos documentos aqui envolvidos e por razões práticas e factuais, as fontes de informação estão armazenadas em suportes heterogéneos: algumas são guardadas em bases de dados relacionais, enquanto outras estão na forma de textos anotados em XML.

Neste artigo pretende-se a integração de *fontes nominativas históricas heterogéneas*, isto é, armazenadas de formas diversas em suportes diferentes através do uso do Metamorphosis, o qual vai representar o conhecimento extraído do SIME na norma ISO 13250 Topic Maps. O objectivo é propiciar à comunidade em geral a extracção do conhecimento a partir das fontes de dados do Museu da Emigração, através da navegação conceptual sobre a informação contida nas seguintes fontes heterogéneas.

Este artigo encontra-se dividido em 4 secções. A Secção 2 descreve as fontes de dados que fazem parte do SIME, apresentando sua estrutura e conteúdo. A Secção 3 define os passos a serem seguidos para obter a interoperabilidade sobre os recursos de informação do Museu da Emigração através do uso do Metamorphosis: o primeiro passo, apresentado na Subsecção 3.2, descreve como o Oveia propicia a geração de um topic map a partir de recursos heterogéneos contendo o conhecimento do universo de discurso do SIME; por fim, a Subsecção 3.3 disponibiliza a visualização do domínio do ME através de páginas HTML geradas pelo Ulisses. A conclusão do caso de estudo em questão se apresenta na Secção 4.

2 Sistema de Informação do Museu da Emigração

Normalmente a informação relevante que se pretende extrair de qualquer sistema está armazenada em formatos distintos. Este foi o caso encontrado no âmbito do SIME. Os dados que alimentam esse sistema estão armazenados em documentos XML (para as fontes de dados referentes aos *Almanaques* e aos *Diários de Viagem*), de onde se pode extrair as histórias da vida e de viagens referentes aos emigrantes. Combinando estes documentos XML com uma base de dados que contém a informação relativa aos seus passaportes, fica-se com um conjunto de informações sobre o percurso de vida e histórias com relevo de emigrantes.

As próximas subsecções descrevem a estrutura e exemplificam o conteúdo de cada fonte de dados pertencente ao SIME.

2.1 Notas Biográficas contidas em Almanaque

Os *almanaques* são publicações, normalmente periódicas e nacionais (embora por vezes possam ter um cariz mais local), com calendário, informações científicas, tabelas, informações úteis sobre o cotidiano e alguns textos mais amenos como contos, notas biográficas, descrição de eventos sociais, entretenimento, etc.

Contudo, os almanaques não se encontravam escritos de uma forma facilmente manipulável (de um ponto de vista informático). De um modo geral, as histórias dos almanaques não possuem um conteúdo semelhante entre si, podendo abranger um grande leque de descrições sociais, psicológicas, físicas, além de conter informação ambígua e parcial.

Para seu armazenamento digital, definiu-se então uma estrutura XML que permita a extracção e catalogação das informações possíveis de ser encontradas nas Notas Biográficas destes almanaques. Abaixo encontra-se a anotação em XML do exemplo de história de vida do emigrante "*José Pereira Leite*".

```
<Notas_Biograficas>
    <fonte>Almanaque de Fafe</fonte>
    <titulo>José Pereira Leite</titulo>
    <tipoMemoria valor="outro"/>
    <MEMORIA>
        <PESSOA ID="ID0000006">
            <nome>José Pereira Leite</nome>
            <profissao>Comércio</profissao>
        </PESSOA>
        <HISTORIA_A>
            <notasAdicionais>
                <perfil valorPerfil="psicologico">Sincero e Dedicado</perfil>
            </notasAdicionais>
            <localNasc>Natural de Chacim, Cabeceiras de Basto</localNasc>
            <paisEmigrou>Brasil</paisEmigrou>
            <notasAdicionais>
                <perfil valorPerfil="profissional">Graças à sua força de vontade e
                actividade, possui uma boa casa de negócio</perfil>
                <relSociais>Tem ao seu serviço o Sr. Joaquim de Macedo Ferraz e Souza,
                conhecidíssimo na vila de Fafe</relSociais>
            </notasAdicionais>
            <docAnexo>foto6</docAnexo>
        </HISTORIA_A>
    </MEMORIA>
</Notas_Biograficas>
```

Esta história anotada em XML possui a mesma informação acerca desse emigrante que a página de um almanaque em papel. A informação está estruturada e explicitamente marcada (para permitir uma interpretação objectiva, não ambígua), tornando viável o seu arquivo electrónico e o processamento deste documento por um parser XML. As notas biográficas retiradas dos Almanaques de Fafe no âmbito do ME estão juntamente armazenadas em ficheiros de texto que seguem o formato acima exemplificado.

2.2 Diários de Viagem

Um *Diário de Viagem* é um documento onde uma pessoa escreve suas jornadas, ou seja, notas de embarque, repouso e o que fez ou não em certos locais. Estas notas são acompanhadas da data do respectivo apontamento. Resumindo, é um diário que apenas se cinge a uma dada viagem efectuada por uma pessoa, contendo informações como a data, o local e uma descrição textual de sítios visitados (monumentos ou paisagens) ou episódios vivenciados.

Os diários de viagem são muito genéricos, sucintos e podem ocorrer grandes intervalos de tempo em que não existe informação (pode não existir uma descrição contínua, metódica e temporal do relato da viagem).

Assim como no caso dos almanaques, foi definido um esquema genérico o suficiente, tornando possível a catalogação da informação contida num diário. Este esquema define a estrutura lógica, em anotação XML, para um diário de viagem, o qual é exemplificado pelo documento abaixo:

```
<Notas_Bio>
    <PESSOA ID="ID000006">
        <nome>José Pereira Leite</nome>
        <idade>25</idade>
    </PESSOA>
    <HISTORIA_V ID="HIST0100">
        <nota data="1905-01-07">Início da viagem à Europa: Amanhã.</nota>
        <nota data="1905-01-08">Sahida hoje do Rio, para a Bahia, ...</nota>
        <nota data="1905-01-09">...</nota>
        <nota data="1905-01-10">...</nota>
    </HISTORIA_V>
</Notas_Bio>
```

Em termos do SIME, todos os diários de viagem recolhidos e pertencentes ao âmbito do ME encontram-se anotados em XML, tal qual o exemplo acima.

2.3 Registos de Passaportes

Os passaportes contém a informação referente às viagens efectuadas por um emigrante, informando qual foi o destino da viagem (podendo ser múltiplos destinos) e se ele foi acompanhado por alguma outra pessoa (podendo ser múltiplos acompanhantes).

A informação referente aos passaportes está estruturada seguindo o modelo relacional em quatro tabelas: *Passaportes* (contém os dados básicos dos passaportes), *Viagem* (contém os dados referentes às viagens efectuadas por determinado emigrante), *Acompanhantes* (contém os dados sobre os acompanhantes dos emigrantes em cada viagem) e *Pessoa* (contém os dados pessoais de cada emigrante). A Figura 1 apresenta estas quatro tabelas e os relacionamentos entre elas, no diagrama entidade-relacionamento dessa base de dados.

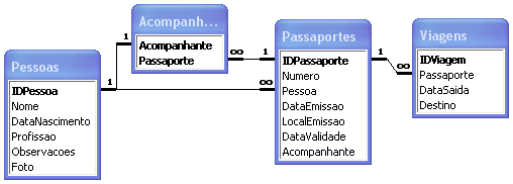


Fig. 1. Diagrama Entidade-Relacionamento da base de dados dos Passaportes.

Os relacionamentos da Figura 1 indicam que uma pessoa pode possuir vários passaportes, assim como ter vários acompanhantes associados a ela. Por sua vez, cada passaporte poderá referenciar vários destinos.

3 Passos para a Integração de Fontes Heterogéneas

A partir da definição de quais fontes de informação compõe o SIME – base de dados dos passaportes e documentos anotados XML para os almanaques e os diários de viagem – utiliza-se o Metamorphosis para se obter uma visão homogénea destas fontes heterogéneas.

O Metamorphosis [3] é um ambiente formado por um conjunto de linguagens de especificação e ferramentas que permitem criar uma interface para integração de informação oriunda de diversas fontes, através do uso de uma ontologia representada em Topic Maps [5]. A partir da descrição das fontes heterogéneas de informação e da especificação da ontologia, o Oveia [4] (um dos componentes do Metamorphosis) extraí automaticamente o respectivo topic map. Depois de guardado, a componente Ulisses [3] gera uma interface Web para manipular o topic map extraído.

Neste processo, o topic map não foi validado sintáctica e semanticamente (face a um conjunto de restrições especificadas numa linguagem apropriada) pelo Processador XTche [3] (outro dos componentes no Metamorphosis) pelo facto do topic map ter sido criado por uma ferramenta de construção automática, o Oveia, reduzindo o número de restrições que possam detectar erros, pois certos casos já são verificados pelo próprio processo de criação. Assim sendo, a validação XTche iria apenas confirmar que o Oveia realizou seu processo de uma maneira correcta.

O processo envolvendo a geração dessa visão é composto por uma série de etapas. As primeiras etapas requerem um especialista no universo de discurso em questão, pois ele será responsável por definir qual a visão que será oferecida aos utilizadores do sistema, ou seja, aos visitantes do Museu Virtual da Emigração. Esse especialista deve possuir sólidas noções sobre a norma Topic Maps, permitindo assim tornar a especificação o mais abrangente possível.

3.1 Por que Topic Maps?

Topic Maps [1] é um formalismo para representar conhecimento acerca da estrutura de um conjunto de recursos de informação e para o organizar em tópicos. Esses tópicos possuem ocorrências e associações que representam e definem relacionamentos entre os tópicos. A informação sobre os tópicos pode ser inferida ao examinar as associações e ocorrências ligadas ao tópico. Uma colecção desses tópicos e associações é designada *topic map*. Topic Maps pode ser visto como um paradigma que permite organizar, manter e navegar através da informação, permitindo transformá-la em conhecimento.

Tendo em vista que os Topic Maps são grafos compostos por tópicos (onde cada tópico representa um tema, identifica recursos e está associado com outros tópicos), o Ulisses foi imaginado e desenvolvido para providenciar navegadores estáticos que permitem percorrer estas redes de conceitos. Esses navegadores são compostos de páginas HTML que descrevem os tópicos e usam a ideia de hiper-ligações para implementar as associações e as ligações às ocorrências.

3.2 Construção do topic map pelo Oveia

Esta secção tem por objectivo demonstrar, passo a passo, o processo de geração do topic map a partir dos dados extraídos das fontes heterogéneas de informação. O topic map gerado segue a ontologia descrita na especificação XS4TM (*XML Specification for Topic Maps*) [4] que descreve o domínio do ME.

Especificação das fontes em XSDS. Esta etapa é responsável por definir a especificação XSDS (*XML Specification for Data Sources*) [4] que determina quais fontes de dados serão integradas e quais os dados que estão em cada fonte. A especificação XSDS divide-se em duas etapas: definição dos recursos físicos a serem interrogados; e selecção das partes de informação contidas nestes recursos necessárias para a construção da rede semântica.

A especificação em XSDS de cada fonte define um elemento <datasource>. Abaixo representa-se a especificação das fontes Almanaques e Passaportes:

```
<resources>
    <datasources>
        <!-- Declaração da Fonte de Dados dos Almanaques -->
        <datasource extratorDriver="br.uneb.dcet.tmbuilder.drivers.XMLFile" name="XML_Almanaque">
            <parameter name="pathDocument">C:\\MUSEU\\Almanaques\\Alm.xml</parameter>
        </datasource>
        <!-- Declaração da Fonte de Dados dos Passaportes -->
        <datasource extratorDriver="br.uneb.dcet.tmbuilder.drivers.DataBase" name="BD_Passaporte">
            <parameter name="connectionURL">jdbc:access://localhost/museu/passaporte</parameter>
            <parameter name="password"/>
            <parameter name="user"/>
            <parameter name="jdbcDriver">sun.jdbc.odbc.JdbcOdbcDriver</parameter>
        </datasource>
    </datasources>
    <datasets/>
</resources>
```

Para o acesso aos documentos XML, basta apenas informar o seu caminho na estrutura de directorias do sistema operativo. Aliado a isto, define-se o nome de cada *dataset* que será criado (no atributo *name*) e informa-se o *driver* responsável pela extracção (no atributo *extratorDriver*). Para o acesso na base de dados relacional Microsoft Access® define-se um nome para o *dataset* a ser criado e do driver de extracção. Além disso, é necessário a criação de quatro parâmetros: a URL de conexão, o utilizador, sua palavra-passe e o *driver* JDBC.

Uma vez definidas as fontes de dados, parte-se para a definição de quais elementos (em documentos XML) ou campos (em base de dados) devem ser extraídos. Para isso, o elemento <dataset> é utilizado, o qual referencia qual fonte de dados (*datasource*) será a matéria-prima para a extracção.

```
<resources>
    <datasources>
        ...
    </datasources>
    <datasets>
        <dataset name="Alm-Emigrante" database="XML_Almanaque">
            <columns>
                <column identify="true">@ID</column>
                <column>nome</column>
                <column>idade</column>
```

```
                <column>profissao</column>
            </columns>
            <statement>//Notas_Biograficas/MEMORIA/PESSOA</statement>
        </dataset>
        <dataset>
            ...
        </dataset>
        ...
    </datasets>
</resources>
```

O *dataset* acima apresentado efectua uma extracção no documento XML dos Almanaques. O conjunto de dados extraídos será referenciado pelo nome de *Alm-Emigrante* e conterá 4 elementos, referidos nos elementos <column>. Cada um desses elementos será encontrado fisicamente através do caminho XPath declarado no elemento <statement>. Para exemplificar, o *dataset* chamado *Alm-Emigrante* formará uma tabela, onde cada linha conterá as seguintes informações extraídas da fonte *XML_Almanaque*:

- //Notas_Biograficas/MEMORIA/PESSOA/@ID – será o identificador deste *dataset*, definido pelo atributo *identify*.
- //Notas_Biograficas/MEMORIA/PESSOA/nome
- //Notas_Biograficas/MEMORIA/PESSOA/idade
- //Notas_Biograficas/MEMORIA/PESSOA/profissao

Para o caso da definição do *dataset* extraído da base de dados dos passaportes (contendo o seu identificador, nome, nome do pai e nome de mãe), o seguinte código elemento <dataset> deve ser definido:

```
        <dataset name="Passaporte-Emigrante" database="BD_Passaporte">
                SELECT ID,Nome,NomePai,NomeMae FROM Pessoas
        </dataset>
```

Percebe-se, deste modo, que a linguagem de interrogação às fontes de dados varia de acordo com a própria fonte: quando a extracção é realizada sobre documentos XML, utiliza-se a linguagem XPath; quando a extracção é realizada sobre base de dados, utiliza-se a linguagem SQL.

Definição da ontologia do universo de discurso. Esta subsecção apresenta a definição da ontologia para o Museu da Emigração, contendo a representação de todo o conhecimento deste domínio. A definição da ontologia consistiu, inicialmente, em determinar quais conceitos eram considerados importantes no domínio dos Almanaques, Diários de Viagem e Passaportes e seguiu a metodologia definida por Gruber [2].

O próximo passo consiste da definição dos relacionamentos entre os conceitos. Então, o especialista identifica quais conceitos estão relacionados e cria uma associação entre ambos.

Para exemplificar, tomamos como exemplo o caso entre os conceitos *Emigrante* e *Histórias de Almanaque*. Ao definir uma associação entre estes dois conceitos, se fez necessário determinar como um conceito se relaciona com o outro, determinando que *um emigrante é referido numa história de almanaque*.

Após a identificação das associações, obteve-se um grafo representando a ontologia, conforme apresentado na Figura 2. Nesta figura, os nodos representam os conceitos, enquanto que as associações são representadas pelos arcos.

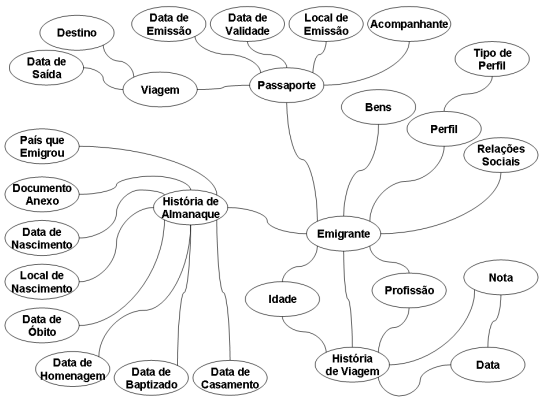


Fig. 2. Ontologia para o Museu da Emigração

Como o conceito *Emigrante* foi encontrado em todas as fontes, acabou por tornar-se um elo de união pois os conceitos *Emigrante* foram fundidos num único. Assim sendo, o conceito resultante mantém todos os relacionamentos que os demais participavam.

Especificação da Ontologia em XS4TM. A partir do momento que todos os conceitos e seus relacionamentos foram definidos, foi então especificado em XS4TM, a ontologia apresentada na subsecção anterior.

A primeira etapa da especificação em XS4TM consistiu na codificação dos tipos de tópicos (ou seja, os conceitos). A partir da sua definição, parte-se para a definição dos tipos das associações. Assim como os tipos de tópicos, os tipos das associações são definidos como sub-elementos do elemento <ontologies>.

Ao final deste passo, a ontologia do Museu da Emigração estará definida no elemento <ontologies>. A mesma consiste basicamente da definição dos tipos de tópicos, dos tipos de associações e dos papéis de actuação em associações.

Instanciação da Ontologia em XS4TM. Uma vez definida a ontologia, o próximo passo consiste de relacioná-la com os *datasets* especificados em XSDS. Esta relação entre conceitos e fonte de informação é realizada no elemento <instances> da especificação XS4TM.

Em XS4TM, cada elemento //instances/topic e //instances/association possui um atributo chamado dataset, o qual determina de qual *dataset* os dados serão obtidos para a construção do tópico ou associação em questão.

Como exemplo, apresenta-se a seguir a especificação da extracção dos tópicos que possuem o tipo *Emigrante*. Neste exemplo, o tópico *Emigrante* está definido para a fonte de dados *almanaques* (*dataset Alm-Emigrante*, linha 4 a 23). As demais fontes (*diários de viagem* (*dataset DViagem-Emigrante*) e *passaportes* (*dataset Passaporte-Emigrante*) terão uma representação similar.

```
1  <xs4tm xmlns="http://www.topicmaps.org/xtm/1.0/" xmlns:xlink="http://www.w3.org/1999/xlink">
2    <ontologies>
3        <!-- Aqui vai a definição da ontologia do SIME -->
4    </ontologies>
5    <instances>
6        <topic dataset="Alm-Emigrante" id="@Alm-Emigrante.@ID">
7            <instanceOf>
8                <topicRef xlink:href="#Emigrante"/>
9            </instanceOf>
10           <baseName>
11               <baseNameString>@Alm-Emigrante.nome</baseNameString>
12           </baseName>
13           <occurrence>
14               <scope>
15                   <topicRef xlink:href="#Idade"/>
16               </scope>
17               <resourceData>@Alm-Emigrante.idade</resourceData>
18           </occurrence>
19           <occurrence>
20               <instanceOf>
21                   <topicRef xlink:href="#Profissao"/>
22               </instanceOf>
23               <resourceRef xlink:href="@Alm-Emigrante.profissao"/>
24           </occurrence>
25       </topic>
26   </instances>
27 </xs4tm>
```

No processo de extracção realizado pelo Oveia, será criado um tópico para cada emigrante encontrado nos *datasets* acima. Num segundo passo, o Oveia faz uma fusão entre os tópicos que possuem o mesmo identificador, de acordo com as regras definidas na norma ISO 13250 [1]. Deste modo, se um mesmo emigrante é encontrado nas três fontes de dados do SIME, um único tópico conterá toda a informação acerca do emigrante.

Esta especificação irá determinar a extracção de todos os tópicos do tipo *Emigrante*, os quais conterão um nome e até quatro ocorrências: idade, profissão (provenientes dos *datasets Alm-Emigrante* e *dataset DViagem-Emigrante*), nome do pai e nome da mãe (originadas do *dataset Passaporte-Emigrante*). Esse processo de associação dos tópicos com os *datasets* deve ser realizado para todos os tipos de tópicos definidos na ontologia desse domínio.

Para a especificação de associações, apresenta-se o exemplo do relacionamento entre *Emigrantes* e *Histórias de Almanaque*. A associação abaixo então relaciona os tais papéis com os tópicos que desempenharão cada função:

```
1          <association dataset="Alm-Emigrante">
2              <instanceOf>
3                  <topicRef xlink:href="#Emigrante-historiaAlmanaque"/>
4              </instanceOf>
5              <member>
6                  <roleSpec>
7                      <topicRef xlink:href="#emigrante-referido-historia"/>
8                  </roleSpec>
```

```
9                <topicRef xlink:href="@Alm-Emigrante.@ID"/>
10           </member>
11           <member>
12               <roleSpec>
13                   <topicRef xlink:href="#historia-refere-emigrante"/>
14               </roleSpec>
15               <topicRef xlink:href="@Alm-HistoriaAlmanaque.pessoa"/>
16           </member>
17       </association>
```

Baseada nesta especificação de associações, será criado uma nova associação toda vez que os campos contidos nos *datasets* @Alm-Emigrante.@ID e @Alm-HistoriaAlmanaque.pessoa forem iguais. Na prática, quando isto ocorrer, indica que um emigrante está sendo citado numa história.

A cardinalidade, neste caso, será definida pelas fontes de dados; ou seja, se um mesmo emigrante está referido em várias histórias de almanaques, será criada uma associação para cada referência encontrada nas fontes. Assim, um mesmo emigrante pode ser citado em várias histórias de almanaque.

Extracção do topic map com o Oveia. O processo de extracção por intermédio do Oveia foi realizado em 4 etapas. Na Etapa 1 são definidas as informações referentes ao projecto de extracção. Nesta etapa, pode-se escolher um projecto existente para re-processar a extracção (caso o mesmo já tenha sido realizado anteriormente) ou criar um novo projecto.

Após definir o projecto, definem-se as configurações de saída do Oveia, na Etapa 2. Para a especificação do armazenamento do topic map, selecciona-se qual é o tipo da base de dados a ser criada; por exemplo: SQL Server, MySQL, Oracle, etc. A seguir, insere-se os `drivers` de conexão com a base de dados, o protocolo utilizado, o endereço do servidor da base de dados, o utilizador e a palavra-passe necessárias para o estabelecimento da conexão. Para a especificação da criação do topic map num ficheiro XTM[6], basta seleccionar a opção *XML file*.

A Etapa 3 consiste no processo de extracção propriamente dito. Nesta etapa realiza-se a execução da extracção. Após o término do processamento, uma estatística completa referente ao processo é fornecida, onde é informado o tempo de execução de todo processo, os elementos extraídos e o número de elementos que não foram extraídos, devido a alguma falha.

Ao final de todo o processamento, foi obtido um ficheiro de acordo com a sintaxe XTM com 35588 linhas, contendo a especificação de 1043 tópicos, 25 tipos de tópicos, 1541 associações e 32 tipos de associações, com um tamanho final de 1,14 MB (1.197.322 bytes). Por sua vez, o Oveia armazenou o topic map gerado em tabelas, o qual contem o mesmo número de tópicos e associações que a versão XTM.

3.3 Navegação no topic map com Ulisses

A partir do topic map gerado pelo Oveia, o Ulisses se encarrega de produzir o website referente ao domínio do ME.

A integração das fontes é percebida na visualização das informações referentes a um emigrante. A Figura 3 apresenta a página HTML correspondente ao emigrante *Sérgio Carvalho*. Além da ocorrência textual do tipo *Profissão*, o emigrante em questão está envolvido em 27 associações, as quais são de 12 tipos distintos. Essas associações foram definidas a partir dos dados extraídos das 3 fontes. Por exemplo:

- a associação "*Sérgio Carvalho é citado em uma nota com a data 1985-08-13*" é oriunda de um diário de viagem, que citava o emigrante;
- a associação "*Sérgio Carvalho é considerado alto, forte e barrigudo*" tem origem num almanaque;
- a associação "*Sérgio Carvalho tinha o passaporte CM420589*" surgiu dos dados obtidos a partir da base de dados dos passaportes.

Analisando desta maneira, toda a informação referente a este emigrante está apresentada de uma forma homogénea, pois para o utilizador que está a navegar no website pouco importa de qual fonte vieram os dados responsáveis por definir uma associação de um conceito; o importante é obter toda a informação disponível nas fontes de dados sobre o conceito desejado.

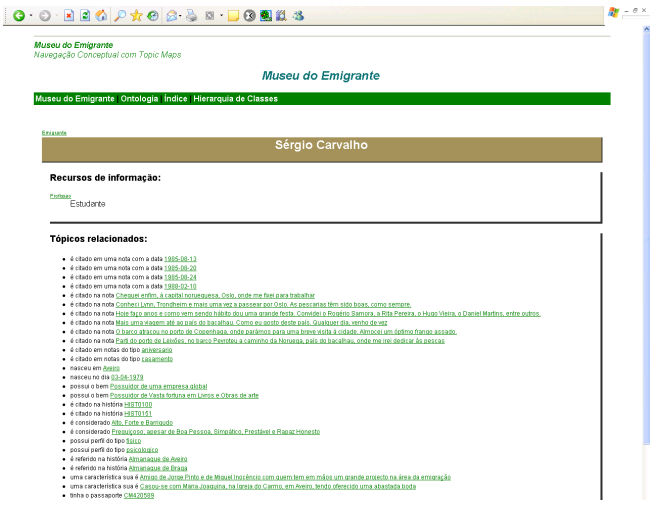


Fig. 3. Página sobre um emigrante

A página referente a um emigrante é o melhor exemplo de uma integração de fontes heterogéneas, pois como se pode observar na Figura 2 (referente à ontologia sobre o domínio do ME), o conceito *Emigrante* está relacionado com histórias de almanaques, passaportes e histórias de viagens, ou seja, todas as fontes de dados podem fornecer informação sobre o mesmo emigrante.

Através dos *links* que descrevem cada associação, pode-se navegar conceptualmente através do grafo formado pelo topic map do ME. Portanto, cada *link*

representa um arco que conecta dois nodos relacionados (as páginas que descrevem os conceitos).

4 Conclusão

O caso de estudo em questão permitiu aplicar o Metamorphosis na combinação de fontes heterogéneas de dados que alimentam o Sistema de Informação do Museu da Emigração (SIME), de modo a facultar aos possíveis visitantes uma navegação sobre os dados extraídos das fontes de uma forma transparente e homogénea.

Em termos práticos, foram utilizados dois dos módulos do Metamorphosis para a obtenção da finalidade pretendida: o Oveia extraiu os dados contidos nas fontes heterogéneas de informação (uma base de dados de passaportes e dois documentos XML de almanaques e diários de viagem), construindo um topic map baseado numa especificação XS4TM; e o Ulisses gerou um website a partir do topic map validado, no qual se pode navegar sobre os recursos do SIME.

A representação do conhecimento do domínio do Museu da Emigração está então descrita num topic map que contém a rede semântica formada pelos conceitos e os seus relacionamentos. Este topic map pode ser intercambiado posteriormente com outras aplicações pois a sintaxe XTM é a mais difundida e aceite pela comunidade académica e empresarial para a representação de Topic Maps.

O intercâmbio do topic map gerado com outras aplicações pode vir a permitir que o conhecimento acerca deste universo de discurso possa ser integrado com outros topic maps que representem domínios relacionados ao Museu da Emigração, aumentando assim a riqueza semântica da rede resultante.

Igualmente importante, é o facto de se poder gerar uma vista completamente diferente dos dados retirados das mesmas fontes, bastando para isso reformular a ontologia a representar, definindo-a em XS4TM.

References

1. Michel Biezunsky, Martin Bryan, and Steve Newcomb. ISO/IEC 13250 - Topic Maps. ISO/IEC JTC 1/SC34, December 1999. http://www.y12.doe.gov/sgml/sc34/document/0129.pdf.
2. Thomas R. Gruber. How to Design an Ontology. In *Web page as part of the Ontolingua guided tour*. Accessed in April 2002, 1995.
3. Giovani Rubert Librelotto. *XML Topic Maps: da Sintaxe à Semântica*. PhD thesis, Departamento de Informática, Escola de Engenharia, Universidade do Minho, 2005.
4. Giovani Rubert Librelotto, Weber Souza, José Carlos Ramalho, and Pedro Rangel Henriques. Using the Ontology Paradigm to Integrate Information Systems. In *International Conference on Knowledge Engineering and Decision Support*, pages 497–504, Porto, Portugal, 2004.
5. Jack Park and Sam Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*, volume ISBN 0-201-74960-2. Addison Wesley, 2003.
6. Steve Pepper and Graham Moore. XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification, August 2001. http://www.topicmaps.org/xtm/1.0/.

Modelação de um Mercado da Pequena Geração Dispersa através de Agentes e Serviços *Web*

Nuno Correia[1], Andreia Malucelli[2,3], Nuno Fidalgo[4,5], Luís Custódio[6], Benedita Malheiro[1,2]

[1] ISEP - Instituto Superior de Engenharia do Porto
[2] LIACC - Laboratório de Inteligência Artificial e Ciências de Computadores
[3] PUCPR - Pontifícia Universidade Católica do Paraná
[4] INESC - Instituto de Engenharia de Sistemas e Computadores do Porto
[5] FEUP - Faculdade de Engenharia da Universidade do Porto
[6] ISR - Instituto de Sistemas e Robótica, IST - Instituto Superior Técnico

Resumo O objectivo deste trabalho é a criação de um modelo do mercado energético da pequena geração dispersa através de serviços *Web*, agentes móveis e leilões. Neste cenário, o mercado, supervisionado pelo leiloeiro, é constituído basicamente por dois tipos de actores: os vendedores – com uma determinada carteira de pequenos produtores de energia, equipados com diversos tipos de geradores, e os compradores – entidades que distribuem e comercializam energia, bem como grandes consumidores.
Apresenta-se a arquitectura adoptada, composta por agentes estáticos e agentes móveis, assim como a metodologia de desenvolvimento integrado elegida. Esta metodologia especifica uma abordagem, suportada pela tecnologia XML, que permite, a partir da informação relativa aos intervenientes, criar uma ontologia comum de representação do conhecimento do domínio, gerar automaticamente os agentes que modelam os intervenientes e, por último, transformá-los em serviços *Web*.
Os agentes compradores e vendedores participam no mercado através de agentes móveis, a quem delegam a sua representação durante o leilão. O trabalho, que está em curso, encontra-se na fase da desenvolvimento dos agentes/serviços *Web*.

1 Introdução

A liberalização do sector eléctrico visa criar um ambiente competitivo, no qual o negócio da geração de energia se encontra totalmente aberto à concorrência. Na Europa, cerca de 70% da totalidade dos consumidores pode actualmente escolher livremente o seu fornecedor de energia e prevê-se que, a médio prazo, a adopção da Electricity Directive 2003/55/EC resulte na liberalização total do Mercado Europeu de Electricidade. A iminente abertura do Mercado Ibérico de Electricidade (MIBEL), prevista para Junho de 2006, pode ser encarada como um primeiro passo neste sentido.

A par da liberalização progressiva dos mercados, espera-se a curto prazo uma proliferação de pequenos geradores distribuídos pela rede eléctrica, baseados

sobretudo em energias renováveis. Esta previsão da proliferação da pequena geração dispersa (PGD) do tipo renovável decorre da meta estabelecida pela Comissão Europeia de, em 2010, 12% da energia produzida no seu espaço passar a ser proveniente de fontes de energia renovável, em particular de geradores eólicos, pilhas de combustível e minihídricas. Outros incentivos para esta ocorrência estão relacionados, por um lado, com a adesão ao protocolo de Quioto, no sentido de reduzir a emissão de CO_2 e, por outro lado, com a necessidade de optimização do investimento nas estruturas de transporte e distribuição de energia. Prevê-se ainda que uma parte considerável da PGD renovável será baseada em sistemas eólicos, os quais apresentam actualmente a melhor relação de custo por MWh produzido [9], [13].

Este trabalho enquadra-se no âmbito do projecto GENEDIS[1] - "Análise do Impacto da Pequena Geração Dispersa sob diferentes Directivas de Regulação". O objectivo é proceder à análise dos efeitos que a PGD terá nos mercados de energia abertos onde os produtores independentes decidirão, de acordo com as directivas de regulação aplicáveis e em função do preço de mercado da energia, quando e quanto produzir. Assim, pode-se afirmar que o GENEDIS apresenta duas vertentes: por um lado, pretende estudar o impacto que diferentes directivas de regulação produzirão nos mercados da PGD e, por outro lado, pode ser interpretado como uma ferramenta de apoio à decisão na implementação da legislação de regulação.

A modelação do mercado da PGD é um problema (i) distribuído – composto por múltiplos actores autónomos com papeis e objectivos distintos; (ii) dinâmico – o preço da energia, que é definido pela lei da oferta e da procura, é influenciado pelas estratégias individuais dos participantes no mercado; (iii) aberto – o número e tipo de participantes varia de sessão para sessão; e complexo - a quantidade de intervenientes, a variedade dos seus perfis e a multiplicidade de cenários de regulação são factores de dificuldade acrescida. Desta forma, o recurso a sistemas multiagente afigura-se como uma alternativa adequada face aos processos de modelação mais convencionais [8]. A escolha de leilões para modelo de negociação automática resultou de dois conjuntos de razões: por um lado, é o modelo utilizado em vários mercados de energia do mundo real e, por outro lado, trata-se de "um dos modelos de negociação mais utilizado no domínio do comércio electrónico [10]". Os leilões são ainda particularmente adequados para "ambientes dinâmicos, dado que permitem, de uma forma muito eficiente, atribuir bens ou serviços às entidades que mais os valorizam [12]". Num leilão, um ou mais agentes leiloeiros (no nosso caso, um leiloeiro) conduzem o processo em que um ou vários agentes realizam ofertas de acordo com o protocolo de interacção imposto, resultando na realização de negócios segundo as leis do mercado.

Este artigo descreve as características do modelo de mercado electrónico proposto. Trata-se de um sistema multiagente baseado em leilões composto pelo agente leiloeiro, no papel de regulador do mercado, e pelos agentes móveis que representam os agentes autónomos (do tipo estático e com interface do tipo

[1] O projecto GENEDIS é financiado pela Fundação para a Ciência e Tecnologia.

serviço *Web*) que modelam os produtores e compradores. Os agentes, as suas características, relações e os serviços que disponibilizam são especificados na ontologia comum do domínio.

A secção 2 descreve, em traços gerais, os moldes da liberalização do Mercado Português de Electricidade e da criação do Mercado Ibérico. São também descritas as áreas de responsabilidade, bem como os organismos criados por cada um dos países com vista à gestão dessas mesmas áreas. As tecnologias utilizadas são descritas na secção 3 e as principais iniciativas XML de suporte ao comércio electrónico são referidas na secção 4. Na secção 5 são apresentadas a abordagem, a arquitectura, a ontologia e a metodologia de negociação propostas. Por último, apresentam-se as conclusões e as direcções futuras deste projecto.

2 Liberalização do Mercado Eléctrico

As regras e o funcionamento dos mercados de energia podem apresentar características bastante distintas consoante o país ou grupo de países. Em [3] é apresentada uma síntese das características dos mercados liberalizados europeus. Em todos estes mercados o leilão constitui uma das actividades fundamentais do funcionamento em regime aberto, quer dentro do país, quer em termos de trocas energéticas entre países. A Union of Electricity Industry - Eurelectric[2] considera mesmo que os leilões constituem os únicos métodos baseados em mercados (*market-based*) para alocação da capacidade de interligação [2].

O processo de liberalização do mercado eléctrico representará uma mudança profunda para a realidade portuguesa. Com a criação do MIBEL as empresas portuguesas e espanholas passarão a operar num sistema de bolsa, com concorrência alargada – com um número estimado de 50 milhões de consumidores, será o quarto maior mercado de energia da União Europeia – e, no caso de Portugal, para um mercado totalmente liberalizado (em Espanha a liberalização teve lugar em 1997) que integra os sistemas eléctricos ibéricos, sem distinção operacional de fronteiras.

O MIBEL caracteriza-se por uma bolsa de energia única, que contempla a possibilidade de operações de mercado a prazo, operações diárias e intradiárias. A gestão das transacções diárias e intradiárias ficará a cargo do Operador do Mercado Ibérico de Espanha (OMIE), enquanto que o Operador do Mercado Ibérico de Portugal (OMIP) efectuará a gestão das operações a prazo. A organização do mercado propriamente dito ficará a cargo do Operador do Mercado Eléctrico (OMEL). Os produtores de energia eléctrica afixarão a sua oferta de produção no mercado, sendo esta cruzada com a procura, materializada nas propostas de compra efectuadas pelos compradores. O preço encontrado para a energia é ainda supervisionado por um organismo de cada um dos países, que exercerá a tarefa de regulador do mercado liberalizado e velará pelo seu correcto funcionamento.

O acesso à bolsa de energia será aberto às empresas de distribuição e comercialização de energia, bem como a consumidores qualificados.

[2] http://www.eurelectric.org

3 Tecnologias Utilizadas

Os agentes foram implementados usando o Java Agent DEvelopment Framework[3] (JADE). O JADE é uma plataforma de desenvolvimento de agentes que cumpre os padrões da Foundation for Intelligent Physical Agents[4] (FIPA) e que possibilita a implementação de sistemas multiagente, disponibilizando também um conjunto de ferramentas de gestão, monitorização e desenvolvimento.

A ontologia de representação do conhecimento do domínio foi especificada recorrendo ao editor de ontologias Protégé[5], tendo os agentes sido criados, automaticamente, usando um módulo adicional – BeanGenerator[6] – que gera, a partir de uma ontologia comum, classes Java que podem ser usadas na definição de agentes JADE.

Os agentes resultantes assumem a forma de serviços *Web* recorrendo ao Web Services Integration Gateway[7] (WSIG). O WSIG é um módulo adicional do JADE que integra as tecnologias de agentes e serviços *Web* de forma bidireccional, i.e., tanto permite aos agentes invocar serviços *Web* em máquinas remotas, como disponibilizar as suas funcionalidades sob a forma de serviços *Web*. Esta integração é conseguida através de um agente – o Web Service Gateway Agent – que inspecciona os serviços publicados no Directory Facilitator[8] (DF) da sua plataforma de execução. Sempre que um serviço for publicado com o tipo "webservice", este agente cria, a partir da descrição no formato Agent Communication Language (ACL), um ficheiro de descrição do serviço no formato Web Services Description Language (WDSL) e afixa-o num serviço de registo conforme com o padrão Universal Description, Discovery and Integration (UDDI) [14]. A partir desse momento, o serviço passa a poder ser invocado por clientes externos à arquitectura. O processo de invocação do serviço é também garantido pelo Gateway Agent. Ao receber uma mensagem Simple Object Application Protocol (SOAP) de invocação de um serviço *Web* que assegura, o Gateway Agent encarrega-se de a traduzir para uma mensagem FIPARequest, contendo os parâmetros que extraiu da mensagem original, e de a enviar ao agente que disponibiliza o serviço. Ao receber a resposta – sob a forma de uma mensagem FIPAInform – converte-a novamente para o formato SOAP e envia-a para o agente cliente.

O caso inverso – registo de um serviço *Web* sob a forma de um serviço de agente – tem uma mecânica reversa mas análoga.

4 Iniciativas XML para Comércio Electrónico

Nos últimos anos surgiram múltiplas iniciativas de apoio ao desenvolvimento do comércio electrónico suportadas pela tecnologia XML, designadamente a

[3] http://jade.tilab.com
[4] http://www.fipa.org
[5] http://protege.stanford.edu
[6] http://acklin.nl/page.php?id=34
[7] http://jade.tilab.com/news-art.php?id=29
[8] O Directory Facilitator é o serviço de registo e descoberta da plataforma JADE.

Electronic Business XML (ebXML), RosettaNet, BizTalk Framework e as classificações de produtos e serviços da United Nations Standard Products and Services Codes[9] (UNSPSC), North American Industry Classification System[10] (NAICS), Standard Classification of Transported Goods[11] (SCTG) e E-cl@ss[12]. A maioria destas propostas destinam-se ao *business–to–business* (B2B), i.e., ao comércio electrónico entre empresas. De entre as mais simples destacam-se as taxonomias de classificação de produtos e serviços da UNSPSC, NAICS, SCTG e E-cl@ss, e entre as mais completas encontram-se os enquadramentos propostos pelas iniciativas ebXML, RosettaNet e BizTalk Framework.

A Electronic Business XML[13] (ebXML) providencia um enquadramento destinado ao B2B baseado na partilha de serviços empresariais através da *Web*. A ebXML suporta sequências coreografadas de trocas entre serviços de negócio que estão definidos em diversas especificações, nomeadamente de serviços de mensagens, serviços de registo e repositório, perfis de protocolos de colaboração e acordos, metodologias para processos de negócio, etc.

A RosettaNet[14] define padrões para a implementação de processos de comércio electrónico abertos e de âmbito industrial tais como catálogos, facturas e encomendas. Este objectivo é alcançado através da definição do padrão XML RosettaNet Document Type Definition (DTD) e de *schemata* do tipo XML Data Reduced (XDR). A RosettaNet encontra-se organizada em camadas que incluem serviços de mensagem, serviços de registo e repositório, processos e dicionários de negócio e uma especificação universal da arquitectura.

A proposta BizTalk Framework[15] especifica uma arquitectura de referência e um roteiro para o desenvolvimento de soluções de integração de aplicações. A arquitectura baseia-se na troca de componentes entre aplicações e organizações através de mensagens XML. As mensagens XML são definidas através de um XML Schema e pelo conjunto de elementos XML que podem ser utilizados nas mensagens. O BizTalk define ainda regras para formatação, transmissão, recepção e processamento de mensagens padrão XML.

5 Plataforma de Modelação do Mercado

A plataforma de modelação do mercado electrónico da PGD encontra-se na fase de desenvolvimento. Dado que se trata de um problema distribuído, dinâmico e complexo, optou-se por uma arquitectura do tipo sistema multiagente constituída por agentes autónomos que competem no mercado para alcançar os seus objectivos. Em relação ao protocolo de negociação, escolheu-se um protocolo do tipo leilão, uma vez que se trata de um problema onde é necessário adoptar

[9] http://www.unspsc.org
[10] http://www.census.gov/epcd/www/naics.html
[11] http://www.statcan.ca/english/Subjects/Standard/sctg/sctg-intro.htm
[12] http://www.eclass-online.com
[13] http://www.ebxml.org
[14] http://www.rosettanet.org
[15] http://www.microsoft.com/biztalk/techinfo/framwork20.asp

uma estratégia que permita determinar de forma eficiente o custo dos produtos/serviços segundo a lei da oferta e procura e as regras do mercado. Actualmente, o trabalho encontra-se na etapa final do desenvolvimento dos agentes estáticos, faltando ainda implementar a criação do mercado.

5.1 Abordagem

Foi adoptada uma metodologia de desenvolvimento integrado com base nas ferramentas de desenvolvimento seleccionadas e nos objectivos deste trabalho. A concepção do protótipo foi iniciada após o estudo das propriedades dos diferentes tipos de unidades de produção utilizados na PGD e do funcionamento dos mercados energéticos.

Começou-se pela construção de uma ontologia do conhecimento do domínio, onde se especificaram os diversos intervenientes (compradores, vendedores e leiloeiro), as suas características, acções e as suas relações. A partir desta ontologia comum foram gerados, de forma automática, os códigos dos agentes JADE correspondentes. Os agentes estáticos que modelam os compradores e vendedores podem residir em plataformas distintas da plataforma do mercado. Os agentes compradores e vendedores foram ainda enriquecidos com uma interface do tipo serviço *Web*, para se poderem registar num serviço de registo público do tipo UDDI. Esta opção permite ao agente leiloeiro encontrar todos os agentes compradores e vendedores que pretendem participar no leilão, independentemente da plataforma onde vai decorrer o leilão.

As regras que definem o tipo do leilão assim como a especificação dos perfis comportamentais que os agentes (produtores e clientes) podem exibir no mercado encontram-se especificadas em ficheiros externos. Compete ao utilizador, aquando do lançamento dos agentes, seleccionar o tipo pretendido. Esta decisão permite: (i) adoptar uma filosofia de desenvolvimento modular; (ii) realizar experiências com diferentes cenários de regulação; (iii) analisar o comportamento dos agentes com perfis competitivos distintos; (iv) manter privada a informação que configura a estratégia individual dos agentes. Esta última opção foi tomada para assegurar a correcta modelação do cenário competitivo dos mercados da electricidade. Assim, no âmbito da ontologia comum do conhecimento do domínio apenas se define a estrutura genérica (conceitos, atributos e relações) deste tipo de conhecimento, i.e., não se definem quaisquer valores.

5.2 Arquitectura

A arquitectura proposta para a modelação do mercado energético da PGD é um sistema multiagente composto por agentes autónomos do tipo competitivo. A figura 1 ilustra a arquitectura proposta. Existem duas categorias de agentes: (i) os agentes que modelam os consumidores e os fornecedores de energia; e (ii) os agentes que participam directamente no mercado. Enquanto os primeiros são os agentes estáticos equipados com uma interface do tipo serviço *Web*, os segundos são o agente leiloeiro e os agentes móveis que representam agentes estáticos convocados para o leilão.

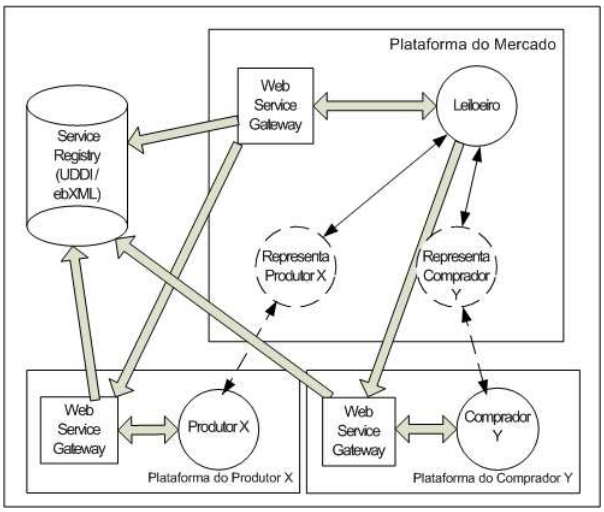


Figura 1. Arquitectura do Sistema.

Agente Leiloeiro O leiloeiro está incumbido de desencadear os leilões. Nos momentos pré-determinados pelas regras do mercado, o leiloeiro verifica se existem simultaneamente clientes registados com necessidades energéticas e produtores em condições de satisfazer a procura. Em caso afirmativo, convoca-os para o leilão, resultando na criação de um conjunto de agentes móveis delegados. Cabe ainda ao leiloeiro o papel da condução do leilão segundo as regras previstas para o mercado. Findo o leilão, os agentes móveis reportam aos respectivos agentes estáticos os resultados alcançados através das interfaces serviço *Web* previstas.

Agente Produtor O agente produtor implementa um serviço destinado a informar o leiloeiro acerca da sua disponibilidade para efectuar o fornecimento de uma dada quantidade de energia num determinado intervalo de tempo. A resposta deste serviço determina a elegibilidade do produtor para o leilão. Em caso de disponibilidade por parte do produtor, este é convocado para o leilão através da invocação do serviço *Web* respectivo, resultando na criação e envio de um agente móvel para o leilão. O comportamento que o agente móvel irá exibir é especificado pelo agente produtor.

Agente Cliente O agente cliente informa, através do serviço *Web* pré-definido, acerca da quantidade de energia e da duração do fornecimento que necessita. O agente leiloeiro invoca este serviço para determinar as necessidades energéticas do agente cliente. Havendo condições para realizar o leilão, o agente leiloeiro convoca o agente para o leilão, resultando na criação e envio de um agente móvel para o leilão. O comportamento que o agente móvel irá exibir é definido pelo agente cliente.

5.3 Ontologia

A interacção entre agentes num sistema multiagente requer uma plataforma de comunicação, uma linguagem de comunicação e uma ontologia comum. Por ontologia entende-se a "representação do conhecimento de algum domínio, que é deixada disponível para todos os componentes de um sistema de informação [7]". O comércio electrónico, enquanto área de enorme potencial de crescimento onde a interacção entre sistemas autónomos heterogéneos é sistemática, constitui um domínio de aplicação de ontologias por excelência. Acresce ainda o facto de "os mercados electrónicos estarem continuamente a fornecer novos tipos de serviços de interacção entre fornecedores e compradores [4]". Assim, é essencial adoptar metodologias que permitam melhorar a interoperabilidade entre agentes, i.e., recorrer a ontologias que providenciem uma interpretação comum do vocabulário usado durante a comunicação.

É neste contexto de comércio electrónico, mais especificamente no contexto do mercado electrónico da electricidade que se insere o trabalho proposto. O mercado electrónico resultante, apesar de apenas envolver empresas, é misto porque, entre as empresas compradoras, coexistem empresas revendedoras de energia – comércio do tipo *business–to–business* (B2B) – e empresas que desempenham o papel de clientes finais – comércio do tipo *business–to–customer* (B2C). Tanto quanto sabemos não existem disponíveis ontologias para este tipo de domínio. Por esta razão fomos confrontados com a necessidade de desenvolver de raiz uma ontologia para o domínio do comércio electrónico da PGD – uma ontologia suportada pela tecnologia XML.

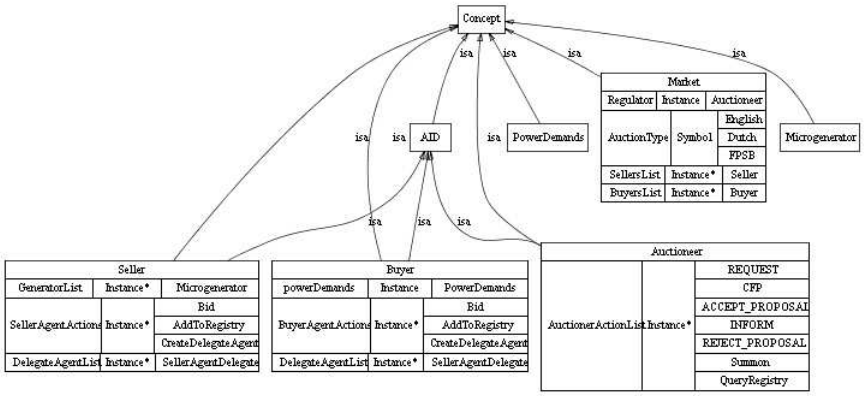


Figura 2. Ontologia do Sistema.

Criou-se uma ontologia comum de representação do conhecimento do domínio do mercado da PGD (ver a figura 2) a partir do conhecimento prévio das características dos elementos envolvidos nas transacções electrónicas do mercado

de electricidade. De uma forma resumida apresentam-se alguns dos elementos especificados na ontologia:

- Definição das entidades Produtor, Comprador, Leiloeiro, Delegado e Mercado;
- Definição das acções dos agentes: consulta e afixação no serviço de registo, convocação para o leilão, criação de delegado, licitação, compra e venda. As acções de interacção no mercado correspondem às primitivas FIPA-ACL do tipo de leilão seleccionado;
- Características dos agentes Produtor: tipo de gerador (microturbina, *fuelcell*, minihídrica ou fotovoltaico), estimativa da produção (horas cheias, de vazio e de ponta) e características genéricas da função de remuneração;
- Características dos agentes Cliente: estimativa do consumo (necessidades energéticas das próximas 24 horas) e características genéricas da função de licitação;
- Características do agente Leiloeiro: tipo de leilão, processo de licitação, regras do leilão, criação de um histórico a ser entregue a cada um dos representantes dos agentes/serviços *Web*, etc.

Esta ontologia foi desenvolvida com o editor de ontologias Protégé [5]. A escolha do Protégé deveu-se principalmente à sua extensibilidade - trata-se de um sistema com uma arquitectura que permite o desenvolvimento e integração de módulos adicionais. Três destes módulos são utilizados neste trabalho: (i) o módulo Ontology Web Language[16] (OWL), que permite exportar e importar ontologias descritas em OWL; (ii) o módulo BeanGenerator que mapeia objectos do modelo Protégé para as correspondentes classes Java, as quais podem ser posteriormente utilizadas pelos agentes JADE; e (iii) o módulo OntoViz que permite a visualização gráfica das ontologias criadas. A ontologia resultante é uma ontologia em OWL, i.e., uma ontologia suportada pela tecnologia XML.

5.4 Negociação

A negociação automática que vai ocorrer entre fornecedores e consumidores tem por objectivo a celebração de contratos que, dadas as condições do mercado, satisfaçam ambas as partes. Neste cenário de comércio electrónico, os modelos de negociação automática utilizados encontram-se divididos em duas categorias: leilões e negociações bilaterais [6]. A negociação pode ser do tipo um-para-um, um-para-muitos ou muitos-para-muitos e, consequentemente, diferentes protocolos de negociação podem ser concebidos [11].

O tipo de protocolo de negociação escolhido é, pelas razões já referidas, o leilão. Os leilões podem ser agrupados em duas grandes classes, os leilões unilaterais e os bilaterais. Nos leilões unilaterais apenas um dos tipos de agentes envolvidos pode ofertar, i.e., os vendedores colocam ofertas de venda ou os compradores efectuam ofertas de compra. Nos leilões bilaterais, os vendedores podem

[16] http://www.w3.org/TR/owl-features

efectuar ofertas de venda e os compradores podem realizar ofertas de compra em simultâneo.

Apesar de no mercado da PGD concorrerem simultaneamente múltiplos compradores e vendedores, são usualmente aplicadas um conjunto de regras pelo operador do mercado (*vide* as regras do Mercado Espanhol[17]) que simplificam o problema: (i) o leilão é efectuado na véspera e diz respeito aos 24 períodos de uma hora do dia seguinte; e (ii) o preço apurado para cada período mantém-se fixo durante todo o período. Estas simplificações permitem conduzir o mercado através de leilões unilaterais do tipo um-para-muitos. Existem quatro tipos de leilões unilaterais: o Leilão Inglês (as ofertas são públicas, efectuam-se por ordem ascendente e ganha a última oferta – a mais elevada), o Leilão de Ofertas Seladas (ganha a oferta mais elevada), o Leilão de Vickrey (as ofertas são seladas e ganha a segunda oferta mais elevada) e o Leilão Holandês (o leilão é conduzido por ordem descendente do valor do artigo, as ofertas são públicas e ganha a primeira oferta que for efectuada).

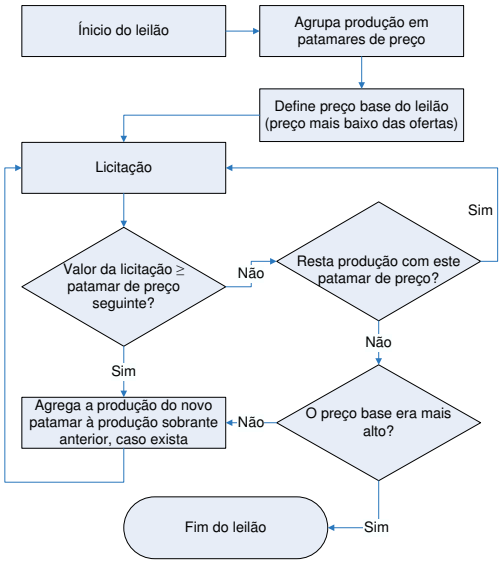


Figura 3. Processo de Licitação.

Na nossa plataforma, o leiloeiro convocará todos os interessados (compradores e produtores) registados no servidor de registos para que se façam representar no leilão. Os agentes com intenção de participar vão delegar em agentes móveis a sua representação na sala de leilão virtual. Antes de se iniciar o leilão, os produtores indicam ao leiloeiro a produção que pretendem vender e o preço por unidade de produção (MWh). O leiloeiro procederá à licitação dos 24 intervalos

[17] http://www.omel.es

de fornecimento de forma consecutiva. Com uma perspectiva global da oferta, o leiloeiro define patamares de agrupamento de produção em função dos preços propostos pelos produtores, cabendo aos compradores efectuar a licitação.

No caso do Leilão Inglês, a licitação tem início com a oferta de energia ao preço mais baixo. O leilão decorre segundo as regras de leilão de preço único, em que o preço é determinado através do cruzamento da oferta com a procura. Assim, o preço a que será negociada a energia será o do produtor mais caro que seja necessário incluir para satisfazer a procura prevista para o período em causa. Sempre que o valor da licitação ultrapasse um patamar de preço, a totalidade das produções com esse preço será adicionada à oferta. O procedimento repete-se até que se escoe a produção de todos os patamares de preço ou a procura esteja totalmente satisfeita. O fluxograma da figura 3 ilustra o funcionamento descrito. Serão realizadas experiências com os diferentes tipos de leilões unilaterais.

6 Conclusões

O modelo proposto para o mercado energético da PGD apresenta um conjunto de propriedades que importa realçar:

1. Permite replicar o comportamento dos mercados físicos – caso do mercado Espanhol, Inglês e do País de Gales [1].
2. Possibilita análises do tipo *what/if* em relação à aplicação de diferentes regras de mercado assim como à adopção de diversos perfis de negociação. Esta característica vai ao encontro dos objectivos do projecto GENEDIS, onde se pretende analisar o impacto produzido por diferentes directivas de regulação nos mercados da PGD.
3. Os agentes (produtores e clientes) que modelam os intervenientes no mercado apresentam interfaces padrão do tipo serviço *Web*, possibilitando uma distribuição transparente por múltiplas plataformas e um elevado nível de conectividade.
4. A decisão de se delegar a participação no mercado em agentes móveis permite aos agentes produtores e clientes participar simultaneamente em múltiplos mercados.
5. A adopção da metodologia de desenvolvimento descrita possibilita uma integração eficiente desde a criação da ontologia comum de representação do conhecimento do domínio até à criação do sistema multiagente. Consequentes actualizações da ontologia são também facilmente repercutidas no sistema.

Num futuro próximo pretende-se terminar a implementação deste modelo através da criação do mercado electrónico. Nas etapas subsequentes prevê-se a realização de experiências envolvendo agentes com diferentes estratégias e leiloeiros que implementam regras de mercado distintas. Também se antevê a modelação de agentes vendedores detentores de uma carteira constituída por múltiplos pequenos produtores de energia – cenário de agrupamento de produção.

Trata-se, do ponto de vista funcional, de um sistema de apoio à decisão.

Referências

1. Ebadi, A., Hersch, M., Rolaz, L.: *Agent-Based Modelling of Electricity Market*, École Polytechnique Fédérale de Lausanne, June (2002)
2. Eurelectric Position Paper, *Improving Interconnection Capacity Allocation*, August (2005). Available at http://www.europa.int/energy/electricity/florence/doc/florence_12/eurelectric_capacity_allocation.pdf
3. Eurelectric Working Group Trading, *Regulatory Aspects of Electricity Trading in Europe*, February (2003). Available at http://public.eurelectric.org
4. Fensel, D.: *Ontologies and Electronic Commerce*, IEEE Intelligent Systems, pg 8. January/February (2001)
5. Gennari, J. , Musen, M.A., Fergerson, R.W. Grosso, W.E., Crubézy, M. Eriksson, H. Noy, N.F., Tu, S.W.: *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*, Technical Report, SMI Report Number: SMI-2002-0943 (2002)
6. He, M., Jennings, N. R., Leung, Ho-Fung: *On Agent-Mediated Electronic Commerce*, IEEE Transactions on Knowledge and Data Engineering, Vol. 15, N.º 4, July/August (2003)
7. Huhns, M.N., Singh, M.P.: *Readings in Agents*, Morgan Kaufmann Publishers, INC. San Francisco, California (1997)
8. Jennings, N. R., Wooldridge, M.: *Applications of Intelligent Agents* in Agent Technology: Foundations, Applications and Markets, Eds. N. R. Jennings and M. J. Wooldridge, SpringerVerlag, (1998).
9. Parbor, P.: *Texas Renewable Energy*, Presentation, Energy Planning Council Meeting, Austin, Texas, USA, (2004). Available at http://www.rrc.state.tx.us/tepc/092404meeting.html
10. Parkes, D.C., Ungar, L. H., Foster, D. P.: *Accounting for Cognitive Costs in On-Line Auction Design*, Agent Mediated Electronic Commerce, Eds. Noriega, P., Sierra, C., Lecture Notes In Computer Science; Vol. 1571, Springer-Verlag, (1999).
11. Trastour, D., Bartolini, C., Preist, C.: *Semantic Web Support for the Business-to-Business E-Commerce Lifecycle*, Proceedings of International WWW Conference, Honolulu, Hawaii, USA (2002)
12. Wurman, P.R., *Dynamic Pricing in the Virtual Marketplace*, IEEE Internet Computing, Vol. 5, March/April, (2001)
13. *Wind Power Economics: Wind Energy Costs - Investment Factors*, European Wind Energy Association, (2003). Available at http://www.ewea.org/fileadmin/ewea_documents/documents/publications/factsheets/factsheet_economy2.pdf
14. Van Aart, C., Pels, R., Caire, G.: *Creation and Use of Agent Message Content Ontologies*, (2002). Available at http://members.home.nl/r.f.pels/articles/index.html

7 Agradecimentos

Os autores agradecem a Susana Silva, Bolseira do Projecto GENEDIS, pela informação cedida relativa aos diferentes tipos de unidades de geração.

Especificação e Geração Automática de Navegadores para Redes Semânticas baseados em Interfaces Web

Miguel Domingues[1], José Carlos Ramalho[2]

[1] Universidade do Minho, Mestrado em Informática
{migaldo}@netcabo.pt
[2] Universidade do Minho, Departamento de Informática
{jcr}@di.uminho.pt

Resumo. Os Topic Maps são um conjunto de standards que resultam da investigação contemporânea numa nova área – "Web Semântica". A premissa deste trabalho foi investigar a teoria subjacente à norma ISO/IEC 13250, mais conhecida por Topic Maps, e propor uma arquitectura aplicacional para o desenvolvimento de um navegador Web de Topic Maps, dinamicamente sustentado por um modelo relacional de dados e especialmente concebido para o armazenamento de documentos Xml Topic Maps (XTM), o standard mais difundido da família. O estudo da norma, a apresentação e discussão das questões estruturais mais importantes, e a concepção de uma framework de Topic Maps que alicerce o desenvolvimento de aplicações baseadas no paradigma, constituem simultaneamente o objectivo e o resultado final deste trabalho.

1 Introdução

Em pleno século XX, entre muitas das inovações tecnológicas, assistimos ao nascimento da Internet (1969-1983). Desde a Revolução Industrial, por volta da segunda metade do século XVIII, que a humanidade não dava um passo tão grande rumo ao futuro. A Internet e a rápida divulgação dos seus principais serviços originava uma cadeia de acontecimentos tecnológicos sem precendentes que contribuiu para o desenvolvimento da "Sociedade de Informação". O resultado de uma acção continuada (1989-1996) de proliferação de conteúdos, de forma desmesurada e sem qualquer planeamento de médio ou longo prazo, deu origem ao principal serviço da Internet que hoje se conhece por World Wide Web (WWW).

Nos últimos vinte anos, esta tecnologia permitiu-nos partilhar activa e globalmente um conjunto de informação anteriormente auto-contida em distintos domínios culturais e sociais. Se até então éramos prejudicados pela escassez de recursos ou pela sua reduzida divulgação, hoje enfrentamos o problema do excesso de informação (infoglut). Nesta sequência histórica, a dispersão dos serviços de internet e a ausência de mecanismos de regulamentação universais, justificaram a criação de organizações responsáveis pela normalização do sector. Em 1994, foi fundado o World Wide Web

Consortium (W3C)[1] como o responsável máximo pela recomendação dos standards necessários à regularização da informação na Web. Organizações como esta começaram a discutir os problemas emergentes e apresentaram as primeiras recomendações para o restabelecimento de uma estrutura de conhecimento organizada baseada nos conteúdos da Web.

No contexto desta temática contemporânea denominada de "Web Semântica", nos finais da década de 90 foi publicada a norma ISO/IEC 13250, mais conhecida por Topic Maps, como uma eventual solução para o problema da ausência de meta-informação na Web. O termo Topic Maps é a designação de um paradigma de unificação de conhecimento e informação, especialmente concebido para descrever estruturas com relações complexas e não lineares. Um topic map corresponde à substanciação de uma ontologia em elementos concretos de informação. O conceito aparece na International Organization for Standardization (ISO) por volta de 1998. Muitas vezes referidos como o sistema GPS de informação do futuro, prometem revolucionar a navegação e a pesquisa de informação digital, oferecendo outro ponto de partida válido para a construção de ferramentas baseadas em modelos de conhecimento e gestão de informação semântica.

O estudo meramente teórico de uma norma como os Topic Maps pode revelar-se incompleto se não tivermos à nossa disposição uma aplicação capaz de demonstrar as suas reais potencialidades e que nos permita por à prova a sua aplicação prática com exemplos concretos. O desafio deste trabalho começou precisamente aqui, com a pretensão de contribuir para a utilização mais expressiva desta norma ao nível do desenvolvimento aplicacional, porque acreditamos que mais importante do que criar uma excelente norma é conseguir difundir a sua aplicação por toda a comunidade envolvente. Com base neste princípio, desenvolvemos a teoria de que seria possível ajudar os produtores e os consumidores de topic maps a compreenderem melhor o paradigma através da utilização de um navegador Web de Topic Maps, com interfaces textuais simples e objectivas.

2 O Paradigma Topic Maps

Desde há muito tempo que o homem tenta organizar o conhecimento. O livro, um dos mais antigos elementos da nossa cultura, tem-nos permitido partilhar e transportar o conhecimento da sociedade ao longo dos séculos. O livro é constituído por um índice e um conjunto de capítulos que estruturam o conhecimento transmitido pelos seus autores. O índice é um mapa de informação que, para cada item relevante, nos indica a página e o capítulo onde podemos ler mais acerca do mesmo assunto. É dos primeiros e mais básicos sistemas de indexação de informação que, na sua forma mais

[1] Por esta altura, organizações com objectivos semelhantes noutros sectores já haviam sido fundadas: International Organization for Standardization (1947). Tim Berners-Lee foi o primeiro Director da W3C e ainda hoje é considerado o inventor da World Wide Web.

complexa e evoluída, apresenta conceitos mais ricos como os índices remissivos, os glossários e os thesauri.

Como resultado da evolução cultural e tecnológica, os livros adquiriram uma nova dimensão enriquecendo o seu próprio significado. Um livro é também hoje um espaço de informação digital, como o conjunto de documentos que circulam diariamente na nossa empresa ou, num caso mais geral, a Internet a que todos temos acesso. Mas se por um lado o seu significado se tornou mais abrangente, por outro a transposição das suas características essenciais para os novos significantes está ainda em desenvolvimento. Esta versão moderna carece de algumas das propriedades fundamentais do seu irmão mais velho. Senão vejamos, onde estão os tais índices e glossários? Em resposta a esta questão, os Topic Maps [1] surgem como um mecanismo inovador que materializa a extensão e aplicação dos conceitos subjacentes ao índice remissivo de um livro a um novo domínio de dados, com o objectivo de agilizar a pesquisa e a navegação sobre recursos de informação digital [2]. Neste novo âmbito, o índice é o mapa que elicita o conhecimento sob a forma de uma lista de tópicos, que estão relacionados entre si através de associações, e que indica as ocorrências dos mesmos ao longo de um espaço de informação. Estes três conceitos, simples e concisos, são o centro do paradigma - Tópicos, Associações e Ocorrências. Portanto, sem sabermos, já utilizávamos topic maps desde há muito tempo.

Os Topic Maps podem desempenhar um papel importante, não só ao permitir reunir recursos de informação dispersos e não normalizados num mapa de conhecimento, mas, e sobretudo, ao permitir a cada um dos potenciais receptores da informação encontrar com relativa facilidade os dados factuais mais relevantes, abstraindo-os com precisão de um imenso repositório de recursos. Os topic maps são um dos caminhos a percorrer para a gestão e construção de conhecimento [3] e, tal como o livro, o objectivo continua a ser o mesmo, resolver problemas de gestão de conhecimento.

Nesta área, tradicionalmente confinada aos domínios da inteligência artificial, por exemplo com as redes semânticas, os Topic Maps apresentam-se como uma alternativa que facilita e potencia o desenvolvimento de ferramentas de gestão de conhecimento, mas com mais vantagens comparativamente, na medida em que envolvem no mesmo modelo aquilo que antes existia em modelos separados: o conhecimento e a informação. Essa ponte é conseguida através do conceito de "ocorrência". A título de exemplo, e no contexto de uma empresa, poderíamos considerar que entidades como os produtos, as pessoas e os departamentos podem ser representados como tópicos; as relações entre as entidades podem ser representadas como associações; e, finalmente, os documentos, as normas internas e todos os recursos de informação que se possam imaginar, podem ser representados por ocorrências de tópicos. A simplicidade destes termos - Tópicos, Associações e Ocorrências, aliada à flexibilidade do modelo, conferem aos Topic Maps uma vantagem competitiva na sua utilização como um tecnologia para o futuro. A informação torna-se pesquisável atribuindo aos conceitos uma identidade própria a partir da qual se constroem caminhos múltiplos e redundantes de navegação sobre o

espaço multidimensional de informação. Os caminhos são semânticos e todos os cruzamentos intermédios são claramente identificados com nomes e tipos que nos permitem saber, em qualquer ponto de intersecção, onde estamos e para onde podemos ir – *"you always know where you are and where do you want to go ... today"*. Como é uma camada independente dos dados, é um mecanismo facilmente escalável que adiciona níveis de significados distintos a documentos estruturados ou não estruturados.

A aplicação dos topic maps é muito vasta, desde a criação de índices documentais, tabelas virtuais de conteúdos, sistemas de navegação inteligentes e *knowledge-bases*. Estas características permitem-nos observar os Topic Maps como uma tecnologia com potencial suficiente para revolucionar a forma como navegamos pela informação, como a pesquisamos, como a partilhamos, como a exportamos e como inferimos conhecimento do meta-conhecimento. Estamos perante um standard especialmente concebido para descrever qualquer tipo de relacionamento complexo entre quaisquer entidades conceptuais.

3 Arquitectura Aplicacional

Ao pensar na arquitectura ousámos estabelecer um conjunto de objectivos ambiciosos, assumindo desde o início o risco de não sermos bem sucedidos. Mais do que criar uma simples aplicação, propusemo-nos desenvolver uma framework genérica que potenciasse a concepção de outras ferramentas de gestão e visualização de Topic Maps, para além do próprio navegador Web. Esta infra-estrutura, que dará origem a uma framework aplicacional actual, deverá servir o propósito de demonstrador do paradigma Topic Maps, revelando de uma forma mais prática o seu verdadeiro potencial, por exemplo com a exploração de exemplos online, onde a comunidade possa descobrir rapidamente as vantagens de utilizar esta tecnologia.

Actualmente, existem alguns clientes de topic maps com os quais poderíamos estabelecer uma comparação com base nas características de visualização, no desempenho, na interactividade e facilidade de utilização, na capacidade de actualização de dados, na interoperabilidade aplicacional, no custo, nos pontos de extensão, etc... Por estranho que possa parecer, as soluções[2] para a Web não são assim tantas e destas resolvemos salientar o melhor de dois mundos: Ulisses [4] e Ontopia Omnigator [5].

O Ulisses é uma ferramenta não comercial e Open-source desenvolvida por um grupo de investigação[3] do Departamento de Informática da Universidade do Minho. Na prática, é um compilador que gera um conjunto de páginas HTML a partir de um topic map em formato XTM, aplicando-lhe transformações XSLT. A geração deve ser repetida sempre que o topic map fonte sofrer alterações. É um processo

[2] Outras soluções: http://www.topicmap.com/topicmap/tools.html
[3] Homepage: http://www.di.uminho.pt/gepl

assíncrono, manual e pouco eficiente, contudo não é um aspecto muito importante a ter em conta quando esta operação ocorre raras vezes. A solução final é um Site completamente estático que pode ser consultado com um browser Web cliente. O desempenho da navegação é obviamente excepcional. Apesar das vantagens mencionadas, este gerador de navegadores conceptuais apresenta uma desvantagem considerável: os navegadores gerados são estáticos, situação que não é a mais adequada para cenários em que o topic map ou mesmo os recursos de informação possam estar em constante alteração.

O Omnigator[4] é uma ferramenta comercial, de código proprietário, desenvolvida pela empresa Ontopia. Embora o licenciamento seja Freeware, está fortemente dependente de outros blocos funcionais que compõem um pacote ainda maior – Ontopia Knowledge Suite (OKS) [6], este sim, um produto Shareware. Esta aplicação, depois de carregar um topic map em formato XTM, permite navegar sobre ele através de um Web browser. Na prática, é um interpretador que dinamicamente gera as páginas de navegação a partir de um conjunto de templates HTML. O que começou por ser um projecto de demonstração das potencialidades dos topic maps, rapidamente se tornou num produto de carácter comercial, extremamente completo e eficiente. É uma solução óptima para cenários de constante re-alimentação dos topic maps, contudo é parte constituinte de uma framework fechada e comercial – Ontopia Navigator Framework.

Na nossa opinião, um conceito tão recente e promissor como o dos topic maps teria de ser apoiado com outras soluções, mais completas e mais abertas, que realmente pudessem potenciar a criação de aplicações futuras baseadas nesta tecnologia. A partir da observação do melhor de dois mundos, esta foi a principal razão pela qual decidimos criar uma arquitectura de topic maps como uma framework aberta, desprovida de carácter comercial, e, por outro lado, com as principais mais-valias funcionais de cada um. Em suma, e da observação das soluções apresentadas, retiramos alguns aspectos importantes que incorporamos na nossa arquitectura como objectivos:

- Adoptar uma arquitectura de 3 camadas com um navegador Web de topic maps (Topic Maps Web Navigator) dinamicamente apoiado por um modelo relacional de dados especialmente concebido para o armazenamento de documentos Xml Topic Maps (XTM), o standard mais difundido da família;

- Implementar uma interface aplicacional (Topic Maps API) capaz de potenciar o desenvolvimento de outras aplicações baseadas neste paradigma, para além do próprio navegador, e que permita aos níveis superiores da arquitectura (Clientes) interagir com o modelo de dados através de uma abstração "Orientada a Objectos" proporcionando o máximo de interoperabilidade;

[4] Homepage: http://www.ontopia.net/omnigator

- Criar um modelo relacional de dados (Topic Maps Relational Data Model) para topic maps que demonstre a sua aplicabilidade generalizada no âmbito de outras aplicações;

Nem sempre é fácil cumprir estes requisitos completamente, mas estas foram hipóteses colocadas à partida que poderiam muito bem constituir as métricas de qualidade deste projecto.

4 Modelo de Dados

O desenho de um modelo relacional parte da análise de uma descrição contextual para identificar as entidades, os relacionamentos e os atributos. O aspecto que está aqui em discussão é a referência inicial a partir da qual vamos iniciar a análise. No caso dos Topic Maps, já existem um conjunto de representações formais que nos podem encurtar o processo de transposição, como por exemplo o *xtm1.dtd* da norma XTM 1.0 [7], que é um *Document Type Definition* de validação. Este documento é suficientemente formal e sintético para ser utilizado como ponto de partida da análise e, por outro lado, também são conhecidas algumas regras para o mapeamento de DTDs em modelos relacionais de dados [8]. Por estas razões, o documento *xtm1.dtd* será utilizado como referência no desenho do modelo relacional de dados dos Topic Maps.

Para facilitar a sua interpretação construimos um quadro que nos ajuda a visualizar a relação entre os vários elementos e que estabelece o ponto de partida do nosso processo de análise:

topicMap	topic *	instanceOf *	(topicRef | subjectIndicatorRef)		
		subjectIdentity ?	resourceRef ?		
			(topicRef | subjectIndicatorRef) *		
		baseName *	scope ?	(topicRef | subjectIndicatorRef | resourceRef) +	
			baseNameString		
			variant *	parameters	(topicRef | subjectIndicatorRef) +
				variantName ?	(resourceRef | resourceData)
				variant * (recursive)	
		ocurrence *	instanceOf ?	(topicRef | subjectIndicatorRef)	
			scope ?	(topicRef | subjectIndicatorRef | resourceRef) +	
			(resourceRef | resourceData)		
	association *	instanceOf ?	(topicRef | subjectIndicatorRef)		
		scope ?	(topicRef | subjectIndicatorRef | resourceRef) +		
		member +	roleSpec ?	(topicRef | subjectIndicatorRef)	
			(topicRef | subjectIndicatorRef | resourceRef) *		
	mergeMap *	(topicRef | subjectIndicatorRef | resourceRef) *			

R – Relationship: R - Mandatory Single, R+ - Mandatory Repeatable, R? - Optional Single, R* - Optional Repeatable

Figura 1. *xtm1.dtd* – Quadro de Elementos

Uma observação mais atenta ao quadro de elementos revela que os extremos ou folhas da árvore são sempre expressões do tipo:

(topicRef | subjectIndicatorRef | resourceRef)
(topicRef | subjectIndicatorRef)
(resourceRef | resourceData)
resourceRef
baseNameString

Figura 2. *xtm1.dtd* – Folhas da Árvore

A representação das folhas da árvore pode ser idêntica e depende apenas da natureza dos 5 tipos de elementos que as constituem: *topicRef, subjectIndicatorRef, resourceRef, resourceData, baseNameString*. Para guardar qualquer um destes elementos, e por conseguinte qualquer uma das expressões terminais da árvore, apenas são necessários três campos de dados: *id* – para guardar o identificador; *value* – para guardar o valor; *type* – para guardar o tipo do *value*. Com isto consigo representar qualquer valor de um determinado tipo. Exemplo:

Tabela de Valores

id	Type	value
"TR982"	"topicRef"	"#Universidade"
"SIR500"	"subjectIndicatorRef"	"http://www.uminho.pt"
"BNS32"	"baseNameString"	"Universidade do Minho"
"RD444"	"resourceData"	"A Universidade é ..."
"RR486"	"resourceRef"	"http://www.uminho.pt/alunos"

Os primeiros resultados desta análise abrem caminho para um processo de simplificação que consiste na substituição das expressões terminais pela tríade de atributos (*id, type, value*). A tríade de atributos (*id, type, value*) corresponde, no modelo conceptual de dados, a um atributo composto denominado *typedValue*, isto para não confundir o *id* do próprio elemento com o *id* da tríade. Para generalizar ainda mais o modelo, vamos considerar que os atributos *xml:base* e *xlink:href*, respectivamente dos elementos *topicMap* e *mergeMap*, também são representáveis por um *typedValue*.

Assim, a partir desta visão simplificada do DTD, é agora muito mais fácil e intuitivo desenhar o modelo conceptual de dados para os Topic Maps, aplicando as seguintes regras de transformação: criar uma nova entidade por cada elemento; agregar a cada entidade os atributos do elemento correspondente; criar relacionamentos entre as entidades de acordo com as definições e cardinalidades dos elementos correspondentes. Com a elicitação completa das entidades, dos relacionamentos e dos atributos obtém-se o seguinte modelo conceptual de dados:

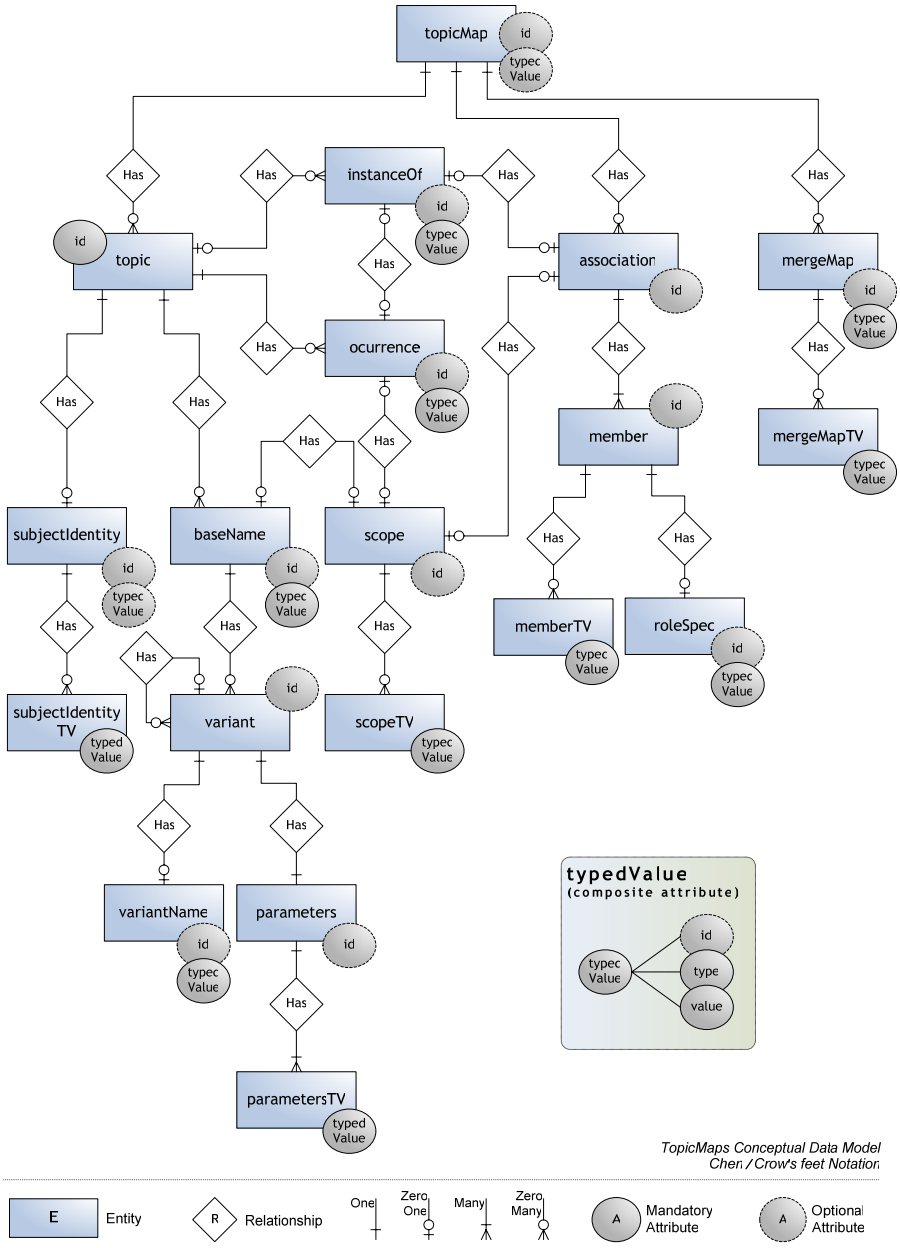


Figura 3. Modelo Conceptual de Dados

O modelo lógico é derivado directamente do modelo conceptual de acordo com a metodologia e as regras de mapeamento do *Entity-Relationship Model* para o *Relational Model* [9].

5 Visualização de Topic Maps

Os topic maps são a ponte entre os domínios da representação do conhecimento e da gestão de informação [10]. Os tópicos e as associações constituem uma rede de conhecimento que se sobrepõe aos recursos de informação e sobre a qual é possível navegar a um nível de abstração muito mais alto. Um topic map define um espaço multi-dimensional de conceitos, em que cada eixo pode representar uma linha de navegação. As camadas superiores podem ser formadas por milhares de tópicos e relações onde os utilizadores facilmente se podem perder.

Podemos comparar os utilizadores a turistas, os tópicos a locais de visita (monumentos, museus, restaurantes, etc) e o topic map à cidade que estão a visitar. O turista quer visitar todos os locais de referência da cidade, mas se, eventualmente durante o passeio, um determinado assunto lhe despertar maior interesse acabará por alterar o seu plano de visitas. Portanto, para que a visita seja bem sucedida o turista deverá conseguir identificar claramente os pontos de interesse e dispor de boas ligações de transporte entre todos eles, para nunca se perder e chegar ao fim da viagem satisfeito. Ainda assim, o turista pode sempre orientar a sua visita por um roteiro, ou deixar-se levar pela aventura da exploração não planeada. Esta parábola contém todos os elementos necesssários à elaboração de um bom navegador de topic maps e as palavras-chave são: Turista, Cidade, Viagem, Transportes, Locais, Roteiros e Aventura.

Os requisitos a ter em especial atenção podem dividir-se em duas categorias principais – Representação e Navegação. Os requisitos de representação reunem todas as características que permitem ao utilizador (Turista) identificar, com maior ou menor detalhe, a informação constante do topic map. Quando o utilizador inicia a navegação (Viagem) por um topic map (Cidade) necessita de uma vista global onde possa analisar a sua estrutura ontológica e consultar uma lista com os tópicos principais (Locais). Depois deve poder escolher um eixo (Roteiros) e partir para uma navegação orientada visitando todos os tópicos que lhe suscitem maior interesse. Cada tópico tem as suas próprias características, como o nome e as ocorrências, e tem também um conjunto de hiperligações (Transportes) para outros tópicos (Locais) relacionados, que poderá querer visitar por interesse ou mera curiosidade (Aventura). A navegação deve ser intuitiva e de fácil dedução, porque é através dela que vamos ajudar o utilizador a construir os seus próprios mapas cognitivos e aumentar a velocidade a que este apreende informação. É o conjunto de uma representação e navegação bem conseguidas que gera a sensação de satisfação ao fim da viagem.

Na sequência da metáfora utilizada, decidimos baptizar o nosso navegador de *"TopicMaps Discovery" (TMD)*, porque nos permite descobrir a informação e o conhecimento presentes num topic map. O TMD é uma ferramenta *multi-TopicMap*, onde é possível a coexistência de vários topic maps, com uma navegação tipicamente orientada ao tópico. Para demonstrar o seu potencial, aproveitamos o exemplo do topic map da "Família do João e da Joana" conceptualizado na imagem seguinte:

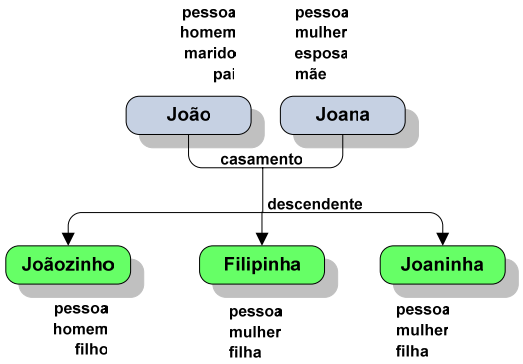


Figura 4. "A Família do João e da Joana"

A "Família do João e da Joana" aparece muito melhor caracterizada desde o momento em que introduzimos um conjunto de termos que classificam cada indivíduo. A esse conjunto de novos conceitos, que nos permitiram "enfeitar" a figura e enriquecê-la com mais informação, chamamos de Ontologia do Topic Map. Depois de seleccionar este topic map no nosso navegador podemos observar a respectiva Ontologia (*"Ontology Index"*):

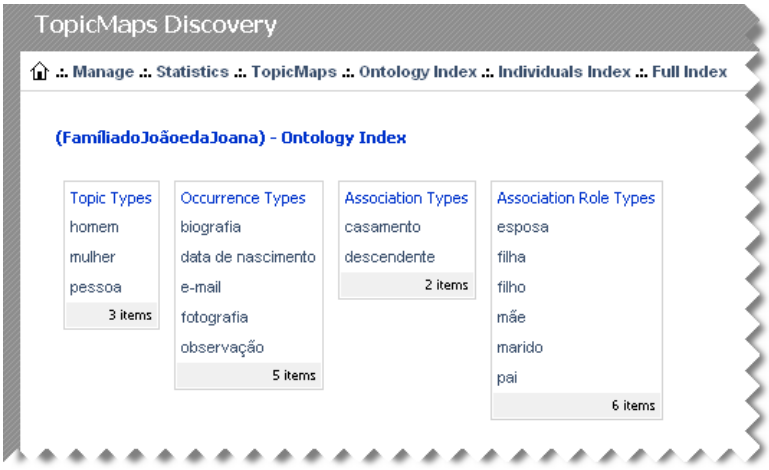


Figura 5. TMD – "Ontology Index"

A partir de qualquer das interfaces do TMD é possível seleccionar um dos tópicos e iniciar uma exploração mais pormenorizada. Por exemplo, na interface dos tópicos individuais (*"Individuals Index"*), se escolhêssemos o tópico "Joana" poderíamos visualizar as suas características tal como se pode observar na imagem seguinte:

Figura 6. TMD – *"Topic"*

Numa interpretação resumida da informação apresentada, podemos ver que o tópico "Joana" é uma "mulher" que nasceu a "4 de Março de 1970" no "Brasil", e o seu email é "joana@hotmail.com". A "Joana" tem um "marido", de nome "João", com o qual celebrou um "casamento". Para além disso, tem ainda três "descendentes" que são as "filhas" "Joaninha" e "Filipinha", e o "filho" "Joãozinho". Também podemos ver uma "fotografia" da "Joana" no endereço "http://(...)/joana.jpg".

Ao visualizar toda esta informação acerca da "Joana" poderíamos ser assaltados pela curiosidade de visitar outros tópicos como "mulher", "casamento", "marido", "João", "descendente", "filha", "Joaninha", "Filipinha", "filho", "Joãozinho", "biografia", "data de nascimento", "email" e "fotografia". Começamos então a ter a sensação de estar a percorrer os caminhos de uma rede de conhecimento, mas de forma intuitiva e organizada. Isto mostra-nos como é extremamente importante disponibilizar não só a informação detalhada do tópico, mas também a possibilidade de navegar directamente para os tópicos que formam a "Vizinhança de Informação". Esta é uma característica singular dos sistemas de navegação semântica, e são exemplos como este que demonstram inequivocamente o potencial da norma Topic Maps.

6 Conclusão

Em retrospectiva, os Topic Maps são uma família de standards, ainda em crescimento, que teve o seu início nos finais da década de 90, com a publicação da norma ISO/IEC 13250. Um topic map é um indexador ontológico que consiste no acoplamento de uma camada de meta-dados a um conjunto de recursos de informação dando origem a uma rede semântica de conhecimento. É um modelo poderoso e inteligente de representação de conhecimento para repositórios de informação continuamente crescentes, como por exemplo a Web. Tal como as ontologias, podem ser usados para acoplar informação conceptual e semântica aos documentos da Web, de modo a que estes se tornem mais úteis.

A partir do estudo inicial do paradigma e em especial da análise detalhada da norma mais difundida (XTM), propusemos uma arquitectura aplicacional para a construção de um navegador Web de topic maps. Projectámos uma arquitectura modular, baseada em três camadas, que deu origem a uma *framework* aberta, extensível e tecnologicamente actual. Desenhámos um modelo relacional de dados, integralmente compatível com a norma XTM 1.0, que nos permitiu extrair apenas a informação relevante em determinado momento, optimizando as operações de consulta e actualização de dados com garantias de integridade e coerência. Sobre este modelo, desenvolvemos uma estrutura de objectos intermédia (Topic Maps API), independente do SGBD e especializada no processamento de operações de dados, para actuar como interface entre os clientes da camada de apresentação, como por exemplo o navegador Web, e o modelo relacional de dados. A camada intermédia foi especificada de forma a poder ser utilizada por outras aplicações, adaptando ou estendendo as suas funcionalidades, de modo a potenciar o desenvolvimento aplicacional nesta área. Finalmente, construímos uma aplicação Web modelo capaz de gerar dinamicamente e em tempo real as páginas de navegação de um topic map, no contexto de um modelo Web cliente-servidor.

Na visualização dos topic maps o sucesso de qualquer ferramenta tem muito ver com os requisitos de “Representação” e de “Navegação”. Quanto ao nosso navegador já tínhamos decidido que seria uma ferramenta essencialmente textual, com uma navegação intuitiva, para ajudar os utilizadores a construirem os seus próprios mapas cognitivos, e com uma representação muito objectiva capaz de mostrar com clareza os detalhes de cada elemento do topic map. Estes dois factores, quando bem conjugados, retribuem ao utilizador uma “satisfação” na atitude exploratória e intuitiva da descoberta de conhecimento. Na prática, é quando um utilizador está a “visitar” um determinado tópico e sente curiosidade em “visitar” outros tópicos que lhe são inteligentemente apresentados, que lhe damos a sensação de estar a “viajar” por uma rede de conhecimento bem organizada. A possibilidade de navegar directamente para a “vizinhança da informação” é uma característica própria dos sistemas de navegação semântica que rapidamente demonstram a potencialidade das redes de conhecimento como os topic maps.

Os topic maps pertencem à classe de objectos a que denominámos de *"Knowledge Webs"*, da qual também fazem parte os *conceptual maps* (mapas de conceitos), as *semantic networks* (redes semânticas), os *cluster maps*, os *mind maps*, os *circle diagrams* e os *flow charts* [11]. O conceito transversal a todos eles é o de que para além das ideias também é necessário integrar na mesma rede de conhecimento as relações existentes entre elas. É a articulação destes dois elementos que define um espaço semântico completo com formatos (texto, imagem, som, vídeo) e modelos (*semantic networks*, *topic maps*, *etc.*) plurais. As comunidades que se preocupam com estas matérias, inclusivé a comunidade dos Topic Maps, esperam que no futuro haja uma maior diversidade de modelos e formatos a par de uma utilização prática mais acentuada, de modo a tornar realidade o grande sonho da *"Semantic Web"*.

Num cenário mundial onde o problema de conectividade está praticamente resolvido, devemos reconhecer que o fenómeno do *infoglut* é o último e mais formidável inimigo do intercâmbio de conhecimento global (*global knowledge interchange*). O Paradigma dos topic maps é mais um passo em frente no caminho da troca de conhecimento global e certamente não será o último, mas para já é bastante significativo. Será a especificação do XTM completa e definitiva? Não, ainda existem muitos detalhes a rever, mas já é uma norma com valor suficiente para ser utilizada.

7 Referências e Bibliografia

[1] L. M. Garshol, "What are Topic Maps?" Xml.Com: O'Reilly, 2002.
http://www.xml.com/pub/a/2002/09/11/topicmaps.html
[2] S. Pepper, "The TAO of Topic Maps - Finding the way in the age of infoglut," Ontopia SA, 2002.
http://www.ontopia.net/topicmaps/materials/tao.html
[3] E. Freese, "Using Topic Maps for the representation, management and discovery of knowledge." XML Europe 2000 Conference, Paris, France, 2000.
http://www.gca.org/papers/xmleurope2000/papers/s22-01.html
[4] G. R. Librelotto, J. C. Ramalho, and P. R. Henriques, "Ontology driven Web sites with Topic Maps." Ovied - Spain: The International Conference on Web Engineering, 2003.
http://www.di.uminho.pt/gepl/IIpCS
[5] "Ontopia Omnigator," Ontopia SA, 2003.
http://www.ontopia.net/omnigator
[6] "The Ontopia Knowledge Suite," Ontopia SA, 2003.
http://www.ontopia.net/solutions/products.html
[7] S. Pepper and G. Moore, "XML Topic Maps (XTM) 1.0 - Annex D: XTM 1.0 Document Type Declaration (Normative)," TopicMaps.Org Specification, 2001.
http://www.topicmaps.org/xtm/1.0/#dtd
[8] R. Bourret, "Mapping DTDs to Databases." XML.com: O'Reilly, 2004.
http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html
[9] R. Ramakrishnan and J. Gehrke, Database Management Systems, 3rd Edition ed: McGraw-Hill, 2003.
[10] J. Park and S. Hunting, XML Topic Maps - Creating and Using Topic Maps for the Web: Addison Wesley, 2002.

[11] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web." Scientific American, May 2001.
http://www.sciam.com/2001/0501issue/0501berners-lee.html
[12] M. Biezunski, M. Bryan, and S. Newcomb, "ISO/IEC 13250:2000 Topic Maps." Geneva: ISO/IEC JTC 1/SC34, 2000.
http://www.y12.doe.gov/sgml/sc34/document/0129.pdf
[13] S. Pepper and G. Moore, "XML Topic Maps (XTM) 1.0," TopicMaps.Org Specification, 2001.
http://www.topicmaps.org/xtm/1.0
[14] S. R. Newcomb, M. Biezunski, and M. Bryan, "Guide to the topic map standardization," ISO/IEC JTC1/SC34 - N323, 2002.
http://www.y12.doe.gov/sgml/sc34/document/0323.htm
[15] K. Ahmed, "Topic Maps - A Practical Introduction with Case Studies." XML Europe 2002 Conference, Paris, France, 2002.
http://62.231.133.220/idea-eks-nav/papers/03-05-01/03-05-01.html
[16] "Ontopia.Net: Topic Mapping," Ontopia AS, 2004.
http://www.ontopia.net/topicmaps
[17] H. H. Rath, "Topic Maps and The Ontological World," Empolis GmbH, 2001.
http://www.empolis.com, http://onto2001.aifb.uni-karlsruhe.de/tm-talk-hhr.pdf
[18] H. H. Rath and S. Pepper, "Topic Maps: Introduction and Allegro," Ontopia SA, 1999.
http://www.ontopia.net/topicmaps/materials/allegro.pdf
[19] G. R. Librelotto, "XML Topic Maps: da Sintaxe à Semântica." PhD Thesis: Departamento de Informática, Universidade do Minho, 2005.
http://wiki.di.uminho.pt/twiki/bin/view/Doutoramentos/SDDI2004
[20] C. F. Goldfarb and P. Prescod, "XML Handbook," 4th Edition ed: Prenctice Hall, 2001.
[21] W. R. Stanek, XML Pocket Consultant: Microsoft Press, 2002.

Utilização de SVG na Visualização de Sinópticos

Filipe Marinho[1], Paulo Viegas[1], João Correia Lopes[23]

[1] EFACEC Sistemas de Electrónica, Rua Eng. Frederico Ulrich, Apartado 3078, 4471-907 Moreira Maia.
http://www.efacec.pt/
{filipe.marinho,pviegas}@se.efacec.pt
[2] Faculdade de Engenharia da Universidade do Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.fe.up.pt/∼jlopes/
jlopes@fe.up.pt
[3] INESC Porto, R. Dr. Roberto Frias, 4200-465 Porto.
http://www.inescporto.pt/

Resumo O GENESys é um projecto desenvolvido pelo Departamento de Gestão de Redes, Unidade de Automação de Sistemas de Energia da *EFACEC*, em parceria com a EDP (*Energias de Portugal*), que visa a implementação de um sistema *SCADA/DMS* (*Supervisory Control and Data Acquisition/Distribution Management System*) que controle a rede eléctrica nacional.
A disponibilização do GENESys na *Web* traz grandes vantagens para a manutenção do sistema, mas para isso é necessário encontrar uma tecnologia que garanta uma interacção, usabilidade e desempenho semelhante à actual, que assenta em aplicações Java.
Neste artigo descreve-se a utilização da tecnologia *SVG* (linguagem em formato XML que descreve gráficos de duas dimensões) para representar sinópticos e permitir a interacção do utilizador do sistema GENESys. Com base num protótipo desenvolvido, foi feita uma avaliação desta tecnologia no sentido de verificar se cobria todos os requisitos para a visualização de sinópticos na Web.
Este estudo mostrou a viabilidade da utilização da tecnologia *SVG* para visualizar sinópticos e actualizá-los dinamicamente com a recepção de informação dos sistemas *SCADA/DMS*. Para além disso, verificou-se que as capacidades interactivas dos *SVG*s permitem enviar comandos e consequentemente actuar sobre o sistema.

1 Introdução

Os sistemas *SCADA* (*Supervisory Control And Data Acquisition*) são responsáveis pela recolha e análise de informação em tempo real. Actualmente estes sistemas recorrem às mais avançadas tecnologias de computação e comunicação para monitorizar e controlar estruturas ou equipamentos industriais dispersos geograficamente e recorrem a interfaces gráficas sofisticadas para tornar a interacção com o utilizador mais amigável.

Este tipo de sistemas é usado em várias indústrias: telecomunicações, águas e saneamento, energia, gás, combustíveis e transportes. Hoje em dia estes sistemas são distribuídos, com utilizadores e dispositivos dispersos, dispositivos estes que poderão falhar sem prejudicar o sistema, uma ou mais bases de dados e com suporte para um desenvolvimento contínuo em várias plataformas de computação.

Os sistemas DMS (*Distribution Management System*), surgiram como uma evolução dos sistemas *SCADA* usados na supervisão de redes eléctricas. Os sistemas *SCADA* monitorizam pontos (equipamentos) da rede que controlam, possibilitando também o seu controlo, mas não têm noção da interligação destes pontos. Por outro lado, os sistemas DMS dispõem de informação do modelo de conectividade da rede, que permite disponibilizar uma série de ferramentas que ajudam no controlo da rede eléctrica.

O GENESys é o resultado de um projecto conjunto entre a *EFACEC Sistemas de Electrónica S.A.* e a EDP (*Energias de Portugal*) para o desenvolvimento de um sistema *SCADA/DMS* de nova geração, no âmbito da execução de um projecto da EDP mais alargado, o projecto SIRED (*Sistemas Integrados para Redes Eléctricas de Distribuição*). O objectivo deste sistema é reduzir o número de centros de controlo da rede de distribuição nacional detidos pela EDP e aumentar a sua eficácia.

O GENESys monitoriza, em tempo real, todas as medidas relativas à rede eléctrica, tais como a corrente, a voltagem, a potência, etc, apresentando-as graficamente para que possam ser interpretadas pelo operador. Da mesma forma é possível, a qualquer momento, actuar sobre os equipamentos: mudar o estado do equipamento, gerir ordens de manobra, ordens de serviço, planeamento e estudo de procedimentos, alocação de recursos e de equipas de reparação.

Alguns sistemas *SCADA*, têm um elevado grau de complexidade, quer na sua instalação quer na sua manutenção, havendo necessidade de ter pessoal especializado nessas funções. Acompanhando a evolução e tendência das tecnologias de informação, pretende-se disponibilizar estes sistemas na *Web*, tendo como vantagens o acesso ilimitado através de um navegador *Web* e a limitação da necessidade de instalação e configuração só nos servidores do sistema.

Um dos módulos mais críticos na disponibilização do GENESys na *Web* é a visualização de sinópticos. Um Sinóptico é a representação gráfica do sistema, que no caso do GENESys é a rede eléctrica (ver figura 2). O transporte da visualização de sinópticos para a *Web* levanta algumas questões importantes. É preciso encontrar uma tecnologia que garanta uma interacção, usabilidade e desempenho satisfatórios em ambiente *Web*.

Neste trabalho é usada a tecnologia *SVG*, por forma a verificar se esta cobre todos os requisitos para a visualização de sinópticos na *Web*. Pretende-se que a representação do sinóptico em *SVG* permita interagir com o sistema GENESys, ou seja, permita alterar a representação dos símbolos consoante a informação recebida do sistema e controlar o estado dos símbolos através da interacção com o sinóptico.

Para além desta introdução, onde se caracterizou o problema abordado por este projecto, faz-se de seguida uma abordagem ao estado da arte e são des-

critos os trabalhos relacionados com a visualização de diagramas sinópticos de sistemas *SCADA* na *Web*. Na secção 2 é feita uma descrição da arquitectura física e tecnológica do protótipo desenvolvido. Na secção 3 faz-se a descrição da arquitectura que suporta o GENESys e das ferramentas utilizadas para o desenvolvimento do visualizador de sinópticos. Na secção 4 é detalhado o desenvolvimento do visualizador *SVG* e na secção 5 são descritos e analisados os testes de desempenho. Finalmente, na secção 6 são apresentadas as conclusões e os trabalhos futuros.

2 Estado da Arte e Trabalhos Relacionados

Nesta secção faz-se a introdução à realidade do sector da Automação de Sistemas, no âmbito deste artigo. Será feito um resumo das tecnologias que podem ser utilizadas na disponibilização de diagramas sinópticos na *Web* e das soluções existentes no sector nesse contexto.

2.1 Clientes Simples e Clientes Complexos

Numa abordagem baseada em clientes simples (*thin client*), a preocupação reside na utilização de tecnologias naturalmente suportadas pelos navegadores e que, por isso, não incorrem na necessidade de descarregamento de blocos de código pesados, tanto pelo seu tamanho, como pela capacidade computacional que necessitam [LFCJ02] [LSFH00] [SLN99] [ZPMD97].

Já numa abordagem baseada em clientes complexos (*rich client*) existe uma clara utilização da capacidade computacional do cliente, em oposição ao que acontece com as abordagens para clientes simples. Esta característica permite a implementação de blocos substanciais de código para a sua execução, podendo em alguns casos, limitar a utilização deste tipo de soluções [Fer03].

2.2 Solução Baseada numa Ligação VPN

Uma das soluções apresentadas por empresas do sector Automação de Sistemas para a disponibilização de sistemas *SCADA/DMS* na *Web*, é uma aplicação cliente baseada em *Java* (*rich client*) que permite o acesso a todas as funcionalidades do sistema *SCADA/DMS*. A aplicação cliente comunica com o sistema *SCADA/DMS* através de uma ligação VPN ou SSL à rede onde os servidores estão alocados. Esta aplicação não é desenvolvida para a *Web*, não havendo preocupações quanto ao tamanho desta pois é instalada *offline*. Nesta solução, a representação de sinópticos na *Web* é feita através de componentes *Java* incorporados na aplicação cliente.

2.3 Páginas HTML

Nesta solução utiliza-se HTML para a definição da interface, através da inclusão de imagens de diagramas geradas pelo servidor, que vão sendo actualizadas ao longo do tempo, através do refrescamento periódico da página.

A execução de controlos é suportada pela inclusão de formulários (ou diálogos gerados por *Javascript*) que resultam na execução de um CGI ou de uma *servlet* no servidor para actuação no processo. Um mecanismo semelhante poderia ser utilizado com vista à inclusão de facilidades de navegação: pedidos de ampliação e arrastamentos que resultavam na geração de imagens ampliadas ou deslocadas, conforme requerido. As animações poderiam ser conseguidas pela utilização de GIFs ou PNGs animados. No servidor haveria a necessidade de implementação de um sistema de geração de imagens, que ia servindo os pedidos recebidos com base na informação recolhida do sistema *SCADA*.

Os principais problemas desta solução são:

- A geração dinâmica de imagens "rasterizadas" (GIFs e PNGs) com animações é pesada e pouco eficiente;
- Há necessidade de carregamento total da página a cada pedido de refrescamento. Uma vez que o conteúdo de uma página poderia incluir uma ou mais imagens dinâmicas de grandes dimensões, a quantidade de informação trocada entre clientes e servidor poderia ser muito grande e, por isso, condicionada pela largura de banda do canal de transmissão;
- Há necessidade de uma taxa de refrescamento elevada. As características de tempo-real dos sistemas *SCADA* levam à necessidade de actualização frequente da informação visualizada. Este factor, em conjunto com a eventual existência de outros clientes que efectuam acessos simultâneos ao sistema, introduz uma enorme sobrecarga no motor de geração de páginas que, por serem complexas e de grandes dimensões, pode conduzir ao atraso na disponibilização da resposta.

2.4 Visualizador Remoto

Um visualizador remoto consiste numa aplicação que permite a visualização de interfaces gráficas que executam remotamente. Esta aplicação seria integrada no navegador (pela utilização de *Java applets* ou *ActiveX*) e comunicaria com um servidor no centro de comando através de um protocolo de transmissão de informação gráfica [LFCJ02] [LSFH00].

Esta é uma técnica conhecida como *Graphics Pipeline Interception* e o seu princípio de funcionamento consiste na existência de uma aplicação que intercepta os comandos gráficos ocorridos na máquina servidora, enviando a imagem resultante para visualização no cliente. Por seu turno, os eventos ocorridos no cliente são transmitidos ao servidor, que os executa localmente.

O problema associado aos sistemas de visualização remota é a largura de banda necessária para a transmissão de informação gráfica através da rede, mesmo que sejam utilizadas técnicas de compressão ou envio de comandos gráficos de alto nível, em vez do envio total da imagem a visualizar. Por outro lado, estes sistemas apresentam a importante vantagem de poderem ser integrados com sistemas existentes, através da intercepção das rotinas gráficas ao nível do sistema operativo.

Actualmente, empresas do ramo da Automação de Sistemas usam esta tecnologia para potenciar o acesso através da *Web* aos seus sistemas e consequentemente a visualização dos seus sinópticos. Como exemplo temos os *Terminal Services* disponibilizados pelos sistemas operativos *Windows Server 2000* e *Windows Server 2003* [Mic04] que permitem, através de um *thin client*, que qualquer máquina (PDA, PC, TabletPC, etc) com um qualquer sistema operativo (Windows, Unix, Mac OS, etc.) possa aceder através da *Web* qualquer aplicação baseada em Windows (neste caso o sistema *SCADA/DMS*) instalada nos servidores. A comunicação é baseada no RDP (*Remote Desktop Protocol*), o qual assegura a encriptação dos dados.

2.5 Componentes Especializados

Esta solução consiste na implementação de uma biblioteca de componentes gráficos *Java*, para a construção de clientes gráficos *SCADA*, que disponibilizam funcionalidades específicas deste tipo de aplicações. Estes componentes poderiam depois ser conjugados em *applets* que seriam descarregadas pelo navegador para execução local.

O desenvolvimento de componentes gráficos permite que grande parte das questões de interacção com o utilizador sejam resolvidas localmente sem necessidade de comunicação com as aplicações residentes no centro de comando, levando não só à diminuição do fluxo de informação entre clientes e servidores, mas também à possibilidade do aumento do grau de interactividade entre operadores e aplicação.

Esta solução permite que a informação transaccionada entre clientes e servidores seja informação de alto nível, que corresponda à própria semântica do problema, em vez de informação de baixo nível, como a presente nos sistemas de visualização remota, onde circulam imagens e directivas de desenho. Assim, os valores correspondentes a alterações de entidades poderão ser enviados directamente a clientes especializados, com conhecimento suficiente para proceder à sua adequada representação, seja por alteração da colocação, por animação de imagens ou simplesmente por apresentação textual do seu valor.

Em oposição a todas as vantagens que o desenvolvimento em *Java* fornece (multi-plataforma, utilização gratuita, segurança, Swing, etc) a utilização de componentes para a construção de interfaces *SCADA* na *Web* obriga à necessidade destes componentes serem relativamente pequenos, para que o processo de descarregamento das aplicações possa ocorrer num intervalo de tempo aceitável.

Outro problema desta abordagem reside na necessidade de implementação específica de componentes para os vários aspectos dos interfaces *SCADA*: sinópticos, listas, gráficos de tendências e alarmes. Mas, uma vez desenvolvidos, estes poderão ser utilizados para interligação com os sistemas existentes. Assim, a implementação de um sistema baseado em componentes gráficos implica a definição e desenvolvimento de uma arquitectura de software que permita estabelecer a comunicação entre os componentes e as aplicações pertencentes a um sistema *SCADA* [Fer03].

3 Utilização de *SVG* na Visualização de Sinópticos

Como vimos nas secções anteriores, pretendemos disponibilizar a visualização de sinópticos e suportar interactividade com o sistema através da *Web*. Das várias soluções tecnológicas ao nosso dispor, testámos a combinação das tecnologias *SVG* e *Java*, assentes na arquitectura distribuída BUS toolkit (ver secção 3.2). Este teste baseou-se no desenvolvimento de um protótipo que nos permitiu retirar as devidas conclusões.

Nesta secção faz-se a descrição detalhada da arquitectura que suporta o GENESys e das tecnologias e ferramentas utilizadas para o desenvolvimento do visualizador de sinópticos [Mar05a].

3.1 *SVG*

Scalable Vector Graphics é uma linguagem em formato XML que descreve gráficos de duas dimensões. Este formato normalizado pela W3C (*World Wide Web Consortium*) é livre de patentes ou direitos de autor e está *totalmente* documentado, à semelhança de outros W3C standards [Wor05b].

Sendo uma linguagem XML, o *SVG* herda uma série de vantagens: a possibilidade de transformar *SVG* usando técnicas como XSLT, de embeber *SVG* em qualquer documento XML usando *namespaces* ou até de estilizar *SVG* recorrendo a CSS (*Cascade Style Sheets*). De uma forma geral, pode dizer-se que *SVG*s interagem bem com as actuais tecnologias ligadas ao XML e à Web [IBM05].

Os *SVG* podem integrar objectos como formas vectoriais (rectângulos, círculos, "caminhos" constituídos por linhas rectas ou curvilíneas), transparências, filtros (efeitos de luz, sombras, etc), imagens e texto. Estes objectos também podem ser interactivos e dinâmicos, características essas que permitem a definição de animações que poderão ser despoletadas através de declarações embebidas nos *SVG* ou através de *scripting* [Wor05a] [Apa05b].

Esta tecnologia poderá ser utilizada na indústria, nomeadamente no suporte gráfico e interactivo dos sistemas *SCADA*, o que faz com que o operador possa monitorizar, controlar e operar sobre estes sistemas [Gar05].

3.2 *BUS* Toolkit

O *BUS* Toolkit é uma arquitectura publicador/subscritor desenvolvida para ser o "coração" do GENESys da *EFACEC* Sistemas Electrónicos, S.A..

Este tipo de sistemas apresenta alguns aspectos particulares que influenciam a filosofia da arquitectura, tais como: a sua complexidade, o facto de ser distribuída, de ter uma grande tolerância a falhas e uma enorme robustez e de ser configurável [MV01].

No ponto de vista do *BUS* um sistema é um conjunto de componentes (unidade básica de processamento). Estes componentes comunicam entre si através do Modelo *Push* ("impulso" ou "empurrão") e de acordo com o paradigma publicador/subscritor. Neste paradigma os componentes publicam sobre "assuntos" (tópicos) que interessam a outros componentes (e vice-versa). Toda a gestão de

quem publica e subscreve tópicos é feita por um componente especial, o Navigator. Este componente tem a função de informar quem publica e subscreve nos diferentes tópicos.

3.3 *BUS* — Canal Web

A possível manipulação de um sinóptico (ligado a um sistema *SCADA/DMS*) através de qualquer navegador *Web*, implica que a comunicação entre o protótipo e o *BUS* tenha de ser feita através da *Web*, usando um canal disponível na JFRK para o efeito [Gom03].

A finalidade do Canal Web é permitir que um qualquer componente que esteja a correr num navegador *Web* de uma máquina qualquer (numa *applet* ou então numa aplicação separada), comunique com o sistema central (onde estão localizadas as base de dados e principais componentes) através de um Canal *Web*, da mesma forma que comunicaria através de RMI ou IIOP (outros canais de comunicação entre componentes disponibilizados pela JFRK), caso estivesse a ser executado na rede local.

A principal razão da existência de um Canal *Web* é possibilitar a comunicação entre componentes em diferentes *Intranets*, o que obriga à passagem pelas *firewalls* que as protegem. Por norma a configuração das *firewalls* não permite a comunicação via RMI ou IIOP, sendo por isso necessário recorrer ao HTTP (*HyperText Transport Protocol*). A comunicação através do Canal *Web* baseia-se no acesso a um serviço *Web* que fará a ponte entre um componente a correr na *Web* e o GENESys (ver figura 1).

Foram elaborados alguns testes, para verificar a capacidade de comunicação através do canal *Web*. Verificou-se que este suporta o envio de 200 eventos por segundo, no teste elaborado dentro de uma *Intranet* com a velocidade de 100 KB/s. Já no teste efectuado com um acesso por VPN (*Virtual Private Network*) à *Intranet* através de uma ligação de 2 Mb/s, verificou-se um limite de 130 eventos por segundo, com a agravante de ter ligar um tempo excessivo no transporte dos eventos entre componentes do *BUS*, o que origina um "longo" período entre o momento em que os eventos são enviados de um componente e o momento em que são recebidos no outro [Mar05a] [Mar05b].

3.4 Batik *SVG* Toolkit

Batik é um conjunto de bibliotecas baseadas em *Java* que permitem o uso de imagens *SVG* (visualização, geração ou manipulação) em aplicações ou *applets*. O projecto Batik destina-se a fornecer ao programador alguns módulos que permitem desenvolver soluções especificas usando *SVG*. Entre esses módulos destacam-se [Apa05a]:

- *SVG* **Generator** — Modulo que permite a todas as aplicações *Java*, converter facilmente gráficos em *SVG* [Apa05c].
- *SVG* **DOM** — Este módulo permite criar uma representação para um qualquer *SVG* com o intuito de permitir a sua manipulação.

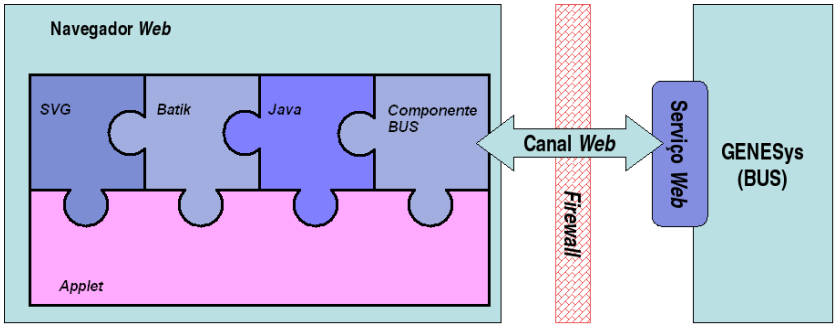


Figura 1. Arquitectura da Solução Proposta

– **J*SVG*Canvas** — Componente gráfico para visualização de *SVG*s e que permite a interacção com estes gráficos (aumentar ou diminuir, rodar, seleccionar texto, etc).

Temos então no Batik uma solução fácil para integrar o *Java* com *SVG*.

4 Visualizador de Sinópticos

Para avaliar as capacidades da tecnologia *SVG* na representação e visualização de diagramas sinópticos na *Web*, optou-se por desenvolver um protótipo que cumprisse os requisitos necessários à visualização de diagramas do GENESys. Este protótipo foi desenvolvido em *Java*, para que possa correr num navegador *Web* (através de uma *applet*) e utilizou-se o Batik *SVG* Toolkit para a visualização e manipulação dos gráficos *SVG*.

Numa fase inicial foram definidos alguns requisitos funcionais, que cobrem os aspectos necessários à visualização de sinópticos na *Web*:

– Criação de um protótipo que permita a interacção com símbolos no *SVG* e que despolete o envio de um evento para o sistema central (simulando assim a possibilidade de actuar sobre o sistema). Esta interacção é feita através de menus ligados ao clique do botão direito do rato.
– O protótipo também deverá receber eventos que mudem o estado dos símbolos e proceder em conformidade (mudar cor, mudar símbolo, etc). Este requisito permite simular a "animação" do diagrama de acordo com a realidade da respectiva rede eléctrica controlada pelo GENESys.
– Cada símbolo no sinóptico tem um menu correspondente. O conteúdo deste menu poderá ser configurado através de um ficheiro XML à parte.

Neste protótipo não foram implementados em *SVG* todos os componentes gráficos que constituem um sinóptico real (ver figura 2). A título exemplificativo foram definidos alguns símbolos para que se pudesse testar a tecnologia. Estes símbolos representam medidas, *switches* e disjuntores nos seus vários estados (ver figura 3).

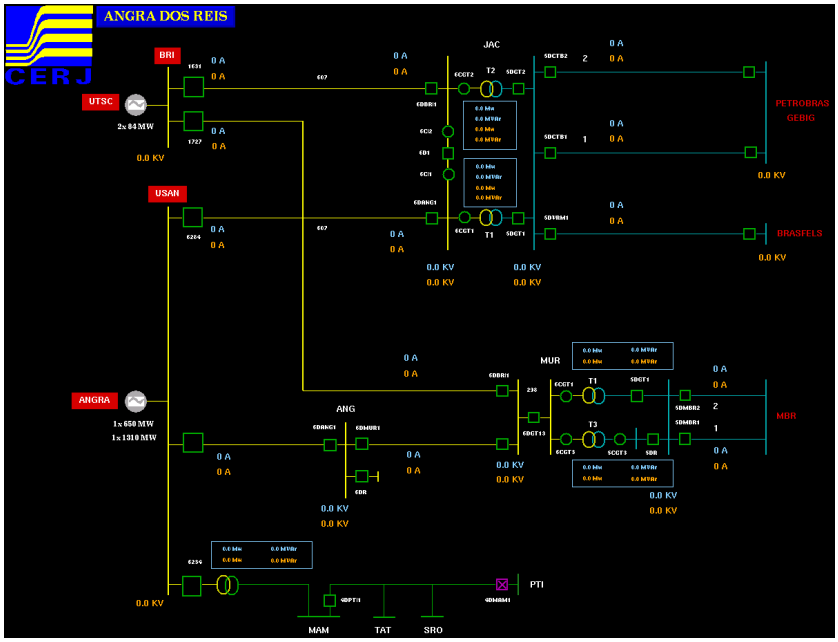


Figura 2. Sinóptico da Área de Angra dos Reis no Brasil

A principal funcionalidade deste protótipo é a de permitir a visualização de sinópticos a partir de qualquer localização, ou seja, através da *Web*, apoiando-se na arquitectura do Canal *Web* do *BUS*. Deste modo, o Visualizador *SVG* será uma *applet*, que correrá sobre um navegador *Web*.

Esta *applet* é um componente *BUS* (ver secção 3.2) que recebe e envia eventos através do Canal *Web*, o que lhe permite actualizar os sinópticos e actuar sobre o sistema (ver figura 4).

A ligação entre a aplicação em *Java* (*applet*) e os gráficos *SVG* é feita através da utilização da ferramenta Batik. Esta ferramenta disponibiliza um painel (*jSVGCanvas*), que permite visualizar e interagir com os gráficos *SVG* (fazer *zoom*, *scroll*, clicar sobre os objectos, etc).

A preocupação no desenvolvimento deste visualizador de sinópticos centrou-se nas funcionalidades, tendo assim uma interface gráfica simples, pensada para facilitar o estudo da tecnologia (ver figura 5)

5 Testes e Avaliação do Visualizador de Sinópticos

O protótipo desenvolvido demonstrou que a tecnologia *SVG* tem capacidades para construir e visualizar sinópticos dinâmicos, permitindo a interacção com o utilizador. Para fazer uma avaliação final sobre esta tecnologia, foi necessário efectuar alguns testes de desempenho. Para isso elaboraram-se alguns cenários que permitiram cobrir todas as áreas que se pretenderam testar: ocupação de

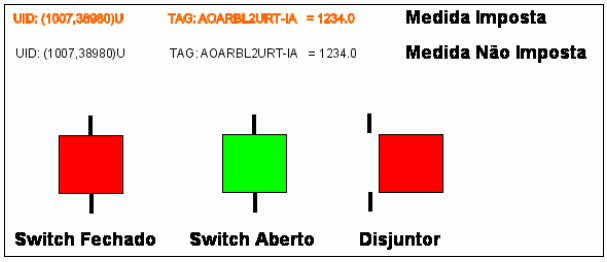


Figura 3. Simbologia Definida para o Protótipo

memória, utilização de *CPU*, desempenho no carregamento e na actualização dos *SVG*s, protótipo multi-plataforma, etc.

A elaboração destes cenários foi suportada pela utilização de dois computadores: um cliente e um servidor. O "servidor" representava o sistema central e permitia o descarregamento (para um navegador) da *applet* do visualizador de sinópticos. O "cliente" tinha a função de descarregar o visualizador e executar todos os testes necessários. O servidor tinha também como função a simulação de eventos que iriam alimentar o diagrama *SVG* no teste de actualização dos símbolos. Para o "cliente" foi usado um computador com um processador Pentium Centrino 1.5 Mhz com 512 MB RAM.

Alguns dos cenários foram efectuados duas vezes mas em modelos diferentes. Um dos modelos foi a elaboração dos testes com os dois computadores ligados à *Intranet* da *EFACEC Sistemas de Electrónica S.A.*, um segundo modelo foi ligar o "cliente" através da *Internet* à *Intranet* da *EFACEC Sistemas de Electrónica S.A.* e consequentemente ao "servidor".

No que diz respeito ao carregamento da *applet* nos vários navegadores testados (*Internet Explorer, Opera, Firefox*), os tempos foram semelhantes numa comparação entre navegadores e no geral aceitáveis, mesmo numa perspectiva de carregamento da *applet* através da *Internet*.

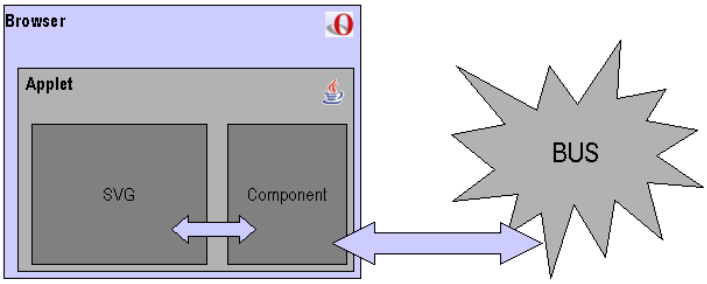


Figura 4. Arquitectura do Protótipo *SVG*

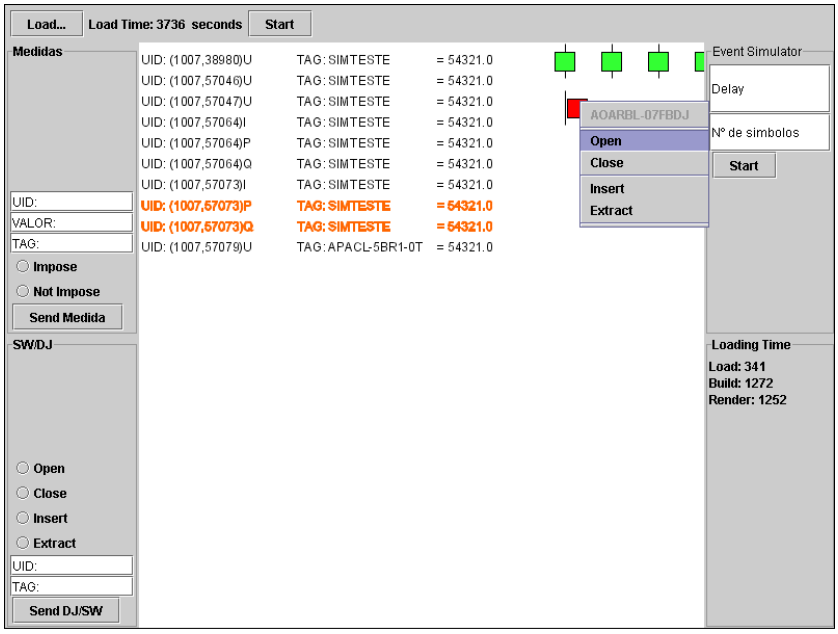


Figura 5. Interface do Visualizador de Sinópticos *SVG*

No carregamento dos diagramas *SVG* os tempos obtidos foram satisfatórios, verificando-se apenas que em diagramas de grande dimensão o carregamento era impossível devido à falta de memória virtual. Este problema foi resolvido com a passagem de parâmetros à VM (*Virtual Machine*) com o intuito de alocar mais memória virtual aos processos *Java*.

O cenário mais crítico foi o teste do desempenho do visualizador na actualização dos símbolos de acordo com o eventos que lhe iam chegando. Os resultados não foram satisfatórios, na medida em que o visualizador só conseguiu tratar eficazmente (tendo carregado um sinóptico de tamanho dentro dos limites impostos pela EDP (500 símbolos)) 8 eventos por segundo. Esta situação deve-se ao facto das operações de manipulação da árvore DOM, que representa o diagrama em memória, serem bastante pesadas (ver figura 6). Este limite coincide com uma utilização máxima do *CPU* do "cliente".

No mesmo cenário, mas correndo o visualizador fora da *Intranet* através de um computador ligado à Internet com uma ligação ADSL de 2Mbits por segundo, a eficácia no tratamento de eventos e consequente actualização dos símbolos manteve-se, tendo o mesmo diagrama carregado. O mesmo não aconteceu aquando a utilização de uma ligação GPRS de 56Kb por segundo, onde os resultados desceram para 4 eventos por segundo. Esta situação deve-se à reduzida largura de banda o que origina um sucessivo atraso no descarregamento dos eventos (agrupados em pacotes) e consequentemente um atraso no seu tratamento.

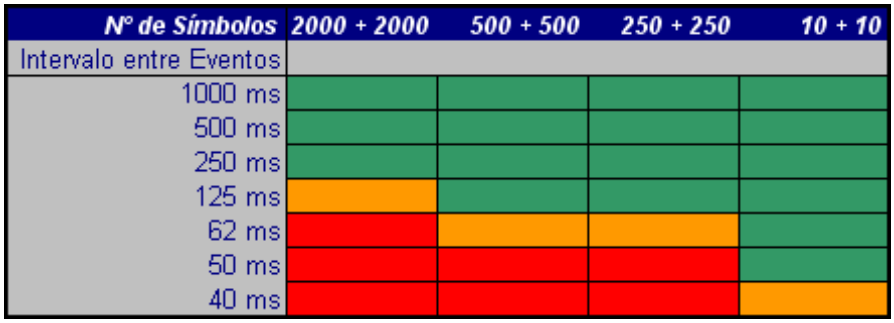


Figura 6. Resultado dos Testes de Desempenho na Actualização dos Símbolos

Verificados estes resultados, houve a necessidade de elaborar o mesmo teste, mas mudando a maneira de actualização dos símbolos. Em vez de substituir cada símbolos de acordo com os eventos recebidos, altera-se apenas uma propriedade do símbolo (mudando a cor, texto, etc). Esta versão do protótipo levou a resultados muitíssimo melhores, chegando a tratar 160 eventos por segundo, com o mesmo diagrama referido anteriormente. A conclusão retirada foi que a operação de substituir a representação de um símbolo do diagrama na árvore DOM é bem mais pesada que a operação de alterar apenas uma sua propriedade.

6 Conclusões

Neste artigo abordou-se o desenvolvimento de um protótipo, com vista a estudar a adequadibilidade da tecnologia *SVG* à visualização de sinópticos na *Web*.

No que diz respeito à capacidade da tecnologia *SVG* para a representação e visualização de diagramas, interacção com o sistema *SCADA/DMS* e desempenho no carregamento dos gráficos *SVG*, os resultados mostraram-se satisfatórios. Por outro lado, quanto à capacidade de actualização dos diagramas em tempo real, de acordo com os eventos que recebe do sistema, os resultados não atingiram o mínimo aceitável. Este facto explica-se pela utilização da API DOM, o que implica uma elevada ocupação de memória e uma baixa performance no tratamento da árvore DOM (associada aos gráficos *SVG*).

Estes testes também demonstraram que esta baixa performance se deve à forma como os símbolos são actualizados no gráfico, ou seja, se a animação se basear na substituição dos símbolos (substituição de nós na árvore DOM) verifica-se uma baixa performance, por outro lado se esta se basear na alteração de propriedades dos símbolos (não há substituição de nós) a performance torna-se bastante elevada.

Em conclusão, os *SVG* mostram-se capazes de suportar a visualização dos diagramas, a comunicação com o sistema (GENESys) e também a interacção com o utilizador.

Como trabalho futuro fica a melhoria na implementação da animação dos diagramas, de acordo com a informação recebida do sistema. Este trabalho de-

verá passar pela alteração da representação dos símbolos em XML, por forma a permitir a animação através de mudanças apenas nos valores de propriedades.

Referências

[Apa05a] Apache. Batik *SVG* Toolkit Architecture. `http://xml.apache.org/batik/architecture.html#coreComponents`, Junho 2005.

[Apa05b] Apache. Batik SVG Toolkit: Scripting. `http://xml.apache.org/batik/scripting.html`, Maio 2005.

[Apa05c] Apache. Batik *SVG* Toolkit: *SVG* Generator. `http://xml.apache.org/batik/svggen.html/`, Junho 2005.

[Fer03] Ricardo Jorge Nogueira Fernandes. Interfaces Web para aplicações SCADA. Mestrado em Ciências de Computadores, Faculdade de Ciências da Universidade do Porto, Fevereiro 2003.

[Gar05] Rodrigo García García. *SVG* for SCADA. Swiss Federal Institute of Technology Lausanne (EPFL). `http://www.svgopen.org/2004/papers/SVGforSCADA/`, Abril 2005.

[Gom03] Miguel Ferreira Pereira Gomes. Canal SCADA na *Web*. Mestrado em Ciências de Computadores, Faculdade de Ciências da Universidade do Porto, Fevereiro 2003.

[IBM05] IBM. Program with SVG. `http://www-128.ibm.com/developerworks/xml/library/x-matters40/`, Maio 2005.

[LFCJ02] Simon Lok, Steven K. Freiner, William M. Chiong, and Yoav J.Hirsch. A graphical user interface toolkit approach to thin-client computing. *Proceedings of the Eleventh International Conference on World Wide Web*, pages 718–725, 2002.

[LSFH00] Sheng Feng Li, Quentin Stafford-Fraser, and Andy Hopper. Integrating synchronous and asynchronous collaboration with virtual networking computing. *Proceedings of the First International Workshop on Intellingent Multimedia Computing and Networking, Atlantic City, USA*, Vol. 2, pages 717–721, Fevereiro–Março 2000.

[Mar05a] Filipe Marinho. *Actividade Sem Concepção — Sinópticos na Web*. Relatório interno, *EFACEC Sistemas de Electrónica S.A.*, Maio 2005.

[Mar05b] Filipe Marinho. *Testes de Sistema — Sinópticos na Web*. Relatório interno, *EFACEC Sistemas de Electrónica S.A.*, Agosto 2005.

[Mic04] Microsoft. Windows server 2003 terminal services. `http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technolo%gies/featured/termserv/default.mspx`, 2004.

[MV01] Dave Marsh and Paulo Viegas. The bus architecture. *IEEE Porto PowerTech 2001* `http://power.inescn.pt/powertech/`, Setembro 2001.

[SLN99] Brian K. Schmidt, Monica S. Lan, and J.Duane Northcutt. The interactive performance of SLIM a stateless, thin-client architecture. *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles (SOSP)*, Vol. 34, pages 32–47, Dezembro 1999.

[Wor05a] W3C World Wide Web Consortium. W3C — About SVG. `http://www.w3.org/TR/SVG/intro.html/`, Abril 2005.

[Wor05b] W3C World Wide Web Consortium. W3C SVG Specification. `http://www.w3.org/TR/SVG11/`, Junho 2005.

[ZPMD97] Debora J. Zukowski, Apratim Purakayastha, Ajay Mohindra, and Murthy Devarakonda. Metis: A thin-client application framework. *Proceedings of the Third Conference on Object-Oriented Technologies and Systems*, pages 103–114, Junho 1997.

Linguagens XML de Descrição de Cenas com Conteúdos Multimédia

Pedro Pinto[1], Isidro Vila Verde[2]

[1] Departamento de Ciências Básicas de Computação
Escola Superior de Tecnologia e Gestão,
Instituto Politécnico de Viana do Castelo
Apartado 574, s/n 4901-908 Viana do Castelo
pedropinto@estg.ipvc.pt
[2] Departamento de Engenharia Electrotécnica e de Computadores
Faculdade de Engenharia da Universidade do Porto,
Rua Dr. Roberto Frias, s/n 4200-465 Porto
jvv@fe.up.pt

Abstract. Neste artigo fazemos uma análise comparativa de formatos de linguagens de descrição de cenas dinâmicas baseadas em XML. Apresenta-se uma descrição resumida de vários formatos, focando as suas principais características. A necessidade de análise comparativa surge num contexto de desenvolvimento de um ambiente de e-monitorização de provas electrónicas de avaliação. Nesta perspectiva, são definidos critérios de avaliação adequados a este contexto e com base nestes critérios é realizado um estudo comparativo, identificado o formato mais adequado para o fim especificado.

1 Introdução

No contexto aplicacional existe já um número significativo de produtos multimédia, quer 2D ou 3D. Por forma a realizar seus objectivos, estes sistemas possuem várias componentes que trabalham para a produção, manipulação e apresentação dos objectos nas cenas multimédia. Na composição destas cenas são utilizadas linguagens específicas para o efeito chamadas linguagens de descrição de cena, que representam uma entidade essencial no processo e tem a intenção de descrever de uma forma eficaz e adequada uma cena multimédia, descrevendo todos os objectos que a compõem e definindo todos os seus estados nesse ambiente. Neste momento podemos encontrar várias linguagens possuindo características e objectivos muito próprios.
É objectivo deste artigo fazer uma análise comparativa de algumas dessas linguagens numa perspectiva particular do desenvolvimento de uma aplicação de monitorização de provas electrónicas. Na secção seguinte é apresentado o contexto no qual surgiu esta necessidade. Na secção 3 são descritas de forma resumida as linguagens analisadas. Na secção 4 formulam-se os critérios de análise orientados para o contexto da aplicação. Na secção 5 apresentam-se os resultados da análise considerando os critérios definidos e as características particulares das linguagens.

Por último, são tiradas conclusões desta análise e é feito o enquadramento da opção escolhida na aplicação em desenvolvimento.

2 Contexto

A necessidade desta análise surge no contexto do desenvolvimento de uma aplicação específica, desenvolvida no âmbito de uma tese de Mestrado. Esta tese tem por base de trabalho, o desenvolvimento de um modelo de monitorização remota, vocacionado especialmente para ambientes de formação. A arquitectura de monitorização para este modelo está apresentada na Fig.1.

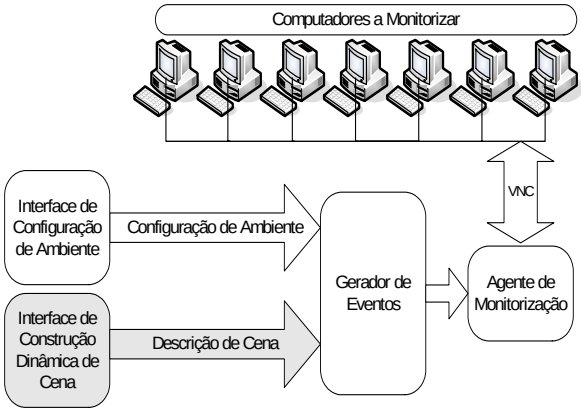


Fig. 1. Diagrama de blocos da arquitectura de monitorização de provas electrónicas de exame

Nesta arquitectura, a responsabilidade de monitorização pertence ao "Agente de Monitorização". A monitorização consiste em duas vertentes, a monitorização dos ecrãs remotos e a monitorização dos processos.
A monitorização dos ecrãs é realizada recorrendo ao protocolo Remote Frame Buffer (RFB)[1] disponibilizado pelo software Real Virtual Network Computing (RealVNC)[2] que trata de estabelecer fluxos deste protocolo para cada computador a monitorizar, fluxos estes que transportam todos os pixels do ambiente de trabalho visto por cada utilizador desses mesmos computadores. Na Fig.2 é apresentado um exemplo de um cena do processo de monitorização.

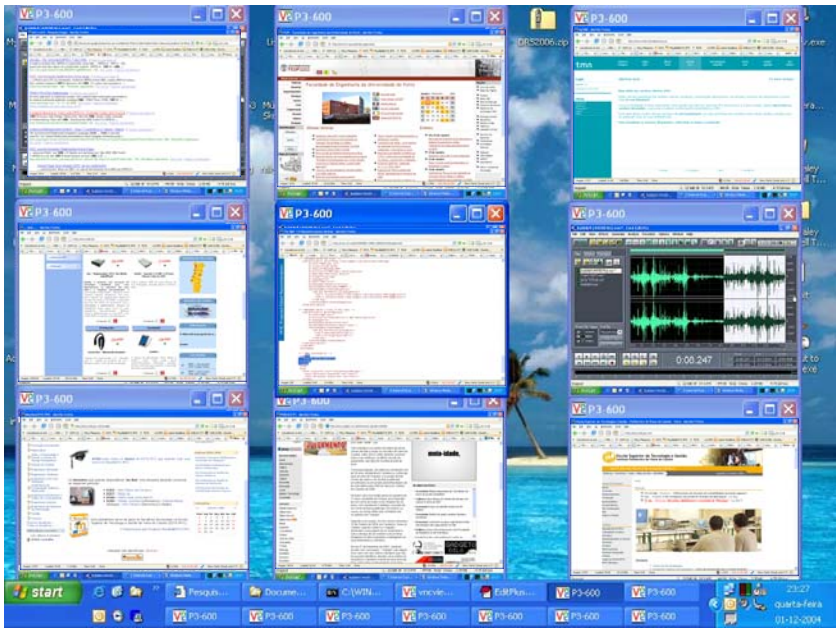


Fig. 2. Exemplo dos objectos presentes na cena multimédia da aplicação a desenvolver

A monitorização dos processos recorre à interface Windows Management Interface (WMI)[3] para configurar a geração de alertas a enviar ao "Agente de Monitorização". Estes alertas são disparados quando são realizadas acções relevantes, por exemplo, quando são iniciadas aplicações ou quando existem acessos não autorizados a determinadas páginas Internet.

Desta forma, implementa-se uma monitorização activa e passiva das provas electrónicas que estão a ser realizadas pelos formandos num determinado ambiente.

Em ambientes de larga escala esta monitorização não pode ser permanente e terá de haver programas de monitorização que automatizem as "rondas" periódicas pelos diversos postos.

O programa de monitorização é gerado automaticamente na entidade "Gerador de Eventos" à custa de um ficheiro de descrição de ambiente e um fluxo de descrição dinâmica da cena. O ficheiro de descrição de ambiente é criado previamente à realização do exame e descreve o ambiente da realização da prova. Esta descrição é feita num formato XML simples, validado segundo um XSD[4] definido, de acordo com a seguinte semântica:

```
- ambiente a monitorizar: "Sala"
            - informação de identificação do aluno: "Pedro Pinto"
            - número identificador do computador (IP): "192.168.0.20"
            - posição relativa do computador no espaço: "15"
```

Quanto ao fluxo de descrição de cena este apresenta-se como um fluxo constante que traduz em cada instante os objectos que devem ser monitorizados e como eles se apresentam na cena de monitorização. Esta descrição apresenta-se como parte fundamental deste sistema e deve estar definida numa linguagem de descrição de cenas com suporte para cenas dinâmicas. Isto é, cenas onde os objectos visualizados mudam em função da decisão humana, em cada instante.
Esta linguagem de descrição de cena terá, por um lado, de traduzir do modo mais fiel construções de cenas possíveis realizadas na "Interface de Construção de Cena" e por outro terá de ser uma linguagem adequada e consistente para a leitura no "Gerador de Eventos". Existem duas informações principais enviadas sob a linguagem de descrição de cena. A primeira informação define o espaço de monitorização, com a definição da divisão do ecrã e a posição de cada janela de monitorização de acordo com a seguinte semântica:

```
- Espaço de divisão do monitor: "4 Colunas 3 Linhas"
- Janela número "1" com posição na "Linha 1 Coluna 3"
- Janela número "2" com posição na "Linha 3 Coluna 2"
```

A segunda informação transporta dados sobre a dinâmica espacial e temporal da cena, apresentados com a seguinte semântica:

```
- imagem do computador da posição "1" visualizado na janela "2" com
início aos "4" segundos e duração de "2" segundos
- imagem do computador da posição "15" visualizado na janela "4" com
início aos "2" segundos e duração de "4" segundos
```

O conjunto destes elementos constitui um fluxo contínuo que traduz em cada instante a posição de todos os elementos na cena de monitorização. Por conseguinte, existe a necessidade de identificar a melhor escolha no formato desta linguagem num conjunto de linguagens possíveis de linguagens de descrição de cena e, por isso, é realizado neste documento um estudo comparativo neste contexto.

3 Linguagens

Das várias linguagens que podem ser encontradas para a componente essencial da aplicação referida, existe um subconjunto específico que é relevante para este estudo: as linguagens de descrição de cena baseadas em XML. O fundamento para o foco do estudo nas linguagens de descrição de cena baseadas em XML é induzido pelas características que herdam desta linguagem base, nomeadamente residente no facto do XML:

– ser uma linguagem de norma não proprietária
– ser independente dos vários tipos de plataformas e sistemas, permitindo um alto grau de interoperabilidade;
– oferecer capacidade descritiva adequada para ferramentas de geração de descrições automáticas ou para uma edição simples em modo texto com linguagem perceptível para o ser humano;

– permitir uma descrição estruturada de meta-informação para selecção, avaliação, e integração de objectos;
– ser apropriada para hierarquias de elementos que representam vários tipos de objectos e conceitos adequados a sistemas gráficos e cenas;

Nesta esfera podem ser encontradas várias linguagens de cena, mas apenas vão ser referenciadas neste estudo aquelas que, por um lado, possuem um uso mais corrente e são aplicadas num número considerável de aplicações com um grau de desenvolvimento e maturidade bem definido e, por outro lado, que se enquadram sob os aspectos gerais nesta aplicação de monitorização em desenvolvimento. Cada uma das linguagens analisadas incorpora um conjunto de mecanismos próprios, que resultam na obtenção de diferentes resultados, no âmbito da aplicação específica descrita na Fig. 1. Nesta fase torna-se imperativo descrever de uma forma minimalista algumas linguagens de descrição de cena que, pelo estudo dos pré-requisitos efectuado, se apresentam como as soluções mais adequadas ao sistema, realçando as suas características individuais relevantes ao contexto. Segundo este propósito, seleccionaram-se as seguintes linguagens:

– XMT-A[5]
– XMT-O[6]
– SMIL[7]

Segue-se uma descrição breve destas linguagens e suas características relevantes para o estudo. Nos tópicos seguintes a designação SMIL depreende a referência à versão 2.0, ou seja, SMIL 2.0.

3.1 Linguagens desenvolvidas no contexto da norma MPEG

A norma Moving Picture Experts Group (MPEG)[8] define uma estrutura onde é possível transmitir vários tipos de objectos numa plataforma comum. Esta norma divide-se em duas partes fundamentais: a codificação dos objectos intervenientes e a possibilidade de definição da apresentação dos objectos pertencentes à cena multimédia. A apresentação, dos objectos que compõem a cena audiovisual, é especificada através de formatos próprios para descrição das cenas. O fluxo que passa este tipo de interacções e que contém as informações das relações entre os objectos multimédia, é chamado fluxo descritor de cenas e é especificado através da linguagem BInary Format for Scene (BIFS)[9]. O BIFS foi inspirado na linguagem VRML[10], e possui características próprias, adequadas para o sistema de transporte MPEG-4. A linguagem BIFS permite a eficiente representação de apresentações dinâmicas e interactivas, incluindo gráficos 2D & 3D, imagens, texto e material audiovisual. Esta representação inclui a descrição de organização no espaço e no tempo dos diferentes elementos de cena, animações e interacções com o utilizador. Ao contrário do VRML, onde os seus objectos e eventos são declarados em modo texto, em BIFS esta estrutura é especificada em código binário. A adopção do formato binário é fundamental para a compressão e posterior distribuição das cenas no MPEG-4, no entanto esse formato dificulta o processo de criação e alteração do conteúdo, embora este facto possa ser contornado com a utilização de ferramentas de edição gráfica, onde o formato binário já se torna transparente para os editores destes conteúdos.

Com intuito de ultrapassar a limitação do formato binário na criação e alteração de conteúdos na linguagem BIFS, o formato eXtensible MPEG-4 Textual (XMT)[11] foi proposto como uma estrutura para especificar descrições de cenas MPEG-4 através de uma sintaxe textual. O XMT é uma extensão textual usada na norma MPEG-4 e tenta combinar vantagens da representação binária BIFS e da representação de alto nível de abstracção de cenas multimédia, SMIL. Sendo assim, o XMT apresenta características novas no contexto MPEG-4 como:

– um intercâmbio transparente entre editores e ferramentas de produção
– interoperabilidade com formatos. O formato XMT pode ser convertido em SMIL, VRML ou em BIFS (para os leitores MPEG-4)
– possibilidade de edição manual do conteúdo

Esta estrutura é instanciada por duas linguagens, com diferentes sintaxes e semânticas XMT-A e XMT-O apresentadas de seguida.

Linguagem XMT-A. A linguagem XMT-A permite a descrição de baixo nível de conteúdos multimédia baseada em Extensible 3D (X3D)[12] e é uma versão em formato texto da linguagem BIFS. A sua estrutura baseia-se em construções XML com representação directa das expressões existentes no formato binário do MPEG-4, e por representar construções existente no BIFS, o XMT-A também se aproxima do tipo de estruturas definidas em VRML. Também no aspecto semântico, esta linguagem apresenta uma equivalência estrita com o BIFS, não acrescentando nenhuma outra funcionalidade adicional. Assim sendo, apresenta o mesmo conjunto de características centrais já oferecidas pelo formato binário, que se apresentam enumeradas desta forma:

– Permite a integração e controlo de conteúdos áudio/vídeo diferentes na mesma cena.
– Tem um conjunto rico de construtores gráficos 2D/3D.
– Possui um conjunto de elementos que permitem a interactividade local e remota.
– Permite animações locais e remotas como por exemplo a alteração de posições de objectos, de cores, e formas.
– Permite a reutilização de conteúdos através de referências para fluxos.

De seguida é apresentado um exemplo de uma descrição de cena no formato XMT-A.

```
<Shape>
    <geometry>
      <Rectangle size="40 30"/>
    </geometry>
    <appearance>
      <Appearance>
        <texture>
          <MovieTexture url="http://site.pt/obj.mpg" loop="false"
          startTime="0.0" stopTime="-1.0"/>
        </texture>
      </Appearance>
    </appearance>
</Shape>
```

Como se pode observar, existe um alto grau de verbosidade para o tipo de exemplo simples que se quer descrever.

Linguagem XMT-O. Com o intuito de diminuir a complexidade existente na linguagem XMT-A na especificação de cenas MPEG-4, foi desenvolvida a linguagem XMT-O. Nesta linguagem, os objectos audiovisuais e as suas transformações são descritas num alto nível de abstracção, facilitando o processo de edição, ao contrário do que ocorre em XMT-A. A linguagem XMT-O tende a uma aproximação forte à linguagem SMIL e por isso apresenta algumas das suas características próprias, conseguindo assim uma interoperabilidade entre estes dois sistemas. Isto é conseguido pelo XMT-O recorrendo a alguns módulos presentes na linguagem SMIL. A estrutura do documento XMT-O também é dividida em duas partes, um cabeçalho e um corpo. No cabeçalho encontram-se as informações gerais do documento, como a forma da apresentação, a meta-informação para definição semântica do documento e as referências para conjuntos de elementos. De seguida é apresentado um exemplo de um cabeçalho no formato XMT-O.

```
<head>
    <layout metrics="pixel" type="xmt/xmt-basic-layout">
      <topLayout width="300" height="300" backgroundColor="branco">
        <region id="regiao_video" size="91 27"/>
      </topLayout>
    </layout>
</head>
```

Em XMT-O, de forma análoga a SMIL, as composições possuem semântica de sincronização e é no corpo do documento XMT-O que se descreve a semântica temporal e sequencial da cena, onde várias composições podem ser definidas, através dos elementos *par*, *seq* e *excl* (tipo de semântica temporal implementada - paralela, sequencial ou exclusiva). Ainda no corpo do documento, as características para apresentação dos objectos multimédia são especificadas nos próprios elementos, através dos seus atributos. Além das características para apresentação, as relações entre esses objectos também podem estar incluídas nos seus atributos. Dessa forma, além de composições com semântica de sincronização, estas relações podem ser estabelecidas através de eventos, especificados nesses atributos. De seguida é apresentado um corpo associado ao cabeçalho acima descrito. Neste exemplo, estes dois elementos formam um documento que descreve uma cena em formato XMT-O.

```
<body><par>
       <video src="video.mp4" region="regiao_video" begin="0s"
       dur="indefinite"/>
       <audio src="audio.mp4" begin="0s" dur="indefinite"/>
</par></body>
```

3.2 SMIL

A linguagem SMIL apresenta-se como uma boa candidata neste estudo por ser uma linguagem normalizada pelo W3C[13] e por ser desenvolvida desde início como uma linguagem para apresentações multimédia. De modo geral, o SMIL permite a gestão de apresentações de audiovisuais interactivas que integram fluxos de áudio, imagens e texto, ou qualquer outro tipo de objecto multimédia. Para isso, possui funções para a definição de sequências, duração, posição e visibilidade dos vários elementos

(objectos) constituintes da cena multimédia, oferecendo a possibilidade de descrever o comportamento temporal da apresentação e descrever a localização espacial dos objectos no ecrã.
A segunda versão desta linguagem (SMIL 2.0) veio acrescentar dados novos importantes neste contexto, nomeadamente a introdução de conceitos como modularização e perfis, que permitem uma expansão da linguagem de um modo relevante. Assim, existem módulos que podem ser utilizados com elementos, atributos e valores de atributos adequados à descrição de uma aplicação particular. Existem vários módulos disponíveis neste contexto, como por exemplo, o módulo de ligação, o módulo de controlo de conteúdo e o módulo de apresentação. Além disso, é possível nesta linguagem seleccionar um perfil que descreve o conjunto de módulos que estão a ser utilizados num determinado documento, com a função de compatibilizar esta descrição com a versão anterior (SMIL 1.0) e assegurar a compatibilidade com a semântica e conteúdos SMIL entre os vários módulos.
No código seguinte, é apresentado um exemplo de descrição de cena em SMIL, com a possibilidade de interacção dos conteúdos .

```
<smil xmlns="http://www.w3.org/2005/SMIL20/Language">
  <head>
    <layout>
      <region id="geral" width="30" height="50"/>
      <region id="especifica" width="300" height="500"/>
    </layout>
 </head>
 <body><par>
     <a href="video1.avi" target="especifica" accesskey="a">
     <video region="geral" src="video1.avi" begin="0s"
     dur="indefinite"/></a>
     <a href="video5.avi" target="especifica" accesskey="a">
     <video region="geral" src="video5.avi" begin="0s"
     dur="indefinite"/></a>
   </par></body></smil>
```

De seguida são apresentados os critérios de análise.

4 Critérios de Análise

Antes da análise comparativa destas tecnologias, torna-se necessário distinguir o conjunto de critérios que se revelaram preponderantes no desenvolvimento da aplicação específica de monitorização. A recolha destes critérios de comparação e análise é de grande importância para a decisão final na escolha da linguagem, uma vez que já se mostrou que a linguagem de descrição de cena é uma componente essencial na aplicação em foco. Para a aplicação de visualização e monitorização em desenvolvimento, são enumerados os seguintes critérios:

1. Conjunto de elementos adequados: Dando lugar à monitorização existirão elementos a monitorizar num determinado ambiente. Na linguagem de descrição terá de existir um conjunto de elementos, atributos e valores associados adequados à descrição e manuseamento dos objectos na cena. Se estes dados não estiverem definidos, terão de ser definidos fora da norma das linguagens.

2. Semântica própria para representar alterações temporais: É necessário que a linguagem possua formas de descrição de alterações temporais na monitorização, assim sendo, é valorizado o modo como estas alterações são representadas para, da melhor forma, se traduzir todas as sequências temporais de monitorização possíveis neste tipo de aplicações. Seria conveniente que esta sequência fosse lógica e simples para melhor compreensão de autores e editores da aplicação referida.
3. Semântica adequada para representar alterações espaciais: O modo como são expressas as transformações na cena também será um critério relevante neste contexto. Torna-se importante a existência de formas básicas para a transformação espacial dos elementos, nomeadamente na sua posição e tamanho, e uma semântica própria também na representação fácil e intuitiva destas alterações.
4. Interpretação na edição de conteúdos: É importante para o autor/editor dos conteúdos ter uma noção da representação da cena apenas por inspecção do ficheiro que a compõem. À partida as descrições terão de ser exprimidas em modo texto para possibilitarem a fácil edição e deverão ter um grau de complexidade baixo para, dessa forma, serem intuitivas e facilmente entendidas na interacção humana.
5. Baixo volume de informação: Para existir facilidade no envio de informação e baixo tempo de atraso entre alterações à monitorização, um dos critérios que podem ser definidos passa pela baixa complexidade e tamanho do ficheiro que contém a linguagem da descrição de cena. Na aplicação em causa exige-se uma transferência semi-contínua de conteúdos que pode ter frequências bastante altas, o que coloca este factor como relevante neste ponto.
6. Escalabilidade e Modularização: Como último critério e não menos importante, convém referir que é intenção no desenvolvimento desta aplicação, utilizar linguagens que possam ser aplicadas no contexto referido e que possam ser extensíveis, permitindo algumas modificações, adaptações ou a utilização de incrementos modulares adequados.

Definidos os critérios que serão avaliados, podemos definir também uma escala de valores que quantifique, de forma objectiva, as linguagens em análise. Sendo assim teremos valores de 1 a 3 que devem corresponder à seguinte interpretação:

1. Linguagem que não satisfaz o critério estabelecido.
2. Linguagem que satisfaz o critério estabelecido com necessidade de algumas alterações adicionais.
3. Linguagem que satisfaz o critério estabelecido sem necessidade de alterações adicionais.

Seguidamente mostra-se os resultados da análise efectuada.

5 Resultado da Análise

Ao realizar este estudo, em primeira instância pôde-se verificar que o XML está a influenciar de várias formas o formato MPEG, formato importante na esfera das aplicações multimédia. Neste momento, tanto o formato BIFS (norma MPEG-4) como o formato BiM[14] (norma MPEG7) estão a ser complementados com esquemas de representação XML levando o MPEG a utilizar conceitos de algumas especificações como SMIL e X3D.

Numa análise mais pormenorizada, pode-se referir que a linguagem de descrição de cena BIFS numa fase anterior à conversão e compressão apresenta um conjunto vasto de elementos de desenho facial e de figuras geométricas, que não são necessários numa aplicação de monitorização no contexto apresentado. Esta linguagem possui também um conjunto variado de funções construtoras de gráficos 2D/3D, que são, no âmbito da análise, igualmente dispensáveis. Mas convém referir nesta altura que o principal factor que leva a rejeitar esta opção como válida no desenvolvimento da aplicação referida, está relacionada com o formato nativo da linguagem. Uma vez que a linguagem BIFS é convertida em formato binário, podemos verificar que não permite a edição ou reedição directa dos conteúdos.

O MPEG redireccionou o formato desta linguagem para outro formato próprio de XML - o XMT que, na norma XMT-A apresenta a mesma forma da versão BIFS mas em formato texto, já permitindo ao autor/editor uma criação e alteração de conteúdos mais directa. Apesar desta vantagem note-se que continua a existir um conjunto de elementos não adequados, herdados das construções geométricas do BIFS. Na realidade, por representar directamente todos os elementos presentes na arquitectura MPEG-4, o XMT-A continua a apresentar-se como uma linguagem algo complexa e extensa para a edição fácil na criação e alteração de conteúdos.

Sendo uma linguagem de menor complexidade, o XMT-O apresenta-se como uma boa solução em relação ao XMT-A. Esta norma é baseada em SMIL e por isso apresenta um conjunto mais adequado de elementos. De qualquer forma, este formato apresenta alguma compatibilidade com o formato de representação da cena em árvore como o BIFS e o XMT-A e, com isto, algumas das características do SMIL são implementadas de forma parcial, como por exemplo na sincronização, nas transições e controlo de conteúdos ou até não implementadas como no caso do tratamento de eventos de entrada (por ex. teclado).

O SMIL apresenta-se como uma linguagem para descrição de cenas de princípio simples e bastante eficiente. É de referir que esta é a linguagem que apresenta os elementos mais adequados ao sistema em desenvolvimento, principalmente porque implementa uma modularização abrangente e, baseado neste conceito, podem ser utilizados alguns módulos orientados para este tipo específico de aplicação em desenvolvimento. Para este efeito os seguintes módulos demonstraram boa adequabilidade:

- Módulo de Ligação
- Módulo de Controlo de Conteúdo
- Módulo de Apresentação

Após o estabelecimento dos critérios de análise e após esta análise particular da adequabilidade destas linguagens, irá ser definido o grau de adequabilidade e a medida da integração destas linguagens nestes sistemas. Ao providenciar esse conjunto possível de critérios e ao cruzar esta informação com as características inerentes às linguagens em análise, é apontada uma solução que será a mais adequada no contexto explicado. Na Tabela 1 é apresentado o conjunto de linguagens em análise e a sua correspondente adequabilidade aos critérios formulados utilizando a escala de valores estabelecida na secção 4.

Tabela 1. Tabela com análise comparativa da adequabilidade das linguagens submetidas

Linguagens	Critério 1	Critério 2	Critério 3	Critério 4	Critério 5	Critério 6
XMT-A	**1**	**2**	**3**	**1**	**3**	**1**
XMT-O	**2**	**3**	**3**	**3**	**3**	**3**
SMIL 2.0	**3**	**3**	**3**	**3**	**3**	**3**

Desta tabela sobressai que as linguagens XMT-O e o SMIL se apresentam como boas candidatas à aplicação em desenvolvimento. Note-se a diferença existente entre o formato XMT-A e XMT-O baseada em primeira instância no critério de adequação de elementos (critério 1), onde se verifica que o XMT-O, oferece um conjunto de elementos mais adequados do que o XMT-A. Em segunda instância também se verificam diferenças notórias na facilidade de interpretação na edição de conteúdos e consequente complexidade (critério 4), onde o XMT-A revelou ser mais complexo do que o formato XMT-O. Quanto ao critério 6 podemos verificar que existe uma discrepância entre o XMT-A e os outros dois formatos. Isto deve-se à estrita ligação do XMT-A com o formato binário, não permitindo qualquer modularização ou capacidade de adaptação por parte deste formato à aplicação em desenvolvimento. Sendo assim, o XMT-O e o SMIL são apresentados como formatos válidos, embora com algumas pequenas diferenças entre eles. Ambos diferem no conjunto de elementos que disponibilizam (critério 1), sendo o SMIL o formato mais adequado, uma vez que, para além dos elementos principais (objectos multimédia) e seus atributos, este permite um conjunto vasto de possibilidades de interacção com o conteúdo e transições adequadas ao contexto em causa. Sob outra perspectiva, no critério 6 também podemos observar algumas diferenças. Neste critério, o SMIL apresenta-se com um conjunto vasto de módulos que permitem adaptações a vários tipos de sistemas. Alguns destes revelaram ser bastante úteis para a aplicação, nomeadamente os módulos de ligação e apresentação. Apesar de alguns destes conceitos também estarem presentes no XMT-O, o SMIL apresenta uma maior adequabilidade à aplicação a desenvolver. Em suma, o SMIL possuí vantagens adicionais em relação ao XMT-O, que passam pela existência de elementos primários adequados (com os seus atributos respectivos) e pela capacidade de modularização disponível, oferecendo um conjunto adicional de elementos que permitem descrever melhor a cena da aplicação em causa e permitindo uma futura escalabilidade do sistema.

6 Conclusões

A decisão na escolha da linguagem da descrição de cena é determinante para o projecto que está a ser desenvolvido e teve como base as conclusões do estudo evidenciado neste documento. Tal como é indicado na secção do resultado da análise, a solução que se demonstrou mais adequada aos critérios exigidos foi a linguagem SMIL. Tendo por base este resultado, no âmbito referido, já foram formulados e testados alguns exemplos de possíveis descrições de cena que poderão existir. O seguinte código apresenta um exemplo simples de uma descrição de cena, onde os objectos centrais da cena são dois objectos vídeo que representam o fluxo das imagens do ambiente de trabalho de cada computador a monitorizar.

```
<smil>
    <head>
      <layout>
        <root-layout width="320" height="240"/>
        <region id="janela1" left="170" top="110"/>
        <region id="janela2" left="50" top="50"/>
      </layout>
    </head>
    <body>
      <par>
        <video src="comp1.avi" begin="5s" region="janela1"/>
        <video src="comp5.avi" begin="2s" region="janela2"/>
      </par>
    </body>
</smil>
```

Neste formato linguagem SMIL, depois de definidos os objectos e seus atributos, poderemos ter ainda interacções entre eles e transformações espaciais e temporais adequadas. Estas transformações, poderão ser definidas no futuro para traduzir níveis de importância de monitorização, de acordo com critérios definidos antes do início da monitorização, ou ao longo do processo de monitorização, de acordo com valores de variáveis que se pretendam monitorar.

Referências

1. "Remote Frame Buffer Protocol", acedida pela última vez em 17 de Novembro de 2005, http://www.realvnc.com/docs/rfbproto.pdf
2. "Software RealVNC", acedida pela última vez em 17 de Novembro de 2005, http://www.realvnc.com/
3. "Windows Management Interface Reference", acedida pela última vez em 17 de Novembro de 2005, http://msdn.microsoft.com/library/en-us/wmisdk/wmi/wmi_reference.asp
4. "XSD", acedida pela última vez em 17 de Novembro de 2005, http://www.w3.org/TR/xmlschema-0/

5. "MPEG-4, eXtensible MPEG-4 Textual Format - XMT-A", acedida pela última vez em 17 de Novembro de 2005, http://www.digitalpreservation.gov/formats/fdd/fdd000156.shtml
6. "MPEG-4, eXtensible MPEG-4 Textual Format - XMT-O" , acedida pela última vez em 17 de Novembro de 2005, http://www.digitalpreservation.gov/formats/fdd/fdd000156.shtml
7. "SMIL - *Synchronized Multimedia Integration Language*", acedida pela última vez em 17 de Novembro de 2005, http://doc.async.com.br/w3-recs/REC-smil/introduction.html
8. "Moving Picture Experts Group", acedida pela última vez em 17 de Novembro de 2005, http://www.mpeg.org/MPEG/index.html
9. "BIFS - BInary Format for Scenes", acedida pela última vez em 17 de Novembro de 2005, http://www.chiariglione.org/mpeg/tutorials/papers/icj-mpeg4-si/05-BIFS_paper/5-BIFS_paper.htm
10. "VRML - Virtual Reality Modeling Language", acedida pela última vez em 17 de Novembro de 2005, http://www.w3.org/MarkUp/VRML/
11. "MPEG-4, eXtensible MPEG-4 Textual Format (XMT)", acedida pela última vez em 17 de Novembro de 2005, http://www.digitalpreservation.gov/formats/fdd/fdd000156.shtml
12. "X3D", Extensible 3D: XML meets VRML, Len Bullard, http://www.xml.com/pub/a/2003/08/06/x3d.html
13. "W3C - The World Wide Web Consortium", acedida pela última vez em 17 de Novembro de 2005, http://www.w3.org/
14. "Les formats du multimédia et des hypermédias", acedida pela última vez em 17 de Novembro de 2005, http://www.enseeiht.fr/~dayre/hypermedias/Formats.html
15. ISO/IEC International Organisation for Standardisation. 14496-11:2004. Coding of Audio Visual Objects – Part 11: Scene description and application engine/Amd 4. XMT & MPEG-J extensions. 2004.

Acesso Interoperável a Informação Geográfica para Disponibilização de Modelos Urbanos 3D em Dispositivos Móveis

Manuel Mesquita T. F. Gomes[1], Artur Jorge da Silva Rocha[1], António Fernando Coelho[2], A. Augusto Sousa[1,3]

[1] INESC-Porto, Rua Dr. Roberto Frias, n.378, 4200-465 Porto, Portugal,
[2] Universidade de Trás-os-Montes e Alto Douro, Quinta de Prados, 5000-801 Vila Real, Portugal,
[3] Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal.
mmgomes@inescporto.pt, artur.rocha@inescporto.pt, acoelho@utad.pt, aas@fe.up.pt

Resumo A evolução tecnológica nas plataformas móveis, nomeadamente no que diz respeito ao aumento da largura de banda das comunicações e da maior capacidade de processamento, assim como a integração de equipamentos GPS, sugerem novas aplicações baseadas na disponibilização de modelos tridimensionais localizados de ambientes urbanos que proporcionem ao utilizador um ambiente virtual georeferenciado com utilização nos domínios da cultura, turismo, manutenção de infraestruturas, planeamento urbano, entre outros.

Este artigo descreve a abordagem utilizada para o desenvolvimento de uma solução de modelação automática e distribuição de modelos de ambientes urbanos virtuais extensos através de redes de comunicação sem fios para dispositivos móveis. É dado especial ênfase ao acesso interoperável aos dados pelo seu subsistema de modelação que, utilizando processos automáticos para a criação de ambientes urbanos virtuais extensos, permite uma prototipagem rápida mas com a possibilidade de refinamento progressivo por forma a atingir o maior nível possível de realismo.

O sistema acede de forma interoperável à informação geográfica proveniente de diversas fontes, utilizando as normas estabelecidas pelo consórcio *OpenGeospatial* (OGC) [14] para a interoperabilidade entre serviços geoespaciais, nomeadamente *Geographic Markup Language* (GML) [16] e *Web Feature Service* (WFS) [18], gerando os modelos virtuais urbanos a disponibilizar, no formato X3D (*Extensible 3D*).

Palavras-chave: Sistemas de Informação Geográfica, OpenGIS, OSF, OWS, WFS, GML, Computação Gráfica, Realidade Virtual, Modelação 3D, Sistemas-L, X3D.

1 Introdução

A evolução da tecnologia das plataformas móveis, como PDAs e telemóveis, com cada vez maiores capacidades de processamento, o aumento da largura de banda disponível

para as comunicações, permitindo a transmissão de maior quantidade de informação, e a cada vez maior precisão e integração de equipamentos GPS neste tipo de dispositivos, tem motivado o aparecimento cada vez maior de aplicações cientes da localização do utilizador e do seu contexto geográfico, como serviços de orientação e navegação.

Embora estes serviços utilizem principalmente mapas interactivos como forma de visualização, derivados dos mapas utilizados durante séculos em papel, existem já modelos tridimensionais para sistemas de navegação automóvel que sugerem que a utilização destes modelos para navegação e exploração de ambientes complexos, como os urbanos, pode ser mais eficiente e compreensível[5].

Nesse e outros contextos, os ambientes urbanos virtuais são um conteúdo relevante. Por exemplo, a geração de modelos virtuais de cidades antigas com base nos dados obtidos pela arqueologia possibilita "viajar no tempo", visitando o passado. Os modelos virtuais do Mosteiro de Santa Clara-a-Velha [26] e de Bracara Augusta [1] são exemplos desse tipo de projectos desenvolvidos em Portugal. A simulação de projectos de urbanização e a avaliação do seu impacto, com recurso a modelos virtuais que recriem ambientes urbanos existentes, evitam alterações dispendiosas realizadas em fases posteriores, e potenciam a optimização da gestão de recursos do ambiente urbano. Neste contexto, salientam-se os casos de Lisboa Virtual [21], de Angra do Heroísmo [7] e do Funchal [8] e, num contexto mais alargado, encontram-se vários bons exemplos, como se pode ver no directório de cidades virtuais [13].

A simples visualização destes ambientes virtuais, tal como sucede nos exemplos anteriores, embora atractiva, não explora todas as suas potencialidades. A sua utilização como ferramenta para a disponibilização de informação georreferenciada de peças do edificado ou mesmo de outro mobiliário urbano pode ser de grande importância, quer para o grande público, quer mesmo para serviços especializados, ainda mais se disponibilizados em plataformas móveis.

O reconhecimento do uso da Internet como meio de difusão de informação geográfica e centro do desenvolvimento dos sistemas de informação geográfica (GIS) pelos principais fornecedores deste tipo de tecnologia[20], tem levado ao aumento de fontes deste tipo de informação disponíveis publicamente.

Tal situação traduz-se também num grande potencial para a modelação tridimensional de ambientes urbanos que, apesar de tudo, requer ainda uma grande intervenção humana. Esta condicionante resulta de problemas relacionados com a utilização de diferentes fontes de dados proprietárias, da redundância geográfica causada pela diversidade de temas sobre a mesma informação geográfica de base e de divergências temporais na recolha dos dados.

Para ultrapassar estes problemas e permitir assim a modelação automática destes ambientes, é necessário encontrar novas formas de interoperabilidade entre sistemas de informação geográfica diferentes, que ofereçam meios de uniformizar a informação neles contida.

No capítulo seguinte deste artigo é apresentado o estado da arte referente à interoperabilidade no domínio dos sistemas de informação geográfica. No capítulo 3 é apresentada a solução proposta para um sistema capaz de modelar ambientes urbanos virtuais, de uma forma automática a partir do acesso interoperável a diferentes fontes de informação. No capítulo 4 descreve-se a implementação da solução apresentada e no

capítulo 5 os testes efectuados e respectivos resultados. Para finalizar, no capítulo 6, são apresentadas conlusões e aponta-se o caminho a seguir no desenvolvimento do conceito descrito neste artigo.

2 Interoperabilidade em GIS

Uma das grandes limitações das aplicações GIS proprietárias centradas na Internet é a adopção, por cada fornecedor, da sua própria arquitectura, estrutura de base de dados e formatos. A informação e capacidades de processamento da informação que cada uma destas soluções fornece não podem ser partilhadas pelas outras, tornando a interoperabilidade entre estes sistemas uma tarefa difícil.

Em 1994 foi constituído consórcio *OpenGeospatial* (OGC)[14] com o objectivo de estudar e especificar meios de interoperabilidade entre sistemas GIS na Internet. Daí têm resultado especificações várias que normalizam, entre outros, as operações de pesquisa e de consulta de mapas e dados geográficos de diferentes fontes e fornecedores.

Em Abril de 2000, na sequência da iniciativa *Web Mapping Testbed, Phase 1* (WMT-1) [28], foi introduzida a primeira versão da especificação de implementação do *Web Map Server* (WMS) [15], que constituiu o primeiro passo no desenvolvimento de componentes da *Spatial Data Infrastructure* (SDI). Seguiram-se desenvolvimentos-piloto deste conceito, tais como o *North Rhein Westphalia Pilot Project* (NRWPP) [22], que deu origem ao actual *Geodata Infrastructure North-Rine Westphalia* (GDI NRW) [29]. Neste e noutros exemplos, como o *Infraestructura de Datos Espaciales de España* (IDEE) [6], utilizam-se WMS para disponibilizar informação geográfica de forma interoperável. Outras especificações da OGC, como *Web Feature Service* (WFS), *Web Coverage Service* (WCS), *Catalogue Service for Web* (CS-W) e *Geography Markup Language* (GML), são também actualmente utilizadas para implementar SDIs globalmente interoperáveis.

Dos resultados obtidos no WMT-1, acabou também por surgir uma *framework*, resumidamente representada na figura 1, baseada em serviços *Web*, denominados de *OpenGIS Web Services* (OWS). Estes serviços são os componentes individuais das aplicações geográficas dinâmicas e estão na base do paradigma *Spatial Web* para a elaboração de soluções de problemas geoespaciais.

Este paradigma impõe restrições conceptuais e de implementação aos OWS, que incluem orientação do serviço, capacidade de distribuição, descrição própria, operações *stateless*, uso de codificações comuns de XML, transporte por HTTP, sintaxe de interface definida e modelos de informação específicos para a descrição dos serviços e de outra metainformação[11].

Os serviços da *OpenGIS Web Services Framework* (OSF) que disponibilizam a informação geográfica aos utilizadores através da Internet são os *Portrayal Services* e os *Data Services*[11]. Os primeiros, cujo principal serviço é o *Web Map Service* (WMS)[17], fornecem mapas que são a representação visual e bidimensional da informação em formatos *raster*, como PNG, GIF ou JPEG. Os *Data Services*, cujo principal serviço é o *Web Feature Service* (WFS)[18], fornecem a informação propriamente dita, geralmente sob a forma de *Geography Markup Language* (GML)[16].

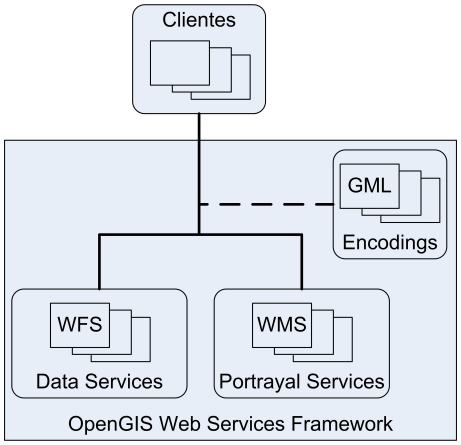


Figura 1. Representação da OpenGIS Web Services Framework (OSF)

O WFS é a especificação desenvolvida pelo OGC para as operações de manipulação e consulta de informação geográfica, permitindo que os utilizadores obtenham dados em GML a partir de diversos tipos de fontes de dados, tais como formatos específicos de fornecedores suportados pelo WFS e até de outros WFS.

O WFS tem como requisitos principais ser capaz de, no mínimo, fornecer informação geográfica utilizando GML e possuir interfaces definidas em XML. Os pedidos e respectivos filtros são definidos em XML e derivados de CQL, conforme definido na especificação do OpenGIS Catalog Interface[19]. As fontes de dados utilizadas para guardar a informação geográfica devem ser opacas às aplicações cliente, cuja única visão da informação será através da interface WFS que utiliza um subconjunto de *XPath* para a referenciação de propriedades.

O processamento de consulta e a manipulação da informação geográfica num WFS são suportados através das seguintes operações: *GetCapabilities*, que permite descrever as capacidades do WFS indicando que tipo de informações pode fornecer a um cliente e as operações suportadas por cada tipo; *DescribeFeatureType*, que permite descrever, a pedido, a estrutura de cada tipo de informação geográfica que pode servir a um cliente; *GetFeature*, que permite fornecer a informação geográfica pedida a um cliente, devendo ser possível a este especificar o tipo de informação que pretende, bem como limitá-la, quer geograficamente, quer não geograficamente; *Transaction*, que permita a um WFS ser capaz de suportar pedidos de transacção, que é composta por pedidos de edição de informação, isto é, por operações que insiram, actualizem ou eliminem informação geográfica; e *LockFeature*, que permite a um WFS ser capaz de processar um pedido de *lock* a um ou mais tipos de informação geográfica durante uma transacção, permitindo o suporte de transacções serializáveis.

Baseado nestas operações, podem definir-se duas classes de WFS: *Basic WFS*, que implementa as operações *GetCapabilities*, *DescribeFeatureType* e *GetFeature* e que se considera apenas de leitura, servindo para consulta de informação geográfica; *Transaction WFS*, que suporta todas as operações de um *Basic WFS*, a operação *Transaction*

e, opcionalmente, a operação *LockFeature*, permitindo, além de consultar informação geográfica, a sua modificação e inserção de nova informação.

Os pedidos mais usuais neste tipo de serviço são o *GetCapabilities* e o *GetFeature*. O primeiro é geralmente efectuado através do método GET do HTTP enquanto o segundo, e dependendo da implementação do WFS, pode ser efectuado quer através do método GET, quer através do método POST.

Actualmente existem diversas implementações da especificação WFS e, em regime de *open source*, destacam-se duas, implementadas em Java: GeoServer[9] e Deegree[23]. Estas apresentam semelhanças ao nível da funcionalidade sendo, no entanto, o GeoServer mais fácil de configurar, através de uma interface *Web*. Joga também a favor do GeoServer o facto de ser a implementação de referência da especificação WFS do OGC[10].

GML[16] é uma codificação XML para o transporte e armazenamento de informação geográfica, incluindo características espaciais e não espaciais e cuja especificação define mecanismos, convenções e uma sintaxe em XML Schema que estabelecem uma *framework* aberta para a definição de esquemas e objectos para aplicações geoespaciais. Além disso, suporta a descrição de esquemas de aplicações geoespaciais para domínios especializados e comunidades de informação, permitindo a criação, manutenção, armazenamento e transporte de esquemas de aplicações geográficas e conjuntos de dados, o que aumenta a capacidade de as organizações partilharem esquemas de aplicações geográficas e a informação que estes descrevem.

A linguagem GML foi desenvolvida pelo OGC para a troca de informação entre os diferentes sistemas e modela o mundo em termos de *features*[1]. A GML foi desenvolvida com um conjunto de objectivos em mente, alguns dos quais de sobrepõem aos próprios objectivos do XML: Fornecer um meio de codificar informação espacial, quer para o armazenamento, quer para o transporte, especialmente num contexto alargado como é a Internet; ser suficientemente expansivo para suportar uma grande variedade de tarefas espaciais, desde a representação até à análise; estabelecer a base da Internet GIS de uma maneira incremental e modular; permitir a criação e manutenção de esquemas de aplicações geográficas e conjuntos de dados; permitir a codificação eficiente de geometria geoespacial; fornecer codificações de informação e relações espaciais fáceis de entender, incluindo as definidas no modelo Simple Features do OGC; conseguir a separação do conteúdo (espacial e não espacial) da apresentação da informação; permitir a integração fácil de informação espacial e não espacial, principalmente nos casos em que esta é expressa em XML; efectuar a ligação de elementos espaciais (geométricos) a outros elementos espaciais ou não espaciais; e fornecer um conjunto comum de objectos de modelação geográfica de forma a permitir a interoperabilidade entre aplicações desenvolvidas independentemente.

A GML suporta a interoperabilidade entre sistemas diferentes através do fornecimento de um conjunto básico de *tags* geométricas, um modelo de dados comum (*features* e/ou propriedades) e mecanismos de criação e partilha de esquemas de aplicação.

[1] *features* são abstracções de fenómenos do mundo real e que são geograficamente associadas a um local da Terra

3 Solução Proposta

O grande potencial de informação tridimensional como via de comunicação e a evolução dos dispositivos móveis sugerem o desenvolvimento de novos serviços de disponibilização de ambientes urbanos virtuais neste tipo de tecnologia usando, para o efeito, a variada informação geográfica que vai sendo publicamente disponível. No entanto, a falta de interoperabilidade entre os sistemas GIS proprietários, bem como as diferenças entre a informação geográfica existente, obriga a grande intervenção humana para a modelação destes ambientes.

Por forma a resolver estes problemas de interoperabilidade e a permitir a modelação automática de ambientes urbanos foi definido o sistema, representado na figura 2. Este sistema é baseado numa arquitectura distribuída de serviços *Web* potenciando a interoperabilidade entre os diversos módulos e aplicações externas e constituído por dois importantes subsistemas: o de modelação e o de distribuição.

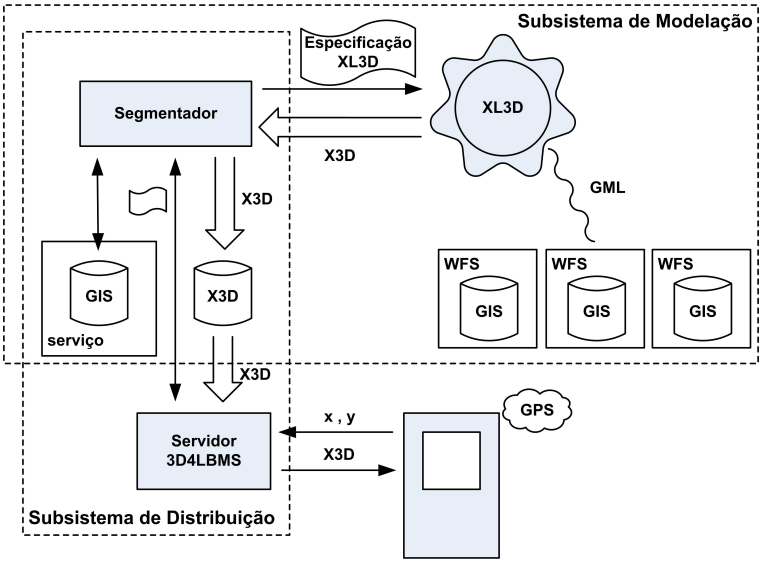


Figura 2. Arquitectura 3D4LBMS

3.1 Subsistema de Modelação

O subsistema de modelação é o responsável pela geração dos modelos tridimensionais de ambientes urbanos, tendo como núcleo principal o modelador XL3D [5]. Este modelador reage em função de uma especificação de modelação, gerando modelos de cenas urbanas no formato X3D[25] e armazenando-os numa base de dados de modelos. Esta é acedida pelo subsistema de distribuição e envia os modelos 3D para os dispositivos móveis[3].

O modelador XL3D utiliza processos automáticos para uma prototipagem rápida do modelo tridimensional, mas dando a possibilidade de efectuar refinamentos posteriores, de forma a atingir-se o maior nível possível de realismo.

3.2 O Papel da Interoperabilidade na Modelação

Para a geração automática de ambientes urbanos virtuais, o subsistema de modelação necessita de informação geográfica da zona a modelar. Essa informação é obtida em diversos formatos e é proveniente de diversas fontes, segundo um modelo interoperável, utilizando principalmente as especifições GML[16] e WFS[18] da OGC como meio de homogenização da informação.

A utilização de servidores WFS permite que a informação se encontre em qualquer dos formatos suportados pela implementação utilizada, sendo transformada, em tempo real, para GML. Sendo a GML uma linguagem derivada da XML, a informação disponibilizada pelo servidor WFS pode ser transformada pelo modelador, através de folhas de estilo XSLT, em cadeias modulares paramétricas. Estas são então concatenadas para compor a cadeia inicial de dados a utilizar no processo de modelação, definido e controlado por um sistema L paramétrico[4].

3.3 Subsistema de Distribuição

O subsistema de distribuição foi concebido seguindo uma arquitectura cliente-servidor e é o responsável por, ao receber um pedido de um dispositivo móvel e respectiva localização (obtida, por exemplo, através de GPS), enviar o respectivo modelo tridimensional da área envolvente ao utilizador. O modelo é extraído da base de dados de modelos X3D[25] ou, no caso de ainda não estar modelado na zona solicitada, poderá ser gerado pelo sistema de modelação em tempo real.

A transmissão do modelo do servidor para o cliente está prevista ser efectuada no formato X3D, embora como prova de conceito e, face à inexistência de visualizadores X3D para dispositivos móveis, esteja neste momento a ser efectuada em VRML[27].

Devido à ainda limitada largura de banda das redes de comunicação móveis, bem como às limitações de processamento e de visualização nos dispositivos móveis, é importante que a informação enviada para o cliente não seja nem precária nem excessiva. Para o efeito, faz também parte do subsistema de distribuição, um segmentador que tem como papel a determinação da informação necessária ao utilizador, não só tendo em conta a sua localização, mas também outros factores como a geografia e topologia do modelo, a resolução do ecran do dispositivo, a largura de banda disponível, ou o perfil do utilizador[3].

4 Implementação

A modelação de ambientes urbanos virtuais é potencialmente facilitada pela existência de informação geográfica de base. Embora a quantidade de informação disponível aumente a possibilidade de aproximação do modelo à realidade, nem sempre é possível, nem necessário, utilizar toda a informação disponível. Estudos demonstram que não é

necessária a totalidade da informação para que o modelo seja minimamente realista, sendo necessária apenas a existência de algumas características chave para que aquele seja reconhecido[2].

Para a realização de uma prova de conceito do sistema apresentado, foram utilizados os seguintes níveis de informação geográfica:

- Edifícios: cor, altura, fachada (imagem se possível) e tipo (habitação, comércio, serviços, etc.);
- Ruas: tipo de pavimento, largura e faixas de circulação;
- Mobiliário urbano: postes de iluminação, bancos, painéis publicitários, etc;
- Jardins: com ou sem árvores, tipo de árvores e características das árvores;
- Monumentos: de preferência, modelos 3D detalhados ou imagens;
- Passeios: tipo de pavimento;
- Sinalização de trânsito vertical e horizontal;

Tendo estes níveis de informação sido considerados suficientes para o estabelecimento de um protótipo com um razoável nível de realismo„ foram configuradas várias fontes de dados, como descrito na figura 3, que utilizam soluções, algumas das quais proprietárias, normalmente utilizadas como bases de dados geográficas. O objectivo foi demonstrar que esta informação, apesar de por vezes se encontrar dispersa, pode ser utilizada como base para a construção de modelos virtuais urbanos 3D.

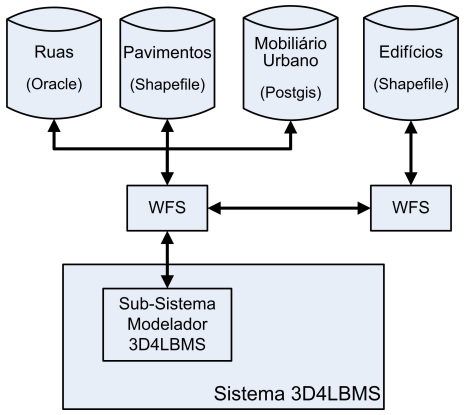


Figura 3. Exemplo da estrutura das fontes de dados disponibilizada

Como interface de acesso aos dados optou-se pela implementação de referência da norma WFS da OGC, o GeoServer[9].

Através de pedidos ao servidor WFS, o subsistema de modelação obtém dados referentes a edifícios, ruas, tipos de pavimento (de ruas e passeios) e mobiliário urbano em GML. Estes pedidos são efectuados através de HTTP, podendo utilizar-se, no caso da implementação GeoServer, quer o método POST quer o método GET. No caso de utilização do método POST e, por exemplo, para obter todos os edifícios configurados no WFS, seria efectuado o seguinte pedido:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<GetFeature outputFormat="GML2"
        xmlns="http://www.opengis.net/wfs"
        xmlns:gml="http://www.opengis.net/gml"
        xmlns:ogc="http://www.opengis.net/ogc">
   <Query typeName="local:edif">
   </Query>
</GetFeature>
```

A resposta do WFS a este pedido corresponderia ao fragmento de GML seguinte:

```
<gml:featureMember>
 <local:edif fid="edif.13904">
  <local:the_geom>
   <gml:MultiPolygon>
    <gml:polygonMember>
     <gml:Polygon>
      <gml:outerBoundaryIs>
       <gml:LinearRing>
        <gml:coordinates decimal="." cs="," ts=" ">
         -40099.224,164391.377 -40084.915,164387.825
         -40086.567,164377.288 -40087.305,164372.578
         -40087.613,164370.646 -40099.224,164391.377
        </gml:coordinates>
       </gml:LinearRing>
      </gml:outerBoundaryIs>
     </gml:Polygon>
    </gml:polygonMember>
   </gml:MultiPolygon>
  </local:the_geom>
  <local:AREA>265.58230450004</local:AREA>
  <local:PERIMETER>65.592984942794</local:PERIMETER>
  <local:CHAO>0</local:CHAO>
  <local:ZVALUE>89.119</local:ZVALUE>
 </local:edif>
</gml:featureMember>
```

À resposta do WFS, o subsistema de modelação aplica a especificação XL3D definida e, recorrendo a transformações XSLT e a processos baseados em sistemas L paramétricos[12], transforma a informação GML em modelos X3D. Estes são armazenados numa base de dados de modelos para serem disponibilizados ao subsistema de distribuição[3].

O subsistema de distribuição é o responsável por receber os pedidos do dispositivo móvel do utilizador e, com base na sua localização e contexto geográfico, por lhe enviar o respectivo ou respectivos modelos tridimensionais da área envolvente.

5 Testes e Resultados

Para testar o potencial da solução em desenvolvimento, foram configuradas diversas fontes de dados, conforme representado na figura 3, em formatos proprietários diferentes. A partir da informação geográfica que as fontes fornecem através do interface WFS, foram feitos alguns pedidos de forma a gerar alguns modelos tridimensionais de ambientes urbanos de forma automática. Estes pedidos foram testados de forma local, embora tenham sido efectuadas experiências bem sucedidas com pedidos através da Internet, para alimentar o modelador.

Para teste, foram efectuados, primeiro, pedidos para a totalidade da informação dos vários tipos disponíveis no servidor (sem filtro). De seguida, numa tentativa de melhorar

a *performance* do sistema, uma vez que nem toda a informação disponível é necessária para a representação de um ambiente urbano, foram realizados testes em que os pedidos efectuados ao WFS apenas contemplavam a informação necessária à modelação (com filtro). Os resultados destes testes estão apresentados na tabela 1.

Tabela 1. Resultados de pedidos ao WFS, contemplando apenas a informação necessária à modelação

	Tamanho da resposta (MB)		*Tempo de acesso (s)*	
Nível de informação	*sem filtro*	*com filtro*	*sem filtro*	*com filtro*
Ruas	7,7	7,2	11,2	9,2
Pavimentos	13,5	4,5	15,8	6,4
Mobiliário urbano	4	1,4	7,3	2,6
Edificios	19	13,5	28,6	15,8

Com estes testes constata-se que a utilização de filtros nos pedidos que devolvam do WFS apenas a informação necessária à modelação dos ambientes urbanos, permite reduzir de forma significativa o tamanho da informação recebida e, logo, os respectivos tempos de acesso.

O acesso interoperável às diferentes fontes de dados disponibilizadas para o sistema em desenvolvimento permitiu, desde já, obter também alguns resultados na modelação automática de ambientes urbanos.

Na figura 4 é apresentado o modelo tradicional de um sistema GIS na forma de um mapa retornado pelo servidor WFS em GML. Paralelamente é exibido o modelo tridimensional respectivo que foi gerado automaticamente pelo modelador XL3D, utilizando regras simples de extrusão dos limites dos edifícios.

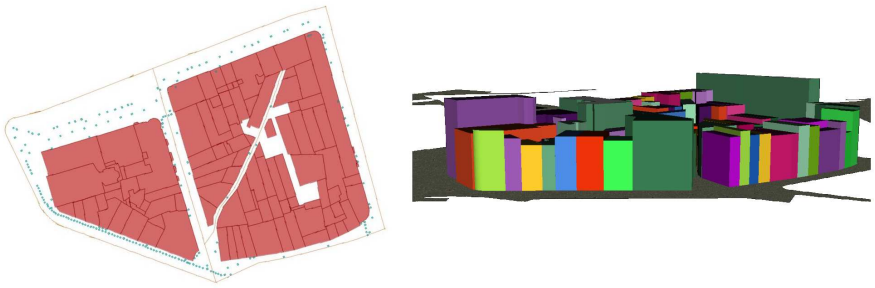


Figura 4. Mapa gerado a partir do GML retornado pelo WFS e respectivo modelo tridimensional gerado automaticamente através de extrusão

Na figura 5 é apresentado o resultado da modelação do mesmo quarteirão, utilizando processos de modelação mais elaborados baseados em sistemas L paramétricos[4]. O resultado incorpora mais detalhes como as fachadas dos edifícios, portas, janelas e varandas, estradas, vegetação e mobiliário urbano.

Figura 5. Vista de um quarteirão mais detalhado gerado automaticamente

Os processos de modelação podem ser ainda mais elaborados, de forma a produzirem objectos mais detalhados e realistas. Embora a fidelidade visual possa ser de extrema importânica em algumas aplicações, esta estará sempre dependente da quantidade e qualidade da informação disponível no processo de modelação.

A diversidade das fontes de informação e dos formatos de dados prevista neste projecto perspectiva, desse ponto de vista, uma adequada infraestrutura, adaptável às circunstâncias e às ofertas de dados, quer em quantidade, quer em qualidade.

6 Conclusões e Trabalho Futuro

Trabalhos recentes apontam na direcção de concluir que a utilização de modelos tridimensionais para a navegação e exploração de ambientes complexos, como os urbanos, é mais eficiente e compreensível que os tradicionais mapas dimensionais [24].

No entanto, a utilização deste tipo de modelos não tem explorado todas as suas potencialidades, principalmente no que diz respeito a informação georreferenciada. A falta de interoperabilidade entre os sistemas GIS proprietários, bem como as diferenças entre a informação geográfica existente, torna difícil a automatização dos processos de modelação de ambientes complexos, obrigando a grande intervenção humana.

As implementações, actualmente existentes, de SDIs globalmente interoperáveis, nomeadamente por recurso às especificações OpenGIS, permitem a disponibilização da informação geográfica proveniente de diversas fontes e em diversos formatos, numa perspectiva de disponibilização de mapas interactivos com informação temática diversificada sobre um pais ou uma região.

No entanto, recorrendo a soluções deste tipo, é possível obter informação geográfica de um modo interoperável, com grande potencial de modelação tridimensional através de processos automáticos. O resultado desta modelação, traduz-se em ambientes urbanos virtuais no formato X3D, que, sendo o formato XML para codificação de informação tridimensional, garante por si só a interoperabilidade dos mesmos.

Com o intuito de disponibilizar estes ambientes urbanos virtuais em plataformas móveis, através de serviços móveis baseados na localização, foi desenvolvido um sistema inovador denominado 3D4LBMS. Este sistema é composto por um subsistema de modelação automática, que possibilita a geração dos ambientes virtuais de acordo com uma determinada especificação de modelação XL3D, e por um subsistema de distribuição, que disponibiliza os modelos tridimensionais às plataformas móveis de acordo com a localização geográfica do utilizador.

O subsistema de modelação 3D encontra-se já operacional, gerando automaticamente, a partir de um acesso interoperável a dados geográficos em diversas fontes e formatos, os modelos virtuais da zona urbana onde o dispositivo móvel se encontra.

De forma a serem obtidos modelos tridimensionais complexos optimizados para a sua transmissão para dispositivos móveis, têm sido efectuados estudos de percepção visual sobre as principais características utilizadas pelos utilizadores ao tentar navegar em ambientes desconhecidos[2]. Percebendo para onde o utilizador olha, pode-se fornecer uma experiência com elevado nível perceptual, sendo este conhecimento utilizado para seleccionar a informação a ser modelada e apresentada ao utilizador.

Como trabalho futuro pretende-se o desenvolvimento de um protótipo de um visualizador tridimensional para dispositivos móveis, capaz de apresentar e permitir a interacção do utilizador com os modelos gerados pelo sistema, com ligações hipermédia a outros tipos de informação, configurável de acordo com o tipo de negócio a explorar.

7 Agradecimentos

Este trabalho é parcialmente apoiado pela Fundação para a Ciência e a Tecnologia (FCT), União Europeia e FEDER através do projecto POSI / CHS / 48220 / 2002 entitulado "3D4LBMS - Modelação Tridimensional de Ambientes Virtuais Urbanos para Serviços Móveis Baseados na Localização".

Referências

[1] P. Bernardes and M. Martins. Computação gráfica e arqueologia urbana: O caso de bracara augusta. In *12º Encontro Português de Computação Gráfica*, pages 63–72, 2003.

[2] Maximino Bessa, António Coelho, and Alan Chalmers. Alternate feature location for rapid navigation using a 3d map on a mobile device. In *MUM2004*, 2004.

[3] Maximino Bessa, António Coelho, João Paulo Moura, José Bulas Cruz, F. Nunes Ferreira, Alan Chalmers, and A. Augusto Sousa. Modelação expedita de ambientes virtuais urbanos para utilização em dispositivos móveis. In *13º Encontro Português de Computação Gráfica*, 2005.

[4] António Coelho, A. Augusto Sousa, and F. Nunes Ferreira. Modelação expedita de cenas urbanas. In *12º Encontro Português de Computação Gráfica*, 2003.

[5] António Coelho, A. Augusto Sousa, and F. Nunes Ferreira. Modelling urban scenes for lbms. *Web3D 2005 Symposium*, 2005.
[6] Infraestructura de Datos Espaciales de España. http://www.idee.es/show.do?to=pideep_wms_generic_viewer.ES.
[7] Angra do Heroísmo Virtual. http://urban-virtual.com.
[8] 3D Funchal. http://www.3dfunchal.com.
[9] GeoServer. http://geoserver.sourceforge.net.
[10] OGC Reference Implementations. http://cite.occamlab.com/reference/.
[11] Open GIS Consortium Inc. Opengis® web services architecture. Technical report, 2003.
[12] A. Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. *Journal of Theoretical Biology*, 18:280–315, 1968.
[13] Virtual Cities Resource Center: Cities on the Web. http://www.casa.ucl.ac.uk/vc/cities.htm.
[14] Inc. Open GIS Consortium. http://www.opengeospatial.org.
[15] Inc. Open Gis Consortium. Opengis web map server interface implementation specification. Technical Report 00-028, OGC, Abril 2000.
[16] Inc. Open Gis Consortium. Geography markup language (gml) 2.0. Technical Report 01-029, OGC, February 2001.
[17] Inc. Open Gis Consortium. Web map service implementation specification. Technical Report 01-068r2, OGC, November 2001.
[18] Inc. Open Gis Consortium. Web feature service implementation specification. Technical Report 02-058, OGC, September 2002.
[19] Inc Open GIS Consortium. *OpenGIS Catalogue Services Implementation Specification*. OGC, 2004.
[20] Zhong-Ren Pen and Chuarong Zhang. The roles of geography markup language (gml), scalable vector graphics (svg), and web feature service (wfs) specifications in the development of internet geographic information systems (gis). *Journal of Geographical Systems*, 2004.
[21] J. Pimentel, N. Baptista, L. Goes, and J. Dionísio. Construção e gestão da complexidade de cenários urbanos 3d em ambientes virtuais imersivos. In *10º Encontro Português de Computação Gráfica*, pages 165–174, 2001.
[22] North Rhein Westphalia Pilot Project. http://www.opengeospatial.org/initiatives/?iid=76.
[23] The Deegree Project. http://deegree.sourceforge.net.
[24] Arne Schilling, Volker Coors, Martin Giersich, and Rune Aasgaard. Introducing 3d gis for the mobile community technical aspects in the case of tellmaris. In *IMC Workshop*, 2003.
[25] X3D Standard. http://www.web3d.org/x3d/specifications/x3d_specification.html.
[26] J. Teixeira, P. Bernardes, A. Diogo, A. Silva, L. Soares, and S. Rodrigues. A cultura e ciência na era dos ambientes virtuais. In *8º Encontro Português de Computação Gráfica*, pages 3–18, 1998.
[27] Virtual Reality Modeling Language (VRML97). http://www.web3d.org/x3d/specifications/vrml/.
[28] Phase 1 Web Mapping Testbed. http://www.opengeospatial.org/initiatives/?iid=101.
[29] Geodata Infrastructure North-Rine Westphalia. http://www.opengeospatial.org/initiatives/?iid=76.

SInBAD – Sistema Integrado de Biblioteca e Arquivo Digital

Pedro Almeida[1], Marco Fernandes[1], Miguel Alho[1], Joaquim Arnaldo Martins[2], Joaquim Sousa Pinto[2]

[1] IEETA, Universidade de Aveiro
3810-193 Aveiro, Portugal
{pma, marcopsf, alho}@ieeta.pt

[2] DET, Universidade de Aveiro
3810-193 Aveiro, Portugal
{jam, jsp}@det.ua.pt

Resumo. O SInBAD – Sistema Integrado de Biblioteca e Arquivo Digital, é um projecto que está a ser desenvolvido na Universidade de Aveiro. O objectivo deste projecto é construir um sistema de informação capaz de armazenar os diferentes tipos de documentos que são propriedade ou são produzidos na Universidade de Aveiro, como por exemplo livros, teses, dissertações, fotografias, vídeos, músicas, etc. Estes conteúdos são acedidos através de diferentes interfaces de acordo com o tipo de informação que está a ser visualizada. Para além da questão crucial do arquivo e armazenamento, estabeleceu-se também como objectivo a criação de um portal de pesquisa para todos estes conteúdos – através de um único ponto de pesquisa, o utilizador pode obter informação sobre livros, teses, vídeos, músicas, etc., que digam respeito a um determinado assunto. A arquitectura do SInBAD é composta por vários subsistemas. Um aspecto a realçar, é que a o componente de acesso à informação foi desenhada de forma a permitir ter um armazenamento de informação distribuído. Ao nível de comunicação entre os diferentes componentes do sistema utilizam-se tecnologias baseadas em XML, nomeadamente WebServices. O modelo de descrição adoptado para os vários tipos de informação tem por base o conjunto de elementos básicos definido pelo grupo Dublin Core. Acontece que esta descrição pode não ser suficiente para caracterizar devidamente os registos, daí utilizar-se para cada descrição, uma extensão com um standard mais apropriado para o material descrito. As descrições são armazenadas no formato XML, tendo em vista a interoperabilidade e a extensibilidade que esta linguagem suporta por natureza.

PALAVRAS-CHAVE

Bibliotecas Digitais, Arquivos Digitais, Sistemas de Informação.

1. INTRODUÇÃO

O SInBAD, Sistema Integrado de Biblioteca e Arquivo Digital, é um projecto que está a ser desenvolvido para a Universidade de Aveiro, financiado por o programa Aveiro-Digital 2003-2006 [1]. Este projecto tem como finalidade construir um sistema de informação que armazene e disponibilize livros, teses, fotografias, vídeos, músicas, etc., no formato digital, sendo que a informação armazenada no sistema diz respeito à vida académica da Universidade de Aveiro. Este repositório de informação vai servir para armazenar, por exemplo, as fotografias de início do ano lectivo, discursos proferidos por personalidades importantes da Universidade de Aveiro, as teses e dissertações apresentadas na Universidade de Aveiro, etc. Para além do registo em formato digital é também necessário armazenar a descrição do mesmo, onde deve constar o nome do autor, assunto, editor, etc. Mas como nem todos os registos são do mesmo formato, é necessário classificar cada documento de acordo com o tipo de registo que está a ser armazenado. Por exemplo, no caso de uma fotografia, convém fazer a distinção entre imagens a cores ou escala cinza, mas no caso de uma música esta descrição não faz sentido. Este é somente um dos aspectos que é necessário ter em consideração, entre vários que vamos apresentar neste artigo.
Uma das inovações associadas a este sistema é a capacidade de armazenar qualquer tipo de registo e disponibilizar mecanismos de integração de conteúdos. Isto porque a informação que consta no SInBAD está dividida, até agora, em três grandes grupos: Biblioteca, Arquivo e Jazz. Estes três grandes grupos reflectem um pouco a organização interna da instituição ao nível da preservação e disponibilização de conteúdos, pois a Biblioteca da Universidade de Aveiro é que está responsável pela organização e classificação dos livros e teses, entre outra grande variedade de documentos, o Arquivo da Universidade de Aveiro está responsável por organizar as fotografias e vídeos referentes à vida académica e, finalmente, o Jazz representa uma iniciativa que está a ser desenvolvida na Universidade, iniciativa esta que pretende divulgar um vasto acervo de Jazz pertencente à Universidade de Aveiro. A integração destes três sistemas é feita através do SInBAD, um portal, um único ponto de pesquisa, onde um utilizador pode fazer pesquisas nos conteúdos da Biblioteca, Arquivo e Jazz. Ao fazer uma pesquisa no SInBAD o utilizador vai obter uma listagem de teses, livros, fotografias, vídeos, músicas, etc., que apresentem conteúdos relacionados com a pesquisa efectuada. Cada registo é apresentado através de uma interface própria, pois cada um destes catálogos tem um subsistema que apresenta a informação num formato amigável. Este artigo apresenta também os objectivos do projecto SInBAD, a arquitectura do sistema e algumas conclusões sobre a investigação desenvolvida até ao momento.

2. TRABALHO RELACIONADO

No artigo [2], Robert Tansley et. al. apresentam o DSpace [3]: um sistema *open-source* que actua como um repositório para material digital educacional e de investigação produzido por uma organização ou instituição. Este sistema foi desenvolvido numa parceria entre a HP e a biblioteca do MIT e foi construído tendo

em conta todas as funcionalidades necessárias ao bom funcionamento de um serviço de repositório digital. Este serviço deve ser um serviço "vivo", sendo o alicerce para as funcionalidades de um repositório e particularmente para resolver as preocupações de armazenamento a longo prazo. Todos os registos que estão armazenados no DSpace podem ser livremente acessíveis, não existindo restrições no acesso à informação. Os aspectos funcionais do DSpace podem ser resumidos da seguinte forma:

1. É definido um modelo de dados para organizar a informação de uma forma básica.
2. Metadados de vários tipos são armazenados pelo sistema.
3. O sistema armazena informação acerca dos utilizadores do sistema. Alguns utilizadores podem não ser humanos mas sim outros sistemas computacionais.
4. Enquanto muito do esforço visa facilitar o acesso ao material digital de uma instituição, simplesmente permitir acesso livre total nem sempre é aceitável. Funções adicionais como depositar ou avaliar devem ser restringidas a determinados indivíduos. Daí que o sistema inclua uma funcionalidade de autorização.
5. O sistema tem que ser capaz de aceitar material submetido, um processo definido com ingestão.
6. Algumas comunidades podem requerer que o material ou metadados relacionados que são submetidos para arquivo sejam verificados por indivíduos designados. Este processo chama-se *workflow*.
7. Tendo em vista que o material armazenado pode ser citado e acedido usando informação de uma citação, um sistema de gestão de identificadores é usado para atribuir identificadores únicos e persistentes aos objectos arquivados.

A nível de funcionalidades de pesquisa e descoberta o DSpace permite que os utilizadores acedam ao conteúdo através de uma referência externa, por exemplo um identificador, procurando por uma ou mais palavras chave, navegando pelos índices de títulos, data e autor. Convém também salientar que o DSpace implementa o protocolo OAI-PMH [5], expondo os metadados Dublin Core dos registos que são públicos.

A arquitectura do DSpace é baseada num modelo de três camadas. Utiliza uma base de dados relacional para armazenar a informação sob a forma de uma cadeia de bits, mantendo os metadados acessíveis pelo sistema. As funcionalidades de indexação, autorização, gestão de utilizadores, etc., estão localizadas na camada lógica, e esta disponibiliza um conjunto de API's para comunicar com aplicações externas. Outra característica importante é que utiliza um serviço do ORNI Handle Server [6] para atribuir identificadores únicos ao registos que armazena.

3. OBJECTIVOS

Durante a fase de análise do projecto analisou-se a possibilidade de adaptar o DSpace para construir o SInBAD, mas devido a algumas limitações do DSpace optou-se por desenvolver um sistema de raiz. Comparativamente ao DSpace, o SInBAD suporta as funcionalidades 2,3,4,5 e 6 apresentadas na secção 2 e definimos também os seguintes requisitos:

1. Utilizar XML [8] e tecnologias baseadas em XML para armazenar e disponibilizar o acesso à informação existente no sistema.
2. Adoptar uma arquitectura distribuída para suportar diversos repositórios em vez de um repositório único.
3. Suporte para descrições multinível.
4. Estrutura de organização de informação especificada em cada subsistema.
5. Apresentação dos documentos de uma forma estruturada.

4. ARQUITECTURA

Os objectivos mencionados, foram também definidos tendo em consideração a integração do SInBAD com os vários sistemas que existem na Universidade de Aveiro, nomeadamente o sistema Aleph [9] da Biblioteca, onde são armazenadas as descrições dos livros, revistas, etc., e o sistema PACO [10] onde são armazenadas as informações sobre os alunos, como por exemplo as disciplinas a que estão inscritos.

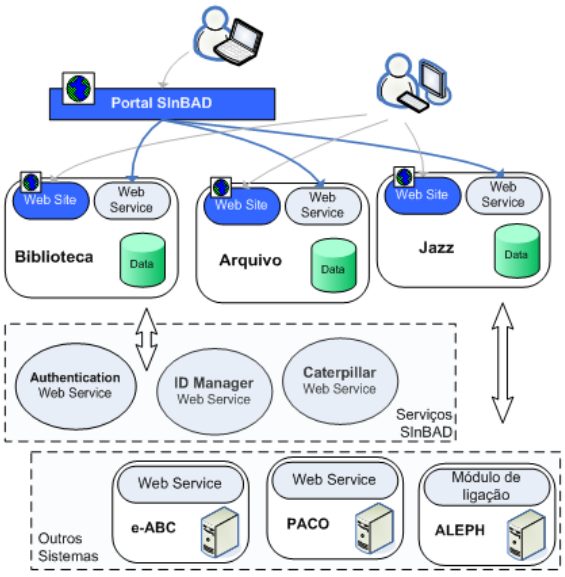


Figura 1 - Arquitectura do SInBAD

A arquitectura do SInBAD utiliza WebServices [11] por forma a permitir uma fácil integração dos subsistemas do SInBAD em outros portais de informação. A Figura 1 ilustra a arquitectura do sistema.
O SInBAD em si é composto por vários subsistemas, nomeadamente o subsistema da Biblioteca, o subsistema do Arquivo e o subsistema do Jazz.

4.1 Subsistemas

Cada um dos subsistemas do SInBAD poderia existir isolado, daí que para cada subsistema exista um Web Site individual. O bloco que aparece com o portal SInBAD faz a integração de todos os subsistemas existentes. O portal funciona como um ponto de acesso a partir do qual se podem fazer pesquisas em todos os subsistemas que existem no portal, Biblioteca e Arquivo e Jazz. Desta forma, caso sejam criados mais subsistemas dentro do SInBAD, como por exemplo um subsistema de informação para museus, desde de que este seja construído segundo a arquitectura do SInBAD facilmente se integraria este novo subsistema no portal, permitindo fazer pesquisas simultâneas na Biblioteca, Arquivo, Jazz e Museu Digital.

Figura 2 - Interface do SInBAD

Após efectuar uma pesquisa no portal SInBAD este devolve uma listagem com documentos dos vários subsistemas, mas quando o utilizador selecciona um desses documentos é a interface do respectivo subsistema que vai mostrar a informação. Desta forma garante-se que a informação é apresentada de uma forma estruturada ao

utilizador, isto é, da melhor forma possível, uma vez que cada subsistema pode implementar as suas regras de apresentação dos conteúdos.
Já foi desenvolvido um protótipo do SInBAD e a Figura 2 ilustra o resultado de uma pesquisa efectuada no sistema. Como se pode observar, o sistema devolve registos de teses, pertencendo estas ao subsistema da Biblioteca Digital, e registos do Arquivo Fotográfico. Este protótipo permite ver perfeitamente o tipo de resultados que se pode obter com este sistema e a integração de pesquisas em vários sistemas de informação.

4.2 Arquitectura dos subsistemas

Cada subsistema é composto por uma série de componentes que formam a arquitectura dos subsistemas, como se pode observar na Figura 3. Esta divisão por subsistemas permite que cada um defina as regras de consulta e as regras de acesso aos seus conteúdos. O bloco *query manager* é responsável por implementar as funções de acesso aos dados, nomeadamente consultar, pesquisar, eliminar e alterar registos.
Um aspecto a destacar é que cada subsistema implementa o seu próprio modelo de organização e descrição de dados. Este factor é importante, pois os arquivistas, bibliotecários, museólogos, etc., não trabalham da mesma forma e cada um usa os seus próprios standards de descrição e regras de organização da informação. Isto permite que os subsistemas se adaptem aos requisitos do utilizador e não que o utilizador se adapte ao modo de funcionamento dos sistemas.
Cada subsistema contém um módulo de gestão de direitos, o bloco *Rights Manager*. Existem diversos perfis de utilizadores do SInBAD e para cada perfil é necessário verificar os direitos do utilizador. Por exemplo se um utilizador tentar aceder a uma tese em formato digital tem que estar autenticado como aluno, docente ou funcionário da Universidade de Aveiro, caso contrário não pode ver o documento em formato digital. Cada documento inserido no SinBAD tem associada uma descrição dos direitos de visualização por perfil de utilizador.

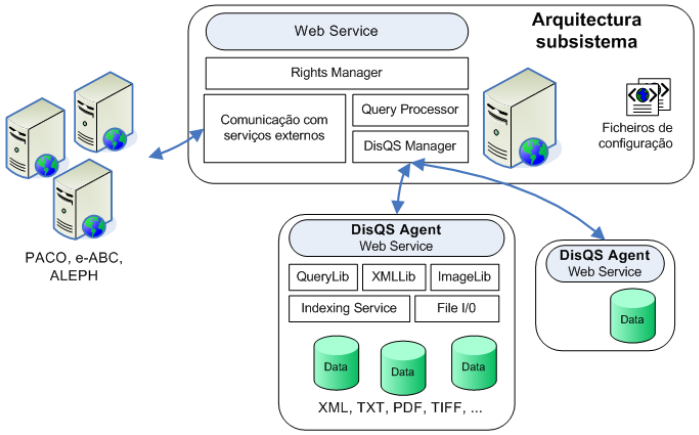


Figura 3 - Arquitectura dos subsistemas

4.2.1 Acesso e armazenamento de informação

As funções de acesso e manipulação de dados são geridas através de Web Services, aumentando desta forma a interoperabilidade do sistema. Cada subsistema deve implementar o seu próprio Web Service. Sempre que um novo documento é inserido num dos subsistemas o módulo *ID Manager Web Service* atribui-lhe um identificador único no sistema.
O armazenamento da informação, isto é, o documento propriamente dito e os metadados associados ao documento, é efectuado pelo módulo *DisQS Manager*, um sistema de armazenamento distribuído. O *DisQS Manager* faz a gestão de um conjunto agentes que armazenam as descrições em formato XML e os documentos, sendo que cada documento pode ser armazenado em um ou mais agentes DisQS, sempre com o mesmo identificador que foi atribuído pelo módulo *Id Manager*. Estes agentes, funcionam como um conjunto de diferentes repositórios que actuam como um repositório virtual único. Este repositório virtual único, tem como objectivo disponibilizar um conjunto de funcionalidades que visam aumentar a fiabilidade e desempenho do sistema global, conforme se descreve detalhadamente em [12]. Neste módulo a informação é armazenada de forma redundante, existindo um mesmo ficheiro em diferentes repositórios. Quando é efectuada uma consulta ao *DisQS Manager*, este agrupa os resultados dos vários agentes e elimina os itens repetidos.

4.2.2 Transformação de dados

O SInBAD suporta o armazenamento de diversos tipos de ficheiros, mas os formatos preferenciais são o PDF [13] e o TIFF [14]. Ambos estes formatos têm um problema, é que não são suportados directamente por os browsers web. Sendo assim, muitas vezes é necessário converter imagens para um formato diferente daí a existência do *Caterpillar Web Service*, um módulo que tem um conjunto de funcionalidades associadas, como por exemplo a conversão de imagens TIFF para PNG, e a extracção de texto e imagens de documentos PDF. Depois de transformadas as imagens são colocadas numa cache, por forma a que em novos pedidos não seja necessário converter novamente a imagem, aumentando desta forma o desempenho do sistema.

4.2.3 Interfaces com outros sistemas

A arquitectura dos subsistemas também contempla interfaces de comunicação com outros sistemas da Universidade, nomeadamente o PACO e o ALEPH, por forma a reutilizar informação da catalogação das teses, livros, etc., e comunicar com o sistema da secretaria da Universidade por forma a saber quais as disciplinas em que os alunos estão inscritos, quais as disciplinas de que um docente é responsável, o plano de estudos da disciplina, etc., sendo depois esta informação utilizada para atribuir as permissões de acesso à informação e para criar serviços de valor acrescentado. Por exemplo, após fazer *login* no SInBAD, um aluno pode saber quais as disciplinas a que está inscrito e pode aceder directamente à bibliografia da disciplina em formato digital.

5. MODELO DE DESCRIÇÃO DOS REGISTOS

Cada subsistema do SInBAD utiliza diferentes standards de descrição dos documentos de acordo com as características do material armazenado. No entanto, para além da descrição pormenorizada existe também uma descrição elementar para cada um dos registos segundo a norma Dublin Core. O formato de descrição Dublin Core é composto por um conjunto reduzido de elementos que existem em todos os tipos de registos: o título, autor, assunto, descrição, editor, etc.. O modelo de descrição adoptado no caso da Biblioteca Digital utiliza Dublin Core para fazer a descrição base e DC-Lib [15] para fazer a descrição pormenorizada. No caso do Arquivo Digital existem três tipos de registos: textos, imagens e audiovisuais. Cada um destes registos deve utilizar standards de descrição diferentes de acordo com a natureza do documento. Caso se trate de um registo audiovisual a descrição é armazenada segundo a norma MPEG-7 [16], no caso de ser uma fotografia a descrição é armazenada segundo a norma VRACore [17] e no caso de se tratar de um documento de arquivo usa-se a norma ISAD(G) [18]. Estas descrições armazenam as características específicas dos registos, sendo que o resto da informação é armazenada segundo a norma Dublin Core. A Figura 4 ilustra o modelo de descrição adoptado para o Arquivo Digital: uma descrição base segundo a norma Dublin Core e a extensão dos restantes standards de acordo com a natureza do documento associado.

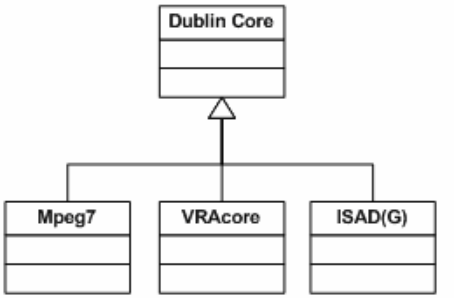


Figura 4 - Modelo de descrição do Arquivo Digital

Uma vez que todos os registos têm uma descrição básica Dublin Core, independentemente de pertencerem ao Arquivo ou à Biblioteca Digita e mesmo independentemente do formato, pesquisando por um dos campos Dublin Core pode-se obter uma listagem de registos pertencentes a qualquer um dos sistemas, sendo que este registo pode ser um vídeo, um livro, uma tese, etc.. O *DisQS Manager* é que indica de onde veio o registo e que tipo de informação é que contém, isto é, se é um vídeo, uma fotografia ou um documento.
A Figura 5 representa um exemplo de uma descrição de um cartaz de Jazz. Como se pode observar, existe um conjunto de elementos Dublin Core que descrevem as características gerais do cartaz e alguns elementos VRA Core que descrevem características específicas, como por exemplo as dimensões do cartaz.

```
<document xmlns:sinbad="http://sinbad.ua.pt" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dclib="http://purl.org/dc/terms/" xmlns:dcterms="http://purl.org/dc/terms2/"
xmlns:vracore="http://www.vraweb.org/vracore3.htm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ua.pt/SInBAD/Cartazes.xsd"
xmlns="http://www.ua.pt/SInBAD/Cartazes">
      <dc:date>2002</dc:date>
      <dc:description>Texto do cartaz: 25 Out. a 2 Nov., Grande Auditório do Centro de Arte e
    Espectáculos. Identificação dos músicos/intérpretes dos diversos concertos a realizar e
    respectivas datas.</dc:description>
      <vracore:Measurements.Dimensions>30 x 21 cm.</vracore:Measurements.Dimensions>
      <vracore:IdNumber.FormerRepository>00190216</vracore:IdNumber.FormerRepository>
      <dc:language>por</dc:language>
      <dc:publisher>Câmara Municipal</dc:publisher>
      <vracore:subject>Cartazes de concertos</vracore:subject>
      <dc:title>FozJazz 2002 : 1º Festival Internacional de Jazz da Figueira da Foz</dc:title>
      <dcterms:spatial>Figueira da Foz</dcterms:spatial>
      <dc:identifier
    xsi:type="dcterms:URI">http://sinbad.ua.pt/cartazes/CEJ_JD_CT_I_18</dc:identifier>
      <dc:subject>Música popular</dc:subject>
      <dc:subject>Jazz</dc:subject>
      <dc:subject>Concertos de jazz</dc:subject>
      <dc:type>Cartaz</dc:type>
      <dc:rights><sinbad:visible/></dc:rights>
      <sinbad:changed date="21-12-2005 15:23:42">jazzedit</sinbad:changed>
    </document>
```

Figura 5 – Exemplo de uma descrição XML de um cartaz

6. CONCLUSÃO

Da análise do SInBAD no seu estado actual podemos concluir que o objectivo principal, integrar o sistema de Biblioteca e Arquivo Digital da Universidade de Aveiro foi alcançado. Como vantagens deste sistema sobre os demais existentes destacamos os factos de usar standards baseados em XML, uma linguagem extensível, para o armazenamento das descrições, Web Services para a comunicação entre sistemas por forma a garantir a interoperabilidade com outros sistemas existentes na Universidade, a utilização de subsistemas por forma a aumentar a flexibilidade no que diz respeito à apresentação e organização da informação.

Utilizando a arquitectura SInBAD é possível criar um grande repositório com toda a informação de uma instituição, integrando diversas fontes de informação num único portal. Neste caso concreto foram integrados a Biblioteca, o Arquivo Digital e o Jazz, mas facilmente poderiam ser agregados mais subsistemas.

O modelo de descrição adoptado para o sistema permite que os subsistemas utilizem os standards de descrição adequados. Independentemente da natureza dos documentos utiliza-se sempre uma descrição Dublin Core que permite fazer pesquisas em todos os registos do SInBAD, obtendo deste modo uma funcionalidade de

integração básica, mas com valor acrescentado para grandes repositórios de informação heterogénea.

REFERÊNCIAS

[1] Aveiro Digital, 2004. http://www.aveiro-digital.pt, último acesso em Novembro 2004.

[2] Tansley, R. et al, 2003, The DSpace Institutional Digital Repository System: Current Functionality, *Proceedings of the Joint Conference on Digital Libraries (JCDL'03)*, Houston, USA.

[3] MIT, 2003. *DSpace Federation*, http://www.dspace.org/, último acesso em Março de 2005.

[4] Dublin Core Metadata Initiative, 2004. *Dublin Core Metadata Element Set, Version 1.1: Reference Description*, http://www.dublincore.org/documents/dces/, último acesso em Novembro de 2005.

[5] OAI, 2005. *The Open Archives Initiative Protocol for Metadata Harvesting*, http://www.openarchives.org/OAI/openarchivesprotocol.html, último acesso em Novembro de 2005.

[6] S. Sun, L. Lannom, B. Boesch CNRI, 2003. *Handle System Overview*, http://www.ietf.org/rfc/rfc3650.txt, último acesso em Novembro de 2005.

[7] University of Waikato, 2005. *The Greenstone Digital Library Software*, http://www.greenstone.org, último acesso em Março de 2005.

[8] W3C, 2005. *Extensible Markup Language (XML)*, http://www.w3.org/XML/, último acesso em Novembro de 2005.

[9] Universidade de Aveiro, 2005. *Universidade de Aveiro - Serviços de Documentação*, http://www.doc.ua.pt/, último acesso em Novembro de 2005.

[10] Universidade de Aveiro, 2005. *Portal Académico Online*, http://paco.ua.pt/, último acesso em Novembro de 2005.

[11] W3C, 2004. *Web Services Architecture*, http://www.w3.org/TR/ws-arch/, último acesso em Novembro de 2005.

[12] Almeida, P. et al, 2005, DisQS - Distributed Query System Based on Web Services for Digital Librarie, *Proceedings of the First International Conference on Internet Technologies and Applications (ITA 05)*, Wrexham, UK.

[13] Adobe Developers Association, 2005. *PDF Reference*, http://partners.adobe.com/public/developer/pdf/index_reference.html, último acesso em Novembro de 2005.

[14] Adobe Developers Association, 1992. *TIFF Revision 6.0 Final*, http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf, último acesso em Novembro de 2005.

[15] Dublin Core Metadata Initiative, 2004. *DC-Library Application Profile (DC-Lib)*, http://www.dublincore.org/documents/library-application-profile/, último acesso em Novembro de 2005.

[16] International Organization for Standardization, 2003, *MPEG-7 Overview (version 9)*, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, último acesso em Novembro de 2005.

[17] Visual Resources Association Data Standards Committee, 2002. *VRA Core Categories, Version 3.0*, http://www.vraweb.org/vracore3.htm, último acesso em Novembro de 2005.

[18] ICA - International Council on Archives, 1999, *ISAD(G): General International Standard Archival Description*, Second Edition, Stockholm, Sweden, http://www.ica.org/biblio/cds/isad_g_2e.pdf, último acesso em Novembro de 2005.

An Automated Process to Create Preservation and Publishing Copies of Digitized Works at the BND

José Borbinha[1], João Gil[2], Gilberto Pedrosa[3], João Penas[4]

INESC-ID – Instituto de Engenharia de Sistemas e Computadores, Rua Alves Redol 9, Apartado 13069, 1000-029 Lisboa, Portugal
[1]jlb@ist.utl.pt, [2]jgil@ext.bn.pt, [3]gfsp@ext.bn.pt, [4]jpenas@ext.bn.pt

Abstract. This paper describes the automated process to create structured master and access copies for the digitised works at the BND – National Digital Library. The BND created during 2004 and 2005 nearly half a million of digitized images, from more than 25.000 titles of printed works, manuscripts, drawings and maps. The resulting of the digitisation process is a group of TIFF image files representing the surfaces of the original works, which needs yet to be processed in order to be stored and published. Doing that manually would be a very complex and expensive task, with risks for the uniformity of the results, so it was need to develop an automated solution. To create the technical metadata, apply image processing actions and OCR, create derived copies for access in PNG, JPG, GIF, and PDF, we developed a tool named SECO. To create the master copies for each of those works, for preservation, and access copies in HTML, we developed a tool named ContentE, which exists as a standalone tool and as a library. Finally the copies are deposited and registered at the BND repository through the service PURL.PT, which assures also the WEB and intranet access control. This complex process is fully automated through several XML schemas for the control of the processes, description of the results (including the OCR outputs), descriptive metadata (in Dublin Core, MARC XML, etc.) and rights and structural metadata (in METS).

1 Introduction

This paper describes the automated process to create structured master and access copies for the digitised works at the BND – National Digital Library [3]. To create the technical metadata and apply image processing actions, we developed a tool named SECO. To create the master and access, we developed a tool named ContentE. Finally the copies are deposited and registered at the BND repository through the service PURL.PT, which assures also the WEB and intranet access control.

All these components interoperate using XML schemas for the control of the processes and description of the results This is fundamental to meet the BND requirement of an open and widely accepted mean to exchange information with heterogeneous systems. Standardized toolsets and libraries are continuously under development, and XML is the most suitable technology for both data representation

and storage. In particular, modern metadata standards and toolsets, such as for METS and Dublin Core, are essentially "XML based".

The next section describes the overall architecture for the system, stressing the main components. Following, we describe the main relevant sub-systems by their sequence in the workflow (SECO, ContentE and PURL.PT). Finally, we present some conclusions and ideas for future work.

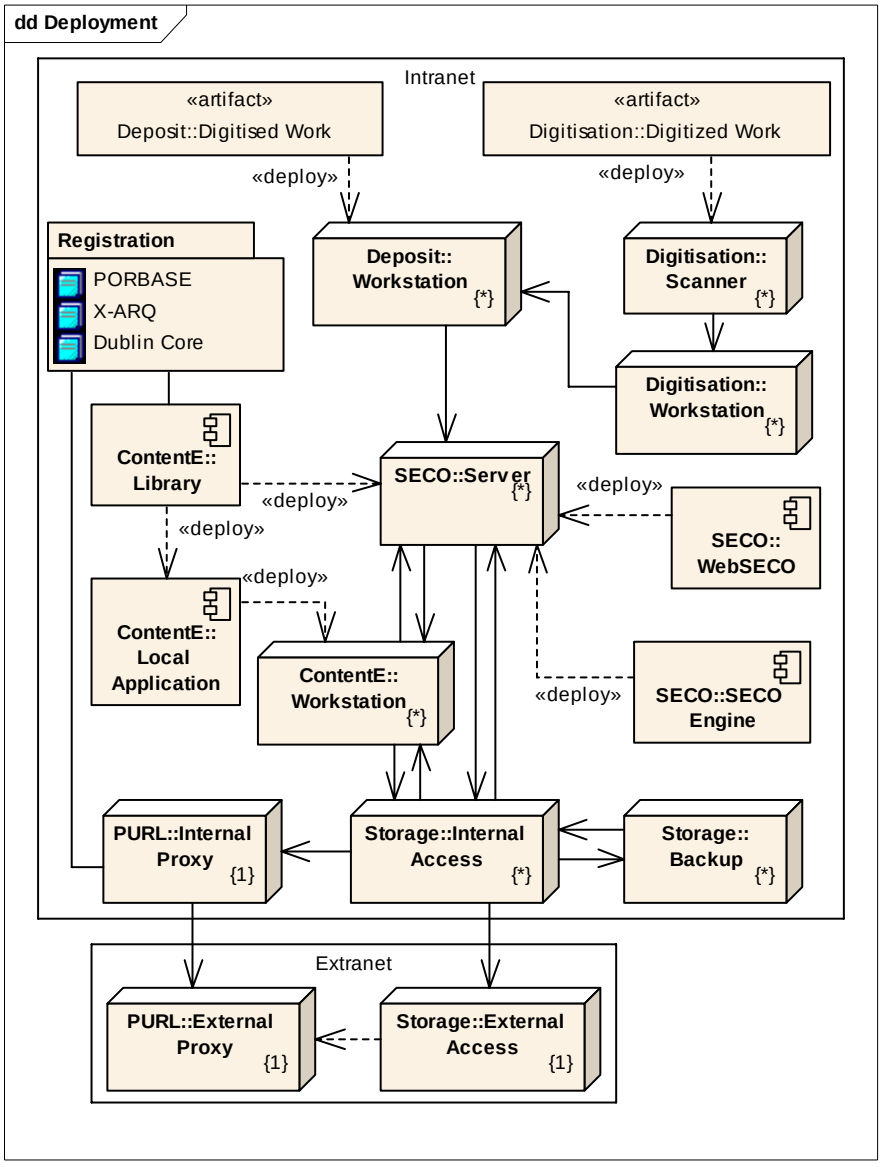


Fig. 1. The overall architecture of the information system supporting the image processing, metadata creation and copies biding for the digitized works at the BND

2 About the overall architecture of the BND

The main purposes of the BND are the development of services for the preservation, registration, discovery and access to digital resources. Those resources comprise digitised and digital born cultural and scientific documental resources. In this paper we are focused only in problem of the digitised resources.

The BND created during 2004 and 2005 nearly half a million of digitized images, from more than 25.000 physical items, comprising printed works, manuscripts, drawings, maps, etc. The resulting of the digitisation of each physical item is a group of TIFF image files representing each surface of the item. In this project nearly 50% of the images, those of items with only text and with good detail, were digitised at the resolution of 300 dpi. The other 50% were digitised at 600 dpi. All the images have 24 bit of colour dept. The biggest images, representing large maps with sizes closer to A1, have more than 2 GBytes.

The master images are perfect for preservation and high quality reproduction, but are too much for access, especially in the Internet. Also, each group of master images representing the same original needs to be described according to that before can be sent to storage. All of this represents a very heavy processing. Doing that manually would be a very complex and expensive task, with risks for the uniformity of the results, so it was identified the need to develop an automated solution. This resulted in a system made of multiple interoperable components, as represented in the Fig. 1 (this image shows only the architecture and components relevant for the scope of this paper, as the overall architecture of the BND comprised also other components here not represented).

In this description we can see that all the process starts with the submission of a digitised work, which can be created internally at the BN, or by an external entity and deposited at the BN. After any of those works is submitted to a local workstation, and after an initial simple inspection, it is sent to SECO. SECO is a service for the creation of the technical metadata, applying of image processing actions, including OCR, as also the creation of derived copies for access in PNG, JPG, GIF, and PDF.

After the processing in this system, SECO calls the ContentE library to create the METS structure and the preservation and access copies. The results are then inspected by a professional, and if they accepted, they are sent directly to storage. However, if it is detect any unusual problem, or if it is decided to publish a specific work in a fancier way, we can use a ContentE Workstation for that. Here the structure can be tuned trough a powerful user interface (creating more indexes and applying a customised style sheet).

Finally the copies are deposited and registered at the BND repository through the service PURL.PT, which assures also the WEB and intranet access control. The PURL.PT service interoperates also with the other traditional registration services, such as the PORBASE – *Base Nacional de Dados Bibliográficos* [9] for the bibliographic works, X-ARQ for the archival works, etc.

This complex process is fully automated through several XML schemas for the control of the processes, description of the results (including the OCR outputs), descriptive metadata (in Dublin Core [10], UNIMARC, coded in MARCXML [7], etc.), as also rights and structural metadata, expressed in METS [8].

3 SECO – Serial Converter

The publishing process of digitized works at BND is essentially geared towards online availability. Non-standard formats and platform dependencies are avoided so that the widest possible range of devices and software can access the information (this is also an important requirement for long term preservation, anyway). For public web-based access, data formats are carefully chosen in order to provide good quality within current technological limitations – bandwidth in particular. On the local intranet (inside the network of the BN – Biblioteca Nacional), however, quality can and should be optimized through larger files and richer content.

To achieve these goals, the publishing process begins with the submission of master high-resolution images to the SECO system. These are usually acquired through top quality digitization techniques and equipment and stored as uncompressed TIFF files at 24 bit colour depth. The typical resolution is 600dpi, or at least 300dpi for good quality printed works containing only text. Consequently, the resulting files can be quite large, often reaching sizes above 1 GByte, which is obviously a major concern in the system.

At its core, the SECO system performs image processing operations. The most basic of these operations is format conversion. TIFF files are not commonly supported by web browsers, which will be the main access tools for the published works. SECO can generate JPEG, GIF, PNG and PDF image files. Only the latter is not natively supported by current-generation browsers. It is nevertheless a nearly ubiquitous multi-page format and viewers such as the Adobe Acrobat Reader [1] are freely available for most platforms.

SECO also performs scaling and re-sampling, as well as colour depth reduction. This allows creating manageable image files for consumer-level hardware at 150dpi and 72dpi, common resolutions, and bitmap (1 bit colour depth). These images are also compress very effectively in PNG and the PDF formats. The nature and content of the digitized works must be weighed when selecting these parameters.

Additionally, OCR can be applied to the master images with textual contents, producing standard text files as also textual PDF copies. In general, the results of the OCR are not good enough to be immediately published, and proofreading the extracted text can be very time-consuming (but we do it for some special works). However, this data can be used to build word indexes for the digitized pages. These indexes can later be read by automated indexing tools and complement standard search procedures. Because we store also the word locations (coordinates) in the images, sophisticated presentation mechanisms can be built upon this information. These word indices are kept in XML files, such as represented in the Fig. 2.

For very large and highly detailed images (e.g. maps), creative solutions must be sought to cater for ordinary network connections. For this SECO supports also automated image slicing, this can break a large image into a tree of lightweight parts at different detail levels. This concept is realized afterwards when generating the actual website for the digitised work.

After master images are sent to the SECO server through a network file sharing protocol, all further operations are configured and executed through the SECO web-based user interface known as WebSECO. Fig. 3 shows a screenshot of this interface, representing the process's list with an assortment of available options.

This web interface allows viewing bibliographic records, reports processing status and provides file and process management operations. It also includes a cropping tool to remove unwanted elements from images, such as colour charts and excess margins. The user specifies the crop area graphically on low resolution versions and the system scales the coordinates to the full resolution masters. This way, no manual processing of the heavy high resolution images is necessary. The low resolution copies for this purpose are generated by an optimized rescaling implementation.

```
<word recognized="ahi">
        <image bottom="236" left="457" right="484" top="219">1-64003-p_0033_64-65_t0.TIF</image>
        <image bottom="683" left="255" right="287" top="665">1-64003-p_0036_70-71_t0.TIF</image>
        <image bottom="508" left="181" right="206" top="491">1-64003-p_0037_72-73_t0.TIF</image>
</word>
<word recognized="ainda">
        <image bottom="628" left="666" right="711" top="611">1-64003-p_0034_66-67_p0.tif</image>
        <image bottom="731" left="681" right="728" top="714">1-64003-p_0035_rosto_t0.TIF</image>
        <image bottom="404" left="754" right="800" top="387">1-64003-p_0036_70-71_t0.TIF</image>
</word>
<word recognized="alarido">
        <image bottom="268" left="213" right="272" top="251">1-64003-p_0036_70-71_t0.TIF</image>
</word>
<word recognized="alcançado">
        <image bottom="706" left="396" right="480" top="685">1-64003-p_0037_72-73_t0.TIF</image>
</word>
<word recognized="alcançaram">
        <image bottom="242" left="271" right="368" top="222">1-64003-p_0037_72-73_t0.TIF</image>
</word>
<word recognized="alcácer">
        <image bottom="335" left="165" right="224" top="317">1-64003-p_0035_rosto_t0.TIF</image>
        <image bottom="585" left="710" right="770" top="568">1-64003-p_0035_rosto_t0.TIF</image>
        <image bottom="535" left="379" right="439" top="517">1-64003-p_0036_70-71_t0.TIF</image>
        <image bottom="831" left="146" right="206" top="813">1-64003-p_0036_70-71_t0.TIF</image>
</word>
```

Fig. 2. Example of a word index formated in XML

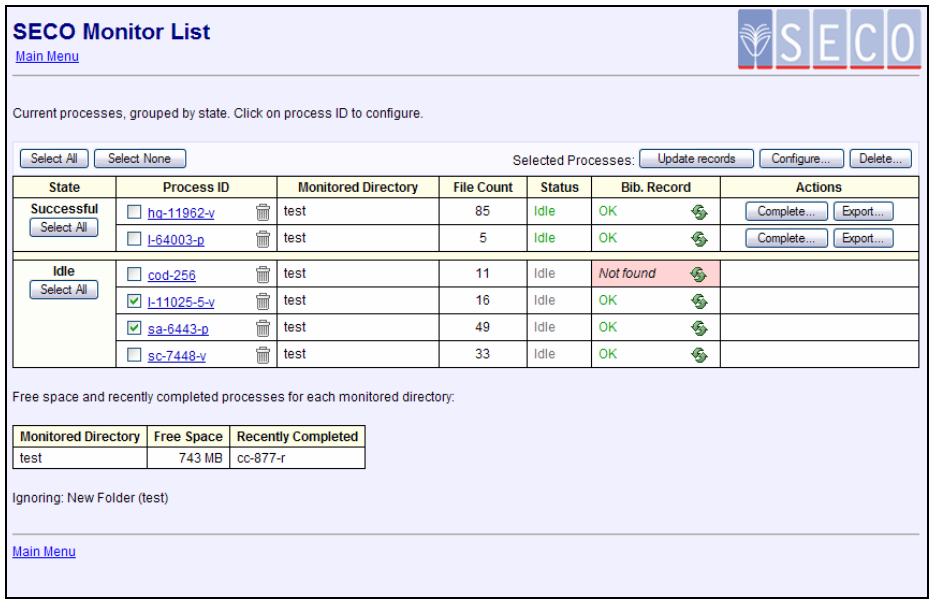


Fig. 3. Example of WebSECO, the on-line interface for SECO

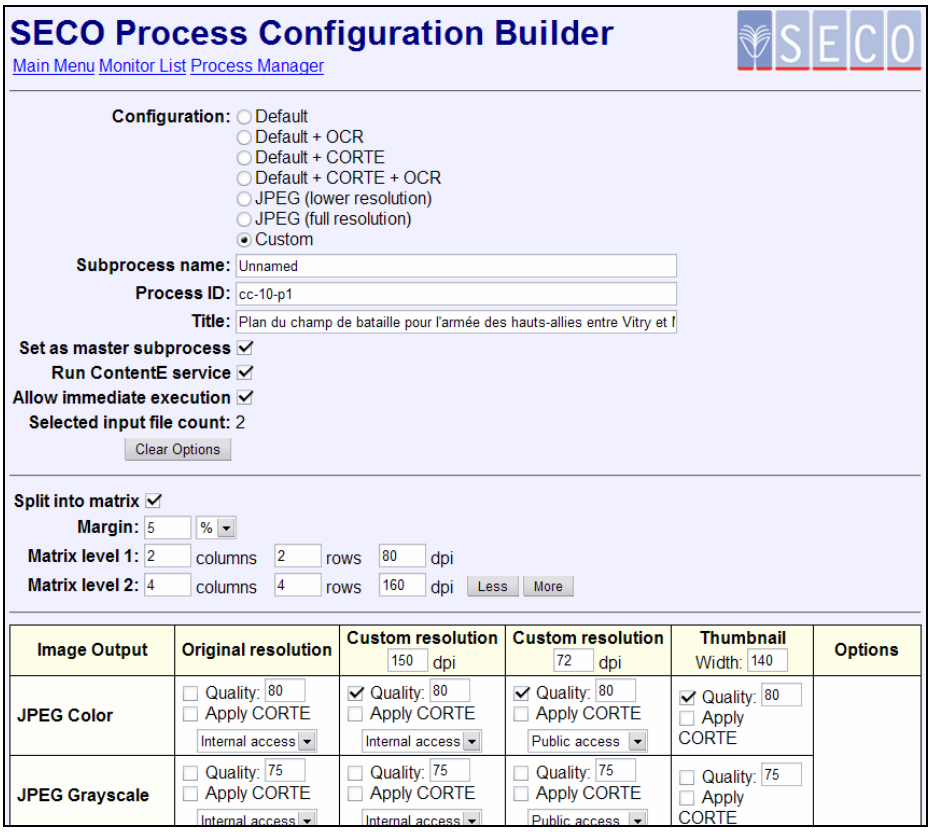


Fig. 4. The process configuration page of the WebSECO interface

The most complex area of the user interface is, however, the process configuration page, partially shown in Fig. 4. All the available formats and sub-formats are listed in a table, along with compression quality, resolution, slicing and multi-page parameters. Several presets are available as configuration templates for most common cases, usually requiring little or no customization for each individual process. To further enhance productivity, a single configuration can be shared by a group of works, so that a user can quickly setup and launch an arbitrarily large set of similar tasks.

After the configuration is set, the scheduling component can execute the process immediately of only in a pre-determinate moment, avoiding overloading the server with simultaneous processor, memory and disk intensive image operations.

After the processing of the images, the process continues by generating XHTML bindings for the access copies. For this, SECO invokes the ContentE Service. It is an integrated software module on which the ContentE Local Application is also based. A detailed description will be given in the next section.

The final stage at the SECO processing is quality control. If the automated execution results are deemed satisfactory, which is true for the majority of the cases, the finished work is flagged as complete and the SECO system notifies the central storage service, which moves the data to the preservation and online access areas. On

the other hand, if the results have insufficient quality, the user can reconfigure the process attempting to correct detected problems. A third option is exporting the generated data to a workspace where it can be manually adjusted, retouched or corrected. This option supports the creation of detailed indexes through the ContentE Local Application.

To store all the data behind the SECO execution, no traditional data base is used. Instead, the system relies on XML extensively, writing information in XML files. This allows for dynamic and flexible data formats and the ability to keep work metadata linked to its main files. Works to be published can be moved, copied and distributed while including relevant properties for indexing, archiving, technical description and additional processing.

The configurations of the SECO processing core are also stored in XML files. The available presets, parameters, processing modules and configuration templates are themselves described in that way. Than SECO architecture can work as building blocks to fit the needs of each image set. Other information stored by SECO in XML files include pre-processing and crop data, generated file properties and statistics, OCR results, configuration history, process persistence data structures and detailed logs of all the processes.

4 ContentE

The processes based on ContentE are described with detail in [4]. Here we'll make only a resume of it! ContentE is made essentially of two components: a library, to be used by other systems, and a local application that uses the same library and provides a powerful user interface for advanced usage.

The ContentE library is used by SECO to process the results of the previous steps and produce master copies for preservation and copies for access. The master copies are just a folder organised inside in a set of other folders, one for each MIME type existing for the object. One typical MIME type that is always present is TIFF. Other types are usually JPEG, PNG, GIF, PDF and TXT. For all of this, ContentE creates also structural descriptions in METS, as also indexes. One index that is always created automatically is the original physical index, representing the images by their natural order. More complex indexes that can be also automatically created by SECO are tree indexes for images that are split in multiple areas, for better visualisation. Other complex indexes, such as authors, parts, chapters, etc., can be created only using the ContentE Local Application, or they can be also created trough SECO but in this case the respective XML descriptions (in METS) have to be provided with the submission of the images.

ContentE creates one access copy for each derived MIME type. Each of these copies is represented as an XHTML site, where the homepage shows the descriptive metadata and possibly also the indexes and thumbnails of the cover pages, etc. The indexes can be also textual or they can use thumbnails to represent the access points. To create these XHTML copies, ContentE can apply multiple XSL visual styles that are already integrated in the library. Anyway, any new style can be used at any moment, providing that it can be applied to METS. Finally, the ContentE Library can

retrieve descriptive metadata from external systems, such as PORBASE and X-ARQ, or import it from local files (in Dublin Core [10] or UNIMARC [11], coded in MARCXML [7]). Each access copy has also an access rights metadata structure, whose parameters can be chosen in the SECO's interface.

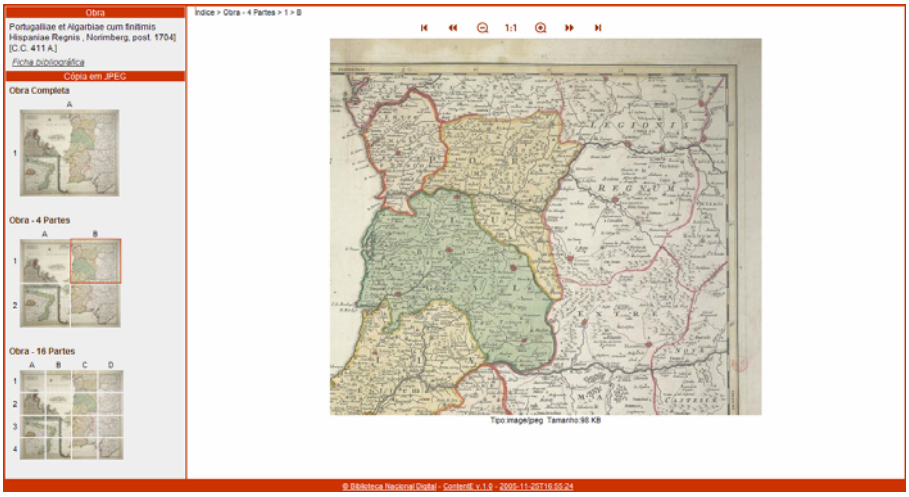


Fig. 5. Example of an access copy of a digitised map structured by ContentE

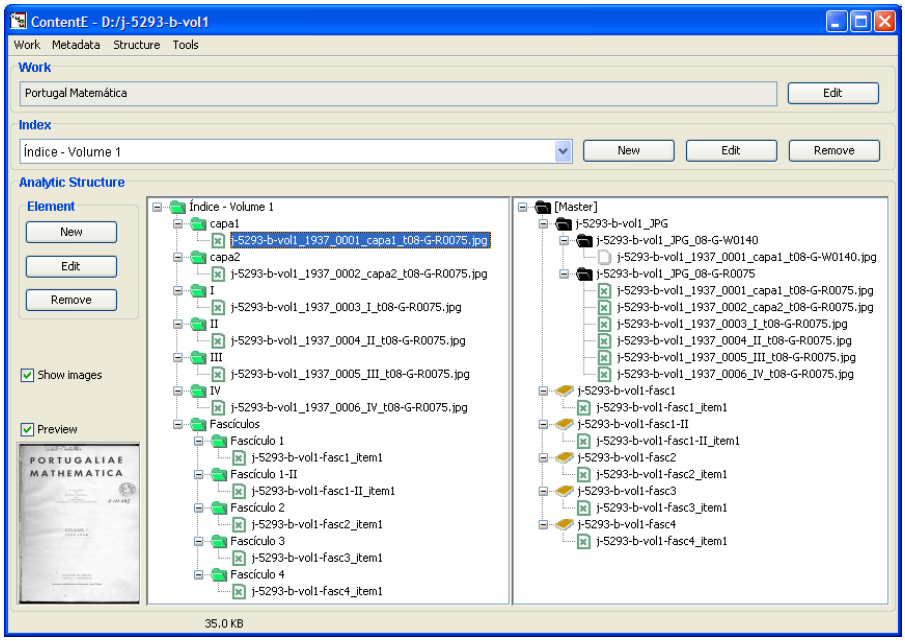


Fig. 6. Example of a session of the ContentE Local Application structuring a digitised journal.

Fig. 5 shows an example of an access copy of a work produced by SECO and ContentE using these procedures. It is a classical example of a map, for which it as asked to be created an index tree of three levels, where the first shows all the map, the second splits it in four parts (all with the same resolution of the image of the upper level), and the third splits it in 16 images. All was done automatically, once configured in SECO. Fig. 6 shows another example of a work being processed using the ContentE Local Application. In this case it is a journal. In cases like this, we want usually detailed indexes. As SECO produces only sequential indexes, manual work is required for the more complex structuring.

5 PURL.PT

The PURL.PT service allows digital or digitalized works to be registered and accessed through unique and persistent URLs. These URL have the syntax "http://purl.pt/xpto", where "xpto" is a simple sequential number that started in 1 and grows up for each new work. These identifiers give access to a "home page" of the work, where a user can see descriptive and technical metadata (with references to the MIME formats of each copy), and have also access to the copies. The URLs for the copies have the syntax "http://purl.pt/xpto/copy", where "copy" is again a sequential number. The value zero is always reserved for the master copy, while the other copies are numbered sequentially without any particular order.

The system has an administration interface, a set of web services to interact with other applications. It works also as a proxy system to allow access control, with the architecture show in Fig. 7.

The works are registered in the PURL.PT service by the submission of a XML file created by ContentE. This file has a METS structure and contains all needed information for register the work and its copies. This file doesn't contain descriptive information about the copies; it only has references for them. Therefore, each copy has its own METS file with the description of its structure.

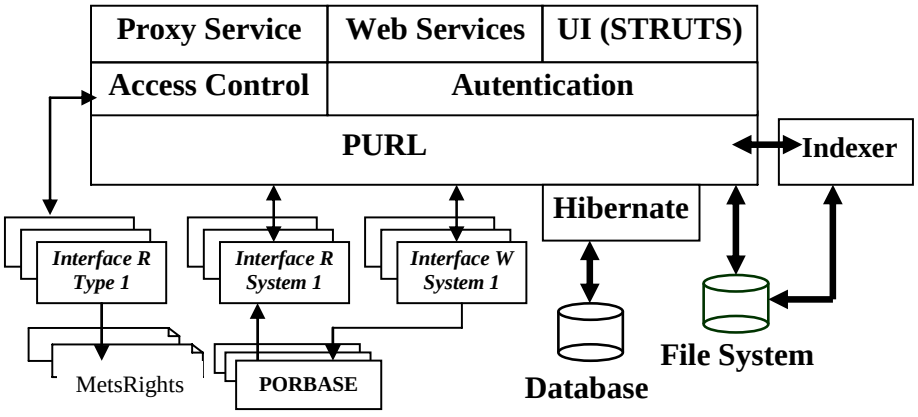


Fig. 7. The architecture of the service PURL.PT

Fig. 8. An example of a rights metadata file

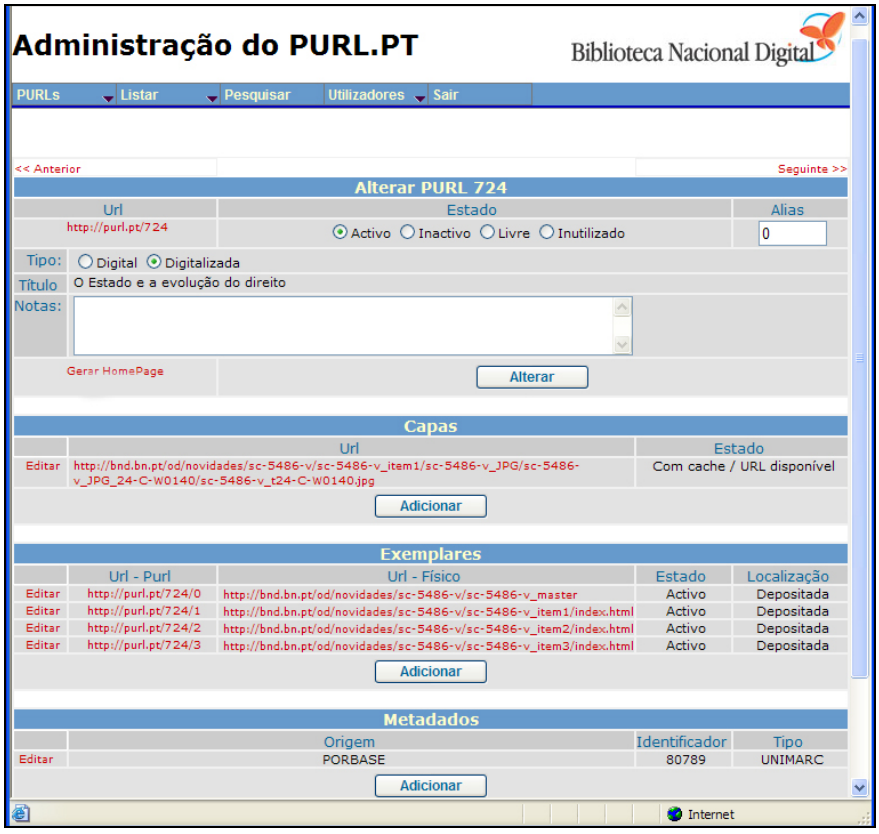


Fig. 9. Administrative interface for the service PURL.PT

The PURL.PT service can process multiple descriptive metadata formats (UNIMARC, Dublin Core and others custom). Through the use of different style sheets (according to the format of the record), a descriptive text is created for the homepage of the work. This page also contains information regarding to the work's several digital items and its properties, as well as for its physical items references. The METS' master copy also enables the creation of a HTML page with technical information about the different MIME formats that can be found in the copies.

Each copy contains a XML rights file, referred through the METS file, from which the PURL.PT extracts the terms for access control. In BND we defined tree types of conditions: private, internal or public. Private items can not be accessed by normal users, which are the case of all the master items; internal copies can only be accessed at the intranet; public copies are accessible to all users, including the Internet. The actual schema for this rights format is available online[1], with a sample shown in the Fig. 8.

In the Fig. 9 we can see an example of the administrative interface of the PURL.PT service, and in Fig. 10 we can see an example of a descriptive page created automatically for a digitised work with one master and three copies for access. It is interesting to see also that due to the interoperability with PORBASE, we can see not only the references to the physical item that was originally digitised, but also the references to other copies existing in other libraries members of PORBASE.

Fig. 10. Descriptive page for a digitised work created automatically by the PURL.PT service

[1] http://schemas.bn.pt/right/v1/rightsv1.xsd

6 Conclusions

In this paper we explained how we harnessed a very complex problem with a strategy based on the development of a processing system integrating several functional blocks by the effective use of XML in the interfaces and transport of data. The overall system is actually in production at the BND, and the experiences had so far make us to believe that it'll make it possible to publish, in a short time and with fair human intervention, the more than 20.000 digitised titles that are still waiting in the queue.

All the code was developed in JAVA, and all the results are available in open-source. We have also plans to continue the developments, especially to reinforce SECO with more image processing features (such as to cut margins and improve the legibility in images), add more schemas to ContentE (such as the DOCBOOK [6] and the Digital Talking Book [2] defined by the DAISY Consortium [5], which will make it possible to process also object with sound, for visual impaired persons). Finally, the PURL.PT service has many other features that were not mentioned in this paper, especially to support browsing in the BND and the publishing of collections and profiles, all supported easily thanks to the global usage of well defined XML schemas. All of this will make it possible to support new services, such as, for example, a user space under development, where users will be able to register and take advantage of new customised services.

References

[1] Adobe Acrobat Reader <http://www.adobe.com/products/acrobat/>
[2] ANSI/NISO Z39.86-2005. Specifications for the Digital Talking Book. ISSN: 1041-5653 <http://www.daisy.org/z3986/2005/z3986-2005.html>
[3] BND. Biblioteca Nacional Digital. <http://bnd.bn.pt>
[4] Borbinha, José; Pedrosa, Gilberto; Penas, João; Gil, João. A gestão de obras digitalizadas na BND. XML: Aplicações e Tecnologias Associadas. 10 e 11 de Fevereiro de 2005, Casa da Torre, Vila Verde, Braga.
[5] DAISY Consortium <http://www.daisy.org/>
[6] DOCBOOK. <http://www.docbook.org/>
[7] MARCXML. MARC 21 XML Schema. <http://www.loc.gov/standards/marcxml/>
[8] METS. Metadada Encoding and Transmission Standard. <http://www.loc.gov/standards/mets>
[9] PORBASE. Base Nacional de Dados Bibliográficos. <http://www.porbase.org>
[10] The Dublin Core Metadata Initiative <http://www.dublincore.org>
[11] UNIMARC <http://www.unimarc.info>

REPOX – Uma Infra-estrutura XML para a Base de Dados Bibliográfica Nacional

José Borbinha[1], Nuno Freire[2]

INESC-ID – Instituto de Engenharia de Sistemas e Computadores, Rua Alves Redol 9, Apartado 13069, 1000-029 Lisboa, Portugal
[1]jlb@ist.utl.pt
[2]Nuno.Freire@bn.pt

Resumo. A PORBASE – Base de Dados Bibliográfica Nacional, é um catálogo colectivo em linha de 150 bibliotecas Portuguesas, incluindo a BN – Biblioteca Nacional, a entidade que tem a responsabilidade da sua coordenação. O sistema integrado de gestão de bibliotecas que suporta a PORBASE armazena os dados num sistema de gestão de base de dados relacional. A PORBASE é uma importante fonte de informação cujo potencial tem sido subaproveitado, pretendendo a BN por isso desenvolver novos serviços. Tal tem-se mostrado no entanto difícil devido ao facto da actual estrutura da base de dados, que está definida para um fim específico nem sempre compatível com os novos serviços que se pretendem desenvolver. O sistema REPOX – Repositório XML da PORBASE pretende ser uma infra-estrutura para resolver esse problema. O objectivo é permitir ter toda a informação da PORBASE num repositório XML, em redor do qual se poderão desenvolver os novos serviços. Outra mais valia é a persistência do historial diário dos dados e suas alterações, que não é possível manter actualmente. Ainda, o REPOX representará assim uma cópia de segurança expressa em XML dos dados da PORBASE, que assim ficarão independentes de qualquer sistema específico. Este artigo descreve a infra-estrutura do sistema REPOX, os resultados e conclusões da sua utilização em ambiente de produção de apoio a uma nova gama de serviços, e ainda os planos de desenvolvimentos futuros.

1 Introdução

A PORBASE – Base de Dados Bibliográfica Nacional [9], é um catálogo colectivo em linha das bibliotecas portuguesas, sendo actualmente a maior base de dados bibliográficos do país, reflectindo as colecções de mais de 150 bibliotecas Portuguesas, incluindo a BN – Biblioteca Nacional [1], a entidade que tem a responsabilidade da sua coordenação. O sistema integrado de gestão de bibliotecas (SIGB) que suporta a PORBASE armazena os dados num sistema de gestão de base de dados relacional (SGBD) da Sybase [11]. É também sobre este SGBD que assenta o catálogo em linha (OPAC - *Online Public Access Catalogue*) que disponibiliza a PORBASE para consulta através da Internet (Fig. 1).

A PORBASE é uma importante fonte de informação cujo potencial tem sido subaproveitado, problema que a BN pretende encarar desenvolvendo novos serviços.

Mas para isso é necessário considerar outra infra-estrutura para os dados da PORBASE. O esquema actual da base de dados suporta um sistema de informação proprietário (sistema HORIZON [2]), e está por isso definida segundo os requisitos tradicionais de suporte às actividades de catalogação desse sistema e de pesquisa por OPAC [19] (Fig. 1). Esse esquema não é assim sempre é compatível com os requisitos dos novos serviços que se pretendem desenvolver, nem será boa prática alterar o esquema pelo risco de conflito potencial que tal implicaria com a empresa responsável pela sua manutenção. Para além disso a possível utilização do mesmo servidor que é usado para catalogação e pesquisa pelo público iria sobrecarregar esse para tarefas que poderá muitas vezes ser perfeitamente feitas em *batch* noutro servidor. Finalmente, a replicação dos dados, codificados num esquema aberto e armazenados numa infra-estrutura objectiva, será uma resposta às questões de segurança e preservação.

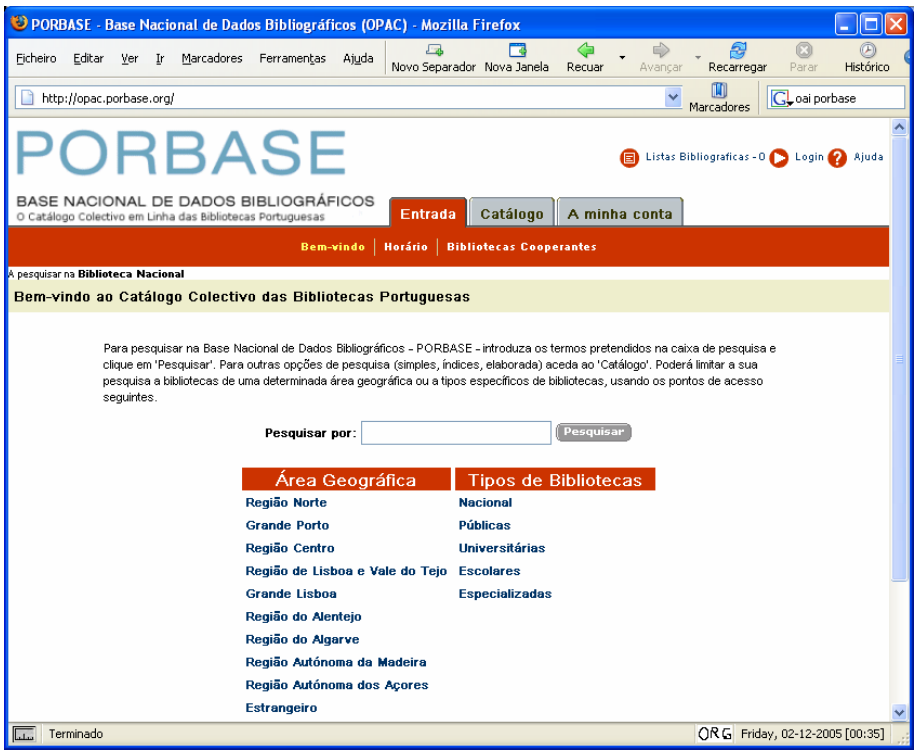


Fig. 1. OPAC da PORBASE.

O objectivo da infra-estrutura REPOX – Repositório XML da PORBASE é assim o de permitir concentrar toda a informação da PORBASE num repositório XML, em torno do qual se podem então planear o desenvolvimento dos novos serviços, tais como de produção de estatísticas, controlo de qualidade, e todo o tipo de serviços úteis de suporte à decisão, explorar novas formas de pesquisa, etc. Nalguns casos pontuais tal poderá ser feito directamente sobre o REPOX, mas o normal é que, de

acordo com os requisitos do serviço em causa, os registos possam ser indexados em índices próprios, importados para bases de dados dedicadas ou sistemas de *data wharehousing*, etc.

Este artigo procede com uma descrição do desenho e arquitectura da solução, seguindo-se uma descrição dos serviços até ao momento já desenvolvidos.

2 Desenho

A infra-estrutura REPOX gere, como primeiro objectivo, o armazenamento em XML de registos da PORBASE, mas o mesmo modelo pode ser igualmente explorado para outros registos da BN. Um registo, na perspectiva REPOX, é uma entidade que se encontra num repositório externo, tem um identificador unívoco nesse repositório, e pode ser representado segundo um esquema XML, tal como ilustrado na Fig. 2. Para o caso da PORBASE, o REPOX gere o armazenamento de duas classes de entidades: os registos bibliográficos e os registos de autoridade.

O REPOX mantém um histórico de todas as versões recolhidas de todos os registos, com a respectiva data, o conteúdo tal como se encontrava nessa data codificado em XML, e ainda uma lista dos eventos que ocorreram sobre esta versão do registo. Um evento pode indicar qual o actor interveniente (um utilizador do repositório externo, ou uma aplicação) e o tipo de acção (criação do registo, alteração, remoção, alteração de registo relacionado, etc.).

Apesar de na fase actual o sistema REPOX estar apenas em funcionamento interno na BN suportando serviços exclusivos da PORBASE, o projecto prevê a existência de vários repositórios externos com dados bibliográficos, sobre autoridades ou mesmo de registos de sistemas de gestão de arquivo (em oposição aos registos documentais da PORBASE). Os registos serão recolhidos periodicamente e guardados num repositório XML, destinando-se a ser posteriormente recuperados e manipulados por serviços externos, indexadores e gestores de colecções (Fig. 3).

Os serviços externos têm acesso ao repositório XML e são notificados quando termina a recolha de um repositório externo, o que despoleta a sua execução. Um exemplo de um serviço já desenvolvido é o sistema QualiCat, que faz o controle de qualidade, com uma periodicidade diária, para os registos bibliográficos da PORBASE segundo os requisitos do formato UNIMARC [14], das Regras Portuguesas de Catalogação, e ainda das especificidades de preenchimento da rede de cooperação PORBASE.

Os indexadores são programas que processam os registos armazenados, preparando-os para serem pesquisados de forma eficiente. No caso dos registos bibliográficos da PORBASE, em formato UNIMARC, estes indexam o título, os autores, os identificadores (ISBN, ISSN, cotas dos exemplares e Número de Depósito Legal), assim como outra informação que seja pertinente para os serviços que num dado momento a necessite.

O REPOX organiza os dados recolhidos em colecções, que não são mutuamente exclusivas, ou seja, um registo no REPOX pode pertencer a várias colecções. A atribuição dos registos às colecções é levada a cabo por gestores de colecção, os quais podem avaliar os registos segundo vários critérios. Estes critérios podem ser

informação de assunto (história, ciências, artes, etc.), o local ou língua de publicação, o ano de nascimento dos autores, a data de publicação, etc.

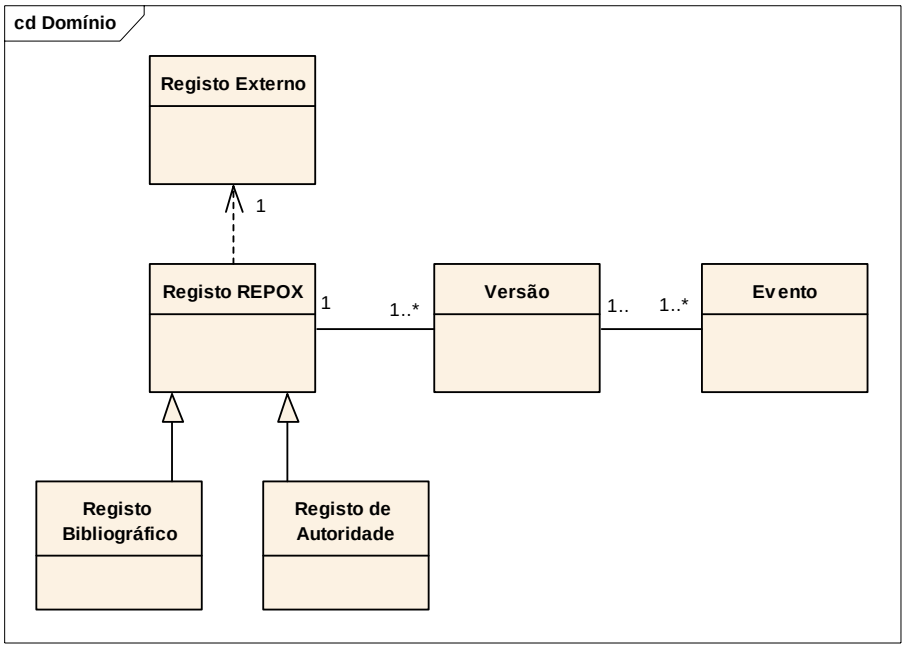


Fig. 2. O conceito de registo na perspectiva do REPOX

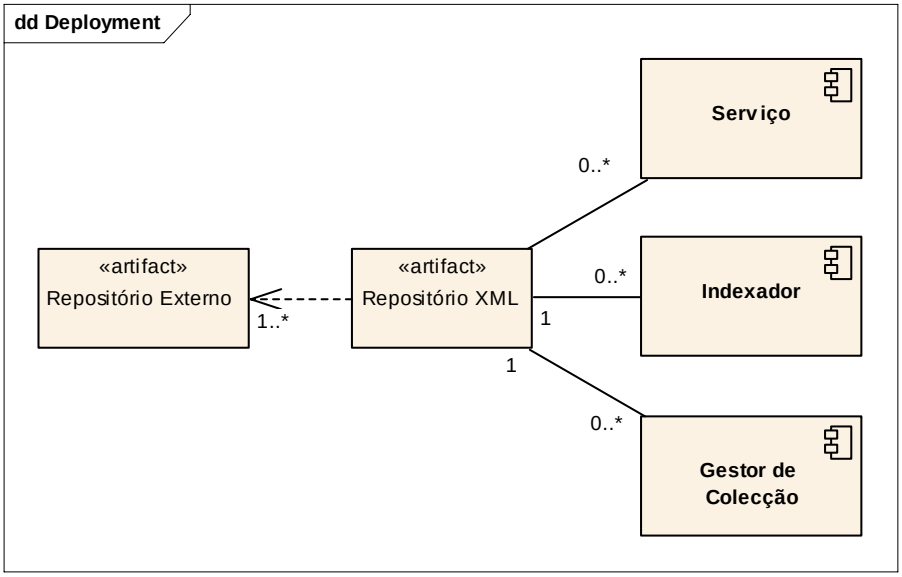


Fig. 3. O repositório XML e principais entidades relacionadas

3 Arquitectura REPOX

A infra-estrutura REPOX é constituída por três componentes principais: o gestor do repositório XML, um SGBD relacional MySql [7] e um servidor de aplicações Tomcat 5.5 [3]. A respectiva arquitectura está representada na Fig. 4.

A versão actual recolhe diariamente os registos bibliográficos e de autoridade da PORBASE. Esta é gerida pelo sistema integrado de gestão de bibliotecas HORIZON, utilizando um SGBD relacional SYBASE [11]. O REPOX gere um serviço de sincronização que recolhe os registos através de uma ligação ao SGBD por JDBC [7]. Cada registo recolhido é guardado num ficheiro XML segundo um esquema definido para o REPOX. Este formato contém uma secção onde são codificadas as várias versões do registo, segundo um esquema próprio para cada tipo de registo. O formato para a troca de registos bibliográficos utilizado quase exclusivamente em Portugal, e consequentemente na PORBASE, é o UNIMARC, sendo por isso utilizado o formato MARCXML [6] para codificar esses registos no REPOX, numa estrutura como se mostra no exemplo da Fig. 5.

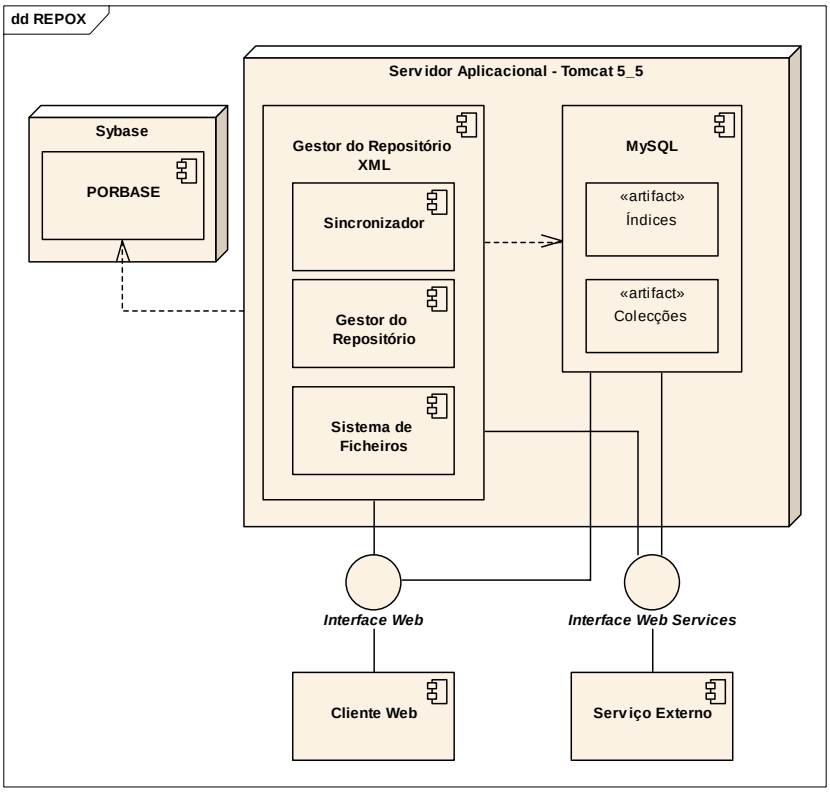
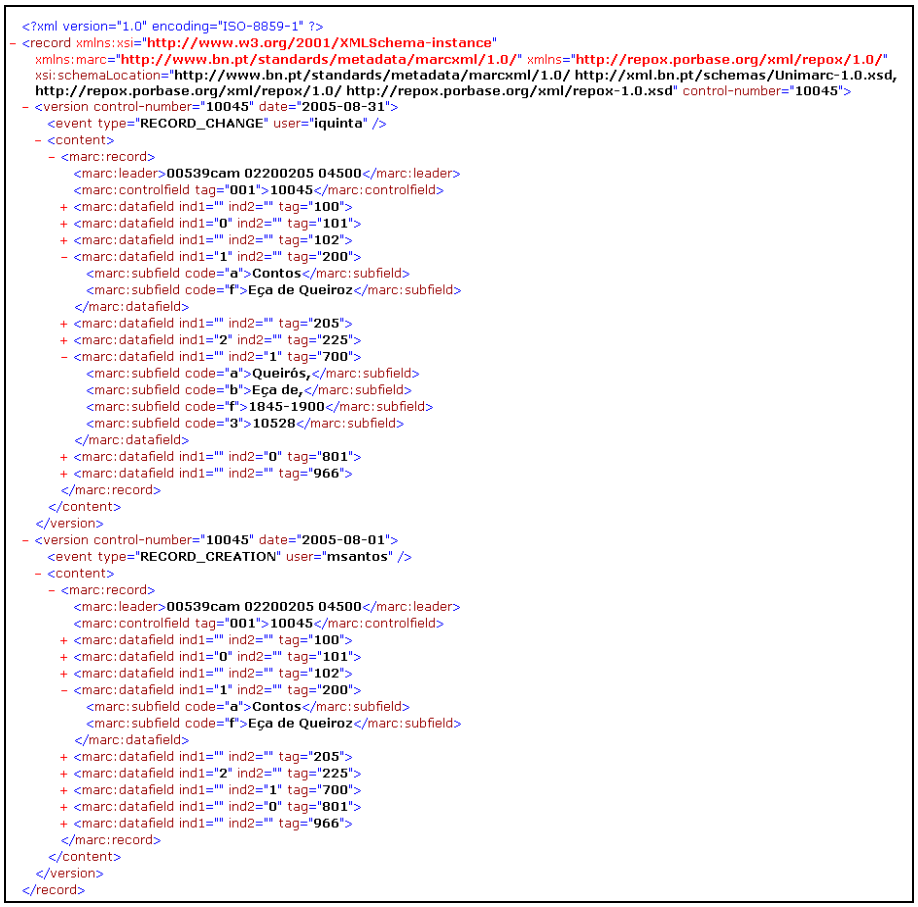


Fig. 4. Os principais componentes do sistema REPOX



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <record xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:marc="http://www.bn.pt/standards/metadata/marcxml/1.0/" xmlns="http://repox.porbase.org/xml/repox/1.0/"
  xsi:schemaLocation="http://www.bn.pt/standards/metadata/marcxml/1.0/ http://xml.bn.pt/schemas/Unimarc-1.0.xsd,
  http://repox.porbase.org/xml/repox/1.0/ http://repox.porbase.org/xml/repox-1.0.xsd" control-number="10045">
 - <version control-number="10045" date="2005-08-31">
    <event type="RECORD_CHANGE" user="iquinta" />
   - <content>
     - <marc:record>
        <marc:leader>00539cam 02200205 04500</marc:leader>
        <marc:controlfield tag="001">10045</marc:controlfield>
       + <marc:datafield ind1="" ind2="" tag="100">
       + <marc:datafield ind1="0" ind2="" tag="101">
       + <marc:datafield ind1="" ind2="" tag="102">
       - <marc:datafield ind1="1" ind2="" tag="200">
          <marc:subfield code="a">Contos</marc:subfield>
          <marc:subfield code="f">Eça de Queiroz</marc:subfield>
        </marc:datafield>
       + <marc:datafield ind1="" ind2="" tag="205">
       + <marc:datafield ind1="2" ind2="" tag="225">
       - <marc:datafield ind1="" ind2="1" tag="700">
          <marc:subfield code="a">Queirós,</marc:subfield>
          <marc:subfield code="b">Eça de,</marc:subfield>
          <marc:subfield code="f">1845-1900</marc:subfield>
          <marc:subfield code="3">10528</marc:subfield>
        </marc:datafield>
       + <marc:datafield ind1="" ind2="0" tag="801">
       + <marc:datafield ind1="" ind2="" tag="966">
        </marc:record>
    </content>
  </version>
 - <version control-number="10045" date="2005-08-01">
    <event type="RECORD_CREATION" user="msantos" />
   - <content>
     - <marc:record>
        <marc:leader>00539cam 02200205 04500</marc:leader>
        <marc:controlfield tag="001">10045</marc:controlfield>
       + <marc:datafield ind1="" ind2="" tag="100">
       + <marc:datafield ind1="0" ind2="" tag="101">
       + <marc:datafield ind1="" ind2="" tag="102">
       - <marc:datafield ind1="1" ind2="" tag="200">
          <marc:subfield code="a">Contos</marc:subfield>
          <marc:subfield code="f">Eça de Queiroz</marc:subfield>
        </marc:datafield>
       + <marc:datafield ind1="" ind2="" tag="205">
       + <marc:datafield ind1="2" ind2="" tag="225">
       + <marc:datafield ind1="" ind2="1" tag="700">
       + <marc:datafield ind1="" ind2="0" tag="801">
       + <marc:datafield ind1="" ind2="" tag="966">
        </marc:record>
    </content>
  </version>
</record>
```


Fig. 5. Um registo bibliográfico em MARCXML

Todos os ficheiros XML do REPOX são guardados no sistema de ficheiros do servidor. Para efeitos de eficiência, é utilizado um SGBD relacional MySql para guardar e indexar alguns dados dos registos com o objectivo de permitir a sua pesquisa. Os indexadores do REPOX processam os registos extraindo e preparando esses dados que sejam relevantes para pesquisa.

Os gestores de colecções utilizam, tal como os indexadores, uma base de dados no mesmo SGBD, na qual são inseridas e indexadas as relações entre os registos e as colecções a que pertencem.

Existem ainda várias ferramentas de gestão do repositório. Estas, através de uma interface por linha de comandos, permitem ao administrador criar, actualizar ou eliminar índices e colecções.

Para acesso ao repositório por serviços externos, o REPOX disponibiliza ainda uma interface por *Web Services* [15], a qual oferece várias funções, tais como:

- Obter a versão mais recente de um registo;
- Obter um registo tal como esse se encontrava em determinada data;
- Obter todo o histórico de um registo;
- Obter uma lista de todos os registos alterados entre duas datas;
- Obter listas de registos através de pesquisas nos índices.

O REPOX disponibiliza ainda uma interface WEB para os utilizadores. Esta permite pesquisar, navegar e consultar registos individualmente, mostrando toda a informação existente, incluindo todas as versões dos registos, os eventos ocorridos e identificação exacta das alterações aos registos entre versões.

O repositório XML da PORBASE conta neste momento com 2,6 milhões de registos (entre registos bibliográficos e de autoridade) correspondendo a cerca de 6 milhões de ficheiros XML que totalizam em tamanho aproximadamente 25 GBytes.

A base de dados relacional que suporta a pesquisa nos dados do repositório contém 21 índices (como o títulos, autores, datas de publicação, editores, ISBN, entre outros). São também mantidas na base de dados 18 colecções de registos bibliográficos e 2 colecções de registos de autoridade. Dependendo da utilização, são frequentemente criados índices e colecções adicionais temporárias. Tal ocorre sempre que, no contexto de alguma actividade da PORBASE, surge a necessidade de processar dados para os quais não existem índices, ou se pretende exportar registos segundo critérios que definem uma colecção virtual adicional.

4 Serviço URN.PORBASE.ORG

Vários serviços da Biblioteca Nacional tiram já proveito eficazmente dos dados do REPOX estarem em XML.

Um desses casos é o serviço URN– Acesso à PORBASE por Identificadores Unívocos [10], que disponibiliza os registos da PORBASE para as bibliotecas portuguesas em vários formatos através de uma interface por http. Na Fig. 7 encontra-se um exemplo da interface para utilizadores deste serviço. O serviço URN tira partido dos índices existentes no REPOX para permitir que os registos sejam obtidos através de vários identificadores (ISBN, ISSN, número de depósito legal, etc.). Os registos são disponibilizados em vários esquemas (UNIMARC, Dublin Core [12]) e em vários formatos de codificação (XML, HTML, texto, etc.). Estes formatos são gerados em tempo real através de transformações XSLT [16] a partir do registo armazenado no REPOX.

Este serviço é acedido principalmente por sistemas de informação de gestão de bibliotecas, especialmente de bibliotecas portuguesas, que importam os registos directamente da PORBASE para as bases de dados locais, facilitando assim o trabalho de catalogação aí levado a cabo.

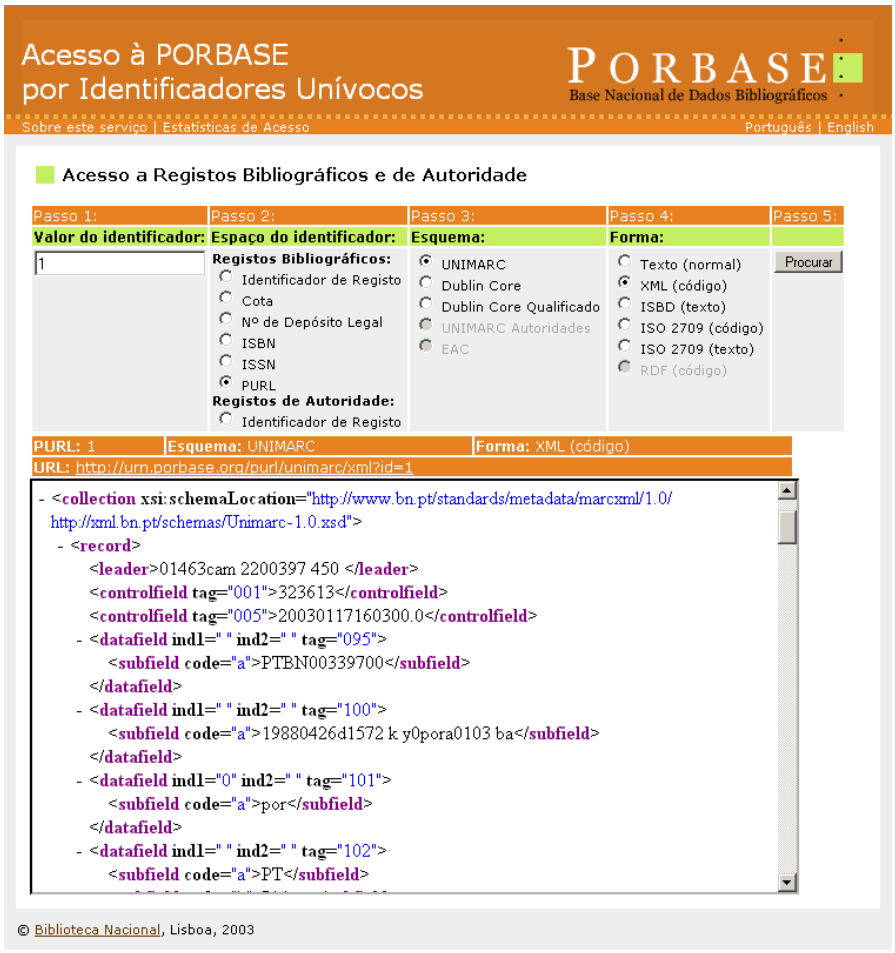


Fig. 6. O serviço URN (http://urn.porbase.org)

5 Serviço OAI.PORBASE.ORG

Um outro serviço suportado pelo REPOX é o serviço OAI-PMH – *Open Archives Iniciative Protocol for Metadata Harvesting* [8].

Este serviço disponibiliza os registos da PORBASE em projectos de cooperação, permitindo que cópias de registos existentes num outro local sejam mantidas actualizadas. É assim por exemplo através deste serviço que o portal da TEL – *The European Library* [13], recolhe regularmente os registos da PORBASE, e os disponibiliza para pesquisa e acesso (Fig. 8).

O servidor OAI-PMH da PORBASE obtém os registos através do REPOX, organizados por colecções que são geridas pelos gestores de colecções. No caso da

colecção para o portal TEL, os registos em MARCXML do REPOX são transformados para o formato de dados da TEL, no perfil para bibliotecas do formato Dublin Core, o DC-Lib [5]. Esta transformação é naturalmente obtida por XSLT.

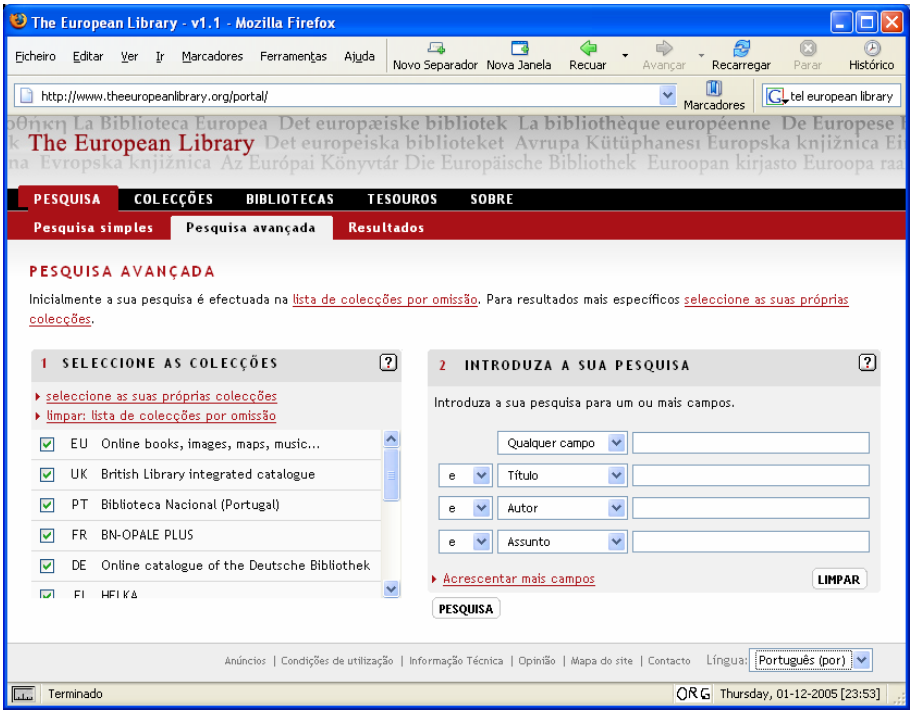


Fig. 7. Serviço TEL, onde a PORBASE está disponível para pesquisa (mostra-se a interface em Português).

Relatório resumido de: 2005-11-25

Ver relatório detalhado

Utilizador	Bibliográficos Criados	Bibliográficos Alterados	Bibliográficos Apagados	Autoridades Criadas	Autoridades Alteradas	Autoridades Apagadas	Itens Criados	Itens Alterados	Itens Apagados	Total
aalves	0	28	0	10	4	0	0	0	0	42
acarol	0	13	0	14	14	0	0	0	0	41
apires	0	18	0	11	7	0	0	0	0	36
asantos	33	1	0	0	0	0	61	0	0	95
barbara	0	44	0	15	3	0	0	25	0	87
bpmp02	0	15	0	3	0	0	30	1	0	49
bpmp03	2	0	0	0	0	0	27	0	0	29
bpmp04	3	0	0	1	0	0	10	0	0	14
bpmp05	0	0	0	0	0	0	4	0	0	4
bpmp06	8	5	0	1	0	0	30	23	0	67
caetano	0	8	0	10	12	0	0	8	0	38
dfontes	0	9	0	2	0	0	0	0	0	11
ecalapa	0	14	0	10	0	0	0	10	0	34
fatima	22	0	0	0	0	0	49	0	0	71
freitas	5	1	0	0	0	0	10	0	0	16
goreti	4	8	0	1	0	0	8	2	0	23
icosta	0	10	0	4	3	0	0	3	0	20
ines	0	73	0	0	1	0	0	1	0	75
ipuga	0	6	0	13	8	0	0	0	0	27

Fig. 8. Exemplo de um relatório de actividade diária na PORBASE criado pelo REPOX.

6 Serviço de Relatórios de Actividade da PORBASE

Finalmente, outro serviço criado baseado na infra-estrutura REPOX é o serviço de relatórios de actividade diária da PORBASE.

Este serviço notifica diariamente os utilizadores do sistema HORIZON de todas as operações sobre os registos, enviando um e-mail contendo um relatório das operações efectuadas nas 24 horas anteriores. A mesma informação fica ainda acessível em linha, num servidor da rede interna, como ilustrado na Fig. 9.

Mensalmente é ainda criado e enviado aos utilizadores e responsáveis de áreas e serviços um relatório consolidado.

7 Conclusão

O sistema REPOX entrou em funcionamento operacional no dia 1 de Outubro de 2005. De imediato melhorou consideravelmente a disponibilidade e o desempenho do acesso aos dados pelos variados serviços da Biblioteca Nacional e das bibliotecas cooperantes da PORBASE. O sistema de gestão de bibliotecas da PORBASE ficou também livre de processar os acessos aos registos por parte de outras bibliotecas e de vários serviços que o sobrecarregavam já de modo bastante significativo.

Para além destes efeitos imediatos, o REPOX abre novas perspectivas. A possibilidade de estabelecer uma ligação ao REPOX através de um URL que identifica o estado do registo numa determinada data, por exemplo, permite a simplificação de relatórios sobre registos da PORBASE que são gerados em outros sistemas. Estes sistemas já não necessitam de guardar localmente cópias dos registos, bastando-lhes guardar uma referência para o registo na data pretendida.

Devido a esta solução, tem também sido possível fazer análise dos registos da PORBASE sobre campos que não estão indexados no sistema HORIZON. É possível descobrir assim com mais celeridade falhas de preenchimento dos registos bibliográficos, bem como determinar as formas de as corrigir automaticamente sem necessidade de intervenção humana.

Um outro factor de extraordinária importância é o facto de desta forma o REPOX representar ainda uma cópia de segurança dos dados da PORBASE, expressa em XML. Neste momento tal representa já um total de cerca de 2,6 milhões de registos, que assim estão preservados, independentes de qualquer software ou hardware específico.

A utilização de tecnologia XML no repositório permite ainda que formatos de registos heterogéneos sejam geridos pelo REPOX de forma relativamente transparente. Isto permite a sua fácil adaptação a outros esquemas de dados. O próximo repositório de dados a ser assim integrado no REPOX será o do Arquivo da Cultura Portuguesa Contemporânea (http://acpc.bn.pt), cujos metadados estão representados num outro sistema de informação, num esquema definido segundo as ISAD(G) [17]. Neste caso os registos deverão vir a ser codificados segundo o esquema EAD [18].

Finalmente, estão em fase de análise de requisitos e desenho de serviços de estatísticas mais avançados da PORBASE (e do REPOX em geral), de controlo de

qualidade (com por exemplo validações sintácticas e semânticas de acordo com as regras do UNIMARC), bem como o estudo de soluções de *data warehouse* para suportar serviços gerais de *data mining* e especificamente de suporte à decisão.

Referências

[1] Biblioteca Nacional: http://www.bn.pt
[2] Horizon: http://www.novabase.pt/ConteudosHTML/Horizon.pdf
[3] Jakarta Tomcat: http://tomcat.apache.org/
[4] JDBC: http://java.sun.com/products/jdbc/
[5] Library Application Profile: http://dublincore.org/documents/library-application-profile/
[6] MARCXML. MARC 21 XML Schema: http://www.loc.gov/standards/marcxml/
[7] MySQL: http://www.mysql.com
[8] OAI-PMH. Open Archives Initiative – Protocol for Metadata Harvesting http://www.openarchives.org/
[9] PORBASE – Base Nacional de Dados Bibliográficos: http://www.porbase.org
[10] *Serviço URN:* http://urn.porbase.org
[11] SyBase: http://www.sybase.pt/gvsview/gvs/sybase-pt/home/index.html
[12] The Dublin Core Metadata Initiative: http://dublincore.org
[13] The European Library: http://www.theeuropeanlibrary.org/
[14] UNIMARC http://www.unimarc.info
[15] *Web Services*: http://www.w3.org/2002/ws/
[16] XSLT: http://www.w3.org/TR/xslt
[17] ISAD(G) - General International Standard Archival Description: http://www.ica.org/biblio/cds/isad_g_2e.pdf
[18] EAD – Encoded Archival Description: http://www.loc.gov/ead/
[19] IFLA Guidelines for Online Public Access Catalogue (OPAC) Displays - Final Report May 2005. IFLA Series on Bibliographic Control; vol. 27. Saur, 2005. ISBN 3-598-24276-X (rascunho disponível em http://www.ifla.org/VII/s13/guide/opacguide03.pdf)

CRiB: A service oriented architecture for digital preservation outsourcing

Miguel Ferreira[1], Ana Alice Baptista[1], José Carlos Ramalho[2]

[1] Department of Information Systems,
University of Minho, Guimarães, Portugal
{mferreira, analice}@dsi.uminho.pt
[2] Department of Informatics,
University of Minho, Braga, Portugal
jcr@di.uminho.pt

Abstract. This paper identifies some of the most prominent issues present in today's digital repository systems, which hinder the long-term preservation of digital materials. In order to address some of those issues, we propose the CRiB system, a service oriented architecture (SOA) supported by Web services' technology, which will enable institutions to outsource part of the functionality that is necessary to carry out effective long-term digital preservation. The proposed system delivers a set of services that client applications will be able to invoke in order to perform complex format migrations; evaluate the outcome of those migrations according to multiple criteria (e.g. data loss and performance); and obtain detailed migration reports for documenting the preservation intervention.

1 Introduction

The number of archives and libraries responsible for managing and safeguarding large collections of digital materials is growing at a startling rate [1]. Several reasons can be outlined which may explain this phenomenon: most information items created today are crafted with the help of digital authoring tools; the physical space required to store those items is nearly insignificant when compared with the requirements for storing conventional analogue-based materials and the possibility of disseminating such content over the Internet entitles these institutions with a whole new branch of business opportunities. Moreover, a large part of such materials can not be adequately represented in conventional analogue media like paper or microfilm (e.g. 3D model, Web page). As a result, classic techniques for preserving information can not be applied in the digital domain.

In addition, digital archiving technology is now affordable to most institutions. Several products have been developed which are both reliable and free of charge (e.g. DSpace [2, 3], Eprints [4], Fedora [5], Greenstone [6]). Most of these products incorporate, off-the-shelf, an assortment of standards developed mostly by the library and archival communities which promote trust and facilitate interoperability between these systems. Among these are the Open Archival Information System Reference

This work was supported by the FCT under the grant SFRH/BD/17334/2004.

Model (OAIS) [7, 8]; standards for describing and structuring information items (e.g. Dublin Core [9], EAD [10], MARC [11], METS [12]); protocols for disseminating metadata (e.g. OAI-PMH [13]) and standards to produce identifiers that are more reliable and persistent than the traditional URL (e.g. CNRI Handle System [14], PURL [15]).

Although current repository software[1] performs a remarkable job at storing, managing and disseminating digital materials, they are not truly capable of assuring the long-term preservation of those materials. The key problem in the design of those systems is that the period of time that the materials are expected to be interpreted is much longer than the lifetime of individual storage media, hardware and software components, as well as the formats in which the information is encoded [16]. As hardware and software turn obsolete, digital materials become prisoners of their own encodings.

In this context, digital preservation is defined as the set of processes and activities that ensure the continued access to information and all kinds of cultural heritage existing in digital formats [17]. A digital object is "(…) an information object, of any type of information or any format, that is expressed in digital form" [18]. Text documents, digital photographs, databases, virtual reality models, Web pages and computer games are just a few examples of digital objects.

This paper introduces a Service Oriented Architecture (SOA) [19] which will enable institutions, as well as individuals, to preserve collections of digital materials without having to go through complex development projects in order to implement all the necessary functionality. The proposed system will deliver a set of services that institutions will be able to invoke in order to convert their digital materials from near obsolete formats to newer encodings that most users will be able to interpret. The proposed system will also be capable of providing suggestions of best suitable preservation alternatives by taking into consideration the individual requirements of each client institution.

This paper is organised as follows: section 2 provides background on the most significant migration-based preservation strategies; section 3 outlines a few of the most prominent issues in today's digital repository software; section 4 describes the proposed system; and section 4 summarises the ongoing research.

2 Format migration as a preservation strategy

Over the last decade, the research community has come up with a considerable number of strategies aiming at solving the problem of digital preservation and technological obsolescence. Among these is migration[2].

Migration consists of a "(…) set of organized tasks designed to achieve the periodic transfer of digital materials from one hardware/software configuration to another or from one generation of computer technology to a subsequent generation." [20]. Contrary to other preservation strategies (e.g. emulation, encapsulation, etc.), migration techniques do not attempt to preserve digital objects in their original

[1] Also known as Content Management Systems (CMS).
[2] Also known as conversion.

formats. Alternatively, they intentionally transform objects from near obsolete formats into up-to-date encodings that most users are able to interpret on their personal computers.

The main disadvantage of this approach is that when a digital object is migrated, there is a high probability that of some of its inner properties will not be correctly transferred to the target format (i.e. some data loss is expected to take place). The reason for this is twofold: there may be structural incompatibilities between the source and the target formats or the converter may be faulty and incapable of performing its tasks appropriately. Nevertheless, migration is by far the most widely used preservation strategy and the only one that has actually worked to date [21].

The most advanced endeavours in the field of migration are based on networks of conversion services [22-26]. In such strategies, a set of well known protocols, such as the ones associated with Web Services technology [27], are used to support the discovery and invocation of procedures capable of carrying out format migrations.

This type of migration entails several advantages over more traditional solutions: the use of Web Services hides the complexity of the conversion software that is being used underneath and promotes interoperability by cloaking the peculiarities of its supporting platform; the development of redundant services insures that the network remains functional during situations of partial break down, while at the same time, facilitates distribution of the workload; ultimately, the possibility of having multiple migration paths enables this solution to cope with the gradual disappearing of converters. In addition, this approach is compatible with several variants of migration, such as migration on-request [28] and normalisation [18, 21, 29-33].

3. Missing elements in today's digital repository systems

Although advances in digital repository software have come a long way there are still a reasonable amount of issues regarding the long-term preservation of digital materials which demand full attention from the R&D community. Some of these issues are outlined next.

Limited preservation functionality. Digital repositories perform an outstanding job ingesting, storing and disseminating materials over the Internet. However, most digital repository systems are not yet capable of effectively performing long-term preservation of those materials. A few repository systems already incorporate some preservation functionality, but in most cases this is restricted to the definition of ingest policies such as the limitation of accepted formats or the normalisation of received objects to formats more suitable for preservation [32-34]. However, when formats in the data store become obsolete, an assortment of more sophisticated techniques are usually required. In such cases, it is necessary to develop specialized migration or emulation tools which often require significant human intervention in order to be fully exploited [35].

Authenticity. Another important issue has to do with authenticity. Authenticity is defined as the quality of a digital object being what it purports to be [29, 36-41]. In a

preservation environment, where tampering is admissible as a form of preservation (e.g. when migration strategies are in place), the issue of authenticity assumes even greater importance. Although most requirements for ensuring authenticity and trustworthiness in a preservation environment have already been identified [37, 40-43], most digital repository systems do not yet implement all the necessary functionality to support them. Technical approaches for assuring authenticity involve maintaining, within the archival system, detailed information about the provenance of the digital object, the context in which it was created, the identification of all agents in the chain of custody and the continuous logging of activities that take place within archive which may affect the preserved object in a significant way, e.g. a migration, updates on the metadata, object accesses, etc. [36, 42]. The management of all these metadata combined with the preservation of the bit-wise original object constitute a handful of mechanisms for ensuring authenticity [41].

Cost management. Effective digital preservation is not cheap. It requires investment in a robust technological infrastructure capable of coping with preservation issues at all possible levels failure. At the physical level, storage media is expected to degrade with time, hardware is subject to failures and the network infrastructure may loose its connectivity. Replication of these components reduces the risk of a single point of failure [16]. At the logical level, software components are also expected to become obsolete as hardware and operating systems are replaced by newer versions. Although the bits in which data are encoded may still be accessible, if the necessary decoding software is no longer available the information is bound to become prisoner of its own encodings [16]. There are also concerns at the social level. Operators are expected to commit errors. Functionality which limits the number of irrecoverable human errors is also an important requirement for digital repository software [16].

Furthermore, preservation requires planning and pre-emptive decision making. Monitoring the external environment for new standards or product updates is quintessential to avoid unexpected changes in the technology market [44]. Additionally, preservation activities should be automated as much as possible in order to reduce costs and liberate human resources. Examples of such functionality could be: an autonomous service that monitors the Web and detects changes in the general public behaviour towards a given format or a repository system that automatically updates formats in its data store whenever a new version of format is released.

4 The CRiB system

This paper proposes a service oriented architecture (SOA) [19] to address the array of issues outlined in the previous section that are currently hindering digital repository software from carrying out effective digital preservation. The proposed system, recently named CRiB[3], will provide a set of Web methods that will entitle client applications to carry out the following activities:

a) Perform complex format migrations;

[3] CRiB stands for Conversion and Recommendation of Digital Object Formats.

b) Determine the amount of data loss resulting from those migrations;
c) Document preservation interventions;
d) Obtain suggestions of migration alternatives to adequately preserve collections of digital objects.

The general architecture of the proposed system is depicted in Fig. 1. The application layer illustrates client applications that may take advantage of the services provided by the CRiB system. Examples of such applications are: digital repository systems (e.g. DSpace, Fedora or Eprints) and custom applications developed by individual users.

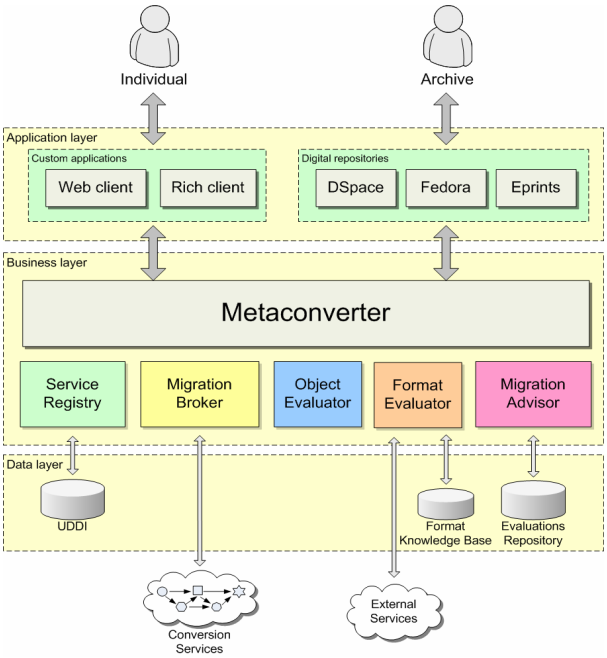


Fig. 1 - Overview of the CRiB architecture.

The middle layer illustrates the whole set of components that constitute the actual CRiB system. The Metaconverter acts as the mediator between client applications and the rest of the system. Furthermore, it is responsible for generating and coordinating all the messages within the system that are necessary to adequately carry out its activities. The Service Registry comprises information to support the discovery of conversion services. The Migration Broker handles invocations to local and remote conversion services. The Object Evaluator is responsible for detecting any loss of information that may take place during the migration process. The Format Evaluator provides information about the current status of digital formats. Finally, the Migration Advisor combines all that information to generate suggestions of migration alternatives that maximise users' satisfaction.

The data layer outlines the sources of information that support the CRiB system (e.g. UDDI server, Format Knowledge Base, Evaluations' Repository). Some external sources of information may also be used, such as remote conversion services or live sources of information, such as Google's own set of Web services.

A possible use scenario where the CRiB system may play an important role is described next:

A large company decides that all technical reports produced in the course of its activity should be available to every employee at the distance of a mouse-click. For that reason, a person was hired to set up and maintain a digital repository system for safekeeping and providing access to those reports. At that time, all existing technical reports were produced using Microsoft Word 95 but, as time went by, new versions of Word begun to be exploited within the company. At the same time, some employees, more fond of the whole open-source movement, preferred to use OpenOffice to produce their technical reports. As a consequence, the number of formats available in the repository became so heterogeneous that barely anyone in the company was able to read a single project's collection of technical reports without having to install additional software in their personal computers. Meanwhile, Microsoft announces that the next version of Office will not support Word 95.

The person responsible for managing the digital repository decides that something had to be done, so he launches an application that is capable of communicating with the CRiB system. First, he informs the CRiB of the format that he soon expects to become obsolete – i.e. the Word 95. The CRiB system responds with a list of criteria that it is capable of evaluating for assessing the quality of the preservation intervention. The manager is expected to weight, according to what he feels is more important for the company, all the criteria that the CRiB has provided. Among these criteria are items such as: textual content preservation, layout, cost, conversion throughput (Kb/s), etc.

The manager decides that the textual content and the page layout are very important items and should be preserved at all cost. After informing the CRiB of the requirements, the system responds with a list of possible formats to which Word 95 files could be converted to. Among these are PDF, Word 2003 and OpenOffice 2, being PDF the top choice suggested by the system.

The manager decides to follow the system's suggestion and requests a list of possible migration services. Based on the weights previously assigned by the manager, the system suggests a rather expensive conversion service but which is able to perform very high quality migrations from Word 95 to PDF. The manager sends its Word 95 files to the CRiB system and the migration process is initiated.

After each conversion the manager receives a PDF version of the technical report and a metadata record that describes the preservation intervention. In that metadata record is included information such as: a description of the conversion services involved in the migration, the date and time of the conversion and other interesting information such as the list of properties of the original Word file that were not properly preserved when converted to PDF. He then uses these reports do document the preservation intervention and validate the conversion.

After running the same procedure for all other formats available in the repository, the manager realises that PDF is often suggested as the preferred format. He then

decides to write a ingest policy stating that all technical reports should be converted to PDF before entering the repository.

The following sections describe the inner workings of the components that constitute the CRiB system.

4.1 Service Registry

The Service Registry is responsible for managing all the metadata, necessary to support the discovery of conversion services. Additionally it facilitates the calculation of composite migrations (i.e. conversions that involve more than one migration service). The metadata elements used within this component are based on the Universal Description, Discovery and Integration (UDDI) standard [45]. The UDDI outlines three basic entities to describe Web services: a service entity that contains information about the service it self (e.g. name, description, etc.); a business entity to describe the producer/developer of the service (e.g. name, description and contacts); and a set of binding templates which comprise information about how the service may be invoked by a client application (e.g. URL).

In order to facilitate the identification of composite migrations, two new metadata elements were introduced, i.e. the source and target formats. Possible values for these elements are obtained from a controlled vocabulary – the PRONOM Registry, an initiative from the National Archives of the UK which aims at building a registry of information about every existing file format [46].

A third metadata element is included in the service description: the cost. The cost refers to the fee that each client is expected to pay in order to use the conversion service. The rationale for this is to stimulate the development of new conversion services which may be published and sold on CRiB platform. Although this economic model is probably too simplistic to be put into practice it constitutes an important tool for assessing how preservation costs may influence decisions in favour, or against, certain migration alternatives (see Migration Advisor).

4.2 Migration Broker

The Migration Broker is responsible for generating and coordinating all SOAP messages, necessary to correctly perform object migrations. In practice, this component will make sure that composite conversions are carried out atomically from the CRiB's point of view.

Additionally, this component is responsible for measuring the performance of each migration service. Performance will be measured according to the following criteria: availability [47], stability [48], scalability [49-51] and throughput [52]. Afterwards, these evaluations will be forwarded to the Migration Advisor in order to be archived. Later on, they will serve as a basis for ranking migration alternatives (see Migration Advisor).

4.3 Object Evaluator

The Object Evaluator is in charge of judging the quality of the migration's outcome. It accomplishes this by comparing the objects submitted to migration with its converted counterparts. Evaluations will be performed according to multiple criteria. These criteria, also known as significant properties, constitute the set of attributes of an object that should be maintained intact during a preservation intervention [53]. They constitute the range of attributes that characterise an object as a unique intellectual entity, independently of the encoding in which it is being represented. The Bible for example, may exist in many different formats and media, e.g. ASCII text, PDF, written on paper or carved on stone, and still be regarded as the Holy Bible. Considering text documents as an example, some relevant significant properties could be: the textual content, the page size, the number of pages, the graphical layout, the number of characters, the order of those characters, the font type and size, etc.

Work is underway as to produce a general taxonomy of significant properties for various classes of digital objects. A good foundation for this, is the work developed by Rauch and Rauber [54-56]. They have conducted a series of workshops from which they have devised a practical list of significant properties for text documents, audio and video objects.

The evaluations performed by the Object Evaluator will be returned to the client user and stored in the Evaluations Repository. The report sent to the client userwill follow the structure of the Event Entity included in the PREMIS Data Dictionary [41], i.e. a dictionary of metadata elements specifically designed for documenting preservation activities within an archival environment. The Event Entity includes elements for describing the type of event (e.g. Migration), the date and time of its occurrence, the agent that carried out the event and detailed information about the outcome of the event (e.g. the amount of data loss that resulted from the migration).

4.4 Format Evaluator

The Format Evaluator provides information about the current status of digital formats. This information will enable the Migration Advisor to determine the formats to which a given object should be converted to. Examples of such information are: the format's market share, its level of support, if it has an open specification, etc. The Format Evaluator will be able to determine this information by questioning the Format Knowledge Base, i.e. a data store of known facts about digital formats; or resorting to external sources of information such as the PRONOM Registry or Google's Web services.

4.5 Migration Advisor

The Migration Advisor is responsible for generating suggestions of migration alternatives. It accomplishes this by confronting the preservation requirements outlined by the user, with the accumulated knowledge about the performance of each individual conversion service. The quality of each migration service will be measured

according to multiple criteria: expected data loss, current status of involved formats and computational performance. Using this information, the Migration Advisor will be able to rank all possible alternatives and produce an appropriate suggestion for the invoking client.

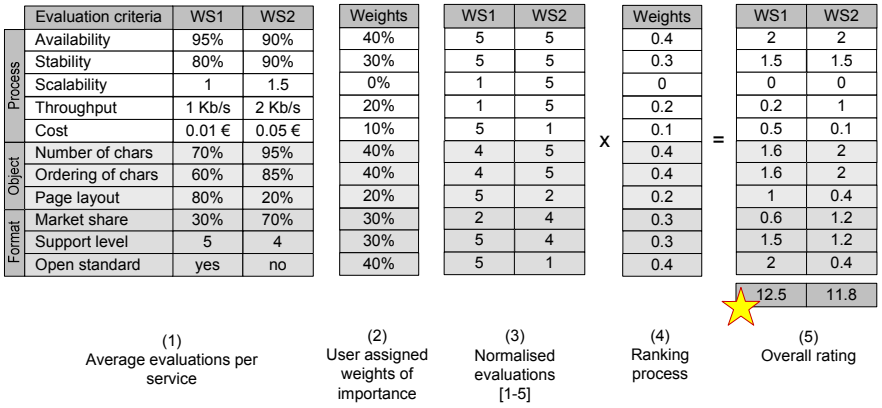


Fig. 2 – Steps involved in the process of ranking migration alternatives. WS1 and WS2 represent two different conversion services.

The ranking process of the alternatives is based on the work developed by Rauch and Rauber [54-57]. They have built a framework which allows users to evaluate, compare and select preservation alternatives according to their individual requirements [54-56]. The process of ranking the alternatives is composed by the following steps (Fig. 2):

1. It begins with the assembly of the evaluation criteria which will be used for assessing the quality of each migration alternative. This set of criteria is regarded as the evaluation taxonomy. All of these criteria fall into one of the following top level categories: object characteristics (handled by the Object Evaluator), format characteristics (handled by the Format Evaluator) and process characteristics (handled by the Migration Broker). For each of the evaluation criteria present in the taxonomy, the system will devise an average performance rating based on previous logged evaluations (Fig. 2, step 1).
2. In order to obtain an adequate suggestion, the user will have to inform the system about which format he/she wishes to migrate from. For that format, the Migration Advisor will return the collection of criteria that it is capable of evaluating (i.e. the evaluation taxonomy). The user manifests its preservation preferences by assigning weights to all the criteria included in the collection. For example, a user might feel that preserving both textual content and graphic layout of a document is fundamental for the success of the preservation intervention. He/she might not care about how much it will cost in terms of money or execution time as long as those characteristics are well preserved. Another user, with a more limited budged, might give up on the graphic layout in order to reduce preservation costs (Fig. 2, step 2).

3. After receiving the weighted list of criteria the Advisor will normalise the average performance of each alternative to a scale of 0 to 5. Different scales are also possible, e.g. 0-100. The important factor here is that all evaluations are converted to comparable values (Fig. 2, step 3).
4. The ranking process works by multiplying the user assigned weights by the normalised values of each evaluation criterion (Fig. 2, step 4).
5. In the end, an overall rating is calculated for each of the migration alternatives (WS1 and WS2 in Fig. 2) by summing up all individual ratings (Fig. 2, step 5).

5 Conclusions

This paper begins by identifying some of the issues that are currently hindering digital repository systems from performing effective long-term preservation of digital materials. To address those issues, we propose a system supported by Web services' technology which enables client institutions to carry out the following activities:

1. Convert digital objects from near obsolete formats to up-to-date encodings that most users will be able to interpret;
2. Evaluate the outcome of a migration by comparing the original digital object with its converted counterpart and identifying the significant properties that have not been adequately preserved;
3. Obtain migration reports in appropriate forms for inclusion in the preservation metadata of the migrated object;
4. Request suggestions of best suitable migration alternatives taking into consideration the preservation requirements of the client institution;

The proposed system addresses the problem of limited preservation functionality in today's digital repository systems by enabling any client application capable of invoking Web services to perform complex format migrations. It addresses authenticity by delivering detailed migration reports that fully document the preservation intervention. It reduces preservation costs by providing suggestion mechanisms that automate preservation activities such as planning while at the same time, facilitating other preservation processes such as documentation and process evaluation.

Parallel contributions are also expected from this research. Developers will have the possibility of publishing and selling their conversion applications. Conversion software published on the CRiB platform will be automatically compared and benchmarked according to multiple quality criteria.

Additionally, sharing experiences of practical use of recently created metadata schemas such as the PREMIS Data Dictionary [41] may contribute to increase its adoption, while the same time, push creators to improve future versions and accelerate the development of XML bindings.

The project presented in this paper is now entering its first stages of development. A group of students is currently working on the development of several migration services for text documents and digital images. Future work will be centred in the construction of a complete prototype of the proposed architecture for proof-of-concept.

References

1. Brody, T., *Analysis*, from http://archives.eprints.org/index.php?action=analysis.
2. Smith, M. *DSpace: An Institutional Repository from the MIT Libraries and Hewlett Packard Laboratories*. in *6th European Conference on Research and Advanced Technology for Digital Libraries*. 2002. Roma, Italy: Springer-Verlag.
3. Hewlett-Packard Company and MIT Libraries, *DSpace Web site*, from http://dspace.org.
4. *EPrints Web site*, from http://www.eprints.org/.
5. University of Virginia and Cornell University, *Fedora Web site*, from http://www.fedora.info/.
6. New Zealand Digital Library Project, *Greenstone Web site*, from http://www.greenstone.org.
7. Consultative Committee for Space Data Systems, *Reference Model for an Open Archival Information System (OAIS) - Blue Book*. 2002, Washington: National Aeronautics and Space Administration.
8. Lavoie, B.F., *The Open Archival Information System Reference Model: Introductory Guide*. 2004, Digital Preservation Coalition: Dublin, USA.
9. Dublin Core Metadata Initiative, *Dublin Core Metadata Initiative*, from http://dublincore.org/.
10. Library of Congress, *EAD - Encoded Archival Description*, Library of Congress, from http://www.loc.gov/ead/.
11. Library of Congress, *MARC Standards*, from http://www.loc.gov/marc/.
12. Library of Congress, *METS - Metadata Encoding & Transmission Standard*, from http://www.loc.gov/standards/mets/.
13. Open Archives Initiative, *The Open Archives Initiative Protocol for Metadata Harvesting*. 2002, from http://www.openarchives.org/OAI/openarchivesprotocol.html.
14. Corporation for National Research Initiatives, *CNRI Handle System*, Corporation for National Research Initiatives, from http://www.handle.net/.
15. Online Computer Library Center, *PURL - Persistent Uniform Resource Locator*, from http://purl.oclc.org/.
16. Rosenthal, D.S.H., et al., *Requirements for Digital Preservation Systems.* D-Lib Magazine, 2005. **11**(11).
17. Webb, C., *Guidelines for the Preservation of Digital Heritage*. 2003, United Nations Educational Scientific and Cultural Organization - Information Society Division.
18. Thibodeau, K. *Overview of Technological Approaches to Digital Preservation and Challenges in Coming Years*. in *The State of Digital Preservation: An International Perspective*. 2002. Washington D.C.: Documentation Abstracts, Inc. - Institutes for Information Science.
19. OASIS SOA Reference Model TC, *OASIS Reference Model for Service Oriented Architectures (Working Draft 10)*. 2005: OASIS.
20. Task Force on Archiving of Digital Information, Commission on Preservation and Access, and Research Libraries Group, *Preserving digital information: report of the Task Force on Archiving of Digital Information*. 1996, Washington, D.C.: Commission on Preservation and Access. 59.
21. Lee, K.-H., et al., *The State of the Art and Practice in Digital Preservation.* Journal of Research of the National Institute of Standards and Technology, 2002. **107**(1): p. 93-106.
22. Ockerbloom, J.M., *Mediating Among Diverse Data Formats*, in *School of Computer Science*. 1998, Carnegie Mellon University: Pittsburg. p. 164, from
23. Walker, F.L. and G.R. Thoma. *A Web-Based Paradigm for File Migration*. in *IS&T's 2004 Archiving Conference*. 2004. San Antonio, Texas, USA.
24. Hunter, J. and S. Choudhury. *A Semi-Automated Digital Preservation System based on Semantic Web Services*. in *Joint ACM/IEEE Conference on Digital Libraries (JCDL'04)*. 2004: ACM.
25. Hunter, J. and S. Choudhury, *Preservation webservices Architecture for Newmedia and Interactive Collections (PANIC)*. 2005, from http://metadata.net/newmedia/.
26. Wing, J.M. and J. Ockerbloom, *Respectful Type Converters.* IEEE Transactions on Software Engeneering, 2000. **26**(7).
27. Graham, S., et al., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*. 2002: Sams Publishing.
28. Mellor, P., P. Wheatley, and D.M. Sergeant. *Migration on Request, a Practical Technique for Preservation*. in *ECDL '02: 6th European Conference on Research and Advanced Technology for Digital Libraries*. 2002. London, UK: Springer-Verlag.
29. Hofman, H. *How to keep digital records understandable and usable through time?* in *Long-Term Preservation of Electronic Records*. 2001. Paris, France.

30. Heslop, H., S. Davis, and A. Wilson, *An Approach to the Preservation of Digital Records*. 2002, National Archives of Australia: Camberra, Australia, from
31. Howel, A.G., *Preserving Digital Information: Challenges and Solutions*. 2004, Cooperative Action by Victorian Academic Libraries, Victorian university libraries, State Library of Victoria.
32. Hodge, G. and E. Frangakis, *Digital Preservation and Permanent Access to Scientific Information: The State of the Practice*. 2004, International Council for Scientific and Technical Information & CENDI.
33. Hedstrom, M., *Digital Preservation: A time bomb for digital libraries.* Computers and the Humanities, 1998. **31**: p. 189-202.
34. Han, Y., *Digital content management: the search for a content management system.* Library Hi Tech, 2004. **22**(4): p. 355-365.
35. Ross, S. and M. Hedstrom, *Preservation research and sustainable digital libraries.* Springer-Verlag, 2005.
36. Lavoie, B. and R. Gartner, *Technology Watch Report - Preservation Metadata*. 2005, Online Computer Library Center Inc., Oxford University Library Services and Digital Preservation Coalition.
37. MacNeil, H., et al., *Authenticity Task Force Report*. 2001, InterPARES Project: Vancouver, Canada.
38. Millar, L., *Authenticity of electronic records: a report prepared for UNESCO and the International Council on Archives*. 2004, International Council on Archives: London, UK.
39. Lynch, C., *Authenticity and Integrity in the Digital Environment: An Exploratory Analysis of the Central Role of Trust*, in *Authenticity in a Digital Environment*. 2000, Council on Library and Information Resources: Washington, DC.
40. Authenticity Task Force, *Requirements for Assessing and Maintaining the Authenticity of Electronic Records*. 2002, InterPARES Project: Vancouver, Canada.
41. PREMIS Working Group, *Data dictionary for preservation metadata: final report of the PREMIS Working Group*. 2005, OCLC Online Computer Library Center & Research Libraries Group: Dublin, Ohio, USA.
42. Diessen, R.J.v. and T.v.d. Werf-Davelaar, *Authenticity in a digital environment*. 2002, Koninklijke Bibliotheek & IBM: Amesterdam, The Netherlands.
43. Cullen, C.T., et al., *Authenticity in a Digital Environment*. 2000, Washington, DC: Council on Library and Information Resources.
44. Anderson, C., *Digital Preservation: Will Your Files Stand the Test of Time?* Library Hi Tech News, 2005. **6**: p. 9-10.
45. OASIS, *Universal Description, Discovery and Integration (UDDI)*. 2005, from http://www.uddi.org/.
46. Darlington, J., *PRONOM - A Practical Online Compendium of File Formats.* RLG DigiNews, 2003. **7**(5).
47. Wikipedia, *Availability*. 2005, from http://en.wikipedia.org/wiki/Availability.
48. Wikipedia, *Stability*. 2005, from http://en.wikipedia.org/wiki/Stability.
49. Deters, R. *Scalability and Multi-Agent Systems*. in *Workshop "Infrastructure for Scalable Multi-Agent Systems" at Agents 2001*. 2001. Montreal.
50. Luke, E.A. *Defining and measuring scalability*. in *Scalable Parallel Libraries Conference*. 1993. Mississippi State, USA.
51. Wikipedia, *Scalability*. 2005, from http://en.wikipedia.org/wiki/Scalability.
52. Wikipedia, *Throughput*. 2005, from http://en.wikipedia.org/wiki/Throughput.
53. Rusbridge, A., *Migration on Request*. 2003, University of Edinburgh - Division of Informatics.
54. Rauch, C. and A. Rauber. *Preserving Digital Media: Towards a Preservation Solution Evaluation Metric*. in *International Conference on Asian Digital Libraries*. 2004. Shanghai, China: Springer.
55. Rauch, C., et al. *Evaluating preservation strategies for audio and video files*. in *DELOS Digital Repositories Workshop*. 2005. Heraklion, Crete.
56. Rauch, C., *Preserving Digital Entities - A Framework for Choosing and Testing Preservation Strategies*, in *Institute for Software Technology and Interactive Systems*. 2004, Vienna University of Technology: Vienna, from
57. Rauch, C., et al., *A Framework for Documenting the Behaviour and Funcionality of Digital Objects and Preservation Strategies*. 2005, DELOS Network of Excellence: Glasgow.

"Em construção": uma análise ao estado actual da plataforma de Serviços Web para negócio electrónico

Miguel Filipe Leitão Pardal[1]

[1] Instituto Superior Técnico, Departamento de Engenharia Informática, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
miguel.pardal@dei.ist.utl.pt
http://mega.ist.utl.pt/~mflpar

Resumo. Este artigo apresenta uma proposta de classificação das normas de serviços Web, que resultou de um levantamento exaustivo. Os serviços Web são a iniciativa das grandes empresas de informática para uma plataforma que permita aos sistemas de informação dar melhor resposta a desafios de negócio. As normas são as especificações técnicas que definem a plataforma, cobrindo os seus vários requisitos. A nossa classificação permite ter uma visão global e independente do conjunto das normas, identificando propostas concorrentes que se sobrepõem ou contradizem. A partir deste levantamento, é possível enquadrar investigação aprofundada e apoiar decisões de adopção desta tecnologia em organizações.

1 Introdução

Os *serviços Web*[1] são a mais recente iniciativa da indústria de tecnologias de informação e comunicação para criar uma *plataforma* universal de negócio electrónico. O seu *objectivo* é permitir a interligação de processos de negócio, dentro e fora das organizações, através da execução integrada das aplicações informáticas e de comunicação suportada por tecnologia Internet.

Sublinhe-se que os serviços Web são aqui apresentados apenas como tecnologia e não como metodologia para desenvolvimento de sistemas de informação. As *arquitecturas orientadas a serviços*[2] são uma proposta estruturada para atingir este fim mais abrangente, pois para construir os sistemas com serviços é necessário adaptá-los aos requisitos do negócio que se destinam a suportar.

Do ponto de vista técnico, um serviço Web define uma *interface* funcional que pode ser invocada remotamente para dar acesso aos recursos informacionais de uma aplicação. A designação Web deve-se à inspiração na World Wide Web e na forma como nela os recursos são referenciados. Cada serviço é identificado por um identificador uniforme de recurso [1].

A plataforma é definida de forma modular através de *normas* de serviços Web, que especificam as regras técnicas que as implementações têm que respeitar.

[1] Em inglês, *Web Services (WS)*
[2] Em inglês, *Service-oriented architectures (SOA)*

Para dar coerência geral a toda a plataforma foi definido um conjunto de *princípios técnicos comuns* [2], que são:

- ⇒ Orientação a mensagens – os serviços comunicam exclusivamente por mensagens, que têm um tempo de vida útil que se pode estender para além do acto de transmissão num dado transporte;
- ⇒ Encapsulamento – a interface do serviço é descrita num contrato normalizado e público, mas a sua implementação é privada;
- ⇒ Autonomia – cada serviço pode ser gerido de forma individual e tem o mínimo de dependências para outros serviços;
- ⇒ Composição de protocolos – os protocolos utilizados pelos serviços são estruturados em blocos que podem ser compostos de acordo com as necessidades efectivas de uma dada utilização;
- ⇒ Interoperabilidade baseada em normas – não se assume nenhum pressuposta para além dos que são explicitados nas normas.

O tema principal deste artigo é a plataforma de serviços Web e as normas que a definem. Na secção do problema apresentamos uma das lacunas actuais da plataforma. No trabalho relacionado fazemos um levantamento de propostas para resolver o problema identificado. Na nossa proposta apresentamos uma classificação das normas de serviços. Na avaliação comparamos a nossa proposta com as outras. Finalmente, as conclusões apontam as principais contribuições e o trabalho futuro.

2 Problema

O *problema* que analisamos neste artigo é a inexistência de uma visão global detalhada da plataforma aceite por todos os seus promotores.

Existem dezenas de processos de normalização actualmente em curso para cobrir todo o espaço de requisitos para aplicações empresariais com serviços Web [3]. As normas são propostas inicialmente por empresas, sendo depois discutidas e finalizadas em organizações de normalização como o IETF[3] [4], o W3C[4] [5], a OASIS[5] [6], o WS-I[6] [7] e outras. Os principais promotores dos serviços Web são a Microsoft [8], a IBM [9], a Sun [10] e a Oracle [11], acompanhadas por muitas outras empresas da indústria de tecnologias de informação e comunicação.

As normas base da plataforma de serviços Web – SOAP [12], WSDL [13] e UDDI [14] – são já suportadas por praticamente todas as ferramentas. As restantes estendem as normas base e são tipicamente designadas pela sigla "WS-" seguida de palavras que referem o seu objectivo e, na sua maioria, estão ainda em estados mais atrasados de desenvolvimento e de aceitação [15].

Recentemente têm surgido propostas de normas que sobrepõem e/ou contradizem outras. Isto acontece porque existem grupos de promotores dos serviços Web que têm

[3] *Internet Engineering Task Force*
[4] *World Wide Web Consortium*
[5] *Organization for the Advancement of Structured Information Standards*
[6] *Web Services Interoperability Organization*

visões detalhadas diferentes da plataforma apesar de partilharem a visão de alto nível. As propostas concorrentes impedem uma visão globalmente coerente e tornam a plataforma instável no seu conjunto. Por sua vez, esta instabilidade é um dos factores que impede a sua adopção pelas organizações, que são os potenciais clientes desta tecnologia.

3 Trabalho relacionado

Existem várias propostas de visão global detalhada para a plataforma de serviços Web. Vamos apresentar resumidamente as propostas que consideramos mais relevantes.

O W3C [16] propõe o esquema de classificação apresentado na figura 1. As categorias de normas são: Tecnologias Base (*Base Technologies*), Comunicação (*Communications*), Mensagem (*Messages*), Descrições (*Descriptions*), Processos (*Processes*), Segurança (*Security*) e Gestão (*Management*). Apesar de promovida pelo W3C, esta proposta não é aceite como norma.

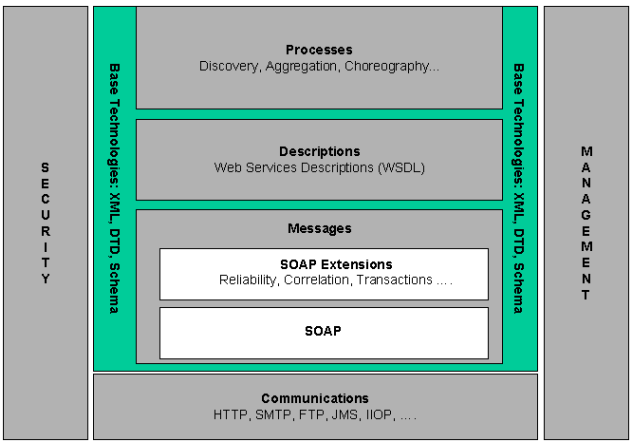


Fig. 1. Arquitectura de normas de serviços Web proposta pelo W3C [16].

A Microsoft [2] propõe as categorias de normas da figura 2, ou seja: XML, Metadados (*Metadata*), Mensagens (*Messaging*), Segurança (*Security*), Mensagens Fiáveis (*Reliable Messaging*) e Transacções (Tr*ansactions*).

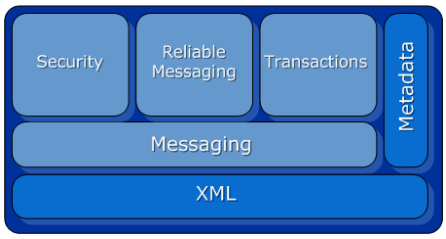


Fig. 2. Categorias de normas de serviços Web propostas pela Microsoft [2].

A IBM [17] define as categorias da figura 3, que incluem: Transporte (*Transports*), Mensagens (*Messaging*), Descrição e Descoberta (*Description and Discovery*), Fiabilidade (*Reliability*), Transacções (*Transactions*), Segurança (*Security*), Processos de Negócio (*Business Processes*) e Gestão (*Management*).

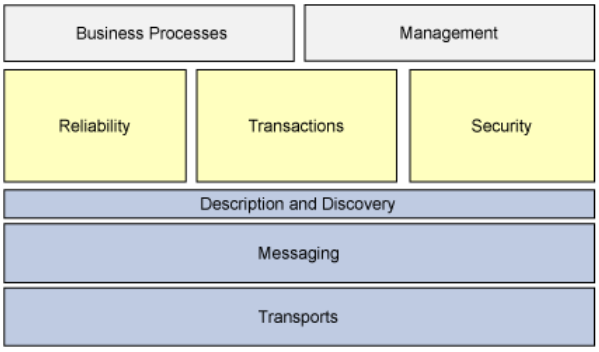


Fig. 3. Categorias de normas de serviços Web segundo a IBM [17].

Kreger [18] propõe categorias numa estrutura mais rica, com dois níveis, apresentada na figura 4. As normas são arrumadas em quatro conjuntos: Comunicação (*Wire*), Descrição (*Description*), Descoberta (*Discovery Agencies*) e Preocupações Abrangentes (*Overarching Concerns*). A descrição dos serviços é muito mais completa do que as outras propostas, abrangendo desde representação de dados até aos acordos de negócio necessários para o serviço. As preocupações abrangentes incluem a gestão, a segurança e outros aspectos de qualidade de serviço, que atravessam toda a plataforma.

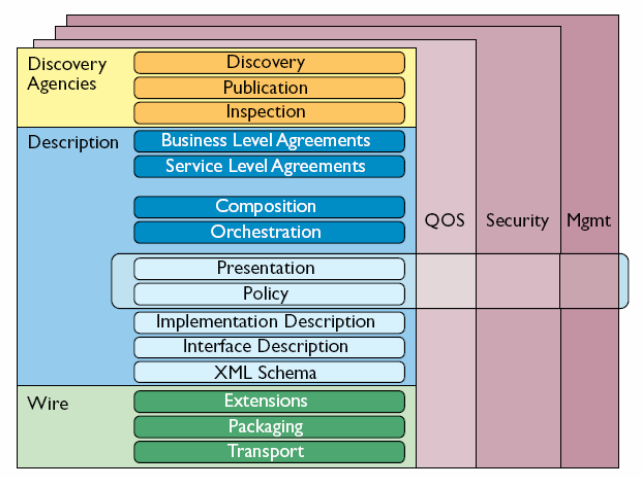


Fig. 4. Categorias de normas de serviços Web propostas por Kreger [18].

Finalmente, Wilkes [3] fez um esforço independente, metódico e exaustivo para identificar as normas e propostas de vários promotores de serviços Web. Os resultados foram um conjunto vasto de tabelas, resumidas na figura 5. As categorias definidas foram: Metadados (*Metadata*), Mensagem (*Messaging*), Transacções e Processos de Negócio (*Transactions and Business Processes*), Apresentação (*Portal and Presentation*), Segurança (*Security*), Gestão (*Management*) e Negócio (*Business Domain*).

Business Domain Specific extensions	Various	Business Domain
Distributed Management	WSDM, WS-Manageability	Management
Provisioning	WS-Provisioning	
Security	WS-Security	Security
Security Policy	WS-SecurityPolicy	
Secure Conversation	WS-SecureConversation	
Trusted Message	WS-Trust	
Federated Identity	WS-Federation	
Portal and Presentation	WSRP	Portal and Presentation
Asynchronous Services	ASAP	Transactions and Business Process
Transaction	WS-Transactions, WS-Coordination, WS-CAF	
Orchestration	BPEL4WS, WS-CDL	
Events and Notification	WS-Eventing, WS-Notification	Messaging
Multiple message Sessions	WS-Enumeration, WS-Transfer	
Routing/Addressing	WS-Addressing, WS-MessageDelivery	
Reliable Messaging	WS-ReliableMessaging, WS-Reliability	
Message Packaging	SOAP, MTOM	
Publication and Discovery	UDDI, WSIL	Metadata
Policy	WS-Policy, WS-PolicyAssertions	
Base Service and Message Description	WSDL	
Metadata Retrieval	WS-MetadataExchange	

Fig. 5. Categorias de normas de serviços Web segundo Wilkes [3].

4 Proposta

4.1 Visão global da plataforma

A nossa proposta para uma *visão global detalhada das normas* que formam a plataforma de serviços Web é apresentada na figura 6 e inclui as seguintes categorias: Representação de dados, Transporte, Mensagem, Contrato, Descoberta, Segurança, Transacções, Processos de negócio, Gestão e Interoperabilidade.

O principal contributo desta proposta face às anteriores é dar uma *visão global, independente do fornecedor de tecnologia*, para apoiar a escolha dos blocos construtores necessários e úteis para satisfazer os diversos requisitos de aplicações, derivados da área de negócio onde se inserem.

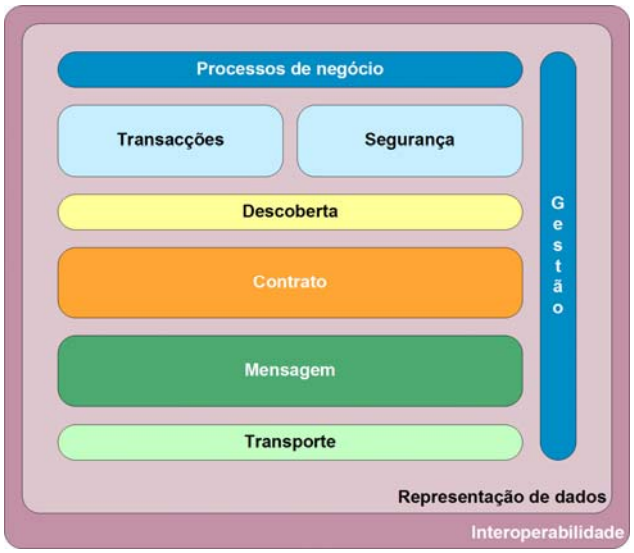


Fig. 6. Proposta de categorias de normas de serviços Web.

O *critério de pertença* de uma norma a uma categoria é que as normas de uma mesma categoria estão dentro do âmbito enunciado para ela e portanto resolvem problemas relacionados.

O arranjo gráfico da figura tem significado associado à estrutura da plataforma, que passamos a explicitar de seguida.

A cada categoria foi atribuída uma *caixa* rectangular com as extremidades arredondadas, devendo ser vista como um *contentor* de normas. A espessura vertical pretende dar uma ideia do volume de informação das normas que contêm, por exemplo, as caixas reflectem visualmente que as normas de Descoberta são menos extensas do que normas de Contrato. As cores próximas representam afinidades dos âmbitos, por exemplo, as normas de Mensagem são próximas das normas de Transporte.

A disposição das caixas deve-se a relações de dependência entre as categorias. Deste ponto de vista, a plataforma é construída de baixo para cima, com as normas superiores a tirarem partido do que está definido abaixo. Por exemplo, no topo está a caixa das normas de Processos de negócio, que tiram partido das normas de Transacções e das normas de Segurança. O posicionamento no topo representa o negócio electrónico como objectivo principal da plataforma. A caixa da Gestão é vertical em vez de horizontal, porque se relaciona com todas as outras, apesar da sua existência poder ser opcional. As caixas da Representação de dados e da Interoperabilidade estão por baixo das outras caixas, pois as suas normas são relacionadas com as outras, mas a sua utilização é obrigatória.

4.2 Categorias

Vamos agora enunciar o *âmbito* de cada categoria de normas da nossa proposta.

O problema da heterogeneidade na *Representação de dados* é transversal a todas as outras normas e foi resolvido com XML[7] [19] como formato canónico e com as tecnologias relacionadas, como a XML Schema [20] que permite especificar formatos de documentos.

O problema da *Interoperabilidade* vai para além da representação de dados e centra-se na compatibilidade das implementações das normas. O ponto de partida para a interoperabilidade são as normas de todas as categorias, que por vezes não têm especificações suficientemente claras, o que obriga a ter mecanismos de verificação da compatibilidade. A WS-I [7] é a organização que junta os principais fornecedores de ferramentas para definir perfis de interoperabilidade. Cada perfil abrange um conjunto de normas e fornece orientações à sua implementação, aplicações de exemplo e testes para aferir o cumprimento das especificações.

As normas de *Transporte* resolvem o problema de estabelecer um canal de comunicação entre o cliente e o prestador do serviço. A comunicação pode ser síncrona ou assíncrona, e adicionalmente pode oferecer garantias de fiabilidade ou segurança [2]. Os transportes mais comuns são o HTTP[8] [21] e o SMTP[9] [22]. Existem também implementações que recorrem a sistemas de filas de mensagens assíncronas.

As normas de *Mensagem* definem a estrutura das unidades de comunicação e as formas como são trocadas entre serviços. O SOAP [12] define mensagens em XML, separando o cabeçalho com dados extensíveis para a plataforma, do corpo com dados para as aplicações. Para transportar dados binários em SOAP usa-se MTOM/XOP[10] [23]. Para fazer o endereçamento e encaminhamento de mensagens de serviços de forma independente do transporte, existem duas propostas concorrentes: a WS-Addressing [24] (IBM, Microsoft) e a WS-MessageDelivery [25] (Oracle, Sun). Para sessões de enumeração de dados sequenciais existe a WS-Enumeration [26]. Para dar fiabilidade à comunicação em aspectos como a entrega garantida, a eliminação de repetições e a correcta ordenação, existem duas propostas concorrentes: a WS-Reliability [27] (Oracle, Sun) e a WS-ReliableMessaging [28] (IBM, Microsoft). Para notificações assíncronas de eventos que permitem evitar consultas sucessivas a serviços existem duas propostas concorrentes: a WS-Eventing [29] (Microsoft) e a WS-Notification [30] (IBM).

As normas de *Contrato* resolvem o problema da descrição da interface, da política e dos recursos do serviço. A interface do serviço é especificada de forma rigorosa com a WSDL[11] [13]. A política do serviço, ou seja, os requisitos que têm que ser cumpridos pelo cliente e pelo prestador do serviço para que a interacção entre ambos possa acontecer, são definidos pela WS-Policy [31]. A auto-descrição do serviço, através do acesso aos documentos WSDL e WS-Policy, é permitida pela WS-MetadataExchange [32]. Para descrever e gerir explicitamente os recursos informa-

[7] *eXtensible Mark-up Language*
[8] *Hyper Text Transfer Protocol*
[9] *Simple Mail Transfer Protocol*
[10] *Message Transmission Optimization Mechanism* / *XML-binary Optimized Packaging*
[11] *Web Services Description Language*

cionais dos serviços existem duas abordagens concorrentes: a WS-Transfer [33] (Microsoft) e a WS-ResourceFramework [34] (IBM).

As normas de *Descoberta* definem formas de publicar e pesquisar serviços. A UDDI[12] [14] define um directório. A WS-Inspection [35] define documentos de inspecção com referências para descrições. A WS-Discovery [36] define protocolos de descoberta através de multi-difusão de mensagens em redes locais.

As normas de *Transacções* permitem ter semânticas bem definidas para os resultados de várias interacções entre serviços com recursos informacionais distribuídos. Para isso assumem modelos de faltas temporárias e recuperáveis para as máquinas e comunicações. Existem dois enquadramentos de normas concorrentes para transacções em serviços Web: a WS-Coordination [37] (Microsoft, IBM) e a WS-CompositeApplicationFramework [38] (Oracle, Sun).

As normas de *Segurança* especificam mecanismos de protecção para que os serviços Web possam ser utilizados em aplicações com valor. As operações de base permitem a troca de itens de segurança e formas de garantir a integridade, a autenticação e a confidencialidade de mensagens. A WS-Security [39] especifica a segurança de mensagens SOAP, enquanto que a WS-SecurityPolicy [40] especializa a WS-Policy para políticas de segurança. O modelo de segurança adoptado abstrai diferentes tecnologias através de itens e de asserções.

As normas de *Processos de negócio* satisfazem a necessidade de ferramentas cujo nível de abstracção de conceitos é menos técnico e mais próximo do negócio e das preocupações das pessoas da organização. Existem abordagens distintas e potencialmente complementares para este problema. Todas elas procuram tirar partido das potencialidades da plataforma, em particular das transacções e da segurança, para satisfazer os requisitos não funcionais de forma mais simples. A WS-BPEL[13] [41] (Microsoft, IBM) é baseada em orquestração, sendo o processo representado por um grafo, em que os nós são actividades e os arcos são fluxos de controlo e informação, permitindo a composição de serviços mais simples. A WS-CDL[14] [42] (Oracle) faz a coreografia de processos de forma declarativa, especificando pré-condições e pós-condições para a execução de actividades, podendo a forma concreta como o processo é executado variar, desde que as condições continuem a ser satisfeitas. O ASAP[15] [43] permite definir processos através do encadeamento de serviços assíncronos, com possibilidade de intervenção humana.

À medida que mais processos de negócio da organização forem sendo suportados por serviços Web, mais importante se tornará a *Gestão* de toda a plataforma. Esta tem duas facetas: a gestão dos serviços Web em si e a gestão dos dispositivos computacionais e das redes de dados que os suportam. Neste momento o esforço de normalização está concentrado na segunda faceta, existindo duas normas concorrentes: a WS-Management [44] (Microsoft) e a WS-DistributedManagement [45] (IBM).

---

[12] *Universal Description, Discovery, and Integration*
[13] *Web Services Business Process Execution Language*
[14] *Web Services Choreography Description Language*
[15] *Asynchronous Services Access Protocol*

4.3 Normas

A figura 7 apresenta o resumo global da nossa proposta, relacionando cada categoria com as normas de serviços Web que nela se encaixam. As normas base, ou seja, aquelas que são suportadas por todas as ferramentas de serviços Web, são apresentadas a sublinhado nas suas categorias de Transporte (HTTP, SMTP), Mensagem (SOAP), Contrato (WSDL) e Descoberta (UDDI). Os pontos de interrogação e setas respectivas identificam normas concorrentes.

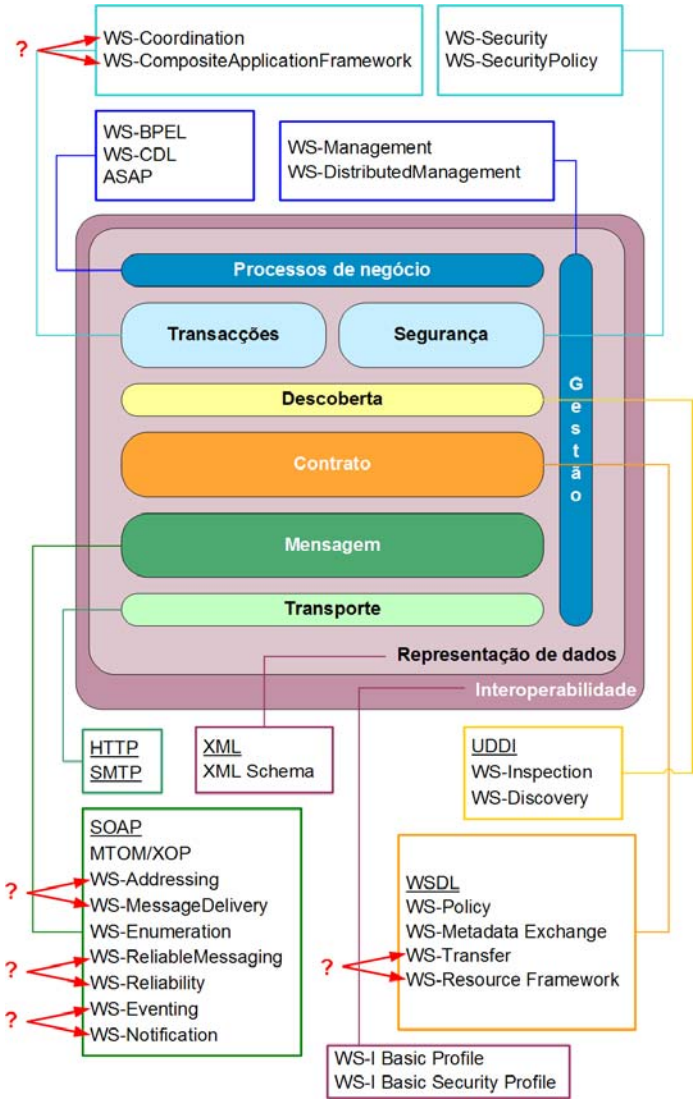


Fig. 7. Normas de serviços Web associadas à categorias da proposta.

5 Avaliação

5.1 Plataforma de serviços Web

Avaliando a plataforma de serviços Web, podemos concluir que os seus princípios técnicos são baseados em ideias de outras tecnologias informáticas, nomeadamente: os sistemas distribuídos (invocações remotas, filas de mensagens e gestão de nomes), as linguagens orientadas a objectos (encapsulação e polimorfismo) e os componentes (uso obrigatório de interfaces e meta-informação disponível em tempo de execução).

Dos princípios fundamentais definidos para a plataforma destaca-se a possibilidade de composição de blocos. A sua vantagem é permitir adequar a plataforma às necessidades efectivas, mantendo a possibilidade de introduzir novos blocos ou de estender os existentes. A desvantagem é que torna mais difícil ter uma visão global coerente da plataforma. Esta dificuldade é amplificada pelo facto de cada promotor ter tendência para ver a plataforma enviesado pelo seu negócio principal, ou seja, dando mais importância relativas aos seus equipamentos ou à prestação dos seus serviços, que são diferentes dos de outros promotores.

5.2 Análise comparativa

O objectivo da nossa proposta foi oferecer uma melhor forma de visualizar globalmente a plataforma no seu conjunto, ou seja, os blocos de normas que a constituem e as formas como esses blocos encaixam. Vamos agora avaliar a proposta em comparação com as propostas do trabalho relacionado.

Booth [16] tem o conceito de tecnologias de base, um núcleo e aspectos abrangentes. O critério de classificação não é muito claro, permitindo que normas de processos com objectivos diferentes sejam agrupados na mesma categoria, o que torna mais difícil encontrar sobreposições de âmbito.

Cabrera [2] tem a vantagem de ter menos categorias de normas, mas em contrapartida as bases não são bem explicitadas. Existe uma categoria XML demasiado abrangente e os meta-dados não diferenciam dois aspectos distintos como a descrição e a descoberta. Além disto, a proposta de Cabrera não inclui a gestão nem os processos de negócio.

A proposta de Wahli [17] é próxima da nossa, pois já detalha as normas base e inclui os processos de negócio. No entanto, não explicita os papéis da representação de dados e, principalmente, da interoperabilidade.

Kreger [18] propõe uma estrutura mais complexa, de dois níveis, que resulta de ter um número elevado de categorias para normas. Consideramos o número excessivo pois, apesar de representar aspectos da plataforma omissos nas restantes propostas, não dá uma visão simples e existem normas que abrangem várias categorias. Apesar da sua abrangência, esta proposta deixa de fora as normas de interoperabilidade.

Wilkes [3] faz uma proposta que resulta de um trabalho de levantamento exaustivo no que respeita a diferentes promotores e que inclui uma categoria para os diferentes

domínios de negócio. No entanto, não explicita o papel do transporte, não destaca o papel da interoperabilidade e define uma categoria para apresentação e portais, que na nossa proposta preferimos incorporar dentro da categoria de processos de negócio.

5.3 Aspectos não contemplados na proposta

Para além dos aspectos comparativos, podemos avaliar os aspectos que não são contemplados na nossa proposta.

Os acordos de serviço e de negócio não foram incluídos, apesar de serem essenciais para que existam transacções de negócio com valor a ser efectivamente suportadas por serviços Web. O acordo de negócio tem que ter uma descrição contratual legalmente aceite do serviço. O acordo de qualidade de serviço tem que definir o desempenho, utilização, custos, métricas e limites que têm que ser cumpridos. Para já não existem propostas de normas para estes acordos, e alguns deles, pela sua especificidade, só fazem sentido em áreas de negócio verticais. Quando surgirem, as normas de acordo de negócio enquadram-se na categoria dos Processos de negócio, e as normas de acordo de qualidade de serviço enquadram-se na categoria de Contrato.

6 Conclusão

6.1 Principais contribuições

A principal contribuição deste artigo é o levantamento do estado actual das normas da plataforma de serviços Web para negócio electrónico, identificando as relações de dependência entre normas, bem como as sobreposições e contradições devidas a propostas concorrentes para o mesmo objectivo.

A visão global detalhada da nossa proposta é completa, pois apesar de ser independente de um ou mais promotores em particular, permite enquadrar as suas visões debaixo da mesma classificação. O papel da interoperabilidade é explicitado como aspecto fundamental dos serviços Web.

Os resultados deste artigo podem ser um ponto de partida para trabalhos de investigação aprofundados em categorias específicas de normas, por exemplo, em segurança, pois ajuda a delinear um quadro mais completo dos desafios de protecção que se colocam às soluções baseadas em serviços.

Este artigo pode também apoiar algumas decisões tecnológicas de empresas em sistemas de informação, pois identifica os promotores e a maturidade das normas, ajudando a optar por ferramentas mais estáveis em detrimento de outras mais experimentais.

6.2 Trabalho futuro

Um dos desafios futuros será manter a actualização do levantamento das normas e a adequação das categorias propostas, de modo a acompanhar os desenvolvimentos que se prevêem nos próximos anos de actividade neste domínio.

Para além de classificar as normas em categorias, será necessário observar os resultados efectivos da sua utilização em sistemas de informação reais, de diferentes áreas de negócio. Isto é necessário para aferir a importância relativa de cada norma no sucesso dos projectos em que são aplicadas, para aferir o seu valor. Uma das formas de o fazer é através da análise de casos de estudo que abranjam as várias fases de desenvolvimento e manutenção dos sistemas.

A nossa proposta cobre os aspectos horizontais da plataforma de serviços, ou seja, aqueles que são comuns a várias áreas de negócio. Futuramente faz sentido enquadrar normas sectoriais que venham a ser propostas – para a Banca, Seguros, Saúde, etc. – numa visão global consistente com os princípios fundamentais da plataforma em particular da composição de blocos.

Referências

1. Rosenberg J., Remy D.: Securing Web Services with WS-Security, SAMS Publishing (2004).
2. Cabrera L. F., Kurt C., Box D.: An Introduction to the Web Services Architecture and Its Specifications, Version 2.0, Microsoft (2004). http://msdn.microsoft.com/webservices/default.aspx?pull=/library/en-us/dnwebsrv/html/introwsa.asp.
3. Wilkes L.: The Web Services Protocol Stack, CBDI (2005). http://roadmap.cbdiforum.com/reports/protocols/.
4. IETF: Overview of the IETF, IETF Web Site (2005). http://www.ietf.org/overview.html.
5. W3C, About the World Wide Web Consortium, W3C Web Site (2005). http://www.w3.org/Consortium/.
6. OASIS, About OASIS, OASIS Web Site (2005). http://www.oasis-open.org/who/.
7. WS-I, About WS-I, WS-I Web Site (2005). http://www.ws-i.org/about/Default.aspx.
8. Microsoft, Company Information, Microsoft Web Site (2005). http://www.microsoft.com/mscorp/info/.
9. IBM: About IBM, IBM Web Site (2005). http://www.ibm.com/ibm/us/.
10. Sun Microsystems, About Sun, Sun Web Site (2005). http://www.sun.com/aboutsun/.
11. Oracle, About Oracle, Oracle Web Site (2005). http://www.oracle.com/corporate/about.html.
12. Gudgin M. (ed.), Hadley M. (ed.), Mendelsohn N. (ed.), Moreau J. J. (ed.), Nielsen H. (ed.): SOAP Version 1.2 Part 1: Messaging Framework, W3C (2003). http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.
13. Christensen E., Curbera F., Meredith G., Weerawarana S.: Web Services Description Language (WSDL) 1.1, W3C (2001). http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
14. Clement L., Hately A., Riegen C., Rogers T.: UDDI Version 3.0.2, OASIS (2004). http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.
15. K. Hogg, P. Chilcott, M. Nolan, Srinivasan B.: An Evaluation of Web Services in the Design of a B2B Application, IBM Global Services Australia, Monash University Faculty

of Information Technology. Proceedings of the 27th conference on Australasian computer science, Dunedin, New Zealand, Volume 26 pages: 331 - 340 (2004).
16. Booth D., Haas H., McCabe F.: Web Services Architecture, W3C (2004). http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.
17. Wahli U., Kjaer T., Robertson B., Satoh F., Schneider F., Szczeponik W., Whyley C.: WebSphere Version 6 - Web Services Handbook - Development and Deployment, IBM (2005)
18. Kreger H.: Fulfilling The Web Services Promise, Communications Of The ACM June 2003/Vol. 46, No. 6 (2003)
19. Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., Yergeau F.: Extensible Markup Language (XML) 1.0 (Third Edition), W3C (2004). http://www.w3.org/TR/2004/REC-xml-20040204.
20. Fallside D., Walmsley P.: XML Schema Part 0: Primer Second Edition, W3C (2004). http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.
21. Fielding R. , Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T.: Hypertext Transfer Protocol -- HTTP/1.1, The Internet Society (1999). http://www.w3.org/Protocols/rfc2616/rfc2616.txt.
22. Klensin J. (ed.): Simple Mail Transfer Protocol, IETF (2001). http://www.ietf.org/rfc/rfc2821.txt.
23. Gudgin M. (ed.), Mendelsohn N. (ed.), Nottingham M. (ed.), Ruellan H. (ed.): SOAP Message Transmission Optimization Mechanism, W3C (2005). http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/.
24. Box D. (ed.), Curbera F. (ed.): Web Services Addressing (WS-Addressing), W3C (2004). http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/.
25. Karmarkar A. (ed.), Yalçinalp Ü. (ed.): WS-MessageDelivery Version 1.0, W3C (2004). http://www.w3.org/Submission/2004/SUBM-ws-messagedelivery-20040426/.
26. Geller A. (ed.): Web Service Enumeration (WS-Enumeration), Microsoft (2004). http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-enumeration.pdf.
27. Iwasa K. (ed.): WS-Reliability 1.1, OASIS (2004). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm.
28. Ferris C. (ed.), Langworthy D. (ed.): Web Services Reliable Messaging Protocol (WS-ReliableMessaging), Microsoft, IBM (2005). http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-ReliableMessaging.pdf.
29. Geller A. (ed.): Web Services Eventing (WS-Eventing), Microsoft (2004). http://www.ibm.com/developerworks/webservices/library/specification/ws-eventing/.
30. Graham S. (ed.), Niblett P. (ed.): Web Services Notification (WS-Notification) Version 1.0, IBM, SAP (2004). http://ifr.sap.com/ws-notification/ws-notification.pdf.
31. Schlimmer J. (ed.): Web Services Policy Framework (WS-Policy), Microsoft, IBM, VeriSign (2004). http://msdn.microsoft.com/webservices/default.aspx? pull=/library/en-us/dnglobspec/html/ws-policy.asp.
32. Curbera F. (ed.), Schlimmer J. (ed.): Web Services Metadata Exchange (WS-MetadataExchange), IBM, Microsoft (2004). http://msdn.microsoft.com/ws/2004/09/ws-metadataexchange/.
33. Geller A. (ed.): Web Service Transfer (WS-Transfer), Microsoft (2004). http://msdn.microsoft.com/ws/2004/09/ws-transfer/.
34. Czajkowski K., Ferguson D., Foster I., Frey J., Graham S., Sedukhin I., Snelling D., Tuecke S., Vambenepe W., The WS-Resource Framework version 1.0, Globus Alliance, IBM, Fujitsu, Computer Associates, Hewlett-Packard (2004). http://www.globus.org/wsrf/specs/ws-wsrf.pdf.

35. Ballinger K., Brittenham P., Malhotra A., Nagy W., Pharies S.: Web Services Inspection Language (WS-Inspection) 1.0, IBM (2001). http://www.ibm.com/developerworks/library/specification/ws-wsilspec/.
36. Schlimmer J. (ed.): Web Services Dynamic Discovery (WS-Discovery), Microsoft (2005). http://msdn.microsoft.com/library/en-us/dnglobspec/html/WS-Discovery.pdf.
37. Feingold M. (ed.): Web Services Coordination (WS-Coordination) Version 1.0, Microsoft, Hitachi, IBM, IONA, BEA (2005). http://www.ibm.com/developerworks/library/specification/ws-tx/.
38. Little M. (ed.), Newcomer E. (ed.): Web Services Composite Application Framework (WS-CAF) Version 1.0, Arjuna, IONA, Oracle, Sun (2003). http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CAF-Primer.pdf.
39. Nadalin A. (ed.), Kaler C. (ed.), Hallam-Baker P. (ed.), Monzillo R. (ed.): Web Services Security: SOAP Message Security 1.0 (WS-Security), OASIS (2004). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.
40. Kaler C. (ed.), Nadalin A. (ed.): Web Services Security Policy Language (WS-SecurityPolicy) Version 1.1, Microsoft, IBM (2005). http://www.ibm.com/developerworks/library/specification/ws-secpol/.
41. Thatte S. (ed.): Business Process Execution Language for Web Services Version 1.1, Microsoft, IBM, BEA (2003). http://www.ibm.com/developerworks/library/specification/ws-bpel/.
42. Kavantzas N. (ed.), Burdett D. (ed.), Ritzinger G. (ed.): Web Services Choreography Description Language Version 1.0, W3C (2004). http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/.
43. Fuller J. (ed.), Krishnan M. (ed.), Swenson K. (ed.), Ricker J. (ed.): Asynchronous Service Access Protocol (ASAP) Version 1.0, OASIS (2005). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=asap.
44. Geller A. (ed.): Web Services for Management (WS-Management), Microsoft (2004). http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-management1004.pdf.
45. Kreger H. (ed.), Bumpus W. (ed.): OASIS Web Services Distributed Management (WSDM) Technical Committee, OASIS (2005). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm.

Anexo: Resumo das normas de serviços Web

Tabela 1. Normas de Representação de dados.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
XML (eXtensible Markup Language)	[19]	Norma	W3C	--
XML Schema	[20]	Norma	W3C	--

Tabela 2. Normas de Transporte.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
HTTP (HyperText Transfer Protocol)	[21]	Norma 1.1	IETF	--
SMTP (Simple Mail Transfer Protocol)	[22]	Norma	IETF	--

Tabela 3. Normas de Mensagem.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
SOAP	[12]	Norma 1.2	W3C	--
SOAP MTOM (Message Transmission Optimization Method)	[23]	Norma	W3C	--
WS-Addressing	[24]	Norma provisória	W3C	BEA, IBM, Microsoft, SAP, Sun
WS-Message Delivery	[25]	Proposta	W3C	Nokia, Oracle, Sun, ...
WS-Enumeration	[26]	Proposta	--	Microsoft, BEA, CA, ...
WS-Reliable Messaging	[28]	Proposta	OASIS	IBM, Microsoft, BEA, TIBCO, ...
WS-Reliability	[27]	Norma 1.1	OASIS	Fujitsu, HP, Hitachi, NEC, Novell, Oracle, Sun, ...
WS-Eventing	[29]	Proposta	--	Microsoft, IBM, TIBCO, BEA, ...
WS-Notification	[30]	Proposta	OASIS	IBM, TIBCO, Akamai, SAP, ...

Tabela 4. Normas de Contrato.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
WSDL (Web Services Description Language)	[13]	Norma 1.1	W3C	--
WS-Policy	[31]	Proposta	--	Microsoft, IBM, BEA, SAP, ...
Web Services Metadata Exchange	[32]	Proposta	--	Microsoft, IBM, BEA, ...
WS-Transfer	[33]	Proposta	--	Microsoft, BEA, ...
WS-Resource Framework	[34]	Proposta	--	IBM, HP, Fujitsu, ...

Tabela 5. Normas de Descoberta.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
UDDI (Universal Description, Discovery, and Integration)	[14]	Norma 3.0	OASIS	IBM, Microsoft, SAP, ...
Web Services Inspection Language	[35]	Proposta	--	IBM, Microsoft
WS-Discovery	[36]	Proposta	--	Microsoft, Intel, Canon, ...

Tabela 6. Normas de Transacções.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
WS-Coordination	[37]	Proposta	--	Microsoft, IBM, Hitachi , IONA, ...
WS-CAF (Composite Application Framework)	[38]	Proposta	OASIS	Fujitsu, IONA, Oracle, Sun, Arjuna

Tabela 7. Normas de Segurança.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
WS-Security	[39]	Norma 1.0	OASIS	IBM, Microsoft, VeriSign, Sun
WS-SecurityPolicy	[40]	Proposta 1.1	OASIS	Microsoft, IBM, Verisign, RSA security

Tabela 8. Normas de Processos de negócio.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
WS-BPEL (Business Process Execution Language)	[41]	Norma provisória 1.1	OASIS	Microsoft, IBM, Siebel, BEA, SAP
WS-CDL (Choreography Description Language)	[42]	Norma provisória 1.0	W3C	Oracle, Commerce One, Novell
ASAP (Asynchronous Service Access Protocol)	[43]	Norma provisória 1.0	OASIS	Cisco, Fujitsu, TIBCO

Tabela 9. Normas de Gestão.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
WS-Management	[44]	Proposta	--	Microsoft, Sun, Intel, AMD, Dell
WS-DistributedManagement	[45]	Norma provisória 1.0	OASIS	IBM, Dell

Tabela 10. Normas de Interoperabilidade.

Norma	Referência	Estado	Organização de normalização	Promotores da proposta
WS-I Profiles	[7]	Norma	WS-I	--

Servicio Web de identificación biométrica sobre FPGA para dispositivos móviles Wi-Fi

David Rodríguez [1], Juan M. Sánchez [1], Arturo Duran[1]

[1] Área de Arquitectura y Tecnología de los Computadores
Esuela Politécnica, Universidad de Extremadura
Campus Universitario S/N 10071 Cáceres
{drlozano, sanperez, aduran}@unex.es
http://atc.unex.es

Abstract. En el presente artículo proponemos una arquitectura completa que cumple el estándar BioAPI, basada en un servicio Web XML y hardware reconfigurable, para la identificación y verificación biométrica móvil de individuos mediante el uso de su huella dactilar. Se plantea un sistema cliente servidor, en el que se utiliza como cliente Web biométrico un Asistente Digital Personal (PDA) provisto de un sensor biométrico y de tecnología inalámbrica WiFi, y de un servidor de aplicación biométrica, soportado sobre un Proveedor de Servicios Biométricos (BSP) parcialmente implementado sobre un dispositivo hardware reconfigurable. BioAPI da acceso a las funciones de *enrolment*, *identification* y *verification* mediante un BSP acelerado por lógica reconfigurable (FPGA).

1 Introducción

La identificación biométrica es la autentificación y/o verificación de la identidad de una persona basado en características de su cuerpo o de su comportamiento, utilizando por ejemplo el iris de su ojo, su voz o su huella dactilar. Los métodos de identificación biométrica son una herramienta eficaz para identificar personas. Las huellas dactilares están reconocidas como prueba fidedigna de identidad, siendo un sistema de autenticación efectivo, cómodo y rápido de aplicar.

Las nuevas tecnologías permiten en la actualidad disponer de dispositivos móviles en forma de ordenadores de mano o PDAs con sensores biométricos para la captura y procesamiento de huellas dactilares. Esto unido a las emergentes tecnologías inalámbricas, permiten desarrollar aplicaciones para la identificación o verificación de individuos en espacios públicos o abiertos, como los aeropuertos, zonas de ocio, centros de enseñanza, etc.

Las PDAs y los nuevos *Smart Phones* (teléfonos con microprocesador, memoria, pantalla y sistema operativo), tienen limitaciones computacionales y de almacenamiento grandes, con lo que se requiere de una arquitectura cliente servidor para poder realizar identificaciones rápidas y fiables de un individuo contra una población grande de huellas, identificación 1:N.

Por otro lado, los servicios Web XML proveen un modelo basado en estándares simples y flexibles, para la interconexión de aplicaciones sobre Internet, aprovechando las infraestructuras existentes, siendo la evolución natural de las arquitecturas distribuidas en Internet [1][2].

Por último, el hardware reconfigurable ha sido utilizado con éxito en múltiples ocasiones para la realización de tareas de cifrado, reconocimiento de patrones, criptografía, etc., que son por naturaleza problemas intratables y de complejidad computacional muy elevada, bien como sistemas embebidos o servidores dedicados [4].

A la vista de estas consideraciones, proponemos el diseño de un sistema completo de autentificación/verificación biométrica de huellas dactilares, basado en una aplicación cliente-servidor para dispositivos móviles (PDA o *Smart Phone*), utilizando servicios Web XML y un BSP acelerado mediante hardware reconfigurable. Las ideas clave del presente trabajo son:

- Introducir la biometría dactilar y el estándar BioAPI, secciones 2 y 3.
- Identificar los elementos que componen la arquitectura propuesta, sección 4.
- Definir la arquitectura y funcionalidades de la aplicación cliente, sección 5.
- Por último, descripción del servicio Web XML de identificación biométrica sobre FPGA, sección 6.

2 Biometría basada en huellas dactilar

La identificación vía huellas dactilares se lleva a cabo mediante la comparación de ciertas características que se extraen de las mismas llamadas minutiaes. Y que son el resultado del análisis de las crestas de fricción (*ridge*) y de los valles (*valley*) que se visualizan en las imágenes de las huellas. Estos minutiaes son clasificados en 18 tipos por algunos métodos, si bien como muestra la figura 1, son combinación de dos tipos básicos: final de cresta (*ending*) y bifurcación de cresta (*bifurcation*) [5].

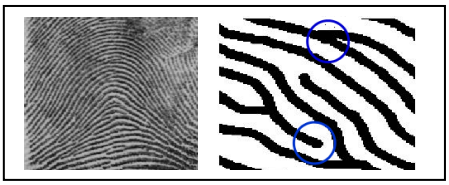


Fig. 1. Huella y detalle de los dos tipos de minutiaes

Los algoritmos de identificación tradicionales están basados en la utilización de funciones de reconocimiento de patrones de puntos para la comparación de minutiaes, y por tanto, los sistemas implementados que los utilizan suelen almacenar únicamente para cada individuo los minutiaes que caracterizan su huella dactilar.

La aplicación a los minutiaes de algoritmos de reconocimiento de patrones de puntos o locales, ofrece problemas en determinados casos, como son imágenes de huellas

de baja calidad de las que se extraen un número pequeño de minutiaes, alteración de las condiciones de humedad o temperatura en la que se capturaron las imágenes, atenuación de los ridge o modificaciones no lineales de las huellas. Todos estos factores hacen que la identificación de individuos basada en este tipo de algoritmos sea tratada como un problema computacionalmente complejo [6].

Este tipo de algoritmos pueden mejorar su efectividad utilizando métodos de reconocimiento de patrones de subáreas o globales, que utilizan la comparación de áreas de especial interés en lugar de puntos. Estas áreas se seleccionan de alrededor de los minutiaes, de áreas que cuenten con ridges que tengan baja curvatura o combinaciones de ridges poco usuales [7].

3 BioAPI

La utilización de todas estas técnicas para desarrollar sistemas automáticos de identificación de individuos, supone problemas de interoperabilidad. La huella de un usuario digitalizada y tratada por uno de los procesos desarrollados, puede no ser útil para su utilización con cualquiera de los algoritmos planteados.

Para solucionar estos problemas, desde diferentes comités internacionales se han desarrollado estándares para diferentes aspectos de la utilización de la biometría, como son: el formato para el intercambio de datos, estructuras de datos, protocolos de comunicación, etc. Uno de ellos es BioAPI y su objetivo es estandarizar el interfaz entre los componentes software y la arquitectura, ocultando en la medida de lo posible los detalles de la tecnología de biometría, facilitando así el desarrollo de aplicaciones que usen esta tecnología.

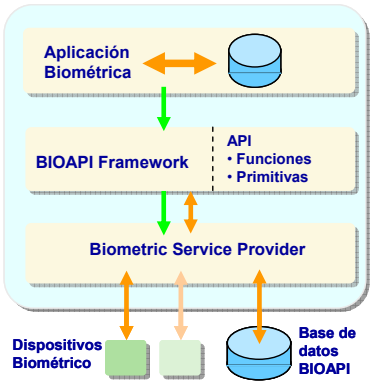


Fig. 2. Arquitectura del estándar BioAPI

La API propuesta por BioAPI ofrece a las aplicaciones funciones de alto nivel como *enrollment*, *verification* e *identification*. Mientras que el proveedor de servicios biométricos o BSP ofrece un conjunto de funciones de bajo nivel como son: *capture*,

process, createtemplate, *verifymatch*, e *identifymatch* orientadas directamente a la tecnología biométrica.

4 Arquitectura XML-BioAPI para entornos móviles

El sistema ha sido diseñado siguiendo una arquitectura de capas como la mostrada en la figura 3, utilizando un esquema cliente servidor, y una red de acceso y tránsito basada en tecnologías IP (red inalámbrica WiFi + Internet/Intranet). Se utiliza tecnología Web estándar para la comunicación entre la aplicación cliente y el servidor, siendo las funciones principales de cada elemento las siguientes:

- Aplicación cliente: se trata de una aplicación basada en un navegador Web y sus extensiones (ActiveX, Applets y JavaScript) la cual actúa como interfaz entre el usuario y el proveedor de servicios biométricos BSP.
- Servidor de aplicación y BSP: consiste en un servicio Web XML con las extensiones software y hardware necesarios para implementar las primitivas propuestas en BioAPI. El BSP diseñado implementa un subconjunto de sus funciones sobre hardware reconfigurable, utilizando una FPGA Virtex de Xillinx sobre una placa de prototipos RC1000 de Celoxica.
- Base de Datos: es compatible con el estándar ISO CBEFF (NISTIR 6529).

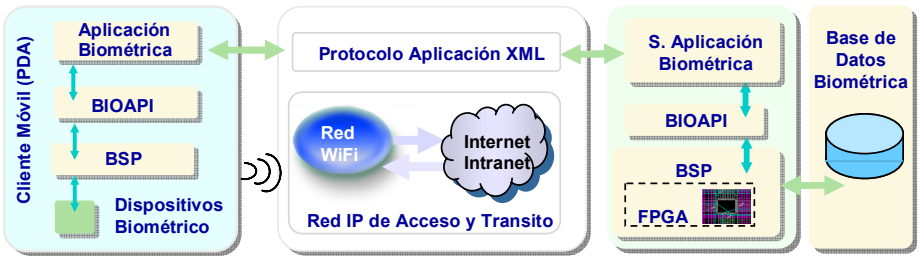


Fig. 3. Visión global de la arquitectura propuesta

Las ventajas de utilizar una arquitectura cliente servidor basada en tecnologías Web sobre una red IP son varias:

- Permite independizar la arquitectura cliente hardware y software del medio de acceso (WiFi, Ethernet, Bluetooth, GPRS/UMTS) y de la la red de transito IP (ATM, Frame Relay, SONET).
- El mantenimiento y distribución de la aplicación cliente se facilita utilizando un servidor Web, la aplicación modificada se descarga de forma automática desde el servidor a los clientes.

5 Aplicación cliente

En el diseño de la aplicación cliente, se ha buscado una arquitectura flexible que permita cubrir tanto un escenario como el propuesto PDA y WiFi, como otros modelos de acceso y aplicaciones, como por ejemplo, un *Smart Phone* y UMTS o un PC con ADSL. También, se ha buscado emplear un interfaz grafico de usuario amigable y sencillo de utilizar, por lo que se decidió utilizar una aplicación basada en un navegador Web estándar.

Para la aplicación cliente, se plantearon dos posibles implementaciones, ambas debían poder utilizar las funciones proporcionadas por las librerías de BioAPI y gestionar la información auxiliar necesaria (nombre, sexo, edad, fotografía…) para realizar el proceso de identificación biométrica de un individuo, las opciones estudiadas han sido:

- Applet Java: utilizando como host un navegador Web equipado con una Maquina Virtual Java (JVM) y como herramientas de desarrollo el *Java Development Kit* (JDK) y el *Abstract Window Toolkit* (AWT). Tiene como ventaja la interoperabilidad entre distintos sistemas operativos y plataformas. Por el contrario es más lento que el código nativo y hay menos soporte y drivers para el acceso al hardware biométrico.
- Control ActiveX: el host es también un navegador Web, que soporte bien de forma nativa o mediante el uso de un *plug-in* controles ActiveX. Tiene como ventaja una ejecución nativa más rápida, más librerías y mayor soporte para el acceso al hardware. En contra una menor interoperabilidad entre sistemas operativos y al requerir Windows o un plug-in compatible.

5.1 Prototipado de la aplicación cliente

El dispositivo seleccionado para soportar la aplicación cliente, es una PDA Ipaq H5550 de Hewlett-Packard, con un microprocesador Intel PXA255 a 400Mhz con tecnología Xscale, 128Mb de memoria SDRAM y 48Mb de memoria Flash ROM, pantalla TFT de 240x320 y 65.000 colores.

Dispone de tecnologías inalámbricas Bluetooth y WLAN 802.11b, así como de un sensor *biométrico* para la lectura de huellas dactilares. La figura 4 muestra la PDA Ipaq H5550 solicitando la identificación del propietario del dispositivo para poder iniciar sesión.

La versión de la PDA es Pocket PC 2003 con sistema operativo Windows CE 4.20. Se ha utilizado el Pack de encriptación de 128 bits para cifrar la transmisión de la información a través de *Internet* o la intranet, añadiendo una capa adicional de seguridad al cifrado WEP.

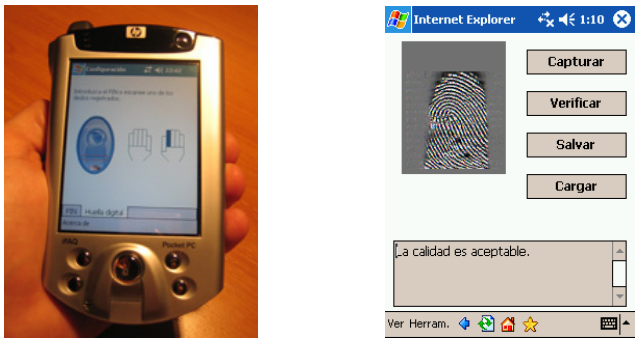


Fig. 4. PDA interfaz gráfico de la aplicación cliente ejecutándose sobre IExplorer

El sensor biométrico AT77C101B instalado en la PDA H5550, denominado FingerChip, es un sensor desarrollado por Atmel Corporation y cuyas características principales son:

- Sensor térmico CMOS.
- Area de escaneado de 0,4 mm x 14 mm.
- Array de la imagen: 8 x 280 = 2240 pixels
- Tamaño de pixel: 50 µm x 50 µm = 500 dpi

El software biométrico de preprocesamiento e identificación local esta desarrollado por Cogent Systems, Inc.

Para realizar una prueba de la arquitectura propuesta, se ha optado por utilizar un cliente basado en una aplicación Windows en forma de un control ActiveX, el cual es invocado desde el navegador Web Internet Explorer para Pocket PC. Para dar soporte a la aplicación biométrica, ha sido necesario instalar en la PDA las librerías:

- HP Ipaq Biometric Toolkit.
- BioAPI Consortium Framework.
- Microsoft .Net Compact Framework

6 Servicio Web XML de identificación biométrica

El servicio Web RC1000_BIOAPI implementa y soporta la conectividad de los clientes remotos a las funciones de alto nivel del estándar BioAPI *enrolment*, *identification* y *verification*.

Las llamadas a las funciones anteriores se traducen en llamadas de bajo nivel soportadas por el BSP. El BSP del prototipo cuenta con parte de estas funciones implementadas sobre la FPGA VIRTEX de la tarjeta RC1000 de Celoxica.

Los clientes inalámbricos pueden consumir los servicios del servidor de aplicación biométrico de dos formas distintas como muestra la figura 5, y descritas en los apartados 6.1 y 6.2.

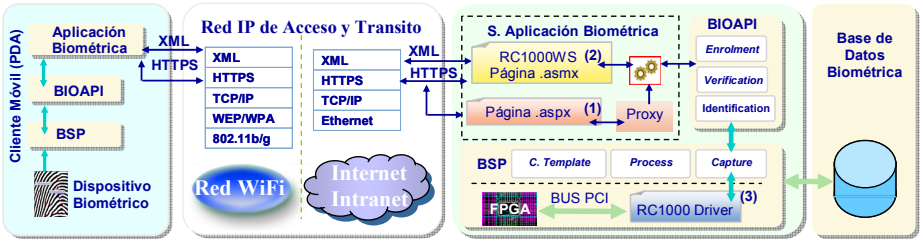


Fig. 5. Arquitectura, protocolos y dispositivos empleados en el prototipo

6.1 Acceso mediante servidor de aplicación

El servidor de aplicación, identificado con un (1) en la figura 5, publica sus recursos mediante páginas Web dinámicas con extensión .aspx, en las cuales se crea una instancia del Proxy que sirve de intermediario entre la página .aspx y el servicio Web que invocará las funciones de BioAPI. El resultado de las operaciones biométricas solicitadas, son devueltos al cliente desde la página .aspx del servidor de aplicación.

El servidor de aplicación controla el acceso a los recursos y puede implementar las funciones de *accounting* de los usuarios. Los protocolos utilizados en el caso (1) son HTTPS para el intercambio entre el Cliente Web y el Servidor de Aplicación, y XML-SOAP entre el Proxy y el Servicio Web.

6.2 Acceso directo al servicio RC1000_BIOAPI

La segunda opción, identificada con un (2) en la figura 5, se basa en la utilización de páginas .asmx de ASP.NET, que permiten un acceso directo al servicio Web. En la página asmx se describen: el espacio de nombres, las clases, propiedades y métodos del servicio Web XML. Para este escenario existen dos posibilidades:

- Primera, la aplicación cliente tipo ActiveX o Java Applet, la cual se comunica directamente mediante SOAP con el servicio Web sin necesidad de utilizar peticiones HTTPS.
- Segunda opción, la aplicación cliente interactúa con el navegador Web que la hospeda mediante JavaScript o VB Script. Siendo el código de la página Web el que se comunica con el servicio Web mediante peticiones HTTPS-GET y HTTPS-POST de páginas .asmx. Las peticiones de los clientes son interceptadas por el servidor Web IIS mediante un filtro ISAPI (Internet Server Application Programming Interface) que inicia un *runtime* que compila y ejecuta el código del servicio Web y devuelve el resultado en XML.

6.3 BSP basado en Hardware Reconfigurable

El BSP implementado en el servidor de aplicación biométrica está parcialmente soportado por algoritmos acelerados por una placa de prototipos RC1000 con una FPGA Virtex. El BSP propuesto en la arquitectura de BioAPI cuenta con cuatro funciones de bajo nivel que son susceptibles de ser implementadas en la FPGA. Estas funciones son: *process*, *createtemplate*, *verifymatch* e *identifymatch*.

La función *capture*, si bien es mantenida en el BSP, no tiene sentido en el lado del servidor si éste no cuenta con un sensor biométrico instalado.

Para el prototipo, se han implementado las funciones de la primitiva *process* del BSP, la cual se encarga del preprocesamiento y extracción de los minutiaes tomando como entrada la imagen de la huella dactilar capturada por el sensor de la aplicación cliente. En estos procesos se aplican algoritmos de tratamiento de imágenes como, filtros, umbralizaciones, adelgazamientos y operaciones morfológicas. Estos algoritmos se han implementado en Handel-C con éxito sobre la plataforma RC1000 en trabajos anteriormente [3].

Respecto a las funciones *verifymatch* e *identifymatch*, su implementación está ligada a la elección de un algoritmo de comparación de minutiaes. Para realizar este tipo de tareas se han utilizado normalmente algoritmos de pattern matching, bien de puntos o de zonas, o ambos simultáneamente.

La función *verifymatch* se encarga de la comparación 1:1 de huellas dactilares. Es decir, se comprueba que una huella (*sample*) coincide con una ya almacenada (*template*).

La función *identifymatch* realiza comparaciones 1:N de huellas dactilares. Comprueba si una determinada huella (*sample*) coincide con alguna de las huellas ya almacenadas en la base de datos biométrica.

Los algoritmos de pattern matching son algoritmos que pueden ser fácilmente pararelizables, existen implementaciones sobre FPGA que desmuestran la viabilidad y mejora en rendiemiento de una solución hardware-software sobre una solución software pura [8].

6.4 Acceso al hardware reconfigurable

Para los métodos de acceso descritos en 6.1 y 6.2 el acceso a los funciones del BSP implementadas en la FPGA, identificado con un (3) en la figura 5, se realiza a través del bus PCI mediante la utilización de las librerías de la tarjeta RC1000 (PP1000.lib) y del driver de la tarjeta.

7 Conclusiones y trabajos futuros

Hemos diseñado e implementado con éxito un prototipo que permite la identificación y verificación biométrica móvil de un individuo mediante un dispositivo PDA con tecnología WiFi y un servidor de aplicación Web XML biométrico acelerado con una FPGA.

Los puntos fuertes de nuestro trabajo incluyen: la utilización de una arquitectura que cumple con BioAPI como interfaz entre la aplicación cliente y la implementación hardware-software de las funciones del BSP, lo cual permite modificar fácilmente y de forma transparente los algoritmos de matching o la lógica reconfigurable. Permitiendo utilizar de forma dinámica componentes IP (Intelelectual Property) ya desarrollados. Otra ventaja es la utilización de tecnologías Web estándar, lo cual permite integrar el sistema diseñado en un gran número de entornos de explotación con mínimas modificaciones.

Los trabajos actuales y futuros incluyen el diseño y programación de la versión Java del cliente biométrico, y la implementación completa y estudio de rendimiento de las funciones *verifymatch* e *identifymatch* sobre la plataforma RC2000 de Celoxica.

Agradecimientos

El presente trabajo ha sido soportado parcialmente por el proyecto coordinado OPLINK TIN2005-08818-C04-03.

Referencias

1. Newcomer E.: Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley (2002).
2. Kaye D.: Loosely Coupled: The Missing Pieces of Web Services. RDS Press (2003).
3. Rodríguez, D., Sánchez J.M., Gómez J.A.: Reconfigurable Hybrid Architecture for web Applications. Proceedings Field-Programmable Logic and Applications, 13th International Conference, FPL'03 (2003) pág. 1091-1094.
4. V.K. Prasanna and A. Dandalis, "FPGA-based Cryptography for Internet Security". Online Symposium for Electronic Engineers. (2000).
5. R.M. Bolle, A.W. Senior, N. K. Ratha and S. Pankanti, "Fingerprint Minutiae: A Constructuive Definition". Lecture Notes in Computer Science.
6. Ying HAO, Tieniu TAN, Yunhong WANG, "An effective Algorithm For Fingerprint Matching".
7. O. Svedin, M. Öbrink, J. Bergenek, "Precise BioMatch™ FingerPrint Technology". White Paper April 2004.
8. Fons M., Fons F., Canyellas N., Lopez M., Cantó E., "Codiseño Hardware-software de un Algoritmo de Matching Biométrico", JCRA 2003, pag. 399-406.

Proposta para um Web Feature Service Temporal

Artur Rocha[1], Alexandre Carvalho[2,1]

[1]INESC Porto, Rua Dr. Roberto Frias 378, 4200-465 Porto
[2]FEUP, Rua Dr. Roberto Frias 378, 4200-465 Porto
{artur.rocha,alexandre.carvalho}@inescporto.pt

Resumo: Este artigo descreve uma abordagem à inclusão de suporte para lógica espaço-temporal em *Web Feature Services* (WFS). Este tipo de *Web Services* permite a interrogação de fontes heterogéneas de informação geográfica, num ambiente distribuído, retornando o resultado em formato *Geography Markup Language* (GML). No entanto, não é possível obter, a partir destes WFS, múltiplos estados para cada entidade (*feature*) geográfica, que resultariam da combinação de filtros espaciais e literais com predicados temporais. O uso destes predicados permite restringir o conjunto de estados contemplados no cálculo da resposta, ou mesmo apresentar um resultado espacial coalescido sobre uma dimensão temporal. Uma vez que a GML 3.1.1 já caracteriza os tempos válidos da informação geográfica, é viável a estruturação temporal das respostas devolvidas pelos WFS, sendo no entanto necessário enriquecer a sintaxe dos seus pedidos com a capacidade de utilizar predicados e operadores temporais. Este artigo propõe alterações à norma WFS, mais concretamente ao nível da linguagem de filtragem que utiliza, e apresenta uma solução efectiva para o suporte espaço-temporal nos SGBD subjacentes.

1 Introdução

1.1 Caracterização do suporte temporal nos SGBD

As bases de dados tradicionais são utilizadas para armazenar informação inter-relacionada num domínio do mundo real ou sintetizado. No que respeita aos aspectos temporais, os SGBD capturam a essência temporal correspondente à representação mais recente dos "factos" armazenados.

No domínio dos Sistemas de Informação Geográfica (GIS), os SGBD que lhe estão subjacentes manipulam também bases de dados do tipo referido, pelo que este tipo de tecnologia (GIS) é utilizada para armazenar e manipular o estado mais recente dos factos espaciais. A actualização do estado mais recente de um facto acarreta assim a perda da informação que descreve o seu estado anterior.

No sentido de permitir a gestão, numa base de dados, dos aspectos temporais dos factos, Date [1] considera duas aproximações: uma semi-temporal e outra verdadeiramente temporal. Na primeira, a captura da essência temporal de dados históricos é realizada com recurso a atributos temporais do tipo *timestamp*. Na

segunda, cada facto deve ter associado um intervalo que indica o período de tempo em que esse facto foi considerado válido no domínio representado, podendo ainda ser necessário manter o registo temporal da sua actualização na base de dados. Resulta daqui que uma abordagem temporal *full-fledged*, passa pela representação dos factos sob dois domínios temporais designados, respectivamente, por domínio dos tempos válidos [3] e pelo domínio dos tempos de transacção [4].

Importa referir que, segundo Date [1], a abordagem semi-temporal pode conduzir a graves problemas e dificuldades, particularmente na interrogação à base de dados, na medida em que as *queries* realizadas tomam graus de complexidade muito elevada. Estes factos resultam da inadequação dos SGBDs não temporais em fornecer o suporte para que a realização destas operações [5], por exemplo, tipos de dados temporais, operadores temporais (Allen [7], entre outros) e funções temporais [6].

Este trabalho recorre pois à utilização de um SGBD com suporte espaço-bitemporal, em desenvolvimento. A opção por esta solução resulta do facto das implementações temporais mais sólidas da actualidade, TimeDB [17] e TEMPOS [18], não possuírem suporte para dados espaciais.

1.2 Acesso interoperável a Informação Geográfica

A informação geográfica (IG) possui características únicas de referenciação, pelo que pode funcionar como um excelente elemento aglutinador de informação, ainda que esta resida em sistemas geograficamente distribuídos, que não tenham tido na sua concepção a preocupação de interoperarem. Essas capacidades de referenciação, aliadas à crescente utilização da *World Wide Web* como meio preferencial de ligação entre sistemas distribuídos, impulsionaram o desenvolvimento acelerado dos GIS e despertado a sua comunidade de utilizadores para a resolução das questões de interoperabilidade inerentes à heterogeneidade que os caracterizam.

O Consórcio OpenGeospatial (OGC) assumiu um papel preponderante na resolução dos referidos problemas de interoperabilidade, tendo produzido especificações abstractas como o modelo de referência OpenGIS e de implementação como o *Web Map Service* [14] (WMS), o *Web Feature Service* [13] (WFS) e a *Geography Markup Language* [10] (GML) entre muitas outras.

Tanto o WMS como o WFS foram concebidos para suportar o acesso remoto e interoperável a IG, mas enquanto o primeiro (WMS) permite aos clientes a sobreposição de mapas (representações da IG em formato de imagem) provenientes de várias fontes de informação geográfica, o WFS permite a interrogação destas mesmas fontes apresentando o resultado codificado na forma de GML. Se o primeiro é útil para representar grandes volumes de informação de uma forma leve e eficiente (uma imagem que os representa), é no segundo que encontramos o suporte necessário para a elaboração de *queries* com filtros que poderão conter predicados e operações espaço-temporais.

1.3 Suporte temporal nas normas OpenGIS

Actualmente, a dimensão temporal é tratada ainda de forma muito incipiente, pelos WFS e pelos serviços de catálogo (*Catalogue Service* [12]), que encaram esta componente como mais um atributo literal, utilizado nas filtragens que implementam.

Uma vez que a GML contém, na versão 3.1.1, um extensivo suporte temporal (na dimensão *validtime*, uma vez que não inclui tempos de transacção), é possível utilizar as primitivas temporais *gml:TimeInstant* e *gml:TimePeriod*, por forma a obter na resposta a um pedido WFS, *features* dinâmicas (*DynamicFeature* e *DynamicFeatureCollection*) que utilizam a propriedade *history* para expressar o seu desenvolvimento temporal, à custa de *time slices* (*gml:TimeSlice*) que capturam a evolução da *feature* ao longo do tempo.

Poder-se-ia pensar que este suporte é suficiente para lidar com a questão temporal, passando a responsabilidade de efectuar cálculos com lógica temporal (*validtime cross-product*, *validtime selection*, *validtime projection*, entre outros) [3] para a aplicação cliente.

No entanto, é opinião dos autores deste artigo, que a transposição da complexidade necessária para o suporte destas operações, do SGBD para a aplicação cliente, é desencorajadora da sua utilização, para além de se tornar extremamente ineficiente. Para além de os SGBD serem muito mais eficientes a realizar estas operações (possuem *cartridges* espaciais; lógica, operações e índices temporais), na maioria dos casos seriam pedidos ao WFS (e consequentemente transportados pela rede) muito mais dados (*features* geográficas) do que o necessário, uma vez que estes poderiam já vir coalescidos sobre a sua dimensão temporal.

Assim, torna-se necessário dotar os WFS da capacidade de filtrar os pedidos recorrendo a predicados e operadores temporais. Ao fazê-lo, estaremos automaticamente a enriquecer as capacidades dos serviços de catalogo [12], uma vez que ambos remetem para a *Common Query Language* (CQL) a definição de uma gramática para filtros. Neste artigo são propostas alterações à *Filter Encoding Implementation Specification* [11] que é uma codificação em XML derivada da CQL.

Finalmente, com base nas alterações propostas, será realizada uma prova de conceito baseada na adaptação de uma implementação WFS existente e suportado numa implementação de SGBD espaço-temporal.

2 Definição do suporte temporal

Considere-se uma relação *P*, não temporal, com o esquema (*nome*, *capital*, *fronteira*), que permite representar o estado de países. Neste esquema, o atributo *fronteira* possibilita uma representação poligonal bidimensional, enquanto que os atributos *nome* e *capital* correspondem a uma designação textual. Considere-se ainda que a instância de *P* contém três tuplos que representam o estado actual de três países (Portugal, Polónia e República Checa). Esta relação *P*, temporalmente designada por *snapshot*, permite capturar apenas o estado actual dos factos. Por exemplo, a mudança da cidade num destes três países, acarreta uma operação de actualização do tuplo correspondente, do atributo *capital*, que passa a conter a designação da nova capital.

Neste processo ocorre que a designação anterior (que se mantinha válida até ao instante de actualização) é perdida. O mesmo acontece com a alteração dos limites da fronteira de um desses países, facto que corresponde, na instância da relação P, à substituição da descrição geométrica da fronteira para esse país, pela descrição geométrica da nova fronteira.

Se o objectivo consiste em manter a informação de histórico acerca de países, então, uma das soluções consiste em temporalizar os factos. Para este efeito, neste artigo adopta-se o modelo *Bitemporal Conceptual Data Model* (BCDM), proposto por Jensen *et al*. [2], embora desse modelo apenas seja contemplado o domínio dos tempos válidos.

Assim, utilizando o BCDM no domínio dos tempos válidos, a uma relação *R,* caracterizada pelos seu conjunto de (A_1, A_2, ... A_n) é acrescentado um atributo temporal T^v ficando $R^v = (A_1, A_2, ... A_n \mid T^v)$. Cada tuplo desta relação $\{a_1, a_2, ... a_n \mid t^v\}$ contém, associado ao conjunto de valores não temporais (a_i), um valor temporal do domínio dos tempos válidos (t^v). No BCDM cada valor de t^v pode conter um conjunto de instantes e/ou de intervalos passados, em que o facto foi considerado válido, ou futuros, quando que se pensa que um facto será válido futuramente.

A transposição deste modelo conceptual para um modelo lógico de base de dados envolve que sejam criados, para cada facto representado, tantos tuplos quantos os intervalos e instantes distintos, que caracterizam o valor de t^v de um tuplo pertencente a uma relação R^v expressa através do modelo BCDM.

Isto significa que a versão unitemporal da relação P corresponde a P_v = (*nome*, *capital*, *fronteira, t^v*), onde t^v representa o intervalo em que os atributos não temporais são considerados válidos. Numa base de dados unitemporal, o atributo de tempos válidos, t^v de um tuplo é representado através de dois valores numéricos que significam os *chronons* [16] limitadores da validade do facto representado nesse tuplo. A título de exemplo considerem-se alguns dos tuplos pertencentes à relação unitemporal P_v:

{'Portugal', 'Coimbra', <geometria$_1$>, [1143-1200)}

{'Portugal', 'Lisboa', <geometria$_1$>, [1200-1250)}

{'Portugal', 'Lisboa', <geometria$_1$>, [1250-1300)}

{'Portugal', 'Lisboa', <geometria$_2$>, [1300-forever)}

Estes quatro tuplos capturam a realidade de que Portugal teve como capital a cidade de Coimbra entre [1143-1200), e que durante esse tempo as fronteiras do país apresentavam a configuração geométrica expressa por geometria$_1$. Já a partir de 1200 e até 1300 a capital do país foi Lisboa e durante esse período manteve a configuração da sua fronteira. No entanto, esta informação encontra-se representada através de dois tuplos, temporalmente contíguos. Finalmente, a partir de 1300 a fronteira sofreu alterações passando a ter a configuração geometria$_2$ que, pelo facto do limite superior ser um instante futuro, indeterminado, faz deste tuplo o facto actual, isto é, o facto que à data actual se mantém verdadeiro.

Esta abordagem temporal dos factos proporciona que os SGBDs temporais possam manipular múltiplos estados de factos e possibilitam respostas para questões que envolvem cálculos sobre cada estado, mas também cálculos que se realizam sobre vários estados. Se considerarmos a relação P_v contendo factos representados apenas

sobre Portugal, é possível realizar questões espaço-temporais, tais como: (1) Quais as datas em que Coimbra deixou de ser a capital? (2) Em que períodos foi Lisboa capital? (3) Qual é a configuração actual da fronteira Portuguesa?

Se considerarmos a relação P_v contendo factos sobre os países referidos, é possível realizar questões espaço-temporais, tais como: (4) Para o séc. XV, que países eram vizinhos da Polónia? (5) Alguma vez Portugal e a Polónia tiveram fronteiras adjacentes? (6) Quando existiu a Polónia? (7) Ao longo dos tempos, qual foi a maior área territorial da Polónia?

Importa referir que, segundo Snodgrass [5], estas questões podem ser realizadas com recurso a SQL em SGBDs não temporais. No entanto, o autor refere que, apesar da linguagem SQL ser efectivamente poderosa para responder a questões que envolvem o estado actual (numa perspectiva *snapshot*), não proporciona o suporte adequado para *queries* temporais, alterações temporais e definição de restrições temporais. Este autor refere a extrema dificuldade de determinar, em SQL, os resultados correctos para as colunas temporais. Segundo Snodgrass, a solução passa por transferir estes cálculos para o SGBD, através de extensões temporais ao SQL.

1.4 Descrição da semântica temporal

Considerando um SGBD temporal e as relações P e P_v, é importante referir que a resposta ao conjunto de questões levantadas envolve diferentes semânticas temporais, nomeadamente, *upward compatibility*, *temporal upward compatibility*, semântica sequencial e semântica não sequencial. Por exemplo, a questão 3 pode ser realizada quer sobre P, quer sobre P_v, envolvendo o mesmo SQL:

```
SELECT fronteira FROM P WHERE nome = 'Portugal'
```

Neste caso, *upward compatibility*, um SGBD temporal responde a questões realizadas em SQL *standard* sobre relações não temporais como se fosse um SGBD não temporal, proporcionando a migração de aplicações que trabalham em SQL standard para SGBDs temporais sem a necessidade de alteração do código.

Se a pergunta for realizada sobre a relação P_v, a semântica temporal envolvida é *temporal upward compatibility*. A resposta a esta questão é não temporal e corresponde ao cálculo efectuado apenas sobre o estado actual da relação P_v. Todos os outros estados correspondem a factos que, na actualidade, já não são válidos, e são, portanto, ignorados.

Na semântica sequencial, cada estado do resultado é calculado à custa de um estado da relação P_v, sendo que os operadores relacionais são temporais. Estas questões envolvem o uso do predicado temporal, `VALIDTIME`, que pode ser seguido por um período de interesse. Por exemplo:

```
VALIDTIME PERIOD [1400-forever) SELECT * FROM Pv WHERE capital =
'Lisboa'
```

Esta questão é similar à questão 2, mas restringe o período de interesse a partir do ano 1400.

Na semântica não-sequencial a informação de cada estado do resultado é calculada a partir de múltiplos estados da relação P_v. Adicionalmente, os operadores relacionais envolvidos são não temporais. Por exemplo, na questão 7, com o SQL temporal:

```
NONSEQUENCED VALIDTIME SELECT validtime(Pv),
max(sdo_geom.sdo_area(fronteira, 0.005)) FROM Pv WHERE nome =
'Polónia';
```

só é possível determinar qual é a maior área se forem realizadascomparações com as áreas que constam nos vários estados (tuplos), para os factos que dizem respeito à Polónia. As questões que utilizam semântica não-sequencial obrigam à utilização de um predicado temporal, `NONSEQUENCED VALIDTIME`, que pode ser seguido por um período de interesse.

3 Suporte temporal em WFS

De modo a dotar um WFS com o suporte temporal descrito torna-se é necessário que o WFS disponibilize a utilização de:

- predicados temporais que definem a semântica temporal com que o GIS temporal irá calcular a resposta;
- operadores temporais entre intervalos e entre instantes que actuam sobre o *validtime* de cada tuplo/*feature* contida no GIS temporal;
- operações temporais que actuam sobre resultados quando estes contêm dimensão temporal, como por exemplo, a operação de *coalesce*;
- suporte para a caracterização temporal dos resultados devolvidos pelo WFS aos clientes, isto é de *features* com um *validtime* (já presente no GML 3.1.1).

O suporte de predicados temporais, adicionados a um pedido, realizado por um cliente, prende-se com a necessidade de definir a semântica temporal com que a resposta deve ser calculada. Os predicados temporais contemplados são `SEQUENCED` e `NONSEQUENCED`, sendo que ambos podem ser seguidos por um intervalo temporal de interesse, tendo esse intervalo semânticas distintas nas duas situações. Na primeira situação – `SEQUENCED` - a inclusão de um intervalo tem o significado de restringir o universo de *features* candidatas a serem incluídas na resposta àquelas cujo *validtime* está contido nesse intervalo. Na segunda situação – `NONSEQUENCED` - a inclusão de um intervalo após o predicado tem o significado de caracterizar temporalmente as *features* que resultam do pedido.

A omissão de predicados temporais corresponde à resolução da pergunta com recurso a operadores relacionais não-temporais, sem utilizar o suporte temporal descrito nesta secção.

A utilização do predicado `SEQUENCED` activa a utilização dos operadores de álgebra relacional temporal [3]. Finalmente, a utilização do predicado `NONSEQUENCED` resulta numa utilização dos operadores relacionais não-temporais, que podem actuar sobre o *validtime* de cada feature, encarando-a desprovida de semântica temporal. O resultado de um pedido calculado com esta semântica é não-temporal excepto se, conforme foi referido, se definir um intervalo temporal após o termo `NONSEQUENCED`.

A utilização dos operadores temporais de Allen [7], tais como *meets*, *contains*, *precedes* e *overlaps*, permite restringir as *features* que resultam de um pedido. Estes operadores actuam sobre o *validtime* de cada *feature* confrontanto-o com instantes e intervalos. Por exemplo, o pedido para obter o nome da cidade que se manteve capital

de Portugal, durante o período [1155-1160), não havendo neste período qualquer alteração de outros atributos, tem como filtro:

```
<ogc:Filter>
  <ogc:Sequenced/>
    <ogc:TContains>
      <ogc:QueryValidtime>ValidTime</ogc:QueryValidtime>
      <gml:TimePeriod>
        <gml:begin>
          <gml:TimeInstant>
            <gml:timePosition>1155</gml:timePosition>
          </gml:TimeInstant>
        </gml:begin>
        <gml:end>
          <gml:TimeInstant>
            <gml:timePosition>1160-31-12T23:59:59.999</gml:timePosition>
          </gml:TimeInstant>
        </gml:end>
      </gml:TimePeriod>
    </ogc: TContains>
</ogc:Filter>
```

Se esta pergunta for realizada sobre a relação unitemporal P_v, a resposta consistirá no tuplo onde está representado o facto de que Coimbra foi capital de Portugal, porque o *validtime* desse tuplo contém (ogc:TContains) o *validtime* definido na *query* – [1155-1160). Também se considera útil a inclusão da operação de *coalesce* no suporte temporal a incluir no WFS pela possibilidade de solicitar resultados coalescidos pela dimensão dos tempos válidos, isto é, para a obtenção dos resultados numa forma em que os intervalos *validtime* são maximizados para *features* que se intersectam temporalmente e são *value-equivalent* para atributos não temporais [8]. Por exemplo a operação de *coalesce* da relação unitemporal P_v resulta em:

{'Portugal', 'Coimbra', <geometria$_1$>, [1143-1200) }

{'Portugal', 'Lisboa', <geometria$_1$>, [1200-1300) }

{'Portugal', 'Lisboa', <geometria$_2$>, [1300-forever) }

O Filtro usado nesta situação deve corresponder a:

```
<ogc:Filter><ogc:Sequenced><ogc:Coalesce/></ogc:Sequenced></ogc:Filter>
```

Onde os tuplos de P_v com o *validtime* de [1200-1250) e de [1250-1300), pelo facto dos seus atributos não temporais, *nome*, *capital* e *fronteira*, serem equivalentes (quando comparado os valores atributo a atributo) resultam num tuplo coalescido temporalmente, [1250-1300).

Finalmente, considera-se necessário contemplar que os resultados devolvidos por um WFS, possam conter, para cada *feature* devolvida, um intervalo temporal *validtime*. Para este efeito, conforme foi referido na introdução, a versão 3.1.1 da especificação GML [10], contempla o suporte para a caracterização temporal de uma *feature* geográfica ou de um outro qualquer objecto, através da inclusão do *schema temporalReferenceSystems.xsd,* que por sua vez inclui o *schema temporal.xsd*. Neste último, está já definido o elemento *gml:validTime*, que compreende a caracterização temporal de um instante e de um período, por meio de múltiplas formas de representação destas grandezas temporais.

4 Especificação das alterações à implementação de um WFS

Propõe-se que o suporte para predicados temporais, operadores temporais e funções temporais seja adicionado à especificação *Filter Encoding Implementation Specification* [11] (*Filter*), sendo esta a definição de uma codificação em XML para a realização de expressões de filtragem, baseada na definição *Backus Naur Form* [15] (BNF) da OCG *Common Catalog Query Language* [12] (CQL), tal como é definida na OGC *Catalogue Service Implementation Specification* [12]. Fundamenta-se esta opção no facto da *Filter*:

- ser utilizada na constituição de filtros que são parte integrante de perguntas realizadas a um WFS [13], através de um pedido *GetFeature* (Figura 1). Destas perguntas resultam conjuntos de *features*, que obedecem aos critérios definidos na filtragem.
- ser utilizada nos pedidos *LockFeature, GefFeatureWithLock,* bem como nas operações *Update* e *Delete*, que estarão disponíveis se o WFS apresentar capacidades transaccionais (WFS-T). De facto, verificou-se que a definição dos pedidos identificados compreende a utilização de Filtros de modo a restringir o conjunto de *features* alvo da operação ou do pedido.
- ser utilizada tanto no *Catalogue Service* como no *Web Feature Service*: actualmente, o suporte temporal previsto consiste somente na utilização de filtros literais sobre os metadados que foram utilizados para catalogar os OpenGIS Web Services (OWS) existentes. Através desta funcionalidade é possível, por exemplo, inquirir um *Catalogue Service* sobre os WFS e os WMS que disponibilizam dados hidrológicos sobre Portugal, entre 1 de Janeiro de 1996 e 31 de Dezembro de 2003.

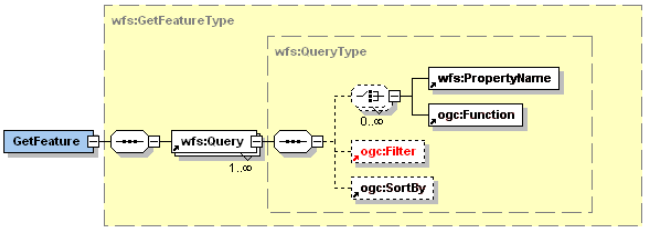


Figura 1: *schema* de um pedido *GetFeature*

Torna-se ainda necessário propor alterações à componente do WFS Capabilites, (Figura 2), que descreve as suas capacidades de filtragem, por forma a informar o cliente que o WFS em questão possui capacidades de filtragem utilizando predicados, operadores e operandos na dimensão unitemporal *validtime*.

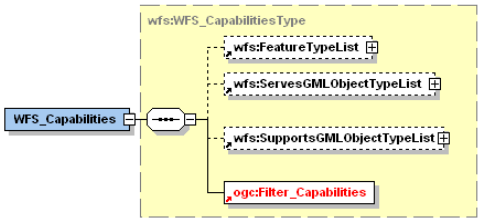


Figura 2: *schema* do pedido *getCapabilities*

Ao nível do *ogc:Filter*, o *schema* permite uma escolha de operações (ver Figura 3) de filtragem espaciais (*ogc:spatialOps*), de operações de filtragem de comparação (*ogc:comparisonOps*), de operações de filtragem pela identificação de features (*ogc:_Id*) e de operações lógicas (*ogc:logicOps*), sendo que estas últimas permitem a composição de várias operações através dos operadores *ogc:and*, *ogc:or* e *ogc:not*.

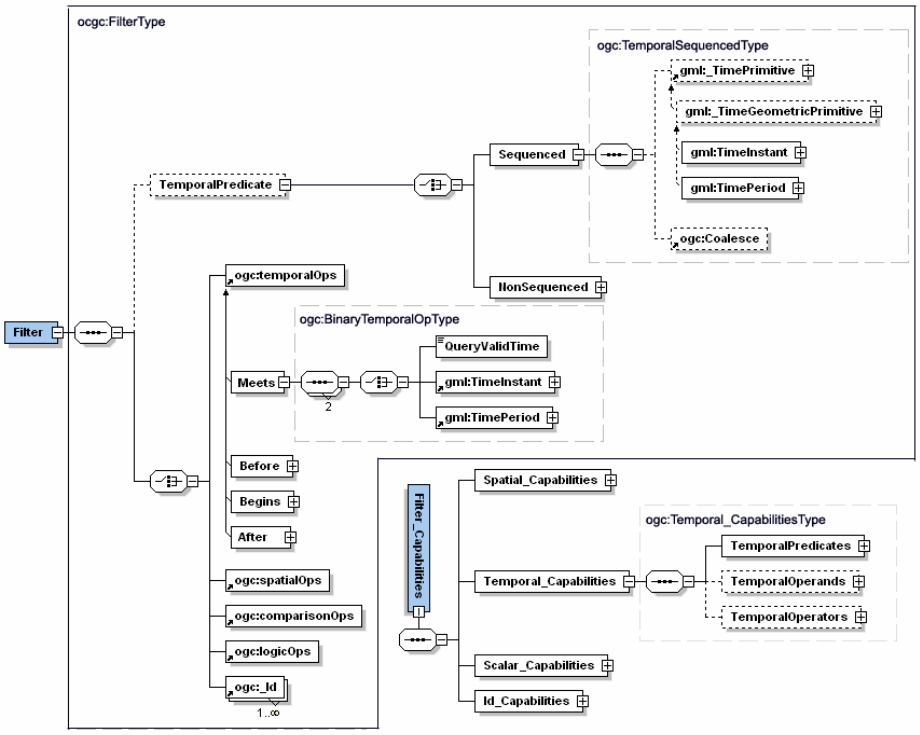


Figura 3: Proposta de alteração dos *schemas Filter* e *Filter_Capabilities*

As operações espaciais permitem filtragem espaciais sobre as propriedades geográficas de uma *feature*, por exemplo, *ogc:BBOX*, *ogc:Disjoint* e *ogc:DWithin*.

As operações de comparação, por exemplo, *PropertyIsEqual*, *PropertyIsLike*, *PropertyIsLessThan*, permitem realizar filtragem de *features* pela comparação entre o valor da propriedade e um ou mais literais (dependendo do número de operandos que a comparação permite).

As operações lógicas compreendidas no *ogc:Filter* permitem realizar operações lógicas binárias ou unárias entre operandos que podem ser operações espaciais, de comparação ou mesmo outras operações lógicas.

Finalmente, o *ogc:Filter* contempla a filtragem de uma *feature* pelo seu atributo identificador, através da definição dos elementos *ogc:FeatureId* (*deprecated*) e *ogc:GmlObjectId*. Sobre esta especificação propomos:

– possibilitar a inclusão de um novo elemento, opcional, designado *ogc:TemporalPredicate,* que precede todas as operações previstas na *ogc:Filter*.

Na constituição de um filtro, a ausência deste elemento equivale à utilização de um WFS não temporal, mantendo assim *backward compatibility*;

- acrescentar um novo tipo de operações temporais, *ogc:TemporalOps*, correspondente aos operadores de Allen [7], binários, que se dividem em operadores entre intervalos e operadores entre instantes. Os primeiros (*After*, *Before*, *Begins*, *BegunBy*, *During*, *EndedBy*, *Ends*, *Meets*, *MeetBy*, *OverlappedBy*, *TContains*, *TOverlaps*), aceitam, como operandos, valores do tipo *gml:TimePeriod.* Já os operadores entre instantes (*Preceedes*, *TOverlaps*, *Meets*, *TContains*) aceitam operandos do tipo *gml:TimeInstant*. Em alguns operadores utiliza-se a letra *'T'* precedendo a designação do operador, por exemplo *ogc:TOverlaps*, por forma a distinguir das operadores espaciais com o mesmo nome.
- definir nas operações lógicas mais um tipo de operandos: as operações temporais.

O novo elemento *ogc:TemporalPredicate*, opcional, é composto por uma escolha entre os elementos *ogc:Sequenced* e *ogc:NonSequenced*. A utilização do primeiro equivale à especificação de semântica temporal `SEQUENCED` no cálculo da resposta, e pode conter dois elementos opcionais: um *gml:_TimePrimitive* e um elemento *ogc:Coalesce*. Neste contexto, a utilização do *gml:_TimePrimitive*, que pode ser um *gml:TimeInstant* ou um *gml:TimePeriod*, tem o significado de restringir temporalmente o cálculo da resposta às *features* cujo *validtime* está contido nesse elemento temporal. A utilização do elemento *ogc:Coalesce* significa que os resultados devem vir *coalescidos* sobre a dimensão dos tempos válidos.

O elemento *ogc:NonSequenced* equivale à especificação de semântica temporal `NONSEQUENCED`, pelo que o SGBD subjacente deve calcular a resposta usando lógica não temporal, tratando o atributo *validtime* das *features* como um qualquer atributo não temporal. O elemento *ogc:NonSequenced* pode ainda conter um elemento *gml:_TimePrimitive*. Nesta situação, apesar do cálculo ser à custa de lógica não temporal, cada *feature* do resultados deve ser temporalizada com um *validtime* equivalente ao elemento representado pela *gml:_TimePrimitive*. Este é um processo de promover resultados *snapshot* a resultados *validtime*.

Ao nível do *schema* do *ogc:Filter_Capabilities* a proposta de alteração é uma consequência das alterações definidas para o *ogc:Filter*, isto é, no *Filter_capabilities* propõe-se adicionar mais um elemento, *ogc:TemporalCapabilities*, que lista as capacidades temporais do filtro, e que consistem nos predicados temporais (*ogc:TemporalPredicates*), nos operandos temporais (*ogc:TemporalOperands*) e nas operações temporais (*ogc:TemporalOperators*). Neste *schema* está ainda definido que a utilização de capacidades temporais obriga à utilização de um predicado temporal.

A Figura 3 resume as propostas apresentadas neste artigo, de alteração à definição dos *schemas ogc:Filter* e *ogc:Filter_Capabilites*, no sentido de incluir o suporte temporal na dimensão de tempos válidos. Todos os *schemas* de base utilizados poderão ser encontrados no endereço *http://schemas.opengis.net/*. As alterações propostas, baseiam-se na versão 1.1.0 da *Filter* e poderão ser obtidas na sua totalidade a partir do endereço *http://gis.inescporto.pt/schemas/filter/1.1.0/*.

5 Resultados esperados

Por forma a ilustrar o que seria esperado de um filtro espaço-temporal, que seria utilizado num pedido WFS, considere-se o seguinte exemplo.

```
<ogc:Filter>
  <ogc:Sequenced>
        <ogc:Coalesce/>
  </ogc:Sequenced>
  <ogc:and>
    <ogc:during>
      <ogc:QueryValidtime>ValidTime</ogc:QueryValidtime>
      <gml:TimePeriod>
        <gml:begin>
          <gml:TimeInstant>
            <gml:timePosition>1901-01-01</gml:timePosition>
          </gml:TimeInstant>
        </gml:begin>
        <gml:end>
          <gml:TimeInstant>
            <gml:timePosition frame="#ISO-8601">2000-31-12T23:59:59.999
            </gml:timePosition>
          </gml:TimeInstant>
        </gml:end>
      </gml:TimePeriod>
    </ogc:during>
    <ogc:BBOX>
      <gml:PropertyName>Fronteira</gml:PropertyName>
      <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
         <gml:lowerCorner>13.345 31.213</gml:lowerCorner>
         <gml:upperCorner>35.345 42.573</gml:upperCorner>
      </gml:Envelope>
      </ogc:BBOX>
  </ogc:and>
</ogc:Filter>
```

Este filtro utiliza lógica temporal sobre a dimensão *validtime* e restringe as *features* resultantes àquelas cujo *validtime* contém o século XX (no calendário Gregoriano, ISO-8601) e cuja localização geográfica se encontra compreendida na *bounding box*, [{13.345 31.213}, {35.345 42.573}], expressa em coordenadas geográficas no SRS WGS84 (EPSG:4326). Pretende-se ainda que os resultados devolvidos pelo WFS sejam coalescidos sobre a dimensão dos tempos válidos.

6 Conclusões

Este artigo apresenta uma proposta de inclusão de suporte unitemporal `VALIDTIME` nos filtros de WFS. Este suporte caracteriza-se pela utilização de predicados, operandos e operadores temporais, sobre a dimensão dos tempos válidos e deve estar contemplado na definição dos *schemas ogc:Filter* e *ogc:Filter_Capabilites*. Estes *schemas* são ainda utilizados pelo *Catalogue Service*, por forma a filtrar os OpenGIS Web Services (OWS) registados num serviço de catálogo pelos seus metadados, providenciando assim um suporte temporal transversal a todos os OWS considerados.

A obtenção dos resultados (*Features* e *FeatureCollections*) temporais encontra-se garantida, na medida em que o GML 3.1.1, disponibiliza suporte temporal através de tipos temporais (*TimeInstant* e *TimePeriod*) e de *features* com atributos dinâmicos.

As alterações propostas serão validadas por recurso a uma implementação específica de um "Temporal WFS" por forma a confirmar os testes realizados sobre a implementação espaço-temporal já realizada ao nível do SGBD.

7 Bibliografia

[1] Date, C., *An Introduction to Database Systems*, 7th edition, Addison-Wesley, Capítulo 22, pp. 730-768, 2000

[2] Jensen, C., *Temporal Database Management*, PhD. Thesis, 2000

[3] Snodgrass, R., Böhlen, M., Jensen, C., Steiner, A., *Adding Valid Time to SQL/Temporal*, SQL/Temporal Change Proposal, ANSI X3H2-96-501r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-146r2, 1996.

[4] Snodgrass, R., Böhlen, M., Jensen, C., Steiner, A., *Adding Transaction time to SQL/Temporal*, SQL/Temporal Change Proposal, ANSI X3H2-96-502r2, ISO/IEC JTC1/SC21/WG3 DBL MAD-147r2, 1996

[5] Zaniolo, C., Ceri, S., Faloustos, C., Snodgrass, R. Subrahmanian, V., Zicari, R*., Advanced Database Systems*, Morgan Kaufman, 1997.

[6] Steiner, A., *A Generalization Approach to temporal Data Models and Their Implementations*, Phd, Zurich, 1998

[7] Allen, J. *Maintaining Knowledge about Temporal Intervals*, Comm. ACM 26(11), 1983.

[8] Böhlen, M., Snodgrass R., Soo, M., *Coalescing in Temporal Databases*, Proceedings of International Conference on Very Large Databases, 1996.

[9] ISO 19108:2002, *Geographic Information – Temporal Schema*, Ref: N1224, 2002

[10] ISO/TC 211/WG 4/PT 19136, OGC GML RWG, *Geographic information – Geography Markup Language (GML)*, versão 3.1, 2004

[11] OGC 04-095, *Filter Encoding Implementation Specification*, versão 1.1.0 , Vretanos, P., 2005

[12] OGC 04-021r3, *OGC Catalogue Services Specification*, versão 2.0.0, Nebert, D., Whiteside , A., Vretanos, P., 2005

[13] OGC 04-094, *OGC Web Feature Service Implementation Specification*, versão 1.1.0, A., Vretanos, P., 2005

[14] OGC 04-024, *Web Map Service*, versão 1.3, Beaujardiere, J., 2004

[15] ISO/IEC 14977:1996, *Information technology - Syntactic metalanguage - Extended BNF*, 1996

[16] Jensen, C., *et al.*, *A consensus glossary of temporal data base concepts*, ACM SIGMOD Records, vol. 23, 1994

[17] Steiner, A., *A Generalization Approach to temporal Data Models and Their Implementations*, Phd, Zurich, 1998

[18] Dumas, M., Fauvet, C., Scholl, P., *TEMPOS: A Temporal Database Model Seamless Extending ODMG*, LSR-IMAG, University of Grenoble, Phd, 2000

Utilização da Tecnologia XML no Desenvolvimento de Arquitecturas Específicas

Rui Rodrigues[1], Ricardo Ferreira[2], João M. P. Cardoso[1,3]

[1] Universidade do Algarve, Campus de Gambelas, 8000-117, Faro, Portugal
[2] Departamento de Informática, Universidade Federal de Viçosa,
Viçosa 36570 000, Brazil
[3] INESC-ID, 1000-029, Lisboa, Portugal
{rrodrigues@ualg.pt, cacau@dpi.ufv.br , jmpc@acm.org}

Resumo. Este artigo apresenta dois exemplos da utilização de XML na investigação e desenvolvimento de arquitecturas computacionais específicas. A tecnologia XML tem sido utilizada pelos autores deste artigo para descrever estruturas computacionais, e como meta-linguagem para criação de dialectos XML que facilitam a especificação de determinadas funcionalidades. Transformadores XSL têm sido utilizados para a geração de descrições em Java, VHDL, etc. As experiências têm mostrado muitas vantagens na utilização da tecnologia XML nos contextos abordados. O artigo ilustra a utilização de XML, foca as vantagens principais, e comenta as facilidades consideradas mais importantes.

1. Introdução

A utilização de XML [1] tem sido apreciada por diversos autores em cenários para os quais a linguagem dificilmente foi considerada. Tal deve-se de facto à facilidade de criar linguagens específicas baseadas em XML e às capacidades da tecnologia XML em transformar especificações em outras representações. Como exemplos do que acaba de ser dito, refira-se que o XML tem sido utilizado para descrever comportamentos [2], máquinas de estados finitos [3], arquitecturas do conjunto de instruções em microprocessadores [4], etc.

A integração na tecnologia XML de um motor de transformação (XSLT [5]) fornece uma solução final adequada ao desenvolvimento incremental de especificações e de mecanismos de transformação necessários. Note-se contudo que as linguagens baseadas em XML são, neste contexto, apenas indicadas para representações intermédias, pois a sua legibilidade sofre imenso com o número de elementos no ficheiro XML.

No nosso caso, a tecnologia XML tem sido utilizada como suporte à investigação de técnicas de compilação para arquitecturas específicas [6][7][8] e em ambiente de investigação de novas arquitecturas reconfiguráveis, baseadas em agregados de células em que cada célula é um elemento de processamento [9][10]. Nestes dois casos o XML tem sido utilizado para representar comportamentos e estruturas e tem-

se revelado extremamente importante. Nos ambientes de projecto de arquitecturas específicas e reconfiguráveis, várias ferramentas (compiladores, simuladores, etc.) são utilizadas. O XML tem sido também utilizado como formato intermediário de comunicação entre ferramentas. Como as tecnologias e os padrões da área estão em constante refinamento, o uso de um formato incremental, baseado em XML, garante uma maior independência de tecnologia, capacidade de reutilização e aumenta das possibilidades de expansão e readaptação dos ambientes de projecto.

A investigação de arquitecturas específicas tem o papel fundamental nos actuais e nos futuros sistemas embebidos [11][12]. Tal deve-se em parte à demanda de altos níveis de desempenho e de economia de consumo de energia muitas das vezes apenas compatíveis com soluções especializadas.

Este artigo está organizado da seguinte forma. A próxima secção apresenta a utilização de XML para arquitecturas específicas. A secção 3 ilustra a utilização de XML como suporte de representação das estruturas finais na compilação de Java bytecodes para FPGAs (*Field-Programable Gate Arrays*). Por último, a secção 4 conclui o artigo e traça alguns comentários acerca das potencialidades desta tecnologia.

2. Avaliação antes do Fabrico de Arquitecturas de Agregados com Granulosidade Grossa

As arquitecturas reconfiguráveis, baseadas em agregados de células de granulosidade grossa, têm provado a apetência para aceleração do desempenho de sistemas computacionais [13]. Tem sido dado ultimamente algum relevo a arquitecturas com comportamento *data-driven* [14]. Nestas, as operações são despoletadas aquando da presença de novos dados nos operandos, à semelhança das célebres máquinas *dataflow* [15]. Desta forma, o controlo acaba por ser distribuído e a computação em pipelining natural.

A **Fig. 1**(a) apresenta as entradas e saídas de uma unidade funcional (FU) com comportamento *data-driven* típica. Para além das entradas/saídas usuais existem os sinais que controlam o funcionamento *data-driven*. Estes sinais constituem os sinais necessários para implementar o protocolo de *handshake* ("aperto-de-mão") utilizado. A **Fig. 1**(b) ilustra a integração da FU numa célula hexagonal da arquitectura e a **Fig. 1**(c) ilustra um exemplo de uma arquitectura hexagonal. Em arquitecturas deste tipo existem várias topologias possíveis e várias formas de comunicação de dados entre as células. São reconhecidas para as diferentes topologias vantagens e desvantagens, mas o impacto destas no desempenho final necessita de esquemas de avaliação rápida, e por isso se revela de extrema importância a investigação e desenvolvimento em ambientes adequados a essa finalidade.

Embora estas arquitecturas tenham vindo a demonstrar algumas vantagens, não têm sido realizados estudos sistemáticos que permitam avaliar certas propriedades antes da opção por um determinado aspecto ao implementar a arquitectura. Neste âmbito temos vindo a desenvolver um ambiente, designado por EDA, que permita o estudo prévio deste género de arquitecturas [9][10]. Na **Fig. 2** é apresentada uma vista geral desse ambiente. Para o desenvolvimento do ambiente temo-nos socorrido da

tecnologia XML de forma a especificar a arquitectura, os parâmetros possíveis de exploração, e os exemplos de teste.

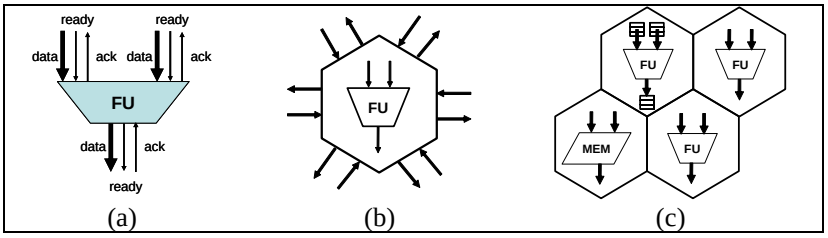


Fig. 1. (a) Unidade funcional *data-driven* (FU) com duas entradas e uma saída; (b) Célula hexagonal com a FU; (c) Agregado hexagonal (FUs podem ter FIFOs nas entradas e nas saídas)

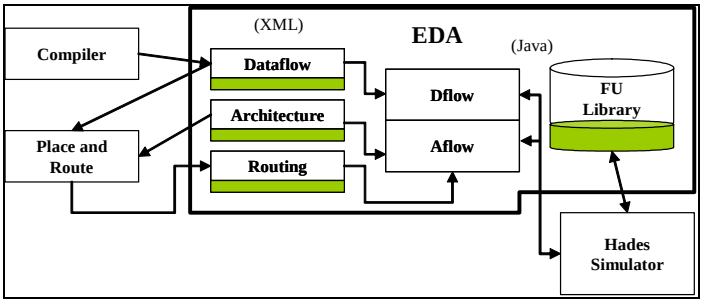


Fig. 2. Ambiente de exploração de arquitecturas do tipo *data-driven* (fonte: [10])

A especificação do *dataflow* é realizada em XML. O XML é também utilizado para especificar a colocação e o encaminhamento dessa especificação na arquitectura sob teste. Cada operador no *dataflow* é directamente implementado pelas unidades funcionais (FUs) existentes nas células da arquitectura. A **Fig. 3** apresenta um exemplo *dataflow*. Na **Fig. 4** pode-se ver a representação em XML do exemplo. A **Fig. 5** ilustra uma arquitectura hexagonal, parte da sua representação XML, o mapeamento do exemplo da **Fig. 3** na arquitectura apresentada e a descrição XML do encaminhamento para esse mapeamento.

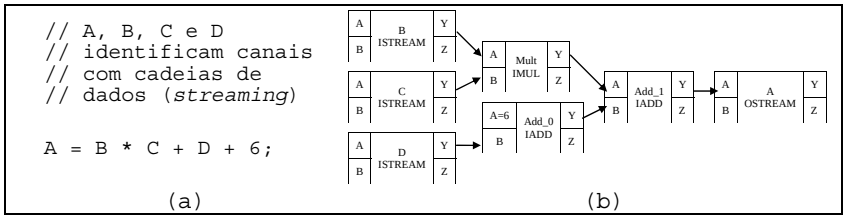


Fig. 3. Exemplo simples: (a) código do exemplo; (b) grafo de uma implementação *dataflow*

Para especificar o comportamento de cada FU, o ambiente utiliza a linguagem Java. A simulação da arquitectura é realizada pela ferramenta HADES [16][17][18] tendo por base uma biblioteca desenvolvida de FUs com comportamento *data-driven*.

```
…
<DESIGN  trace="true">
  <COMPONENT unit="IO" operation="ISTREAM" name="B" file="Data_B.txt"/>
  <COMPONENT unit="IO" operation="ISTREAM" name="C" file="Data_C.txt"/>
  <COMPONENT unit="IO" operation="ISTREAM" name="D" file="Data_D.txt"/>
  <COMPONENT unit="ALU" operation="IMUL" name="Mult"/>
  <COMPONENT unit="ALU" operation="IADD" name="Add_0">
      <PORT name="A" value="6" />
  </COMPONENT>
  <COMPONENT unit="ALU" operation="IADD" name="Add_1"/>
  <COMPONENT unit="IO" operation="OSTREAM" name="A" file="Data_A.txt"/>
  <SIGNAL name="wire1">
      <SOURCE name="B" port="Y"/>
      <SINK name="Mult" port="A"/>
  </SIGNAL>
  …
  <SIGNAL name="wire6">
      <SOURCE name="Add_1" port="Y"/>
      <SINK name="A" port="A"/>
  </SIGNAL>
</DESIGN>
```

Fig. 4. Exemplo simples: parte da representação XML do grafo da **Fig. 3**(b)

Como se pode ver pela **Fig. 5**(b), a definição da arquitectura utiliza regras subjacentes a elementos XML que permitem especificar agregados de forma simplificada. O elemento:

```
<ARRAY type=”HEXA”>
```

utiliza a propriedade “HEXA” que define a topologia da arquitectura. As topologias aceites até ao momento englobam também a “OCTAL” e a “QUADRANGULAR”.

Existem regras que definem vários elementos da arquitectura homogéneos:

```
<LAYOUT length=”3” width=”3” symbol=”I”/>
```

que define uma arquitectura de 3x3 elementos do símbolo “I” (este símbolo havia sido definido em linhas de XML anteriores como representante de células de I/O, ver **Fig. 5**(b)).

As duas linhas seguintes:

```
<LAYOUT x="2" y="0" symbol="M"/>

<LAYOUT x="2" y="2" symbol="M"/>
```

redefinem as células (2, 0) e (2, 2) como sendo do símbolo “M” que anteriormente havia sido definido como correspondente às células do tipo MEM (ver **Fig. 5**(b)).

O elemento final:

```
<LAYOUT x="1" symbol="A"/>
```

indica que todas as células da linha 1 são do tipo “A”, previamente definido como ALU. Desta forma obtém-se a arquitectura hexagonal representada na **Fig. 5**(a). A definição da arquitectura também inclui as propriedades referentes ao tipo de

comunicação entre células da arquitectura (de entrada, de saída, bidireccional, etc.) e ao número de interligações. É considerado que os recursos de comunicação entre células são semelhantes para todos os lados de cada célula da arquitectura.

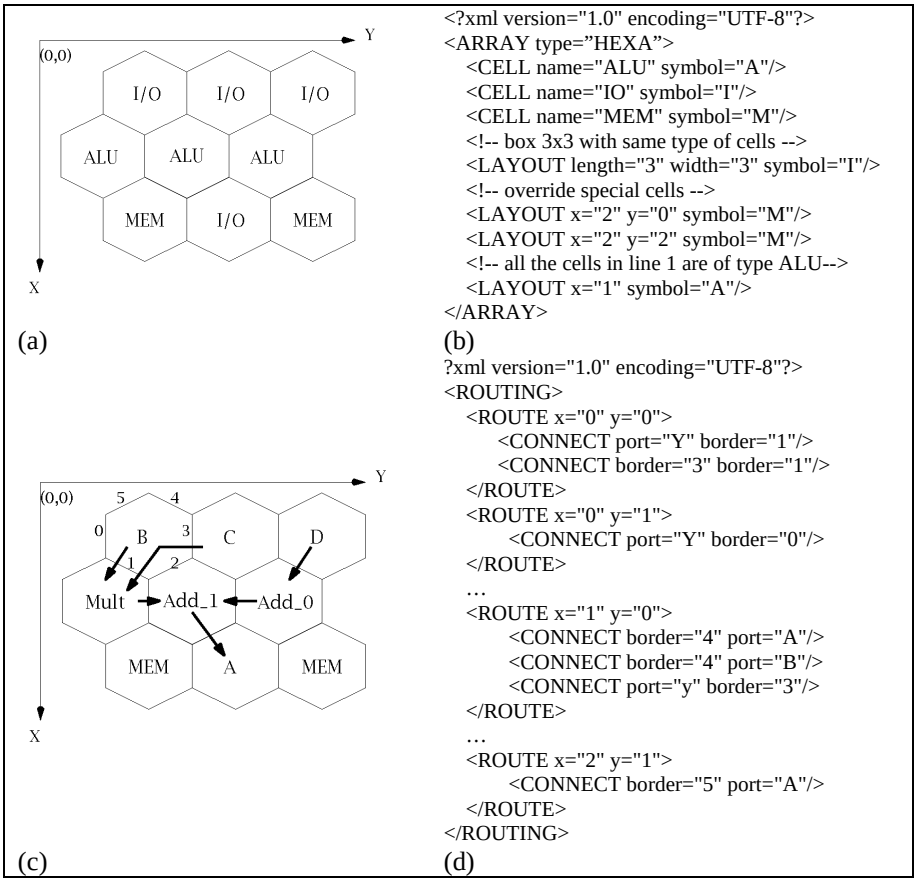


Fig. 5. (a) Arquitectura hexagonal simples; (b) parte do ficheiro XML que especifica a arquitectura; (c) exemplo *dataflow*; (d) descrição XML que especifica o encaminhamento do exemplo *dataflow* na arquitectura hexagonal anterior

Mediante a utilização de transformadores XSL, as especificações descritas anteriormente são transformadas em código Java responsável pela modelação da arquitectura a simular utilizando o HADES. A programação da arquitectura, tendo por base um determinado exemplo e a informação relativa à colocação e encaminhamentos necessários para interligação das unidades funcionais, descritos em XML, é realizada por código Java especifico que é gerado por transformadores XSLT. O código Java gerado tem por base uma biblioteca de classes Java relacionadas com elementos da arquitectura.

3. Arquitecturas Específicas para FPGAs

A compilação de programas em linguagens de software, como por exemplo de Java [19], para arquitecturas computacionais reconfiguráveis é um tópico de investigação extremamente importante. Tal deve-se ao facto de haver indícios claros de que os sistemas embebidos do futuro utilizarão arquitecturas computacionais mais variadas do que os microprocessadores tradicionais baseados no modelo de von-Neumann. Tendo os FPGAs se tornado em verdadeiras soluções plataforma – poderemos ter num único integrado sistemas completos que podem incluir um ou mais microprocessadores e arquitecturas específicas – é provável que o componente principal de sistemas embebidos complexos possa ser baseado em FPGAs. Neste caso, a geração de uma arquitectura específica que esteja adaptada ao problema que pretende resolver é uma etapa importante no desenvolvimento deste tipo de sistemas. Contudo, o projecto de arquitecturas específicas para este tipo de dispositivos é bastante complicado e necessita de conhecimentos profundos e de experiência consolidada no projecto de sistemas digitais para estes dispositivos. Neste momento não é possível a um programador tirar partido do FPGA sem essa experiência. É por estas razões que a compilação de algoritmos descritos em linguagem alto-nível é fundamental.

Um compilador desse tipo gera uma arquitectura específica a partir do algoritmo descrito. Essa arquitectura é depois mapeada nos recursos existentes do FPGA. O mapeamento é normalmente efectuado por ferramentas disponibilizadas pelos fabricantes deste tipo de dispositivos.

A arquitectura gerada pelo compilador é constituída por um *datapath* (unidade de dados) e por uma unidade de controlo (constituída por uma ou mais FSMs). Estas unidades são representadas internamente sob a forma de grafos que mais tarde são passados às ferramentas de mapeamento utilizando descrições aceites por estas. Normalmente são utilizadas linguagens de descrição de hardware (VHDL, por exemplo) das quais são depois obtidos os circuitos finais utilizando ferramentas de síntese. É ainda gerado um grafo de transições de reconfiguração (*rtg*), que tem como função especificar o fluxo de configurações de forma a que a execução das partições temporais que definem o mapeamento do algoritmo possa fornecer a funcionalidade inicial. O comportamento representado por este grafo pode depois ser implementado por um microprocessador ou por uma unidade de controlo de reconfigurações.

A representação destas unidades pode ser feita utilizando XML. A **Fig. 6** ilustra os ficheiros XML gerados pelo compilador (a sombreado) e os transformadores utilizados. A utilização de XML nesta fase traduz-se nas seguintes vantagens:

– O compilador fornece uma representação da arquitectura independente das ferramentas de mapeamento utilizadas;
– A utilização de transformadores XSL permite a obtenção de representações finais em vários formatos adequados para a finalidade que se pretende.

O ponto anterior pode ser importante para a utilização de por exemplo ferramentas de mapeamento que aceitem outras linguagens. Neste caso o utilizador poderá utilizar o seu próprio transformador XSL.

No ambiente utilizado estão disponíveis vários transformadores XSL:

– para gerar representações de visualização das estruturas geradas pelo compilador. É utilizada a linguagem *dot* e a ferramenta Graphviz [20];
– para gerar os modelos VHDL das estruturas que podem ser depois sintetizados pelas ferramentas comerciais, como o ISE da Xilinx, por exemplo;
– para gerar representações das estruturas em Java para serem simuladas pelo simulador HADES [12].

A infra-estrutura de teste assenta em dois componentes principais. O primeiro é o ambiente de simulação e edição de sistemas lógicos desenvolvido em Java denominado de HADES. Este simulador recebe como entrada um ficheiro descrevendo a rede do circuito a simular, circuito este que utiliza como elementos lógicos básicos as unidades funcionais implementadas pelo compilador (multiplicadores, somadores, RAMs, etc.). Estes elementos por sua vez encontram-se numa biblioteca de operadores descritos em Java e prontos a serem utilizados pelo HADES.

Por outro lado, todo o processo de teste é automatizado pela aplicação ANT [21] que interpreta um ficheiro XML com a descrição das tarefas a executar. A primeira etapa consiste na tradução destes ficheiros de entrada para um conjunto de representações intermédias. Isto é conseguido mediante a utilização de transformadores XSL. O ficheiro descritor do caminho de dados é então traduzido para o formato interpretado pelo HADES. A máquina de estados é traduzida para Java e é interpretada pelo simulador como sendo mais uma das unidades funcionais presentes na arquitectura. Por sua vez o RTG é também traduzido para Java e irá controlar o fluxo de simulação de cada partição temporal.

As secções seguintes descrevem as unidades principais geradas pelo compilador e as representações XML.

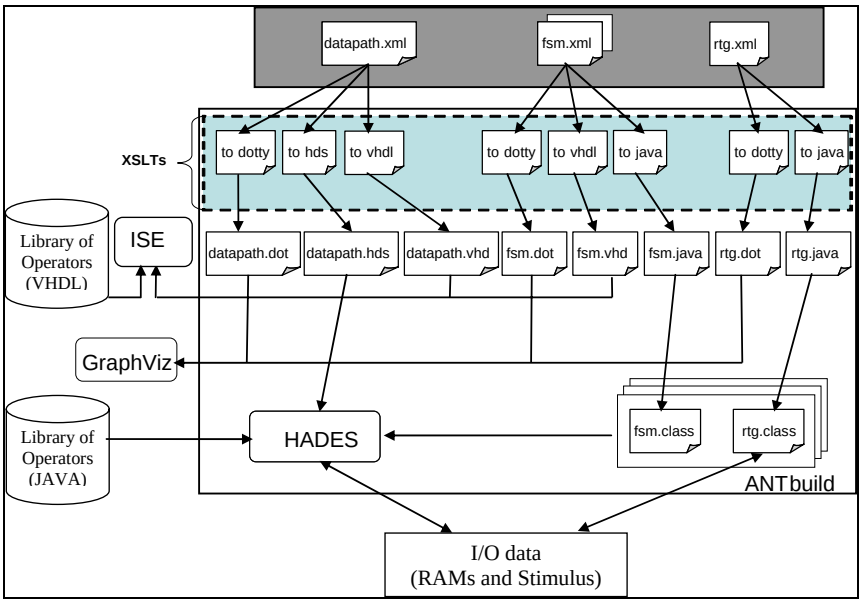


Fig. 6. Utilização da tecnologia XML no *backend* do compilador para arquitecturas específicas

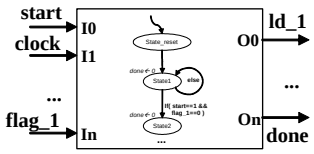
3.1. Especificação da Máquina de Estados

A cada partição temporal corresponde uma unidade de controlo descrita como uma máquina de estados finitos. A **Fig. 7**(a) mostra parte da descrição de uma FSM utilizando o dialecto XML desenvolvido.

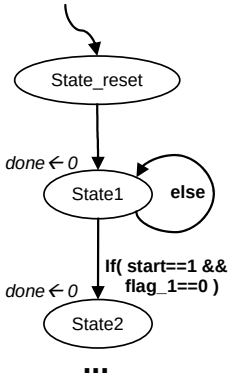
A raiz do dialecto é o elemento FSM. Este é constituido por duas partes distintas definidas por dois elementos filhos, o INTERFACE e o BEHAVIOR, as **Fig. 7**(b) e (c) representam respectivamente esquemas visuais do conteúdo destes elementos no exemplo da **Fig. 7**(a). No INTERFACE são definidos todos os portos de entrada e saída. Isto é feito recorrendo ao elemento PORT, onde são definidos os atributos específicos a cada uma das portas: sentido da ligação (*type*), nome do porto (*name*) e sinal ligado (*signal*). Existe ainda um atributo específico a determinadas portas (*feature*) que indica sinais especiais tais como o sinal de relógio (*clock*) ou o de *reset*.

```
<?xml version="1.0" encoding="UTF-8"?>
<FSM name="fsm0">
  <INTERFACE>
    <PORT type="in" name="I0" signal="start" feature="start"/>
    <PORT type="in" name="I1" signal="clock"
feature="clock"/>
    <PORT type="in" name="In" signal="flag_1"/>
    <PORT type="out" name="O0" signal="ld _1"/>
    …
  </INTERFACE>
  <BEHAVIOR>
    <SET signal="done" value="0"/>
    …
    <STATE name="State_reset" type="reset">
      <NEXT name="State1"/>
    </STATE>
    <STATE name="State1">
      <IF>
        <AND>
          <TEST signal="start" operation="=" value="1"/>
          <TEST signal="flag_1" operation="=" value="0"/>
        </AND>
        <NEXT name="State2"/>
      </IF>
      <ELSE>
        <NEXT name="State1"/>
      </ELSE>
    </STATE>
    …
  </BEHAVIOR>
</FSM>
```

(a)



(b)



(c)

Fig. 7. (a) Descrição parcial de uma FSM. (b) Representação das entradas e saídas da FSM. (c) diagrama de estados da FSM

No elemento BEHAVIOR é descrito o comportamento da máquina. A descrição é iniciada com a declaração de elementos SET de atribuição de valores a sinais de saída. Esta atribuição é válida para todos os estados e no caso do sinal ser definido posteriormente, a última atribuição definida prevalece. Para cada estado é definido um

elemento STATE e o estado inicial é sempre definido por aquele que possuir o atributo *type=”reset”*. A transição de estados é conseguida recorrendo ao elemento NEXT, o qual indica o próximo estado da máquina. O comportamento em cada um dos estados pode ser condicional, neste caso é possível a utilização da estrutura de elementos IF/ELSE em que a condição de controlo é definida pelo elemento TEST. Esta condição pode ser composta utilizando os elementos lógicos AND e OR.

3.2 Especificação do caminho de dados

O caminho de dados de cada partição temporal consiste numa estrutura representando as unidades funcionais que o compõem e as respectivas ligações entre si. Na **Fig. 8**(a) está representado parte do *datapath* correspondente ao cálculo da equação da **Fig. 8**(a), utilizando o dialecto XML desenvolvido. O elemento DESIGN é o elemento raiz do dialecto que é constituído por três tipos de elementos filhos. Um elemento INTERFACE que tal como na descrição da FSM descreve os portos de entrada e saída, um ou mais elementos COMPONENT que representam cada uma das unidades funcionais constituintes e um conjunto de elementos SIGNAL que descrevem as ligações entre os vários componentes.

```
<?xml version="1.0" encoding="UTF-8"?>
<DESIGN name = “design0”>
  <INTERFACE>
    <PORT type="in" name="clock" signal="clock" feature="clock"/>
    <PORT type="in" name="B" signal="b" bitwidth="32"/>
    <PORT type="in" name="C" signal="c” bitwidth="32" />
    <PORT type="in" name="D" signal="d” bitwidth="32" />
    <PORT type="out" name="A" signal="a" />
  </INTERFACE>
  <COMPONENT unit="reg_op" name="r_b" bitwidth="32">
    <PORT type="in" signal="clock" name="clock" feature="clock"/>
  </COMPONENT>
  <COMPONENT unit="add_op" name="a_1" bitwidth="32">
    <PORT name="I0" type="in" />
    <PORT name="I1" type="in" value=”6” />
    <PORT name="O0" type="out" />
  </COMPONENT>
  <COMPONENT unit="add_op" name="a_2" bitwidth="32"/>
  …
  <SIGNAL name="r_d_to_a_1" bitwidth="32">
    <SOURCE name="r_d" port="O0"/>
    <SINK name="a_1" port="I0">
      <PROBE/>
    </SINK>
  </SIGNAL>
  <SIGNAL name="a_1_to_a_2" bitwidth="32">
    <SOURCE name="a_1" port="O0"/>
    <SINK name="a_2" port="I2"/>
    </SIGNAL>
  …
</DESIGN>
```

(a)

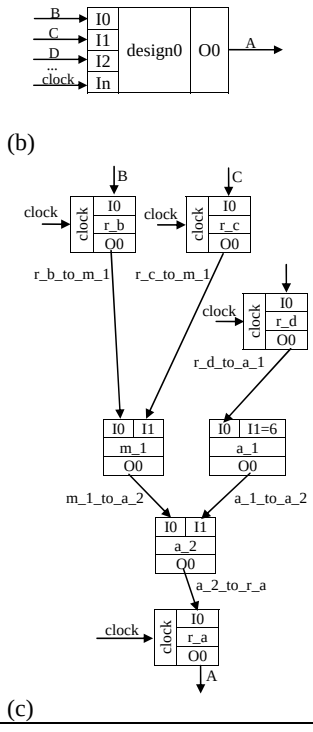


Fig. 8. (a) Descrição parcial do *datapath* correspondente à equação representada na Figura 2(a). (b) Representação do interface de entrada/saída do *datapath*. (c) Diagrama ilustrativo do *datapath*, representando os componentes e sinais de ligação

Cada componente é identificado pela sua unidade funcional (*unit*) e por um nome único (*name*). No caso dos portos de entrada/saída do componente possuírem todos o mesmo tamanho em bits, o elemento *bitwidth* é especificado. Em caso contrário, ou em caso da necessidade de atribuir um valor constante a uma das entradas do componente o elemento PORT terá de ser utilizado para cada um dos seus portos. A atribuição do valor constante ao porto em questão é efectuado utilizando o atributo *value*.

Todos os componentes que necessitem de sinal de relógio terão de ter especificado o porto ao qual o sinal é ligado utilizando o elemento PORT, ver por exemplo o componente de nome *r_b* na **Fig. 8**(a).

As ligações entre portos dos vários componentes são especificadas por elementos SIGNAL. Cada um destes elementos especifica uma ligação, em que é especificado o porto de origem do sinal (SOURCE) e os seus portos de destino (um ou mais elementos SINK). A cada um dos sinais é atribuído o tamanho em bits dos portos de origem e destino.

Para efeitos de teste e despistagem de erros, foi ainda especificado um elemento especial, com a função de indicar quais os sinais que durante a simulação deverão ser escritos para o ficheiro de saída de dados. Este elemento é o PROBE e é especificado como elemento filho do sinal em questão.

4. Conclusões

Este artigo descreve a utilização da tecnologia XML na representação de arquitecturas específicas e de estruturas computacionais. A tecnologia tem permitido os desenvolvimentos incrementais bastante úteis em projectos de investigação, já que novas ideias vão surgindo continuamente e existe a necessidade quase constante de incluir novas propriedades. Embora os transformadores de XSL adicionem atrasos na execução das ferramentas, face por exemplo à geração directa das representações finais, o tempo de execução permanece aceitável.

A linguagem XML tem permitido definir dialectos com semânticas associadas de um modo eficiente e com menores custos de desenvolvimento.

Os transformadores de XSL têm permitido a tradução de uma determinada estrutura ou funcionalidade em representações especificadas na linguagem necessária para a implementação, simulação, visualização, etc.

O trabalho futuro contempla a adição de mais propriedades aos dialectos XML apresentados, de forma a que a exploração de arquitecturas específicas utilizando os trabalhos apresentados neste artigo possa ser ainda mais abrangente e profícua.

Agradecimentos

Os autores agradecem o apoio concedido pela Fundação para a Ciência e Tecnologia (FCT) – programas FEDER e POSI – no âmbito do projecto CHIADO (POSI/CHS/48018/2002).

Referências

1. W3C: Extensible markup language (xml), http://www.w3.org/XML/ (1996-2003).
2. J. E. Coffland, and A. D. Pimentel, "A Software Framework for Efficient System-level Performance Evaluation of Embedded Systems", in *Proc. of the 18th ACM Symposium on Applied Computing (SAC'03)*, Melbourne, Florida, USA, March 2003, pp. 666-671.
3. Eric Keller, Gordon J. Brebner, Philip James-Roxby, "Software Decelerators", in *Proc. of the 13th International Conference on Field Programmable Logic and Applications (FPL'03)*, Lisbon, Portugal, Sept. 2003. Peter Cheung, George Constantinides, José T. Sousa (Editors), LNCS 2778, Springer Verlag, pp. 385-395.
4. Shay P. Seng, Krishna V. Palem, Rodric M. Rabbah, Weng-Fai Wong, Wayne Luk and P.Y.K. Cheung, "PD-XML: Extensible Markup Language For Processor Description," In *Proceedings of the IEEE International Conference on Field-Programmable Technology (ICFPT'02)*, December 2002, pp. 437- 440.
5. W3C: The extensible stylesheet language family (xsl), http://www.w3.org/Style/XSL (1996-2003).
6. João M. P. Cardoso, "CHIADO: compilation of high-level computationally intensive algorithms to dynamically reconfigurable computing systems," in *SPIE Microtechnologies for the New Millennium 2005 Symposium*, Seville, Spain, May 9-11, 2005, SPIE Vol. 5837, pp. 893-901.
7. Rui Rodrigues, and João M. P. Cardoso, "A Test Infrastructure for Compilers Targeting FPGAs," in *International Workshop on Applied Reconfigurable Computing (ARC2005)*, held in conjunction with *IADIS International Conference Applied Computing 2005*, Algarve 22-23, Portugal, IADIS Press, pp. 168-175.
8. Rui Rodrigues, and João M. P. Cardoso, "An Infrastructure to Functionally Test Designs Generated by Compilers Targeting FPGAs," Interactive Presentation at the *Design, Automation and Test in Europe Conference (DATE'05)*, Munich, Germany, March 7-11, 2005, IEEE Computer Society Press, pp. 30-31.
9. Ricardo Ferreira, João M. P. Cardoso, and Horácio C. Neto, "An Environment for Exploring Data-Driven Architectures," in *14th Int'l Conference on Field Programmable Logic and Applications (FPL'04)*, LNCS 3203, Springer-Verlag, 2004, pp. 1022-1026.
10. Ricardo Ferreira, João M. P. Cardoso, Andre Toledo, and Horácio C. Neto, "Data-driven Regular Reconfigurable Arrays: Design Space Exploration and Mapping," in *Embedded Computer Systems: Architectures, Modeling, and Simulation 5th International Workshop*, Timo D. Hämäläinen, Andy D. Pimentel, Jarmo Takala, Stamatis Vassiliadis (Eds.), SAMOS 2005, Samos, Greece, July 18-20, 2005, LNCS 3553 Springer, pp. 41-50.
11. Jürgen Becker, and Reiner Hartenstein, "Configware and morphware going mainstream," in *Journal of Systems Architecture: the EUROMICRO journal*, vol. 49, Issue 4-6, September 2003, pp. 127-142.
12. Francisco Barat, Rudy Lauwereins, and Geert Deconinck, "Reconfigurable Instruction Set Processors from a Hardware/Software Perspective," in *IEEE Transactions on Software Engineering*, vol. 28, no. 9, September 2002, pp. 847-862.
13. R. Hartenstein, "A Decade of Reconfigurable Computing: a Visionary Retrospective," In *Int'l Conference on Design, Automation and Test in Europe (DATE'01)*, Munich, Germany, March 12-15, 2001, pp. 642-649.
14. João M. P. Cardoso, "Data-driven array architectures: a rebirth?," in *SPIE Microtechnologies for the New Millennium 2005 Symposium*, Seville, Spain, May 9-11, 2005, SPIE Vol. 5837, pp. 479-490.
15. A. H. Veen, "Dataflow machine architecture," in *ACM Computing Surveys*, Vol. 18, Issue 4, 1986, pp. 365-396.
16. Norman Hendrich, "HADES: The Hamburg Design System," in *ASA'98, European Academic Software Award/Alt-C Conference*, Oxford, 19-21 Sep. 1998.

17. Norman Hendrich, "A Java-based Framework for Simulation and Teaching," in *3rd European Workshop on Microelectronics Education (EWME'00)*, Aix en Provence, France, 18-19, May 2000, Kluwer Academic Publishers, pp. 285-288.
18. ____, *Hades: Interactive Simulation Framework*, http://tech-www.informatik.uni-hamburg.de/applets/hades/webdemos/index.html
19. João M. P. Cardoso, and Horácio C. Neto, "Compilation for FPGA-Based Reconfigurable Hardware," in *IEEE Design & Test of Computers Magazine*, March/April, 2003, vol. 20, no. 2, pp. 65-75.
20. *Graphviz - open source graph drawing software*: http://www.research.att.com/sw/tools/graphviz/.
21. The apache ant project: http://ant.apache.org/.

Structure Metrics for XML Schema

Joost Visser*

Departamento de Informática
Universidade do Minho
Braga, Portugal
Joost.Visser@di.uminho.pt
http://www.di.uminho.pt/~joost.visser

Abstract. XML schemas are software artifacts claiming an increasingly central role in software construction projects. Schemas are used as interface definitions, data models, protocol specifications, and more. Standards bodies are employing schemas for standards definition and dissemination. Using code generators that accept schemas as input, software components are generated for data interchange and persistence.
With increased reliance on schemas comes the necessity of properly embedding these artifacts in the software engineering process. In particular, schema metrics must be developed to enable quantification of schema size, complexity, quality, and other properties, instrumental to retaining control over the software processes in which they are involved.
In this paper, we propose a suite of metrics for the XML Schema language that measure structural properties. The metrics are mostly adaptations of existing metrics for other software artifacts, such as programs and grammars. Apart from definitions of the metrics, we report on application of these metrics to a series of open source schemas, using our XsdMetz tool. We suggest how the measurement results may be used to assess potential risks in schemas.
Keywords: XML, XSD, Schemas, Software metrics, Document processing, Software quality, Software risk assessment, Dependency graph, Tree impurity, Coupling, Coherence, Component analysis.

1 Introduction

XML Schema was the first separate schema language for XML to achieve Recommendation status by the World Wide Web Consortium (W3C) [8]. It is one of several schema languages proposed to supersede Document Type Definitions (DTDs) for the specification of structure, content, and semantics of XML documents. The main components defined/declared in an XML Schema are *elements*, *attributes*, simple and complex *types*.

XML Schemas are software artifacts that are claiming an increasingly central role in software construction projects. Schemas have come into use as interface

* Supported by the Fundação para a Ciência e a Tecnologia, Portugal, under grant number SFRH/BPD/11609/2002.

definitions, as data models, as protocol specifications, and more. Standards bodies are employing schemas for standards definition and dissemination. Using code generators that accept schemas as input, software components are generated for data interchange and persistence.

With increased reliance on schemas comes the necessity of properly embedding them in the software engineering process. This involves both tool support (e.g. for schema editing, conversion, visualization) and methodology (e.g. schema design patterns and schema style guides), areas in which progress is currently being made at high speed.

An area in which investigation has started only recently is schema *metrics*. In software engineering in general, metrics play a role in monitoring and controlling the software process, or in specifying and improving software quality aspects such as performance or reliability. To wit, the 200-page *Guide to the Software Engineering Body Of Knowledge - SWEBOK* [1] contains about 500 references to the topic of metrics[1]. Over time, an extensive array of software engineering metrics have been defined and applied [4, 3]. Metrics for XML schemas are needed for quantification of schema size, complexity, quality, and other properties, instrumental to control the processes in which they are involved.

The first definition of a suite of metrics for the XML Schema language has been provided by Lämmel *et al.* [6]. Their metrics range from simple counters of various types of schema nodes to more involved metrics such as McCabe, depth, and breadth. Table 1 shows an overview. All these metrics can be seen

Table 1. Overview of XML Schema metrics defined by Lämmel *et al.* [6].

XML-agnostic schema size	
File size	KB or LOC
XSD-agnostic schema size	
Number of all XML nodes	#NODE
Number of all XML nodes for annotation	#ANN
XSD-aware counts	
Number of global, local, or all element declarations	$\#EL_g$,$\#EL_l$,#EL
Number of global, local, or all complex-type definitions	$\#CT_g$,$\#CT_l$,#CT
Number of global, local, or all simple-type definitions	$\#ST_g$,$\#ST_l$,#ST
Number of global, local, or all model-group definitions	$\#MG_g$,$\#MG_l$,#MG
Number of global, local, or all attribute-group definitions	$\#AG_g$,$\#AG_l$,#AG
Number of global, local, or all attribute declarations	$\#AT_g$,$\#AT_l$,#AT
$\#EL_g + \ldots + \#AT_g$	#GLOBAL
McCabe complexity for XSD	
McCabe cyclometric complexity	MCC
Depth and breadth of content models	
Code-oriented and instance-oriented breadth	
Code-oriented and instance-oriented depth	

[1] With search keys like *quantification*, *metric*, *measure*, *statistic*, and their derivations.

as *size* metrics; the property measured by each of them is schema size, albeit measured in many different ways. Even the McCabe metric, originally developed with the intention to measure (program) complexity, is well-known to be strongly and significantly correlated with size, and its use as complexity metric has been criticised [4].

In this paper we propose a number of more advanced schema metrics that may be used to measure other properties than size. The proposed metrics are adaptations of existing metrics for other software artifacts, such as programs and grammars. All metrics are defined over graph representations of schema structure, and can hence be categorized as structure metrics. Apart from definitions of the metrics, we report on application of these metrics to a series of open source schemas, measured using our XsdMetz tool. Also, we discuss how the measurement results may be used to assess potential risks in schemas. Rigorous empirical validation of the metrics as measures for external schema properties falls outside of the scope of this paper.

In Section 2 we define graph representations of schema structure. These graph representations are the basis of the structural metrics which we define in Section 3. In Section 4 we comment on the implementation of the metrics suite in our XsdMetz tool, and we discuss its application to a series of important XML schemas. Section 5 discusses related work, while Section 6 provides concluding remarks and a reflection on future work.

2 Graph representations of schema structure

The various components of an XML schema (such as elements, types, groups, and attributes) can be dependent on each other in the sense that the definition or declaration of one component may mention other components. Whenever such a dependency exists, we say that the mentioned component is an *immediate successor* of the defined/declared component. Based on this immediate successor relation, a graph representation of the schema structure is obtained, where nodes are components, and edges are successor relations. Figure 1 shows a small schema and its associated *successor graph* (SG).

When two nodes in a directed graph can be reached one from the other and *vice versa*, these nodes are called *strongly connected*. In the successor graph of Figure 1, this is the case for the nodes representing the global complex type `EmployeeType` and its local element `Emp`. A set of strongly connected nodes is termed a *strongly connected component*. From any directed graph that potentially contains such cycles, we may derive a directed acyclic graph (DAG) that contains the strongly connected components as nodes, and that contains edges between two components whenever at least one edge exists between a node in one component and a node in the other. For our example, the derived *strongly connected components graph* (CG) is shown in Figure 2. We can regard strongly connected components as a specific notion of *module* for dependency graphs in general, and by extension, for XML schemas in particular.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Emp" type="EmployeeType" />
  <xs:complexType name="EmployeeType">
    <xs:sequence>
      <xs:element name="Emp" type="EmployeeType" />
    </xs:sequence>
    <xs:attribute name="EmployeeID" type="xs:ID" />
    <xs:attribute name="FirstName" type="xs:string"/>
    <xs:attribute name="LastName" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

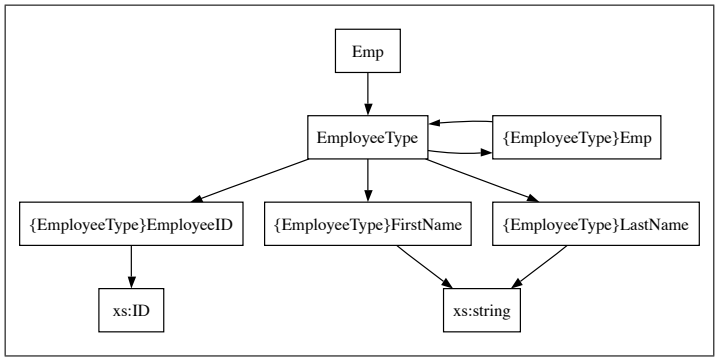


Fig. 1. A simple example schema, adapted from the online *.NET Framework Developer's Guide* (http://msdn.microsoft.com/library/), for representing employee hierarchies. The graph depicts the corresponding successor relation. Local names such as FirstName are qualified with their surrounding global, in this case {EmployeeType}.

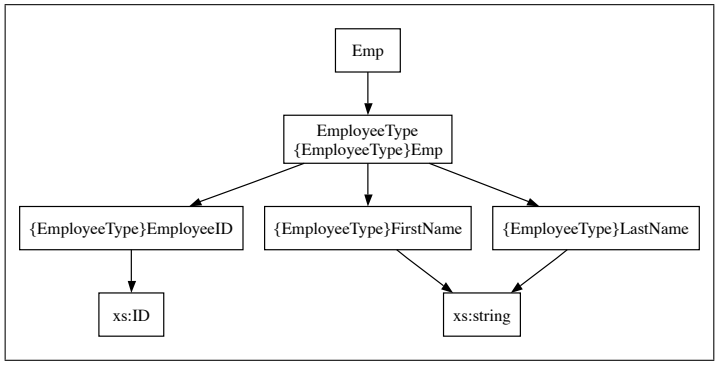


Fig. 2. Graph of strongly connected components derived from the successor graph of Figure 1. The global complex type `EmployeeType` and its local element `Emp`, which are mutually dependent, are wrapped into a single component (module).

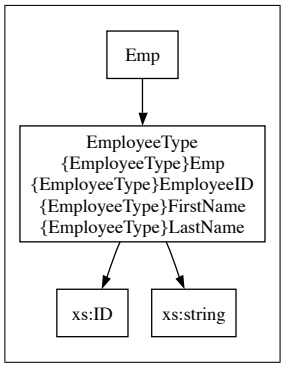


Fig. 3. Graph of global declarations/definitions, derived from the successor graph of Figure 1. The various local components of `EmployeeType` are wrapped into a single component together with that global type itself.

Other, perhaps more intuitive notions of *module* can be employed for XML schemas. For example, namespaces, schema documents (i.e. `xsd` files), or global declarations/definitions can each be employed to group nodes of a successor graph into modules. Similar as in the case of modules as strongly-connected components, each of theses module notions can be used to obtain a derived dependency graph. In this paper, we will take into consideration the dependency graph derived using the notion of modules as global declarations/definitions. An example of such a derived graph of globals is shown in Figure 3.

These three types of dependency graphs are the basis of the structure metrics that we define in the upcoming section.

3 Definition of metrics

We will provide definitions for a range of structure metrics. For each metric, we will show computed values for the example schema of Figure 1.

3.1 Tree impurity

The tree impurity metric indicates to what extent a given graph deviates from a tree structure with the same number of nodes. Fenton *et al.* [4] define tree impurity for connected undirected graphs without self-edges as $\frac{2(e-n+1)}{(n-1)(n-2)} \cdot 100\%$, where n is the number of nodes, and e is the number of edges. A tree impurity of 0% means that the graph is a tree and a tree impurity of 100% means that it is a fully connected graph.

Tree impurity can also be applied to directed graphs that may contain self-edges, simply by disregarding the direction and the self-edges. In this way, tree

impurity is applicable both to successor graphs and to strongly connected components graphs. In particular, for the graphs of Figure 1, 2, and 3, we obtain the values of $TI_{SG} = 4.76\%$, $TI_{CG} = 6.67\%$, and $TI_{GG} = 0\%$, respectively.

3.2 Fan-in, fan-out, and instability

A pair of classic metrics are fan-in and fan-out. The fan-in of a node in a directed graph is the number of its incoming edges. Conversely, the fan-out is the number of outgoing edges of the node. Both metrics are directly applicable to the nodes of each type of dependency graph. For the graphs as a whole, the averages and maximums of these metrics can be relevant. For the graphs of Figure 1 and 2, we have the following values:

SG	avg	max
fan-in	1.125	2
fan-out	1.125	4

CG	avg	max
fan-in	1.0	2
fan-out	1.0	3

GG	avg	max
fan-in	0.75	1
fan-out	0.75	2

The maximum fan-in and fan-out, in particular, can be useful to spot unusual nodes, which subsequently may be inspected to identify the cause of the abnormality, and a possible cause of action to correct the situation.

Based on fan-in and fan-out, a measure called *instability* can be defined as the fan-out fraction of total fan, i.e. as: $\frac{fan\text{-}out}{fan\text{-}in+fan\text{-}out} \cdot 100\%$. For our example graphs, we have the following basic statistics regarding node instability:

SG	avg
instability	45.8

CG	avg
instability	46.4

GG	avg
instability	41.7

The instability metric ranges between 0% (no outgoing edges) and 100% (only outgoing edges). Low instability of a node indicates that it is dependent on few other nodes, while many nodes are dependent on it. Thus, low instability corresponds to a situation where changes to the node will affect relatively many other nodes, and would hence be costly or difficult. In other words, instability may be interpreted as resistance to change.

3.3 Efferent and afferent coupling, and again instability

Coupling is a notion similar to fan, but taking modules into account, where any group of nodes may be viewed as a module. The number of edges from nodes outside the module to nodes inside the module is called *afferent coupling* (Ca). Conversely, the number of edges from nodes inside the module to nodes outside is called *efferent coupling* (Ce). Their sum is simply *coupling*. As in the case of fan-in and fan-out, an instability metric can be defined based on afferent and efferent coupling, as: $\frac{Ce}{Ce+Ca} \cdot 100\%$.

For our example schema, the coupling metrics corresponding to strongly connected components and to global declarations/definitions present the following basic statistics:

CG	avg	max
afferent coupling	1.0	2
efferent coupling	1.0	3
internal edges	0.29	2
instability	46.43	

GG	avg	max
afferent coupling	1.0	2
efferent coupling	1.0	3
internal edges	1.25	5
instability	43.75	

Coupling is often mentioned in one breath with *coherence*, which we discuss next.

3.4 Coherence

Whereas coupling assesses the connections of a module with nodes external to it, the *coherence* of a module concerns the degree to which its internal nodes are connected with each other. As a general coherence metric we propose the ratio of internal edges of a module versus all edges that start and/or end in a node inside the module. Thus, $Ch = \frac{Ci}{Ci+Ca+Ce} \cdot 100\%$, where Ci is the number of edges between nodes inside the module, i.e. the internal edge count. In the limit case of singleton modules, we set $Ch = 100\%$, rather than 0%, expressing that singletons are fully coherent.

Note that this measure of coherence takes external edges into account, not only internal edges. Assuming a stable node count for a module, its coherence increases both with the addition of internal edges and with the removal of external edges. In other words, coherence is compromised both by lack of connections between internal nodes, and by too many connections to the outside, i.e. by breaches of encapsulation.

As an alternative measure of coherence, one may use tree impurity, as defined before, on the level of modules. For our example graphs, we can derive the following basic statistics:

CG	min	avg	max
internal edges	0	0.29	2
coherence	33.3	90.5	100
tree impurity	0	85.7	100

GG	min	avg	max
internal edges	0	1.25	5
coherence	55.6	88.9	100
tree impurity	0	75.0	100

Both Ch and TI range between 0% and 100%, and both increase when internal edges are added. But, whereas Ch also depends on the number of connections to external nodes, TI of a module is independent from the external edge count.

3.5 Normalized count of modules

For each notion of module, we can define a normalized count of modules by expressing the module count as a ratio of *potential* module count, which is the number of nodes in the underlying dependency graph. Thus, $NCM = \frac{\#M}{\#N} \cdot 100\%$. Thus, the more nodes get grouped into modules, the lower the normalized count of modules. A value of 100% indicates that no grouping has occurred, i.e. each node sits in a separate module (full fragmentation). A value approaching 0% indicates that all nodes are grouped together in a very small number of modules (monoliths).

Table 2. Quick reference of structure metrics.

Metric			Measured per ...
TI	tree impurity	%	... graph (SG, CG, or GG), or module (subgraph) of SG.
Fi	fan-in	$\mathbb{N}$	... node (SG), or module (node in CG or GG).
Fo	fan-out	$\mathbb{N}$	... *idem.*
If	instability (based on fan)	%	... *idem.*
Ca	afferent coupling	$\mathbb{N}$	... module (node in CG or GG), by counting edges in SG.
Ce	efferent coupling	$\mathbb{N}$	... *idem.*
Ic	instability (based on coupling)	%	... *idem.*
Ci	internal edges	$\mathbb{N}$	... *idem.*
Ch	coherence	%	... *idem.*
NCM	normalized count of modules	%	... derived graph (CG or GG), in relation to SG.
NM	count of nodes per module	$\mathbb{N}$	... derived graph (CG or GG).

The interpretation of the normalized count of modules depends on the underlying notion of module. In case of the notion of modules as strongly connected components, (mutual) recursion gives rise to non-singleton modules. Correspondingly, *NCM* is a measure of *recursiveness*, where low *NCM* indicates a high degree of mutual recursiveness. In the case of the notion of modules as global definitions/declarations, local elements give rise to non-singleton modules. Correspondingly, *NCM* is a measure of *encapsulation*, where low *NCM* indicates a high degree of encapsulation.

For the running example, we have the following basic statistics: $NCM_{CG} = 87.5\%$ and $NCM_{GG} = 50.0\%$. Note that in the case of XML Schema, mutual recursiveness must always involve global components. Thus, increased mutual dependencies reduce the opportunity for encapsulation, and *vice versa*.

Apart from comparing the total number of modules with the total number of nodes, it may be interesting to count nodes per module (*NM*). For our running example, we have the following extremes: $NM^{max}_{CG} = 2$ and $NM^{max}_{GG} = 5$.

For quick reference, Table 2 lists all metrics defined in this section together with an indication of their scale and of the entities on which they are measured.

4 Data collection

We have implemented support for the XML Schema metrics defined in the preceding section in a prototype tool, called XsdMetz. The tool was implemented in the functional programming language Haskell, using purely functional graph representations and algorithms. The tool shares most of its code with other metrics extraction tools, in particular with the SdfMetz tool which calculates metrics

Table 3. Schemas that were sampled with the XsdMetz tool.

File name	File size	Provider	Purpose
ABCD_2.06	141K	TDWG, CODATA	Access to Biological Collection Data
AWSECommerceService	91K	Amazon	Web service
MARC	6K	Library of Congress	Bibliographic data
PressureVessel	151K	Codeware Inc.	Data on pressure vessels and heat exchangers
WS-ServiceGroup	4K	IBM and others	Web services
XMLSchema	84K	W3C	Schema for XML Schema
diva	86K	Uppsala University	Digital Scientific Archive
ebaySvc	1.9M	eBay	Web services
uddi_v3	39K	OASIS Consortium	Universal Description, Discovery and Integration

from SDF grammar representations [2]. Apart from generating metric reports in several format, the XsdMetz tool exports successor graphs, strongly connected component graphs, and global definition/declaration graphs (see Section 2) in the *dot* format of GraphViz [5], which can be used to render the graphs.

We have applied XsdMetz to a series of freely available XML Schema specifications, listed in Table 3. The scope and character of this paper do not allow the measurement data to be presented here in full. Also, these measurements constitute merely a first step towards empirical validation of the metrics. Still, we will briefly discuss some observations to illustrate potential use of the metrics.

Table 4 shows the values we measured for the normalized count of modules. For almost all schemas, the normalized count of modules for the strong components graph is close to 100%, indicating very low degrees of *recursiveness*. Only the schema for XML Schema itself appears to contain more recursion, with a significantly lower value of 82.2%. This conclusion about recursiveness is cor-

Table 4. Measurement values for normalized counts of modules and number of non-singletong modules for entire graphs, and maximum number of nodes per module.

	NCM_{CG}	NCM_{GG}	NM_{CG}^{max}	NM_{GG}^{max}
ABCD	100	14.6	1	111
AWSE	99.6	14.0	3	236
MARC	100	55.6	1	6
Pres	99.7	9.67	5	132
WSSG	100	62.5	1	4
XMLS	82.2	52.4	38	12
diva	95.0	12.0	8	118
eBay	99.8	31.5	6	89
uddi	100	77.6	1	5

	KB	TI-SG	TI-CG	TI-GG
ABCD	141	0.19	0.19	3.9
AWSE	91	0.18	0.18	2.5
MARC	6	2.0	2.0	6.4
Pres	151	0.13	0.13	4.3
WSSG	4	1.2	1.2	1.1
XMLS	84	0.56	0.58	2.0
diva	86	0.41	0.45	14
eBay	1870	4.3e-2	4.3e-2	0.28
uddi	39	0.38	0.38	0.60

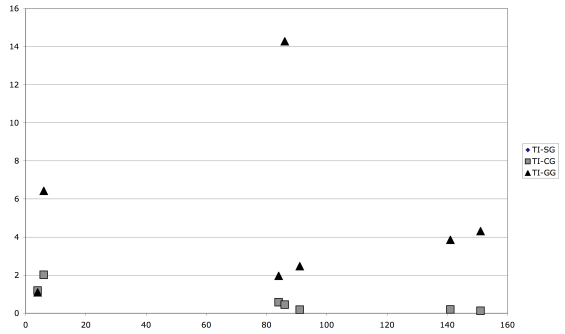


Fig. 4. Measurement values for tree impurity of entire graphs. The chart omits the values for the eBay schema, because its relatively large file size would dwarf the others. The TI-SG series is actually hidden by the TI-CG series, which presents extremely similar values.

roborated by the maximum numbers of nodes per module for strong component graphs (NM_{CG}^{max}), also shown in Table 4. None of the schemas have more than 3 groups of mutually dependent nodes, and the schema for XML Schema has an exceptionally large strong component, containing 38 nodes. The normalized count of modules for the global declarations/definitions graph presents much lower percentages, indicating a fair degree of *encapsulation*. The maximum number of nodes in such declarations/definitions (NM_{GG}^{max}) indicates that the largest module can be found in the Amazon schema, which encapsulates 236 nodes in a single module.

Figure 4 shows the measurement values for tree impurity for each of the dependency graphs. For the successor graph and the strongly-connected components graph, the numbers are very similar, and very low for all sampled schemas. The strong similarity between the tree impurity of SG and CG suggests that the shapes of these two graphs are almost identical, which implies that recursion is uncommon. This corroborates our observations regarding the NCM_{CG} metric. For all schemas, the tree impurity of the global declarations/definitions graph is much higher, with the diva schema as uncontested outlier ($TI_{GG} = 14$). The numbers for SG suggest that schemas in general present a strongly tree-shaped structure, with very little internal reuse. The higher values for GG with respect to SG indicate that most schemas perform extensive encapsulations (using local declarations/definitions), but that dependence on the same nodes is not the criterion for grouping nodes together (otherwise, TI would go down). The degree of encapsulation in the diva schema is exceptional, indicating an uncommon specification style.

The remaining metrics (fan, coupling, coherence, instability), are not applied on complete graphs, but per node or per module only. They are useful for categorizing modules and nodes, and for finding outliers. For instance, in the Amazon schema, values well over 200 are reached for fan-out and for efferent coupling for

both CG and GG. These extremes are traceable to a single element declaration `ItemAttributes` that contains a long list of local elements, ranging from `Actor` to `WirelessMicrophoneFrequency`.

5 Related work

5.1 XML Schema metrics

The first definition of a suite of metrics for the XML Schema language has been provided by Lämmel *et al.* [6]. Their metrics range from simple counters of various types of schema particles to basic complexity metrics such as McCabe, depth, and breadth. The structure metrics proposed by us are not intended to compete with, but rather to complement this suite of metrics. Since structure belongs to the essence of XML schemas, structure metrics are likely to be valueable.

5.2 Grammar metrics

Schema metrics are similar to grammar metrics, in the sense that both schemas and grammars can be regarded as specifying first-order algebraic datatypes, or term algebras. The first definition of a suite of grammar metrics was provided by Power *et al.* [7]. This work was adopted and extended by Alves *et al.* [2] to the feature-rich grammar notation SDF. Our suite of XML Schema metrics has partly evolved out of those SDF grammar metrics. In particular, the tree impurity and normalized count of modules pre-existed this paper as grammar metrics, while the fan, coupling, instability, and coherence metrics were introduced here. As mentioned, the XsdMetz and SdfMetz tools share code libraries for graph representation, transformation, and metrication, and future releases of SdfMetz are likely to include grammar variants of most schema metrics.

5.3 Software metrics

Extensive literature exists on the subject of software metrics (for overviews, see e.g. [4, 3]). The fan and coupling metrics are well-known in their specific application to program modules. We gave a generic graph-based formulation using an abstract notion of modules as groups of nodes. No standard measures of cohesion exist, in spite of its intuitive appeal. We gave two alternative definitions of cohesion of a module: the ratio of internal dependencies versus all dependencies in which the module is involved (Ch), and the tree-impurity of the internal dependency graph. A general definition of tree impurity existed before [4]. We adapted it to directed graphs and applied it on a per-module basis.

6 Concluding remarks

6.1 Contributions

We have defined a suite of eleven structure metrics for the XML Schema language, based on three different graph representations of schema structure: the

successor graph, and the strongly connected component graph and global declaration/definition graph that can be derived from it. We have implemented the metrics suite in a prototype tool, and applied it to freely available schemas ranging up to nearly two megabyte in ascii file size. We have cursorily described the intuitions behind the metrics and their potential use in schema assessment.

6.2 Future work

The suite of metrics presented here is not claimed to be complete in any sense. Many more general software metrics may be investigated to be applicable to XML Schemas.

Before the suite of metrics proposed in this paper can reliably be applied for schema assessment, the proposed metrics must be *validated*. Such validation has been explicitly kept out of the scope of the present paper. In particular, predictive value of the metrics for *external* schema properties must be established. Also, the correlations between the different metrics must be charted out.

Though we have specifically targeted XML Schemas to be measured with the proposed structure metrics, the formulation of the metrics is sufficiently general to attempt their application to other software artifacts. This can be other XML schema notations, grammars, protocols, models, data type definitions, APIs, or any artifact that allows dependency graphs to be extracted and for which appropriate modularization notions exist. In particular, we intend to direct our attention to formats used intermixed with XML Schema, such as SOAP, WSDL, XSL, and further XML formats for specifying application-specific information.

References

1. A. Abran, J.W. Moore, P. Bourque, and R. Dupuis (*eds.*). Guide to the Software Engineering Body of Knowledge - SWEBOK, Version 2004. IEEE Computer Society, www.swebok.org.
2. T. Alves and J. Visser. Metrication of SDF grammars. Technical Report DI-PURe-05.05.01, Universidade do Minho, May 2005.
3. R. Dumke. Software metrics - a subdivided bibliography. http://irb.cs.uni-magdeburg.de/sw-eng/us/bibliography/bib_main.shtml.
4. N. Fenton and S.L. Pfleeger. *Software metrics: a rigorous and practical approach*. PWS Publishing Co., Boston, MA, USA, 1997. 2nd edition, revised printing.
5. E. Koutsofios. Drawing graphs with *dot*. Technical report, AT&T Bell Laboratories, Murray Hill, NJ, USA, November 1996.
6. R. Lämmel, S. Kitsis, and D. Remy. Analysis of XML schema usage. In *Conference Proceedings XML 2005*, November 2005.
7. J.F. Power and B.A. Malloy. A metrics suite for grammar-based software. In *Journal of Software Maintenance and Evolution*, volume 16, pages 405–426. Wiley, November 2004.
8. H.S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema part 1: Structures. W3C Recommendation, May 2001.

Building reusable XML pipelines with APP

Rui Lopes and Luís Carriço

LaSIGE and Department of Informatics
University of Lisbon
{rlopes,lmc}@di.fc.ul.pt

Abstract. XML processing models and respective languages do not reflect the separation of concerns needed in complex XML applications maintenance. As content complexity grows, there is a need for higher abstraction levels on XML processing composition and modularization. This article presents APP, Architecture for Pipelined Processing, focusing on reusable XML pipelines as the way to achieve separation of concerns in complex XML applications. With APP, execution, configuration and development are fully-separated concerns, leveraging component and specifications reuse.

1 Introduction

As XML [1] languages and related technologies mature, processing XML becomes a common task, usually performed with ad-hoc solutions (such as *makefiles*). However, with the introduction of complex document formats, XML processing will be harder within a single processing step, and even, at some cases, impossible to be done.

Consequently, there is a need for higher levels of abstraction regarding XML document processing. XML processing models and languages solve the issue by composing several processing tasks into processing pipelines. This is done by specifying which content is to be processed and what result is expected, forming full-fledged applications, where documents are generated, merged, transformed, augmented and/or serialized.

However, the processing languages that reflect current XML processing models do not fully separate the different concerns in the definition of XML applications. There is a high-coupling between processing tasks and their respective sources (e.g., by hardcoding filenames), as well as a mix of different processing tiers across the processing specification. Therefore, with the introduction of more complex document formats, it becomes harder to develop, configure and maintain complex applications.

As a result, APP [2] emerges as both processing model and language, reflecting the separation of concerns needed for the development of complex XML applications, without sacrificing configuration and management tasks.

This article is decomposed in the following way: on Section 2, we present the requirements on XML processing. Next, on Section 3, related work is discussed. On Section 4, APP's processing model is described. Section 5 presents APP's

processing language. Finally, on Section 6, conclusions are made and future work is delineated.

2 Requirements on XML Processing

2.1 Separation of Concerns

Separation of concerns is a concept introduced in [3], where is it stated as "focussing one's attention upon some aspect". It is widely applied across different software engineering domains, as the way to split a program into distinct features, minimizing the overlapping between them, as described in [4]. This technique has been applied to some XML document formats, such as XHTML [5] coupled with CSS [6] (where the first relates to document structure, and the latter relates to its presentation).

Regarding the production of complex document formats, separation of concerns should be viewed as a way to address different user tasks. Different user profiles can be distinguished, allowing them to perform different tasks on XML processing applications without overlapping their concerns, as follows:

- *Top level users:* little to no knowledge on technologies is required by top level users; given a processing specification, they must be able to apply it to different content sets and perform some minimal configuration (preferably through a graphical user interface);
- *Configuration managers:* some technology expertise should be required by configuration managers; their task relates to the definition of processing specifications, through the composition of available processing tiers and/or XML processing tasks;
- *Developers:* advanced technical skills on XML processing are required by developers, as well as deep understanding of XML processing models, as processing tasks should be componentized and highly reusable.

As separation of concerns must be reflected later into XML processing models and languages, the following requirement set must be fulfilled:

- Processing tasks usage should be decoupled from their specification;
- Different processing tiers on should relate to different concerns;
- Any changes whithin the concerns of a specific user profile should not break the concerns of the other profiles;
- Developers should be able to work in different processing sets, without overlapping.

2.2 Model

XML processing models have been characterized previously in [7], defining how processing tasks interact with given input sets, as well as with each other. The most relevant requirements on XML processing models are:

- The model should be extensible enough so that applications can define new processes and make them a component in a pipeline;
- The model should allow multiple inputs and multiple outputs for a component (as compound documents are starting to appear [8–10]);
- Information should be passed between components in a standard way, for example, as one of the data sets conforming to an industry standard;
- The model should be neutral with respect to implementation language.

Being a critical issue in complex application development, separation of concerns must be reflected into processing models. So, every requirement gathered regarding separation of concerns must be taken into account when specifying a processing model for complex XML applications.

2.3 Language

An XML processing model should have a compliant processing definition language, specifying how to create XML applications compatible with the processing model. Therefore, [7] defines the following requirements towards the definition of an XML processing language:

- The language should be expressed in XML;
- The language should be as small and simple as possible;
- The language must allow the inputs, outputs, and other parameters of a component to be specified;
- Given a set of components and a set of documents, the language must allow the order of processing to be specified;
- The language must be rich enough to address practical interoperability concerns;
- It should be relatively easy to implement a conformant implementation of the language, but it should also be possible to build a sophisticated implementation that can perform parallel operations, lazy or greedy processing, and other optimizations;
- The processing language should be declarative, not based on APIs.

Also, both separation of concerns and processing models requirements, must be taken into account in the definition of an XML processing language.

3 Related Work

Ad-hoc solutions have been the traditional way XML processing has been defined, either through *makefiles*, Ant scripts [11], or even through hard-coded instructions in some programming language. As these methods require a big effort on specification, flexible configurability and maintenance, other solutions were proposed. Several proposals for XML processing have been made, as a way to specify XML applications in a declarative way.

XPL [12] was created to fulfill all requirements of XML processing models defined in [7]. This language defines an XML vocabulary describing a processing model for XML components, specially focused on XML infosets [13]. The operations are composed in a pipeline, where infosets are created, processed and serialized. Each pipeline component describes which inputs and outputs it will be connected to, allowing the direct manipulation of several inputs and outputs infosets. Coupled with operations, some business logic can be applied (iterations and choices), to further enable flexibility of pipeline composition. Specifying parameters to each component is made by adding an input infoset with its description. XPL is a very powerful specification, geared towards both web application architectures, as well as complex offline XML processing. However, as composition of pipelines is not introduced in XPL, it becomes impossible to modularize multiple pipelines with separation of concerns without using external composition tools.

Starting as an XML publishing framework, Cocoon [14] soon evolved to a full-featured XML web development framework. Its concepts focus around component-based web development. It is a flexible framework, being able to work with multiple kinds of data sources (e.g., XML files, relational databases, and others), delivering content in multiple formats (e.g., XML, HTML, PDF). An application created with Cocoon is specified in a sitemap, a group of processing pipelines. Cocoon uses the notion of pipelines as a way to compose web applications without programming. A pipeline is defined as a group of matchers, where each matcher is responsible for the delivery of a single unit of content. A matcher performs three consecutive tasks on a content unit: generate, transform and serialize. Each matcher can depend on other matchers from the same pipeline, thus creating a dependency-based chain of matchers, inside a single pipeline. This dependency-based chaining approach suits well to Cocoon's purposes (i.e., web development tasks). However, this web orientation is single document based, as web users only see one document at a time. This characteristic can be seen as a limitation, when developing complex digital publishing applications, as these tend to lean towards offline content processing and generation, where multiple content generation and processing is a common practice. Cocoon's dependency-based approach is not able to handle secondary outputs, as each matcher only specifies one output. Secondary outputs generated in a matcher are "left in the wild", being unable to reach them from other matchers (thus breaking the dependency-based approach). Also, the separation of concerns achieved in Cocoon relates to each pipeline matcher, not for content processing steps (as these are blackboxed by matchers).

SXPipe [15] is a language for describing simple XML pipelines, towards lightweight processing of XML information sets. It was created as a substitute for general-purposed build tools (such as *make* or Ant), towards simple XML transformations. The pipelines are defined by simple components which perform actions over a given input (document inclusion, validation, transformation and serialization). Implementations of SXPipe are given an input document, process it with the pipeline specification, resulting in an output document. As sim-

plicity is the centre of SXPipe, several issues are left behind, regarding XML processing architectures requirements. When multiple documents are needed as the source of a pipeline, a pre-processing step of inclusion is performed (e.g., XInclude [16]). This feature is acceptable in SXPipe, but not for complex XML processing models. A better clarification of input sources is needed. Another issues relates to multiple output documents. As it is left to implementations how to handle pipeline outputs, it is implicit that each component hides its multiple output serialization issues from SXPipe.

4 APP Processing Model

APP defines its processing model as an application specification applied to a set of inputs, delivering a set of outputs (as seen on figure 1). An application specification can be decomposed successively into different constructs: project, stage, pipeline, and component.

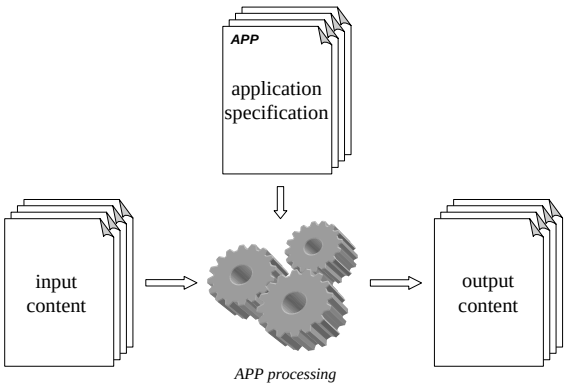


Fig. 1. APP processing model overview

4.1 Project

The processing model of APP defines a project as a sequence of conceptual tiers, named *stages* (see figure 2). Each stage has a well known set of inputs and delivers also a well known set of outputs. The first stage's input is the application input, whereas the application output corresponds to the last stage's output. All other stages use their predecessor's set of outputs as the input for processing. This division in the model is conformant to the separation of concerns requirements.

4.2 Stage

APP defines a stage as a conceptual tier of an XML processing application. Each stage's concern does not overlap any other stage concerns, leveraging its

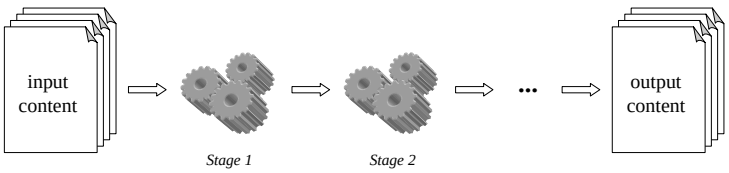


Fig. 2. APP project

reuse in different application specifications. Consequently, a stage used in a given application can be swapped by another stage, as long as they process the same set of inputs and deliver the same set of outputs.

Each stage is decomposed into a set of *processing pipelines*, as seen on figure 3. The stage is responsible on feeding a different subset of its input to each pipeline, and executes their specification. The resulting outputs from each pipeline are aggregated, thus creating the stage's output. Each pipeline is independent from all other pipelines, so that every pipeline output does not collide with each other, within a stage. As long as this rule is not broken, there is no restriction on the number of pipelines a stage can manage.

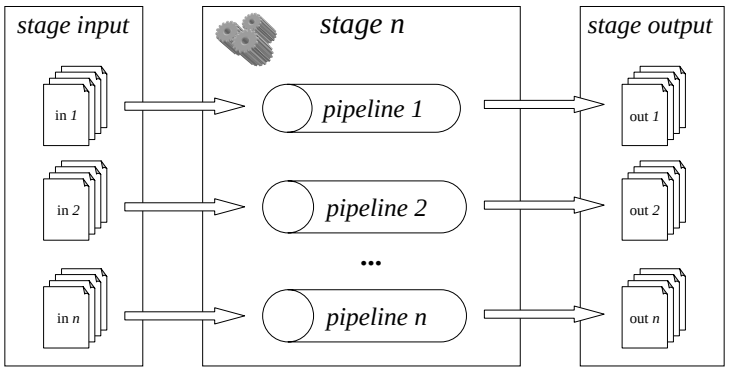


Fig. 3. APP stage

4.3 Pipeline

A pipeline is defined as an acyclic digraph of processing tasks (named *processing components*) applied to a set of inputs, resulting on a set of outputs (see figure 4). Inside a pipeline, each component can process a subset of the pipeline input, as well as generate new contents. Any output from a component can be processed by other components inside the same pipeline. Lastly, all non-processed outputs from each component is aggregated, creating the pipeline's output.

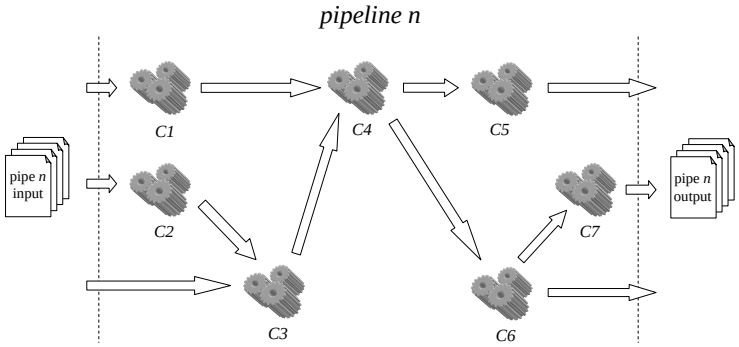


Fig. 4. APP pipeline

A pipeline graph configuration enables processing of multiple inputs and delivering multiple outputs by a single processing component, as stated on XML processing models requirements.

4.4 Component

A processing component is defined by APP processing model as a single processing task applied over a set of inputs, resulting on a set of outputs after its execution. To increase the configuration possibilities on each component, two types of descriptors must be defined: *links* and *parameters*. Links define which set of inputs are fed to the component, as well what outputs the component will deliver. Parameters must be supported as a way to allow higher configurability of the component. Metadata can be also associated to the component and is encouraged when the component count starts to grow, though it is not required.

All this information must be decoupled from the component implementation, as a way to achieve high reusability (see figure 5). This can be done by using unique identifiers (i.e., an URI [17]) over each component interface instead of its implementation. This way, a single implementation could be used within several components (by presenting different configuration levels, for instance).

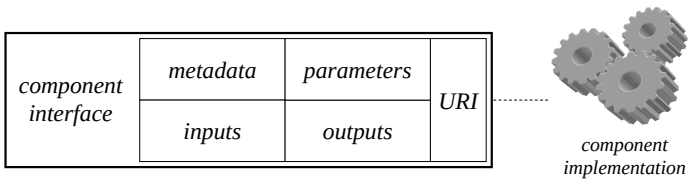


Fig. 5. APP component

As each component description defines which inputs are needed and which outputs are produced, a pipeline can validate if no broken links are found between its processing components, similarly to *pre* and *post* conditions commonly found in "design by contract" software methodologies [18].

5 APP Processing Language

Based on APP's processing model and the requirements of XML processing languages, APP defines its own processing language. Processing applications over APP should be defined in this language, independently of conforming implementations. An application is defined in a project, stating which stages are going to be executed. Each stage defines its pipelines and components separately from each other. Component interfaces are also described separately in a central registry, simplifying the pipeline specification language. With the registry, separation of concerns is achieved regarding the different user profiles (namely configuration managers and developers).

5.1 Project

An APP project is specified in an XML document (as seen on listing 1.1) based on RDF syntax [19], describing the metadata associated with the application, in Dublin Core format [20, 21], as well as the sequence of processing stages. These two descriptions are identified within the document with well-known URIs in *rdf:Description* blocks: *urn:app:project#metadata* for application metadata-related information, whereas *urn:app:project#stages* identifies the stage sequence definition. By splitting stages in different files, separation of concerns is further reached because each developer can centre his/her work independently on each stage, thus stage modularization becomes an easy task.

```
<rdf:RDF xmlns:rdf="..." xmlns:dc="...">
  <rdf:Description rdf:about="urn:app:project#metadata">
    ...
  </rdf:Description>

  <rdf:Description rdf:about="urn:app:project#stages">
    <rdf:Seq>
      <rdf:li rdf:resource="..." />
      ...
    </rdf:Seq>
  </rdf:Description>
</rdf:RDF>
```

Listing 1.1. APP project definition XML

RDF has been used as the standard way to defined metadata (side-by-side with *dc* elements), enabling metadata search by top level users, whithin a GUI tool. This way, a repository of APP applications can be browsed by top level users, easing the selection of appropriate APP applications.

5.2 Stage

Each stage specification is defined in an XML document, with a special purpose language, under the XML namespace [22] *urn:app:stage*. The document root tag *stage* encapsulates all pipeline definitions for the stage (each one defined with *pipeline* tags, enclosing all processing tasks to be executed on that pipeline).

```
<stage xmlns="urn:app:stage" xmlns:reg="urn:app:component:registry">
  <pipeline>
    ...
  </pipeline>
  ...
</stage>
```

Listing 1.2. APP stage definition XML

5.3 Pipeline

As described in APP processing model, a pipeline is an acyclic digraph of processing components. To simplify the speicification of this graph, each component interface description is delegated to the *component registry*, whereas the pipeline description is defined just by a processing component sequence (see listing 1.3). Each component is referenced in the pipeline by its registry identifier (*reg:idref*).

The linearity of a pipeline definition eases the management of complex pipelines, and, consequently, the management of complex XML applications by configuration managers, satisfying separation of concerns requirements. This linearity defines the pipeline's order of processing.

```
<pipeline>
  <component reg:idref="..." />
  <component reg:idref="..." />
  ...
</pipeline>
```

Listing 1.3. APP pipeline definition XML

5.4 Component

A processing component usage inside the definition of a pipeline is straightforward. Each component is defined by the *component* element, referencing its registry identifier (*reg:idref*). Optionally, parameters can be passed to a component, through the *param* element and its *name/value* attributes.

```
<component reg:idref="...">
  <param name="..." value="..." />
  ...
</component>
```

Listing 1.4. APP component definition XML

5.5 Registry

To fully separate processing component interfaces, implementations, and their usage inside a pipeline, a component registry has been defined (see listing 1.5). This registry is RDF-based, where components are registred through *rdf:li* elements, under the *urn:app:registry* URI.

Each component entry in the registry is defined by an identifier (*reg:id* attribute) that will be used in the pipelines, a processing type (*reg:type* attribute) identifying the component type of processing (e.g., an XSLT processor [23], XInclude [16], etc.), and a resource pointer (*reg:resource*). Coupled with this information, a component is defined also by its metadata (under the URI *urn:app:component#metadata*), its input and output sources (URI *urn:app:component#plugs*), and parameters (URI *urn:app:component#params*). Each parameter usage must be identified as *required* or *optional*, through its *use* attribute, having the same semantics of the *use* attribute defined in XML Schema [24].

```
<rdf:RDF xmlns:rdf="..." xmlns:reg="..." xmlns:plug="..." xmlns:dc="...">
  <rdf:Description rdf:about="urn:app:registry">
    <rdf:Bag>
      <rdf:li reg:id="..." reg:type="..." rdf:resource="...">
        <rdf:Description rdf:about="urn:app:component#metadata">
          ...
        </rdf:Description>

        <rdf:Description rdf:about="urn:app:component#plugs">
          <plug:in>
            <rdf:Bag>
              <rdf:li rdf:resource="..." />
              ...
          </plug:in>
          <plug:out>
            <rdf:Bag>
              <rdf:li rdf:resource="..." />
              ...
            </rdf:Bag>
          </plug:out>
        </rdf:Description>

        <rdf:Description rdf:about="urn:app:component#params">
          <rdf:Bag>
            <plug:param name="..." use="..." />
            ...
          </rdf:Bag>
        </rdf:Description>
      </rdf:li>
      ...
    </rdf:Bag>
  </rdf:Description>
</rdf:RDF>
```

Listing 1.5. APP componet registry XML

Regarding the processing type of each component, there should be a standard way to specify it (e.g., XSLT processing could be defined with the type *app:xslt*). However, this specification is out of the scope of this paper.

Having the registry as the way component interfaces are specified leverages separation of concerns between the different users, specially between configu-

ration managers and developers. The registry can act as a catalog for several components, browsed by configuration managers, yet maintained and extended by developers. Also, this separation keeps the graph nature of a pipeline away from the configuration managers, so they can centre their skills just on *what* to process, instead of *how*. The pipeline linearity created by configuration managers can be validated through cycle detection in a pipeline graph composition.

6 Conclusions and Future Work

This paper presented APP (Architecture for Pipelined Processing), a novel approach on complex XML processing, centred on separation of concerns at different levels. As the complexity of XML applications grows, new problems arise in the creation, configuration and maintenance dimensions, as different levels of users can contribute to the definition of an XML application, complex documents must be processed through complex processing tasks, etc.

Consequently, APP defined a new processing model for complex XML applications, based on specific requirements: separation of concerns (e.g., modularization, non-overlapping of user tasks), multiple sources and results on processing tasks, etc. Based on APP processing model, an XML processing language has been defined, reflecting a modularized XML processing approach.

Future directions of APP are being delineated. Porting APP concepts to graphical user interfaces will boost the acceptance of APP, on the different user levels: top level users will not need to manipulate XML files, leveraging the technical skills required; configuration managers can manipulate different configurations easier (e.g., by using drag-and-drop features to create pipelines, easy browsing and searching of processing components that fit their needs); developers will benefit also by having less work on registring a component interface (as its specification may be verbose).

On the APP model and language levels, supporting *iteration* and *choice* constructs will give more flexibility to XML applications by open the way for automatic adaptation inside the pipelines (a *choice* group would select different components based on different inputs, for instance). The dinamics inherent over these new constructs will need a powerful selection language, such as XPath [25], to support higher complexity on XML applications specification.

References

1. Yergeau, F., Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E.: Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation (2004) http://www.w3.org/TR/2004/REC-xml-20040204.
2. Lopes, R., Carriço, L.: APP - Architecture for Pipelined Processing. IADIS International Conference WWW/Internet 2005 (2005)
3. Dijkstra, E.W.: On the role of scientific thought. In: Selected Writings on Computing: A Personal Perspective. Springer-Verlag (1982) 60–66

4. Tarr, P., Ossher, H., Harrison, W., Stanley M. Sutton, J.: N degrees of separation: multi-dimensional separation of concerns. In: ICSE '99: Proceedings of the 21st international conference on Software engineering, Los Alamitos, CA, USA, IEEE Computer Society Press (1999) 107–119
5. Pemberton, S., Austin, D., Axelsson, J., Çelik, T., Dominiak, D., Elenbaas, H., Epperson, B., Ishikawa, M., Matsui, S., McCarron, S., Navarro, A., Peruvemba, S., Relyea, R., Schnitzenbaumer, S., Stark, P.: XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). W3C Recommendation (2002) http://www.w3.org/TR/xhtml1.
6. Bos, B., Çelik, T., Hickson, I., Lie, H.W.: Cascading Style Sheets, level 2 revision 1 – CSS 2.1 Specification. W3C Candidate Recommendation (2004) http://www.w3.org/TR/CSS21.
7. Lenkov, D., Walsh, N.: XML Processing Model Requirements. W3C Working Group Note (2002) http://www.w3.org/TR/proc-model-req.
8. Appelquist, D., Mehrvarz, T., Quint, A.: Compound Document by Reference Use Cases and Requirements Version 1.0. W3C Working Draft (2005) http://www.w3.org/TR/CDRReqs.
9. Newman, D., Patterson, A., Schmitz, P.: XHTML+SMIL Profile. W3C Note (2002) http://www.w3.org/TR/XHTMLplusSMIL.
10. Masayasu, I.: An XHTML + MathML + SVG Profile. W3C Working Draft (2002) http://www.w3.org/TR/XHTMLplusMathMLplusSVG/xhtml-math-svg.html.
11. Apache Foundation: Apache Ant (2000-2004) http://ant.apache.org.
12. Bruchez, E., Vernet, A.: XML Pipeline Language (XPL) Version 1.0 (Draft). W3C Member Submission (2005) http://www.w3.org/Submission/xpl.
13. Cowan, J., Tobin, R.: XML Information Set (Second Edition). W3C Recommendation (2004) http://www.w3.org/TR/xml-infoset.
14. Apache Foundation: Apache Cocoon (1999-2004) http://cocoon.apache.org.
15. Walsh, N.: SXPipe - Simple XML Pipelines. Working Draft (2004) https://sxpipe.dev.java.net/nonav/specs/sxpipe.html.
16. Marsh, J., Orchard, D.: XML Inclusions (XInclude) Version 1.0. W3C Recommendation (2004) http://www.w3.org/TR/xinclude.
17. Berners-Lee, T., Fielding, R., Masinter, L.: RFC2396: Uniform Resource Identifiers (URI): Generic Syntax. Request For Comments (1998) http://www.ietf.org/rfc/rfc2396.txt.
18. Meyer, B.: Object-Oriented Software Construction. 2^{nd} edn. Prentice Hall (1997)
19. Beckett, D., McBride, B.: RDF/XML Syntax Specification (Revised). W3C Recommendation (2004) http://www.w3.org/TR/rdf-syntax-grammar.
20. DCMI Usage Board: DCMI Metadata Terms. DCMI Recommendation (2004) http://dublincore.org/documents/dcmi-terms.
21. Beckett, D., Miller, E., Brickley, D.: Expressing Simple Dublin Core in RDF/XML. DCMI Recommendation (2002) http://dublincore.org/documents/dcmes-xml.
22. Layman, A., Tobin, R., Bray, T., Hollander, D.: Namespaces in XML 1.1. W3C Recommendation (2004) http://www.w3.org/TR/xml-names11.
23. Kay, M.: XSL Transformations (XSLT) Version 2.0. W3C Candidate Recommendation (2005) http://www.w3.org/TR/xslt20.
24. Fallside, D., Walmsley, P.: XML Schema Part 0: Primer Second Edition. W3C Recommendation (2004) http://www.w3.org/TR/xmlschema-0.
25. Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J., Siméon, J.: XML Path Language (XPath) 2.0. W3C Working Draft (2005) http://www.w3.org/TR/xpath20.

O Método de Muenchian revisitado

Isidro Vila Verde[1]

[1] FEUP – Faculdade de Engenharia da Universidade do Porto,
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
jvv@fe.up.pt

Resumo. Um dos problemas clássicos em XSL 1.0 é o agrupamento de Nós. As soluções conhecidas não são triviais e tem âmbitos de aplicação limitados. No presente artigo apresentamos um algoritmo baseado num método conhecido como "método de Muenchian", identificamos estruturas de documentos XML onde este não funciona, apresentamos algumas modificações e evidenciamos a necessidade de, ao definir a regra de agrupamento em XSL, conhecer à partida a estrutura do documento XML fonte, inviabilizando deste modo o uso de bibliotecas. No final, apresentamos uma *stylesheet* genérica de segunda geração para produzir, *on the fly*, algoritmos de agrupamento menos dependentes da estrutura do documento XML.

1. Introdução

Quando em XML temos um conjunto de elementos distintos que partilham o mesmo valor num nó *n* descendente e único no contexto de cada um desses elementos, há por vezes interesse em os agrupar por esse valor, debaixo de novos elementos agregadores, de modo a colocar em evidência essa característica comum e evitar a repetição de ocorrências de nós iguais nesses elementos. Um exemplo simples, agrupar todos os elementos cujo elemento filho *pessoa* contém o valor *x*, debaixo de um novo elemento *pessoa* com um atributo *nome* a assumir esse valor *x* e eliminar o elemento *pessoa* de cada um desses elementos.

Transformar uma estrutura XML noutra estrutura XML onde alguns elementos surgem agrupados por características comuns exige algoritmos de agrupamento eficientes e tanto quanto possível genéricos.

Em XSL 1.0 há várias técnicas de agrupamento [1], [2], [3] e em XSL 2.0, existe mesmo suporte em elementos próprios da linguagem [4]. No entanto, o suporte para XSL 2.0 ainda não está amplamente implementado nos actuais processadores de XSL e, como tal, faz sentido analisar essas técnicas em XSL 1.0. Além disso, mesmo em XSL 2.0, há vantagens em dispor de um XSLT de segunda geração para tirar partido de código fornecido em bibliotecas.

Este artigo surge na sequência da necessidade de criar um XSLT, para transformar em listas HTML ordenadas, os itens de documentos XML que surgem em elementos mistos:

```
<!ELEMENT PARA (#PCDATA|item)*>
```

A procura de uma solução para este problema levou à pesquisa de algoritmos de agrupamento. No decorrer desta pesquisa foram encontrados alguns métodos, mas uma análise mais demorada aos mesmos, permitiu concluir que estes, por um lado estão limitados a determinadas estruturas e, por outro lado, não são passíveis de ser

usados em bibliotecas. Este artigo pretende fazer uma revisão de um desses métodos, apresentar algumas alterações para o tornar mais genérico e propor uma solução para o uso do mesmo em bibliotecas de XSLT's.

Na secção seguinte vai ser revisto um método de agrupamento, apresentando um algoritmo baseado nas técnicas sugeridas por Steve Muench, conhecidas como o método de Muenchian [1]. São evidenciadas as limitações do método exemplificando num caso concreto e é apresentada, na secção 3, uma solução menos restritiva. Com base nessa solução, é proposto na secção 4, um XSLT genérico de segunda geração, para criar dinamicamente *scripts* XSL e, dessa forma, generalizar ainda mais o campo de utilização da solução. Por fim são apresentadas as conclusões e indica-se o trabalho futuro.

2. O método de Muenchian

Suponha-se que temos um documento anotado em XML para descrever um conjunto de pessoas com determinadas propriedades (nome, idade, etc), como o exemplificado na figura 1 e que queremos um XSLT para o transformar num outro documento XML, com os elementos `pessoa` agrupados pelo valor do elemento filho `idade`, com a estrutura exemplificada na figura 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas>
  <pessoa><idade>20</idade><nome>Ana</nome></pessoa>
  <pessoa><idade>25</idade><nome>Joana</nome></pessoa>
  <pessoa><idade>20</idade><nome>Pedro</nome></pessoa>
  <pessoa><idade>25</idade><nome>Sofia</nome></pessoa>
</pessoas>
```

Fig. 1 - Exemplo de estrutura do documento fonte para ser agrupado

```
<?xml version="1.0" encoding="UTF-16"?>
<pessoas>
  <idade anos="20">
    <pessoa><nome>Ana</nome></pessoa>
    <pessoa><nome>Pedro</nome></pessoa>
  </idade>
  <idade anos="25">
    <pessoa><nome>Joana</nome></pessoa>
    <pessoa><nome>Sofia</nome></pessoa>
  </idade>
</pessoas>
```

Fig. 2 - Exemplo da estrutura do documento resultante pretendido

Como se pode verificar os dois documentos descrevem o mesmo conjunto de pessoas mas, no segundo caso, estão agrupadas pelo valor de uma propriedade comum, a idade.

O algoritmo de transformação para converter a primeira estrutura na segunda, usando XSL, não é trivial porque não basta declarar um conjunto de regras que se aplicam a determinados nós. É também necessário identificar os grupos dos elementos

pretendidos (*pessoa*) que contêm o mesmo valor no nó agregador (*idade*) e, é preciso, garantir que há uma regra que será executada para cada um destes grupos.

Uma possível solução, baseada no método de Muenchian, está apresentada na figura 3.

Em prol da clareza do texto convencione-se o seguinte, para o resto deste artigo:

- O elemento que se pretende agrupar será denominado "elemento pretendido"
- O nó pelo qual se pretende efectuar o agrupamento será denominado "nó agregador"
- O elemento pai dos elementos pretendidos será denominado simplesmente por "contexto".
- Pontualmente para relembrar e reforçar a ideia será colocado dentro de parênteses curvos o nome do elemento referido.

Assim na figura 1 o "elemento pretendido" é o elemento *pessoa*, o contexto é o elemento *pessoas* e o nó agregador é o elemento *idade*.

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3       <xsl:key name="k" match="pessoa" use="idade"/>
4       <xsl:template match="*|@*">
5        <xsl:copy>
6         <xsl:apply-templates select="@*|node()"/>
7        </xsl:copy>
8       </xsl:template>
9       <xsl:template match="pessoa[
                count(. | key(' k', idade)[1]) = 1]">
10       <xsl:element name="idade">
11        <xsl:attribute name="anos"><xsl:value-of
    select="idade"/></xsl:attribute>
12        <xsl:for-each select="key('k', idade)">
13         <xsl:copy>
14          <xsl:apply-templates select="@*|node()"/>
15         </xsl:copy>
16        </xsl:for-each>
17       </xsl:element>
18      </xsl:template>
19      <xsl:template match="pessoa"/>
20      <xsl:template match="pessoa/idade"/>
```

Fig. 3 - Um algoritmo de agrupamento baseado no método de Muenchian

Nesta figura, tal como o método de Muenchian propõe, é criada na linha 3 uma chave *k* para os elementos pretendidos (*pessoa*) e indexada pelo valor do nó agregador (*idade*). A cada entrada desta chave vai corresponder o conjunto dos elementos pretendidos que possuem o mesmo valor no nó agregador. Formam-se, desta forma nesta chave, os conjuntos dos elementos que deverão ser agrupados.

Da linha 4 à linha 8 está definida uma regra trivial que se aplica a qualquer nó. Esta regra limita-se a copiar o nó e a aplicar as regras aos seus filhos. Eventualmente, dependendo dos objectivos, estas 4 linhas podem ser omitidas ou substituídas.

Na linha 9 está definido uma regra que, de acordo com o método de Muenchian, se aplica apenas a um dos elementos (neste caso ao primeiro) de cada conjunto indexado

na chave `k`. Garante-se, assim, que há uma regra para cada grupo. É nesta regra que vai efectuar-se o agrupamento dos elementos.

Note-se que no predicado desta regra o cardinal só é um (`[count (…) = 1]`), como é sabido, quando a reunião (`|`) do elemento corrente (`.`) com o primeiro de elemento indexado na chave (`key(' k', idade)[1]`), resulta num conjunto composto por um único elemento, ou seja, quando o elemento corrente é o primeiro elemento devolvido pela indexação na chave. Em todos os outros casos o conjunto reunião conterá dois elementos (o corrente e o primeiro elemento indexado), logo a condição não se verificará. Um método alternativo ao uso da função `count` pode ser visto na referência [1].

Nas linhas 10 a 17 gera-se o novo elemento agregador pretendido (elemento *idade*) com o atributo *anos* a assumir o valor correspondente e que é, nada mais do que, o valor do nó agregador.

Nas linhas 12 a 16 copiam-se todos os elementos pretendidos que contêm, no seu nó agregador, um valor igual ao valor do nó agregador do elemento corrente. A obtenção desses elementos é feita acedendo à chave `k`, usando como índice, o valor do nó agregador (*idade*) do elemento corrente.

Na linha 19 define-se uma regra vazia para todos os outros elementos pretendidos, ou seja, aqueles que não são os primeiros no respectivo conjunto indexado na chave. Garante-se, assim, que estes elementos não são copiados, uma segunda vez, para o resultado, visto já o terem sido no código descrito no parágrafo anterior.

Por fim na linha 20 omite-se a cópia dos nós agregadores, visto a informação, presente nestes nós, ter sido transformada num novo elemento *idade*, criado nas linhas 10 e 11. Note-se que esta é uma decisão que pode não ser tomada noutras circunstâncias.

Por último refira-se que há várias formas de implementar o método de Muenchian. Na referência [1] está um algoritmo ligeiramente diferente do algoritmo aqui descrito. No entanto este é um pouco mais generalista.

3. Método das duas chaves

O método de Muenchian funciona, correctamente, quando os elementos que queremos agrupar pertencem todos ao mesmo contexto particular (isto é, são todos filhos do mesmo elemento pai) e este não contem outros nós filhos para além dos elementos referidos.

Ou seja,

```
count((xpnodes)/parent::*) = 1 and
count((xpnodes)/parent::*/nodes()) = count((xpnodes))
    onde xpnodes é a expressão xpath dos elementos
    que pretendemos agrupar
```

Num caso real a probabilidade disto ocorrer limita bastante a utilização deste método. Será mais provável encontramos os diversos elementos que pretendemos agrupar distribuídos por vários nós da árvore, isto é, pertencentes a contextos particulares diferentes.

Vejamos o caso exemplificado na figura 4, onde temos elementos *pessoa* que pretendemos agrupar, distribuídos por mais de um contexto particular (`count(//pessoa/parent::*) > 1`).

```
<?xml version="1.0" encoding="UTF-8"?>
<pessoas>
  <grupo n="1">
    <pessoa><idade>20</idade><nome>Ana</nome></pessoa>
    <pessoa><idade>25</idade><nome>Joana</nome></pessoa>
    <pessoa><idade>20</idade><nome>Pedro</nome></pessoa>
  </grupo>
  <grupo n="2">
    <pessoa><idade>20</idade><nome>Rita</nome></pessoa>
    <pessoa><idade>20</idade><nome>Tiago</nome></pessoa>
    <pessoa><idade>25</idade><nome>Sofia</nome></pessoa>
  </grupo>
</pessoas>
```

Fig. 4 - Exemplo de uma estrutura para a qual o método de Muenchian não funciona

Neste exemplo temos os elementos pretendidos em dois contextos diferentes. Os elementos no contexto do primeiro elemento *grupo* e os elementos no contexto do segundo elemento *grupo*.
Se o documento XML da figura 4 for transformado pelo XSLT da figura 3, o resultado será ao apresentado na figura 5. Como se pode verificar, os elementos *pessoa* do *grupo* 2 aparecem no resultado como sendo do *grupo* 1. Este resultado dificilmente será o desejado porque deturpa a informação.

```
<pessoas>
  <grupo n="1">
    <idade anos="20">
      <pessoa><nome>Ana</nome></pessoa>
      <pessoa><nome>Pedro</nome></pessoa>
      <pessoa><nome>Rita</nome></pessoa> <!-- errado-->
      <pessoa><nome>Tiago</nome></pessoa> <!-- errado-->
    </idade>
    <idade anos="25">
      <pessoa><nome>Joana</nome></pessoa>
      <pessoa><nome>Sofia</nome></pessoa> <!-- errado-->
    </idade>
  </grupo>
  <grupo n="2" /> <!--vazio por erro na transformação -->
</pessoas>
```

Fig. 5 - Resultado da transformação do XML da figura 4 pelo XSLT da figura 3

```
<pessoas>
  <grupo n="1">
    <idade anos="20">
      <pessoa><nome>Ana</nome></pessoa>
      <pessoa><nome>Pedro</nome></pessoa>
    </idade>
    <idade anos="25">
```

```
      <pessoa><nome>Joana</nome></pessoa>
    </idade>
  </grupo>
  <grupo n="2">
    <idade anos="20">
      <pessoa><nome>Rita</nome></pessoa>
      <pessoa><nome>Tiago</nome></pessoa>
    </idade>
    <idade anos="25">
      <pessoa><nome>Sofia</nome></pessoa>
    </idade>
  </grupo>
```

Fig. 6 - Resultado pretendido para a transformação do XML da figura 4

No caso da figura 4, o resultado que certamente é esperado obter de um algoritmo de agrupamento é o que está representado na figura 6.

O código XSLT da figura 3 não funciona, para esta estrutura, por dois motivos. Primeiro, na selecção do grupo usa-se apenas o primeiro elemento de cada conjunto indexado pela chave *k*, independentemente do contexto, ou seja, são ignorados os grupos cujo valor do nó agregador, já tenha ocorrido num contexto anterior. Segundo, estes conjuntos omitem o contexto desses elementos, logo estes são seleccionados indistintamente e aparecem no primeiro grupo cujo valor seja igual ao valor do seu nó agregador.

Os elementos na chave *k* estão agrupados pelo valor do nó agregador independentemente do contexto onde existem, quando deveriam estar agrupados pelo valor do nó agregador e pelo contexto ao qual pertencem.

Assim é necessário alterar o XSLT para entrar em linha de conta com o contexto. O código XSLT para transformar o XML da figura 4 no XML da figura 6, tendo em consideração o contexto onde existem os elementos pretendidos, está apresentado na figura 7.

```
1     <?xml version="1.0" encoding="ISO-8859-1"?>
2     <xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform
    ">
3a     <xsl:key name="k1"
       match="pessoa[
            not(preceding-sibling::*/idade = idade)]"
       use="generate-id()"/>
3b     <xsl:key name="k2"
       match="pessoa[
              preceding-sibling::*/idade = idade]"
       use="generate-id(preceding-sibling::*[
              idade = current()/idade][last()])"/>
4      <xsl:template match="*|@*">
5       <xsl:copy>
6        <xsl:apply-templates select="@*|node()"/>
7       </xsl:copy>
8      </xsl:template>
9      <xsl:template
            match="pessoa[key('k1', generate-id())]">
```

```
10      <xsl:element name="idade">
11       <xsl:attribute name="anos"><xsl:value-of
                    select="idade"/></xsl:attribute>
12       <xsl:for-each
                  select=".|key('k2', generate-id())">
13        <xsl:copy>
14         <xsl:apply-templates select="@*|node()"/>
15        </xsl:copy>
16       </xsl:for-each>
17      </xsl:element>
18     </xsl:template>
19     <xsl:template match="pessoa"/>
20    </xsl:stylesheet>
```

Fig. 7 - Algoritmo de agrupamento menos restritivo

Este código tem a estrutura base do código da figura 3 mas contém algumas alterações, derivadas de uma aproximação ao problema um pouco diferente. A chave do método de Muenchian é dividida em duas, uma para indexar as primeiras ocorrências do contexto e outra para indexar as restantes ocorrências desse mesmo contexto.

Estas duas chaves vão servir para identificar o grupo (isto é, o primeiro elemento do grupo) e para seleccionar os restantes elementos desse grupo. O modo como as chaves vão ser criadas, vai permitir que os grupos sejam identificados em contexto e não de forma indistinta como acontece no método de Muenchian.

A primeira chave (*k1*) é criada na linha 3a para a primeira ocorrência, nesse contexto, do elemento pretendido (*pessoa*) que contém um valor, no elemento de agrupamento (*idade*), distinto de todos os outros já verificados. A chave é indexada pelo *ID* gerado para o elemento. Este *ID* vai servir para posterior verificação da existência, ou não, de um dado elemento nesta chave. Isto é, vai servir para verificar se um dado elemento identifica um grupo ou não.

Note-se que nesta chave a cada índice corresponde um e um só elemento. Haverá um índice por cada valor distinto do elemento agregador em cada contexto. Se houver *m* contextos e cada contexto tiver *n* valores distintos teremos uma chave com *nxm* índices que identificam outros *nxm* grupos distintos.

A segunda chave (*k2*) contém todos os elementos pretendidos (*pessoa*) que, contendo o mesmo valor no elemento agregador (*idade*) e estando no mesmo contexto, não são as primeiras ocorrências. Isto é conseguido pelo uso do predicado [`preceding-sibling::*/idade = idade`], que garante a existência de, pelo menos, um elemento anterior no mesmo contexto e com o mesmo valor no elemento agregador. Nesta chave os elementos são indexados pelo ID do primeiro elemento ocorrido. Assim quer a chave *k1* quer a chave *k2* tem os mesmos índices e, para um dado índice, a chave *k1* devolve o elemento que ocorre em primeiro lugar, enquanto a chave *k2* devolve o conjunto dos elementos que ocorrem depois do primeiro. Pode-se dizer, em certo sentido, que são complementares.

Para gerar o ID do primeiro elemento ocorrido, para esta chave *k2*, usamos o eixo `preceding-sibling`, com o predicado `[idade = current()/idade]`. Isto devolve o conjunto dos elementos precedentes que contêm o mesmo valor no elemento agregador. O primeiro destes é o elemento que nos interessa mas, como é sabido, quando este conjunto é obtido pelo eixo `preceding-sibling` o primeiro

a ocorrer no documento XML, surge em último, razão pela qual se usa a função `last()` no segundo predicado.

Definidas estas duas chaves é fácil agora definir uma regra XSL para cada grupo e dentro desta seleccionar os elementos do mesmo contexto que contêm o mesmo valor no nó de agrupamento.

Na linha 9 usa-se o predicado para impor a condição do elemento estar presente na chave, ou seja, para garantir que é a primeira ocorrência do grupo. Temos assim uma regra por cada grupo que queremos formar.

A selecção dos elementos a serem copiados para este novo elemento é efectuada na linha 12 usando a segunda chave. Como esta chave está indexada pelo *ID* da primeira ocorrência (ou seja, o elemento corrente) para obter-se a lista de todos os elementos, do grupo, basta indexar esta chave pelo valor do *ID* gerado para elemento corrente. Visto que este também tem de ser copiado e não está presente na chave `k2`, é seleccionada a reunião do elemento corrente com o conjunto de elementos devolvidos pela indexação da chave.

O resto do código, como se pode observar, mantém-se inalterável relativamente ao código presente na figura 3.

Há outras soluções mas que por não recorrerem a chaves têm ordem de complexidade *O(N^2)* (ver argumentação de Michael Kay na *thread* iniciada na referência [2]). Esta solução por se basear em chaves, tal como o método de Muenchian, tem ordem de complexidade equivalente, ou seja *O(N log N)*.

A solução apresentada funciona para documentos com a estrutura exemplificada quer na figura 1 quer na figura 4, ou seja é mais genérica que o método de Muenchian, sem ser um algoritmo mais complexo.

4. Método genérico

A solução apresentada na figura 7 é mais genérica que o método de Muenchian, mas, mesmo esta solução está intrinsecamente associado a documentos XML com uma estrutura onde os nomes dos elementos pretendidos, os nomes ou os tipos dos nós agregadores e a relação de parentesco entre eles são bem conhecidas e inalteráveis.

Se quisermos agrupar outro tipo de documentos onde uma ou mais destas condições não se verifique, teremos no mínimo de copiar este código e altera-lo para reflectir a nova estrutura dos documentos fonte.

Esta situação não é prática por diversas razões, entre as quais a reutilização do código e a manutenção do mesmo. Além disso não é passível de ser utilizado em bibliotecas de software e é propensa a erros de manuseamento do código. Isto para já não falar no tempo despendido por um programador (iniciado nesta problemática) a compreender o método.

Porém, como se pode constatar, a mudança (parcial) de estrutura dos documentos fonte só envolve alterações (na solução da figura 7) nas linhas 3, 9, 12 e eventualmente nas linhas 10 e 11. O desejável era ter um código genérico e parametrizável para cada estrutura particular (dentro de certos limites) dos documentos XML fonte, o qual faria as alterações identificadas.

Este objectivo pode ser conseguido, entre outras formas, pela utilização de um XSLT de segunda geração [5], [6]. Na figura 8, apresenta-se o código correspondente a este XSLT.

Numa XSLT de segunda geração o objectivo é gerar um resultado que seja ele próprio um código XSLT, isto é, os elementos deverão ser criados no espaço de nomes do próprio XSLT que os cria. Isto gera conflitos entre os elementos XSL e os elementos a criar. No parágrafo seguinte faremos uma explicação breve da técnica usual para evitar esses conflitos.

No XSLT da figura 8 é definido o prefixo `dxsl` como sendo um `alias-namespace` (linha 2) e é associado, no resultado da transformação através do uso do prefixo `xsl` (linha 3) já definido na linha 2, ao espaço de nomes `http://www.w3.org/1999/XSL/Transform`. O uso do *alias* garante que os elementos com este prefixo não são interpretados com elementos XSL. A associação, no resultado, garante que este é um documento XSLT definido no espaço de nomes correcto.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <xsl:stylesheet version="1.0"
       xmlns:xsl=http://www.w3.org/1999/XSL/Transform
       xmlns:dxsl="alias-namespace">
3    <xsl:namespace-alias stylesheet-prefix="dxsl"
                                result-prefix="xsl"/>
4    <xsl:param name="ele"/>
5    <xsl:param name="bn"/>
6    <xsl:param name="group" select="'group'"/>
7    <xsl:param name="value" select="'value'"/>
8    <xsl:template match="/">
9     <dxsl:stylesheet version="1.0">
10     <xsl:if test="$ele and $bn">
11      <dxsl:key name="k1"
         match="{$ele}[
              not(preceding-sibling::*/{$bn} = {$bn})]"
         use="generate-id()"/>
12      <dxsl:key name="k2"
         match="{$ele}[
                   preceding-sibling::*/{$bn} = {$bn}]"
         use="generate-id(preceding-sibling::*[
                {$bn} = current()/{$bn}][last()])"/>
13      <dxsl:template
              match="{$ele}[key('k1', generate-id())]">
14       <dxsl:element name="{$group}">
15        <dxsl:attribute name="{$value}">
15         <dxsl:value-of select="{$bn}"/>
17        </dxsl:attribute>
18        <dxsl:for-each
               select=".|key('k2', generate-id())">
19         <dxsl:copy>
20          <dxsl:apply-templates select="@*|node()"/>
21         </dxsl:copy>
22        </dxsl:for-each>
23       </dxsl:element>
24      </dxsl:template>
```

```
25          <dxsl:template match="{$ele}"/>
26        </xsl:if>
27      </dxsl:stylesheet>
28    </xsl:template>
29  </xsl:stylesheet>
```

Fig. 8 - XSLT genérico e parametrizável para gerar XSLT's *on-the-fly*

Este código (figura 8) é composto por uma única regra que transforma a raiz de um qualquer documento fonte (ex: `<dummy/>`) numa regra XSL, parametrizável em função dos valores passados ao *script*.

É efectuado, na linha 10, um teste para verificar se o parâmetro que define o elemento pretendido (`$elem`) e o parâmetro que define o nó agregador (`$bn`) estão definidos. Caso não estejam definidos o *script* é perfeitamente inócuo, pois não gera qualquer elemento para além do elemento raiz (`<xsl:stylesheet/>`).

O elemento agregador e o atributo que define a característica comum são definidos nas linhas 14 e 15 respectivamente. Os nomes destes dois nós são definidos com um valor por omissão nas linhas 6 e 7 respectivamente.

Da linha 11 à linha 25, a menos da troca parcial, do valor dos atributos xsl por parâmetros, o código com prefixo *dxsl* é semelhante ao código da figura 7.

O nome do novo elemento agregador e do seu atributo também podem ser parametrizáveis, mas se forem omitidos assumem os valores omissos definidos nas linhas 6 e 7, respectivamente.

Por fim note-se ainda que duas regras presentes no código da figura 7 estão ausentes aqui. São elas as regras para copiar todos os outros nós e a regra que omite o nó agregador. A razão dessa omissão prende-se com a impossibilidade de saber à priori em que cenários este código vai ser utilizado e se essas regras fazem ou não sentido nesse caso particular. Assim é da responsabilidade do programador que utiliza este código definir (ou não) essas regras.

Na figura 9, está representado um script XSL para agrupar os documentos XML do tipo exemplificado na figura 11. Pretende-se que sejam agrupados os elementos filhos dos filhos do nó raiz, pelo valor do atributo descendente anos.

```
1     <?xml version="1.0" encoding="UTF-8"?>
2     <xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3      <xsl:include href="http://localhost/dxsl.xsl?
         ele=/populacao/*/*&amp;
         bn=.//@anos&amp;
         group=idade&amp;value=anos"/>
4      <xsl:template match="*|@*">
5       <xsl:copy>
6        <xsl:apply-templates select="@*|node()"/>
7       </xsl:copy>
8      </xsl:template>
9      <xsl:template match="idade"/>
10    </xsl:stylesheet>
```

Fig. 9 - Uso do XSLT genérico para agrupar o exemplo da figura 11

Este *script* inclui o resultado do XSLT genérico da figura 8, para o qual passa os parâmetros via interface CGI (nota: foi usado um *pipeline* na plataforma cocoon, o qual permite passar os parâmetros recebido via interface CGI para o *script* XSLT). Usa as regras incluídas para agrupar os nós pretendidos e define as regras que o programador entende serem necessárias para os restantes nós.

Como se pode observar, a expressão xpath para identificar os elementos pretendidos é dado por `populacao/*/*` e para o nó agregador é dada por `.//@anos`. Estas expressões são passadas nos campos *ele* e *bn*.

Na figura 10 está representado o *pipeline* do cocoon usado para disponibilizar este XSLT genérico de segunda geração.

```
<map:pipeline type="noncaching" internal-only="false">
 <map:match pattern="grouping/dxsl.xsl">
  <map:generate src="grouping/dummy.xml"/>
  <map:transform src="grouping/dxsl.xsl">
   <map:parameter
     name="use-request-parameters" value="true"/>
  </map:transform>
  <map:serialize type="xml"/>
 </map:match>
</map:pipeline>
```

Fig. 10 – Pipeline para o XSLT genérico

Neste *pipeline* o ficheiro *grouping/dummy.xml*, como o próprio nome indicia é um documento composto apenas pelo elemento raiz. O código da figura 9 está no ficheiro *grouping/dxsl.xsl*. Ao ser invocado o transformador são passados como parâmetros os campos recebidos do cliente.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<populacao>
  <porto>
    <cedofeita>
      <recenseado>
        <idade anos="20"/><nome>Rita</nome>
      </recenseado>
    </cedofeita>
    <bonfim>
      <idade anos="25"/><nome>João</nome>
    </bonfim>
    <paranhos>
      <recenseado>
        <idade anos="25"/><nome>Marta</nome>
      </recenseado>
    </paranhos>
  </porto>
  <ptlima>
    <arcos>
      <idade anos="20"/><nome>Rui</nome>
    </arcos>
    <moreira>
      <idade anos="20"/><nome>Ana</nome>
    </moreira>
```

```
    <sa>
      <idade anos="25"/><nome>Luis</nome>
    </sa>
  </ptlima>
</populacao>
```

Fig. 11 – Exemplo de um documento XML, a ser agrupado pelo *script* da figura 10

Como se pode ver, pelo exemplo acima, este XSLT genérico pode ser utilizado em diversos tipos de documentos, com estruturas com alguma complexidade, desde que, os elementos pretendidos e os elementos agregadores, possam ser expressos em expressões xpath e que estas possam ser usadas nos atributos XSL utilizados.

De uma forma empírica podemos dizer que o XSTL genérico apresentado funcionará bem para documentos XML que obedeçam ás seguintes condições:

```
count($bn) = 1
no contexto de cada elemento identificado por $elem,

onde $bn e $elem representam as expressões xpath
passadas, como parâmetros, ao script e identificam
respectivamente os nós agregadores e os
elementos pretendidos.
```

5. Conclusões

Neste artigo apresentou-se um algoritmo de agrupamento baseado no método de Muenchian, expusemos dois conjuntos de limitações e apresentamos um método alternativo para solucionar um desses conjuntos de limitações. Para permitir o uso de bibliotecas e simplificar a escrita de *scripts* de agrupamento, apresentamos um XSLT genérico de segunda geração. Por fim exemplificamos a utilização desse XSLT num script para agrupar uma estrutura com alguma complexidade e, com este exemplo, evidenciamos a simplicidade e modularidade que ganhamos.

Com estas duas soluções complementares, são ultrapassados alguns obstáculos à implementação de transformação que envolvam uma ou mais operações de agrupamento.

A primeira solução elimina a necessidade dos elementos pretendidos terem de estar todos definidos no mesmo contexto. A segunda permite ao programador usar uma biblioteca de módulos para gerar dinamicamente regras de agrupamento à medida do tipo de estrutura do documento fonte.

Por vezes o tempo que um programador despende a tentar entender o funcionamento do método de Muenchian é bastante significativo. Com o XSLT genérico, tudo o que o programador precisa de definir são duas expressões xpath.

Por último, falta referir que não foi abordado aqui a solução para agrupar documentos cujos elementos pretendidos, não sejam os únicos filhos do contexto onde existem. Um exemplo é o caso referido na introdução a este artigo e que conduziu a este trabalho. Essa solução será objecto de um outro artigo futuro.

Referências

1. Jeni Tennison;, Grouping Using the Muenchian Method, Publicado em http://www.jenitennison.com/xslt/grouping/muenchian.html
2. Sergiu Ignat , Recursive grouping - simple, XSLT 1.0, fast non-Muenchian grouping method, Publicado em http://www.biglist.com/lists/xsl-list/archives/200412/msg00865.html
3. M. David Peterson, New Alternative to Muenchian Method of Grouping?, Publicado em <xstlblog/> http://www.xsltblog.com/archives/2004/12/new_alternative.html
4. Bob DuCharme, Grouping With XSLT 2.0, Publicado em http://www.xml.com/lpt/a/2003/11/05/tr.html
5. Bob DuCharme, Automating Stylesheet Creation, Publicado em XML.com http://www.xml.com/pub/a/2005/09/07/autogenerating-xslt-stylesheets.html
6. Jirka Kosek,SLT Reflection, Publicado em XML.com http://www.xml.com/pub/a/2003/11/05/xslt.html

Implementação de um modelo baseado em XML para suporte da dinâmica processual de negócio

Gilberto Rocha[1], Isidro Vila Verde[1], Rui Humberto Pereira[2]

[1] Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, 4200-465 Porto Portugal.
{mrs03015,jvv}@fe.up.pt
[2] Instituto Politécnico do Porto
Rua Dr. Roberto Frias, 712 4200-465 Porto Portugal.
rhp@iscap.ipp.pt

Resumo. Este artigo descreve um modelo para implementação de plataformas dinâmicas de negócio, exclusivamente baseadas em tecnologias XML. Apresenta um caso real que implementa as suas diversas camadas, apresentação, lógica e dados, em XML. A comunicação entre camadas é assegurada por serviços Web (WS), tornando esta arquitectura orientada aos serviços (SOA). O modelo proposto sustenta toda a programação do processo de negócio numa linguagem de alto nível, o WS-BPEL, proporcionando, desse modo, condições de adaptação ao dinamismo exigido pelo negócio da organização e à heterogeneidade dos sistemas. O caso real é de uma secretaria electrónica que surge no contexto dos portais Web universitários. O sistema desenvolvido visa oferecer um conjunto de serviços para acesso a informação e para o despoletar de acções computacionais e/ou humanas.

1 Introdução

Nestes últimos anos, a proliferação da Internet, quer a nível tecnológico, quer a nível social, permitiu democratizar o acesso à informação e melhorar a interacção com o mundo da informação.

Associado a este fenómeno, foram aparecendo inúmeros serviços on-line, que evoluem de acordo com os requisitos que lhe vão sendo impostos em função dos objectivos. A continuação desta evolução, está dependente das tecnologias que a suportam, as quais devem permitir uma grande flexibilidade para que seja possível satisfazer e adaptar-se aos requisitos que vão surgindo.

Neste contexto surgiram os portais Web universitários e, na sequência destes, as secretarias electrónicas. A aproximação clássica aos serviços do tipo secretaria electrónica, consiste em fazer um levantamento complexo à lógica do negócio e repercuti-la em código aplicacional. Este modelo de desenvolvimento de código para implementação de lógicas de negócio, tem obviamente desvantagens, a primeira das quais é a adaptabilidade às alterações processuais, isto é, alteração à lógica de negócio.

Por outro lado, hoje em dia as organizações têm uma herança no que respeita aos sistemas de informação já existentes (*legacy*), o que torna complicado desenvolver e acrescentar novas funcionalidades. Se disponibilizarmos estes sistemas como serviços e usarmos o modelo SOA [1], [2], [3] como elemento agregador, conseguimos construir facilmente novas aplicações, e/ou reestruturar aplicações já existentes.

Um serviço on-line, pode ter que interagir com várias aplicações e algumas delas utilizam tecnologias de certa forma obsoletas. Podemos envolver essas aplicações com uma camada aplicacional que interage com a API da aplicação e disponibiliza um serviço Web para atender pedidos de outras aplicações ou serviços Web.

Com base nisto, foi desenvolvido no âmbito da tese de mestrado um Modelo totalmente baseado em XML [4], que procura satisfazer a dinâmica dos requisitos inerentes à evolução do modelo de negócio. O Modelo desenvolvido é constituído por um conjunto de tecnologias de apresentação, lógica e armazenamento baseadas em XML. Pretendeu-se avaliar a exequibilidade destas tecnologias para a agilização dos serviços à mudança.

Este Modelo foi implementado numa secretaria electrónica, que pode conter vários processos, e esses processos podem ter que ser reconfigurados ou alterados ao longo do tempo. Nas aplicações Web actuais, isso implica alterar o código da aplicação. Neste artigo apresenta-se uma possível solução que envolve a utilização de um motor BPEL [5], onde é especificado o processo de negócio.

Na próxima secção, apresentamos a contextualização da necessidade deste modelo e fazemos uma breve descrição das arquitecturas e tecnologias existentes, que visam dar resposta a estes problemas. Na secção três, apresentamos e descrevemos um Modelo que assenta no uso estrito de tecnologias XML, mas permitindo a interface via serviços Web a aplicações convencionais. Na secção quatro, ilustramos a implementação de um módulo de uma secretaria electrónica de acordo com este modelo. Por fim apresentaremos as conclusões resultantes da implementação e das expectativas daí extraídas.

2 Estado de arte

O mundo dos negócios é um ambiente que está sempre em constante alteração. As empresas evoluem, querem expandir-se e melhorar a sua própria estrutura, quer por movimentos de aquisição e fusão, quer por adaptações ou reestruturações internas. Este dinamismo obriga os sistemas de informação a estarem preparados para rapidamente reflectirem as novas realidades e as prováveis alterações nas lógicas de negócio da empresa. Por outro lado, em qualquer sistema de informação, os requisitos são dinâmicos e constantemente se alteram.

Se a estrutura de suporte ao negócio da empresa for orientada ao serviço, torna muito mais fácil ultrapassar as exigências impostas pelas situações em cima explicadas e satisfazer rapidamente os novos objectivos das organizações.

Ao desenvolvermos uma aplicação para suporte de lógica de negócio, ou temos tecnologias que são capazes com eficiência e rapidez de adaptar-se aos processos, ou então estamos a construir uma aplicação que quando estiver totalmente pronta, estará já desactualizada.

As linguagens de alto nível de descrição de processos, permitem definir o núcleo do sistema de negócio, para que seja dinâmico e possibilite a convivência inter-aplicacional num ambiente complexo e heterogéneo. Em ambientes onde existe uma grande diversidade de aplicações, em diferentes tecnologias, a implementação de processos pode ser bastante simplificada com a utilização de linguagens de definição de processos de negócio.

O processo de negócio é uma sequência de interacções entre diversas entidades internas e/ou externas a uma organização, que tem como finalidade modelar o negócio da organização. Um bom processo de negócio deve maximizar e flexibilizar a reestruturação e/ou a integração de novas estruturas de negócio. Isto torna os processos complexos e dinâmicos, implicando constantes alterações com o objectivo da optimização. Mais uma vez, as linguagens de definição processual podem minimizar os inconvenientes do carácter dinâmico dos processos organizacionais.

Uma aplicação convencional, que implementa um processo de negócio, tem embebido no próprio código da aplicação a lógica de negócio. Isto implica que sempre que seja alterado o processo de negócio, é necessário alterar o código da aplicação, tornando complexa a gestão do processo de negócio.

2.1 SOA

A Service Oriented Architecture (SOA) tem como objectivo orientar as aplicações à disponibilização como serviços. Ao disponibilizarmos uma determinada aplicação como serviço, permitimos que essa aplicação consuma, troque e disponibilize informação. Estas aplicações já têm um determinado papel dentro de uma organização, permitindo a sua reutilização e minimizando o impacto da mudança.

A utilização de arquitecturas SOA permite uma maior facilidade de adaptação do sistema a novos requisitos e dá às organizações uma maior capacidade para poderem responder de uma forma rápida e eficiente às exigências do mercado.

As estratégias de desenvolvimento e implementação assentes em SOA's podem variar de acordo com o facto de se ter que alterar serviços existentes, desenvolver novos serviços ou integrar serviços que foram desenvolvidos fora da organização.

2.2 Linguagens de descrição de processos

Uma linguagem de descrição de processos suporta a lógica dos processos de negócio e deve permitir definir a interoperabilidade entre aplicações.

Um processo de negócio pode ser descrito como uma orquestração de interacções, onde existe um ponto central que vai ser encarregue de coordenar as aplicações, ou pode ser descrito como uma coreografia, onde a definição do processo de negócio vai estar distribuído entre as aplicações que estão envolvidas. Numa coreografia, as aplicações agem entre si de acordo com a definição do processo de negócio, mas a implementação é mais difícil, porque coloca problemas de sincronização.

2.3 Base de dados XML

Uma base de dados XML, tem como unidade fundamental de armazenamento, um documento XML, análogo às bases de dados relacionais que utilizam a linha de uma tabela como unidade de armazenamento.

Não precisa de nenhuma sub-camada física de armazenamento em especial, pode ser implementada em cima de uma base de dados relacional (ex: o Oracle disponibiliza isso), hierárquica, orientada ao objecto ou em cima de qualquer formato proprietário de armazenamento.

Como é utilizado XML, e devido à sua extensibilidade, é possível estruturar e adicionar informação de uma forma mais flexível do que a oferecida por um modelo relacional.

De modo a garantir que, a informação contida na base de dados, tem uma determinada estrutura e respeita determinadas condições, pode-se utilizar XML *Schemas* para fazer a validação.

A existência de linguagens de interrogação e interacção, como o XQuery [6],[7],[8] e o XUpdate [9],[10], permitem integrar em plataformas XML a interacção directa com as bases de dados, sem recurso a linguagens de programação convencional.

3 Modelo desenvolvido

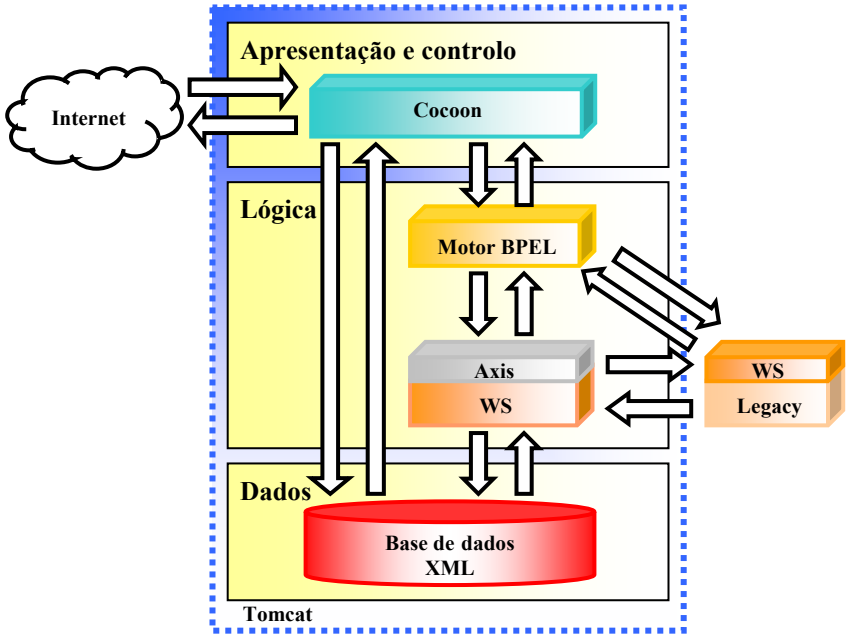


Fig. 1. Arquitectura do modelo

O modelo desenvolvido é constituído pelas camadas de, apresentação e controlo, lógica e dados. A camada de apresentação e controlo, é implementada pelo Cocoon[11],[12] e consiste em atender pedidos Web e interagir com o sistema de negócio. A camada de lógica está distribuída entre o motor BPEL e os serviços Web. É nesta camada de lógica que vai ser especificado e implementado o processo de negócio. A camada de dados é implementada numa base de dados XML.

O módulo do Cocoon é responsável pelas componentes de interacção com o utilizador, interface com os mecanismos de *backend* e pela formatação dos dados para apresentação. Quando o utilizador faz um pedido que envolve uma interacção processual assíncrona, o Cocoon selecciona uma *pipeline* que começa com um *generator* do tipo *Web services* que por sua vez interage com o motor BPEL. Quando o pedido do utilizador é para o acesso a informação já disponível, é usada uma *pipeline* do Cocoon cujo o *generator* é do tipo XQuery e é usado para interrogar directamente a base de dados XML e obter a resposta.

O motor BPEL implementa a lógica de alto nível, e fornece o suporte para as interacções assíncronas (computacionais ou humanas). A lógica de baixo nível, isto é, quando é necessário recorrer a operações ou procedimentos cujo nível de detalhe está muito associado à plataforma tecnológica, é implementado em classes Java disponibilizadas como serviços Web na plataforma Axis [13],[14]. Desta forma conseguimos colocar a lógica do negócio expressa apenas por uma linguagem de alto nível, o BPEL (usando editores gráficos). Remetemos para a programação convencional os pormenores de implementação, que devem ser transparentes para a lógica de negócio. Muitas vezes parte desta camada lógica já existe, em aplicações convencionais, nas plataformas tecnológicas mais diversas e, com este modelo, pode ser reutilizada.

A interacção com aplicações já existentes é efectuada recorrendo ao uso de serviços Web, que fazem a interface a essas aplicações.

Visto que todas as tecnologias usadas são baseadas em XML, e os dados a armazenar são também eles documentos XML, a opção por uma base de dados centrada ao documento, como é o caso das bases de dados XML, parece-nos ser a opção adequada, por já existirem diversas linguagens de interrogação para XML e diversas implementações.

4 Implementação do Modelo

Foi implementado um módulo de uma secretaria electrónica para disponibilização do serviço de emissão de certidões que, além de permitir efectuar os pedidos e acompanhar os estados dos mesmos, envolve ainda um processo de negócio composto pelas acções de pedir, pagar e emitir as certidões.

4.1 Apresentação e controlo

O Cocoon é uma ferramenta desenvolvida em Java que corre num contentor Web (ex: Tomcat) e permite definir *pipelines* compostas por sequências de acções de obtenção, transformação e serialização de documentos XML.

As *pipelines* são definidas num ficheiro de nome *sitemap.xmap* e são seleccionadas pelo Cocoon em função do tipo de pedido HTTP [15].

A selecção da *pipeline* que vai ser executada é feita por uma acção de *matching* entre o URL do pedido e um padrão definido no elemento `<map:match pattern="URL"/>`.

As *pipelines* contêm pelo menos o *generator* que acede à fonte de informação e gera uma sequência de eventos SAX [16],[17]. O *serializer* consome eventos SAX e produz o resultado. Opcionalmente, uma *pipeline* pode conter um ou vários *transformers*.

O Cocoon disponibiliza um conjunto de *generators* de vários tipos, entre os quais, o XQuery e o XSP [18]. O *generator* do tipo XQuery processa documentos definidos em XQuery, e o *generator* do tipo XSP permite gerar dinamicamente, de uma forma análoga aos JSP e aos PHP, documentos XML. Os *transformers* permitem, como o próprio nome indica, transformar o resultado do estágio anterior. Esta operação é normalmente conseguida pelo uso de um ou vários *scripts* escritos em XSLT [19].

4.1.1 Interface

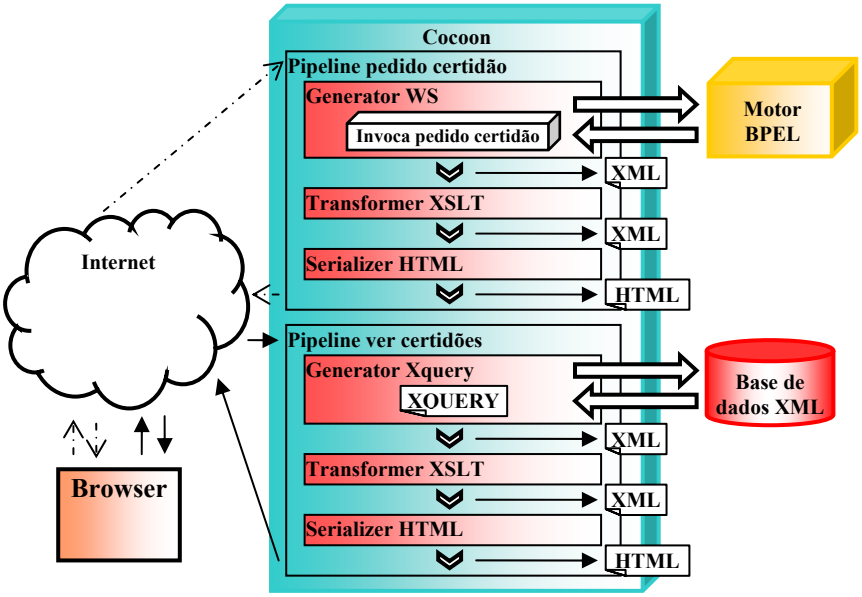


Fig. 2. Módulo de Interface ao utilizador

Quando um utilizador, por intermédio do browser, decide interagir no processo de pedido de certidão, é gerado um pedido HTTP que é recebido pelo Cocoon. Em seguida a *pipeline* referente à interacção começa a ser executada. Primeiro é solicitado o *generator* que invoca um serviço Web, disponibilizado pelo motor BPEL, para se proceder ao desencadear da acção. A resposta será um documento XML que irá ser transformado num documento HTML [20] pelo(s) *transformer(s)* XSLT. Finalmente o *serializer* envia para o browser a página HTML.
Por outro lado quando há um pedido para consulta do estado do pedido de certidão, é desencadeada uma outra *pipeline* cujo o *generator* acede e executa uma XQuery directamente sobre a base de dados. Os resultados desta *query* são, à semelhança do caso anterior, transformados, serializados e devolvidos ao cliente.

Em seguida, vão ser mostrados dois exemplos de pipelines, a primeira utiliza um *generator* que acede directamente à base de dados XML, utilizando XQuery, a segunda, utiliza um *generator* do tipo *serverpages* (XSP), que retorna um documento XML resultante da invocação de um serviço Web, disponibilizado pelo motor BPEL.

```
<?xml version="1.0"?>
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">
<map:pipelines>
  <map:pipeline>
    <map:match pattern="dadosPessoais.x">
        <map:generate src="dadospessoais\dadospessoais.xq"
                      type="xquery"/>
        <map:transform src="dadospessoais\dadospessoais.xsl"/>
        <map:serialize type="html"/>
    </map:match>
  </map:pipeline>
  <map:pipeline>
    <map:match pattern="emitir.x">
        <map:generate src="emitir\emitir.xsp"
                      type="serverpages"/>
        <map:transform src="emitir\emitir.xsl"/>
        <map:serialize type="html"/>
    </map:match>
  </map:pipeline>
</map:pipelines>
</map:sitemap>
```

Fig. 3. Definição das *Pipelines* (Sitemap.xmap)

Na Fig. 3 a primeira *pipeline* é executada sempre que o URL termina em `dadosPessoais.x`. Esta pipeline executa a XQuery definida no ficheiro `dadospessoais.xq`. O resultado desta query é transformado pelo XSLT presente no ficheiro `dadospessoais.xsl`.

A segunda *pipeline* invoca um *generator* do tipo XSP. Como o Cocoon não disponibiliza nenhum *generator* do tipo *Web services* utiliza-se um XSP que invoca um serviço Web. Na Fig. 4 apresenta-se o XSP para invocar o serviço Web disponível em `http://localhost/EmitirService`, e constrói-se a mensagem SOAP [21],[22] para a operação Emitir.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsp:page language="java"
  xmlns:xsp="http://apache.org/xsp"
  xmlns:xsp-request="http://apache.org/xsp/request/2.0"
  xmlns:soap="http://apache.org/xsp/soap/3.0">
  <search-results>
    <soap:call
  url="http://localhost/EmitirService"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
      <Emitir>
        <soap:enc/>
        <numeroUtilizador xsi:type="xsd:int">
          <xsp-request:get-parameter name="numero"/>
        </numeroUtilizador>
        <ref xsi:type="xsd:string">
          <xsp-request:get-parameter name="ref"/>
        </ref>
      </Emitir>
   </soap:call>
 </search-results>
</xsp:page>
```

Fig. 4. XSP para invocar o serviço WEB "EmitirService"

Neste XSP faz-se uso da *SOAP Logicsheet* [24], para definir a mensagem SOAP e invocar o serviço WEB disponibilizado pelo BPEL.

4.2 Lógica

A lógica na nossa implementação está dividida pela lógica do processo de negócio implementada em BPEL e pela lógica de mais baixo nível implementada em serviços Web desenvolvidos em Java.

4.2.1 A lógica do processo de negócio

O BPEL é uma linguagem XML para especificar o comportamento de um processo de negócio. É baseado em serviços Web, suporta interacções assíncronas e permite definir o controlo de fluxo. Um processo BPEL é definido em termos das suas interacções com outros serviços Web, através da definição de parcerias (*partners*). Um parceiro pode disponibilizar serviços Web para o processo BPEL, pode requerer serviços do processo BPEL e pode participar numa interacção bi-direccional (pedido/resposta) com o processo BPEL.

O BPEL orquestra serviços Web, especificando a ordem em que vai chamar uma colecção de serviços e atribui responsabilidades (papel) para cada um dos serviços aos parceiros. A definição de um processo BPEL consiste em definir as relações de parceria e a descrição do processo executável.

Na Fig. 5 apresentamos a lógica do processo de negócio de pedido de certidão que está implementada em BPEL.

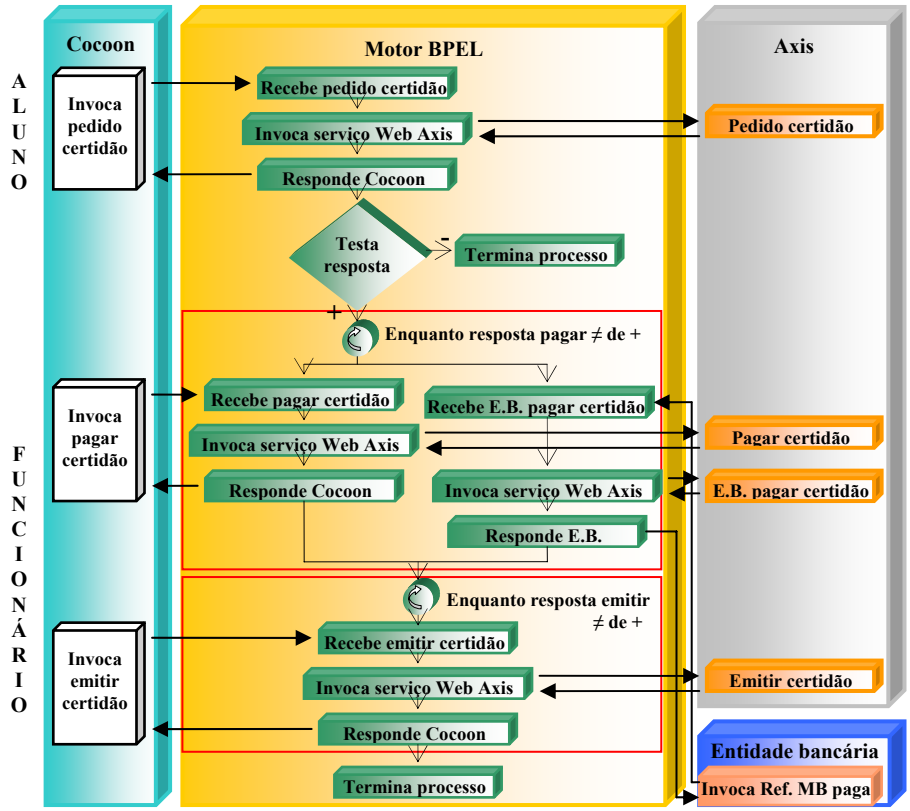


Fig. 5. Diagrama do processo de negócio (pedido de certidão)

Este processo de negócio está descrito num documento XML em linguagem BPEL. Este documento é utilizado pelo motor BPEL para pôr em prática o processo de negócio. Nesta implementação optou-se por utilizar o ActiveBPEL.

Na implementação efectuada, há várias etapas. Começa pelo pedido de certidão, que se traduz na criação de uma nova instância e de seguida invoca o serviço Web, que é responsável por verificar se a certidão tem condições para ser pedida. Qualquer que seja a resposta do serviço Web, é gerado um resultado para o Cocoon. Se a resposta foi negativa o processo BPEL termina de imediato. Como a certidão necessita ser paga, surgem duas soluções de pagamento, ou o utilizador paga a certidão ao balcão da secretaria e, nesse caso, o funcionário ao interagir com a plataforma vai invocar o serviço Web disponibilizado pelo motor BPEL, ou o utilizador paga por Multibanco e, por sua vez, a entidade bancária por intermédio de uma aplicação invoca outro serviço Web disponibilizado pelo motor BPEL. A terceira e última etapa do processo consiste na emissão da certidão. Esta interacção, depois de executar as acções correspondentes, termina o processo BPEL.

Note-se que estas 3 fases, acima descritas (pedido, pagamento e emissão), ocorrem em instantes de tempo distantes uns dos outros. Podem passar-se alguns dias entre cada uma delas.

Na Fig. 6 está apresentado um excerto do código BPEL do processo de negócio acima exposto. Este começa por declarar os *partnersLinks* envolvidos no processo, relativos ao pedido de uma certidão, quer do lado do Cocoon quer do lado do Axis. São definidas duas `correlationSet` para associar cada instância às operações respectivas. A sequência de operações inicia-se com a criação de uma instância do processo, quando o motor BPEL recebe uma mensagem do *partnerLink* `cocoonPedido` invocando a operação `pedirCertidao`. De seguida, o motor BPEL invoca a operação `efectuarPedido` através do *partnerLink* `axisPedido` que é responsável pelos pedidos de certidão, o qual responde com uma mensagem SOAP que é depois encaminhada para o *partnerLink* `cocoonPedido` através de.um `reply` à mensagem recebida.

```
<process name="certidoes"
   targetNamespace="http://gilberto.com/certidoes"
   xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
   xmlns:cocoonPedido="http://gilberto.com/cocoonPedido"
   xmlns:axisPedido="http://gilberto.com/AxisPedirCertidao">
<!—- Definição dos correlationSets -->
    <correlationSets>
        <correlationSet properties="cocoonPedido:ref" name="corrRef"/>
        <correlationSet properties="cocoonPedido:mb" name="corrMb"/>
    </correlationSets>
<!—- Definição dos partnerLinks envolvidos -->
    <partnerLinks>
       <partnerLink name="cocoonPedido"
           partnerLinkType="cocoonPedido:cocoonPedirCertidaoLT"
           myRole="pedirCertidaoProvider"/>
       <partnerLink name="axisPedido"
           partnerLinkType="axisPedido:axisPedirLT"
           partnerRole="pedirCertidaoProvider"/>
              .......
    </partnerLinks>
<!—- Definição das variáveis para guardar mensagens -->
    <variables>
       <variable name="cocoonPedInVariable"
           messageType="cocoonPedido:cocoonPedidoIN"/>
       <variable name="cocoonPedOutVariable"
           messageType="cocoonPedido:cocoonPedidoOUT"/>
       <variable name="axisPedInVariable"
           messageType="axisPedido:efectuarPedidoRequest"/>
       <variable name="axisPedOutVariable"
           messageType="axisPedido:efectuarPedidoResponse"/>
              .......
    </variables>
<!—- Conjunto de actividades executadas sequencialmente -->
    <sequence name="main">
       <receive name="Recebe_cocoon_pedido" partnerLink="cocoonPedido"
           portType="cocoonPedido:cocoonPedirCertidaoPT"
           operation="pedirCertidao" variable="cocoonPedInVariable"
           createInstance="yes"/>
                 .......
       <invoke name="Invoca_axis_pedir_certidao"
           partnerLink="axisPedido"
           portType="axisPedido:AxisPedirCertidao"
           operation="efectuarPedido"
           inputVariable="axisPedInVariable"
           outputVariable="axisPedOutVariable">
              <correlations>
                <correlation set="corrRef" initiate="yes" pattern="in"/>
                <correlation set="corrMb" initiate="yes" pattern="in"/>
              </correlations>
       </invoke>
                 .......
```

```
        <reply name="Responde_cocoon_pedido" operation="pedirCertidao"
            partnerLink="cocoonPedido"
            portType="cocoonPedido:cocoonPedirCertidaoPT"
            variable="cocoonPedOutVariable"/>
               ..............
    </sequence>
</process>
```

Fig. 6. Excerto de código BPEL que processa pedidos de certidões

Na figura 7 apresenta-se uma mensagem SOAP pedido entre o Cocoon e o motor BPEL da operação WSDL `pedirCertidao` no código BPEL anteriormente apresentado.

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <pedirCertidao
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                xmlns:xsd="http://www.w3.org/1999/XMLSchema"
                xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
                  <numeroUtilizador xsi:type="xsd:string">
                      1050001
                  </numeroUtilizador>
                  <tipo xsi:type="xsd:string">
                      Matricula
                  </tipo>
         </pedirCertidao>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fig. 7. Mensagem SOAP enviada pelo Cocoon para o motor BPEL

A figura 8 mostra um excerto do documento WSDL do serviço Web `cocoonPedido`, onde é definida a estrutura das mensagens SOAP de entrada e saída.

```
<wsdl:message name="cocoonPedidoIN">
   <wsdl:part name="numeroUtilizador" type="xsd:string"/>
   <wsdl:part name="tipo" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="cocoonPedidoOUT">
   <wsdl:part name="resposta" type="xsd:string"/>
   <wsdl:part name="respdesc" type="xsd:string"/>
   <wsdl:part name="ref" type="xsd:string"/>
   <wsdl:part name="mb" type="xsd:string"/>
</wsdl:message>
```

Fig. 8. Estrutura das mensagens SOAP de entrada e saída do serviço Web `cocoonPedido`

Por último refira-se que no contexto do sistema implementado, em que os produtores e consumidores de mensagens são únicos e estão muito bem definidos dentro da organização, não foi utilizada a descoberta de serviços Web recorrendo ao UDDI [25]. Estes poderão ser utilizados no contexto organizacional, para introduzir um outro nível de suporte à dinâmica de negócio.

4.2.2 Serviços Web

O Axis é uma plataforma, que possibilita o desenvolvimento de serviços Web. Na nossa implementação, os serviços Web são componentes do sistema de negócio. São eles que representam um determinado papel dentro da lógica de negócio e cujo objectivo é efectuar acções, tarefas, etc.

Na arquitectura da secretaria electrónica, o módulo Axis, disponibiliza estes quatro serviços:

- Pedido de certidão. Recebe o pedido, verifica se há condições e regista o pedido na base de dados XML.
- Pagar certidão. Altera o estado do pedido para pago. É usado apenas para certidões pagas ao balcão.
- Receber pagamento da entidade bancária (E.B.). Altera o estado do pedido para pago. É usado apenas para certidões que foram pagas através de entidades bancárias ou em caixas automáticas (ex: Multibanco).
- Emitir certidão. Altera o estado do pedido para emitido.

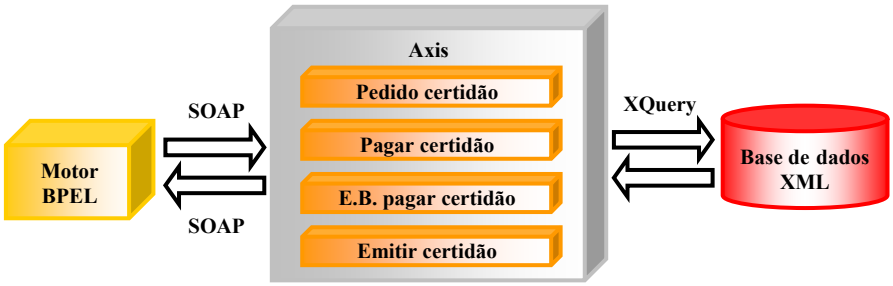


Fig. 9. Serviços Web

Na Fig. 9 estão representados os serviços Web implementados, bem como as interfaces ao motor BPEL e à base de dados XML. Note-se que as mensagens trocadas com motor BPEL, são obviamente mensagens SOAP, mas a interface à base de dados é feita pela execução de *querys* usando a linguagem XQuery.

4.3 Armazenamento de dados

A camada de dados foi implementada recorrendo à base de dados eXist [23]. O eXist é uma base de dados XML, que permite ser acedida por intermédio de interfaces como XML:DB, XML-RPC, SOAP e WebDAV.

As *querys* à base de dados são feitas na linguagem XQuery, quer pelo *generator* do Cocoon, quer pelas classes Java dos serviços Web. As actualizações e inserções são feitas, pelas classes Java, usando a linguagem XUpdate.

5 Conclusões

Foi proposto um modelo de uma plataforma de negócio baseada em XML, que permite integrar tecnologias orientadas ao serviço.

Foi feita uma implementação de um módulo de uma secretaria electrónica seguindo este modelo. A implementação permitiu-nos demonstrar a exequibilidade do mesmo e ser capaz de resolver um problema exemplificativo, o caso do pedido de certidões. Durante o processo de análise, em que a visão do problema foi amadurecendo, foi fácil introduzir alterações à lógica de processo de negócio, actuando na generalidade das situações na lógica de alto nível.

Este tipo de abordagem traz grandes vantagens a nível da reestruturação dos processos de negócio, quer a nível da flexibilidade em adaptar-se as variações dos requisitos, quer pela integração com os sistemas informáticos já existentes.

Por último, resta-nos referir que não foi objectivo deste trabalho averiguar a validade do modelo do ponto de vista da segurança e desempenho. Estes aspectos serão objectivos do trabalho futuro.

Referências

1. Web Services, Service Oriented Architecture (SOA).
http://webservices.xml.com.
2. Mark Colan, Service-Oriented Architecture expands the vision of Web services, 2004.
http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html.
3. IBM, Service-Oriented Architecture (SOA).
http://www-306.ibm.com/software/info/openenvironment/soa/.
4. World Wide Web Consortium, eXtensible Markup Language (XML) 1.0, 2004.
http://www.w3c.org/xml.
5. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, Sanjiva Weerawarana, Business Process Execution Language for Web Services Version 1.1, 2003.
ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf.
6. Dr. Michael Kay, Learn XQuery in 10 Minutes,2005.
http://www.stylusstudio.com/xquery_primer.html.
7. World Wide Web Consortium, XQuery 1.0: An XML Query Language, 2005.
http://www.w3.org/TR/xquery/.
8. Bob DuCharme, Getting Started with XQuery, Part 2, 2005.
http://www.xml.com/lpt/a/2005/03/23/xquery-2.html.
9. Chimezie Ogbuji, Editing XML Data Using XUpdate and HTML Forms (XUpdate), 2002.
http://www.xml.com/pub/a/2002/06/12/xupdate.html.
10. Andreas Laux, Lars Martin, XUpdate Syntax and Document Type Definition.
http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html.
11. Apache Cocoon Project, Cocoon.
http://cocoon.apache.org/.
12. Generating Web content with Cocoon.
http://www.webreference.com/xml/column52/.
13. Apache Web Services, Axis.
http://ws.apache.org/axis/.

14. Dennis Sosnoski, Apache Axis SOAP for Java, 2002.
http://www.sosnoski.com/presents/java-xml/axis/.
15. World Wide Web Consortium, HTTP - Hypertext Transfer Protocol.
http://www.w3.org/Protocols/.
16. Simple API for XML (SAX).
http://www.saxproject.org/.
17. Uche Ogbuji, Using SAX for Proper XML Output, 2003.
http://www.xml.com/pub/a/2003/03/12/py-xml.html.
18. Apache Cocoon Project, eXtensible Server Pages, XSP.
http://cocoon.apache.org/2.1/userdocs/xsp/logicsheet.html.
19. World Wide Web Consortium, XSL Transformations (XSLT) version 1.0, 1999.
http://www.w3.org/TR/xslt.
20. World Wide Web Consortium, HyperText Markup Language (HTML) Home Page.
http://www.w3.org/MarkUp/.
21. W3Schools, XML, SOAP, Web Services.
http://www.w3schools.com.
22. World Wide Web Consortium, XML Protocol Working Group (SOAP).
http://www.w3.org/2000/xp/Group/.
23. Open Source Native XML Database, eXist.
http://exist.sourceforge.net.
24. Apache Cocoon Project, Logicsheet Concepts.
http://cocoon.apache.org/2.1/userdocs/xsp/logicsheet-concepts.html.
25. Using BPEL4WS in a UDDI registry.
http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-bpel-20040725.htm.

Utilização de XML para Desenvolvimento Rápido de Analisadores Morfológicos Flexíveis

Bruno Oliveira[1], Carlos Pona[1], Ricardo Ribeiro[2], and David Martins de Matos[1]

[1] L[2]F/INESC-ID Lisboa/IST
[2] L[2]F/INESC-ID Lisboa/ISCTE
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{bfso,cmpo,rdmr,david}@inesc-id.pt
http://www.l2f.inesc-id.pt/

Resumo Descreve-se a utilização de um repositório lexical e de uma ferramenta de geração morfológica na produção rápida de ferramentas de análise morfológica. A ferramenta criada a partir dos dados XML, denominada XISPA (XML fInite-State-Powered Analyzer), utiliza a tecnologia de autómatos de estados finitos para a realização do processo de análise morfológica. O processo de criação do XISPA beneficiou da estrutura clara e bem definida dos dados produzidos pelo Monge. O uso de XML garantiu a independência relativamente ao processo de gestão dos dados linguísticos, permitindo um elevado nível de flexibilidade na decisão de construção de ferramentas seguindo este processo. Outro aspecto positivo a relevar é o curto intervalo de tempo necessário à produção de um protótipo (o XISPA foi completamente implementado em menos de um dia).

1 Introdução

O uso de ferramentas de processamento de língua natural é cada vez mais ubíquo, seja num simples corrector ortográfico, seja num sistema de diálogo. As aplicações existentes actualmente trabalham, cada uma, sobre um formato de dados específico, havendo a necessidade de introduzir ferramentas de conversão que facilitem a portabilidade da informação linguística. Coloca-se, assim, o problema da manutenção de colecções de dados adequados às diferentes necessidades daquelas aplicações. O Repositório Lexical do Laboratório de Sistemas de Língua Falada do INESC ID Lisboa foi criado para facilitar a gestão e manutenção de dados linguísticos, permitindo a representação de uma informação rica sobre vários níveis de descrição da língua. Este repositório foi definido para ser independente das aplicações que dele fazem uso. Este aspecto levanta, contudo, o problema da adaptabilidade dos dados em utilizações concretas: é necessária a definição de formas e linguagens de interoperação. A ferramenta Monge (Morphological Generator) surge neste contexto: esta ferramenta interage com o Repositório Lexical ao nível da produção e extracção de informação morfológica. O Monge extrai informação morfológica do Repositório e produz um documento XML. A representação em XML garante a independência de processamentos subsequentes e, devidamente validada, garante, em simultâneo, a coerência da informação resultante.

Neste documento descreve-se a utilização de um repositório lexical e de uma ferramenta de geração morfológica (Monge) na produção rápida de ferramentas de análise

morfológica (ferramentas que expressam o radical e conjunto de traços da forma de entrada). A ferramenta criada a partir dos dados XML, denominada XISPA (XML fInite-State-Powered Analyzer), utiliza a tecnologia de autómatos de estados finitos para a realização do processo de análise morfológica.

O documento tem a seguinte estrutura: primeiro apresenta-se o repositório lexical, fonte de dados para as ferramentas descritas; de seguida, descreve-se o gerador morfológico Monge, produtor da informação para a construção do XISPA. A descrição do XISPA, da sua construção, assim como conclusões acerca dos resultados obtidos e comparação com outras ferramentas semelhantes, terminam o texto.

2 Repositório Lexical

A área de processamento da língua apresenta frequentemente o problema bizarro de haver recursos para utilizar, mas não ser possível efectuar cabalmente esse uso. Tal acontece, ou porque os dados codificam informação que não corresponde exactamente às necessidades, ou porque, mesmo havendo essa correspondência, o formato dos dados não é o que seria útil. Estas variações conduzem a situações de incompatibilidade de dados entre ferramentas que realizam a mesma função e impede que os dados de uma possam ser reutilizados noutra, ou até mesmo a simples combinação desses dados para enriquecimento mútuo das diferentes aplicações.

O repositório de dados linguísticos multiusos do L^2F [1,2], é uma solução para o problema da compatibilização e reutilização de dados linguísticos: é capaz de armazenar dados pertencentes a diferentes paradigmas e expressos originalmente em diferentes formatos e cobrindo vários níveis (e.g., morfologia, sintaxe e semântica). Permite ainda expandir a cobertura ou a capacidade de descrição com impacto mínimo nos dados representados. O repositório está descrito em UML/XMI [3], sendo o código a ele associado gerado de forma inteiramente automática. A figura 1 mostra vários exemplos de dados.

Além da capacidade básica de armazenamento, o repositório funciona como ponte entre várias representações de dados, indo além da mera definição de um modelo canónico, definindo também transformações de dados entre vários modelos externos. Estes agentes de tradução constituem um factor positivo na consideração de utilização do repositório numa aplicação. As ferramentas especializadas, das quais as traduções são exemplos, podem ser utilizadas para preparar dados para uso por aplicações previamente existentes e que podem passar a utilizar dados enriquecidos; as aplicações podem ainda recorrer sem intermediários à representação nativa dos dados no repositório, fazendo uso directo dos dados nele armazenados. A figura 2 apresenta os vários grupos de modelos que coexistem para governar o repositório.

O repositório e o seu conteúdo contribuem, assim, tanto para o enriquecimento dos dados utilizados por aplicações existentes (enriquecendo, deste modo, as próprias aplicações), como para a possibilidade de reutilização de dados em aplicações que, de outra forma, não lhes teriam acesso.

PAROLE [4]

```
<mus id="r592" naming="algo" gramcat="adverb"
     autonomy="yes" synulist="usyn23987 usyn23988">
  <gmu range="0" reference="yes" inp="mfgr1">
    <spelling>algo</spelling>
  </gmu>
</mus>
<mus id="pi1" naming="algo" autonomy="yes"
     gramcat="pronoun" gramsubcat="indefinite"
     synulist="usyn23320">
  <gmu range="0" reference="yes" inp="mfgempty">
    <spelling>algo</spelling>
  </gmu>
</mus>
<ginp id="mfgr1" example="abaixo">
  <combmfcif combmf="combtm0">
    <cif stemind="0">
      <removal/><addedbefore/><addedafter/>
    </cif>
  </combmfcif>
</ginp>
<ginp id="mfgempty" comment="empty Mfg">
  <combmfcif combmf="combtmempty">
    <cif stemind="0">
      <removal/><addedbefore/><addedafter/>
    </cif>
  </combmfcif>
</ginp>
<combmf id="combtmempty"/>
<combmf id="combtm0" degree="positive"/>
```

SMorph [5]

```
algo            /pr_i/s/GEN:*/pri .
```

LUSOlex [6]

```
Adv191 <algo> ADVÉRBIO - FlAdv2 <algo>
Pi1 <algo> PRONOME INDEFINIDO - <algo>
FlAdv2  <abaixo>
        __P____ 0       <><>
$
```

EPLexIC [7]

```
algo/R=p/"al~gu/algo
algo/Pi=nn/"al~gu/algo
```

Figura 1. Diferenças entre léxicos na descrição de *algo*.

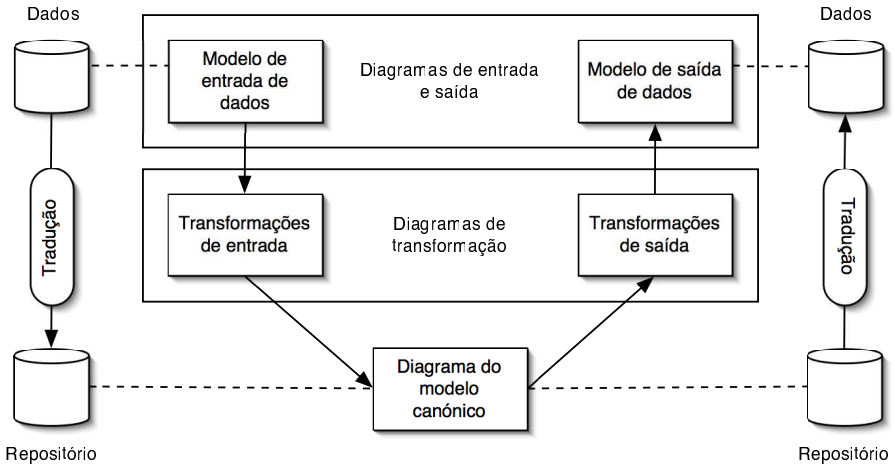


Figura 2. Modelos e fluxos de dados no repositório de recursos linguísticos.

3 Monge

A ferramenta Monge é um gerador morfológico que permite, com base numa raiz e num conjunto de características a ela associadas, obter uma forma (ver figura 3).

3.1 Relação com os dados

O Monge depende intrinsecamente do modelo de dados do Repositório Lexical. Não está, portanto, condicionado pela estrutura de uma base de dados particular. Pode, assim, funcionar com diferentes bases de dados, desde que estas sigam o modelo canónico. A utilização do Monge e da saída XML por ele produzida, como intermediária da representação constante na base de dados, faz com que as aplicações clientes dos dados sejam independentes do modelo do repositório.

Salienta-se também o facto de o Monge transformar os dados em formato relacional, sem estruturação explícita (como tal, de difícil utilização), num modelo estruturado (XML), em que estão patentes as definições subjacentes ao modelo de objectos representado. A utilização de XSD permite, simultaneamente, descrever e validar a estrutura dos dados de saída, facilitando o processo de exportação dos dados.

3.2 Modo de Funcionamento

O Monge recebe como entradas uma raiz e um conjunto opcional de categorizações e/ou traços – categoria, subcategoria, ou qualquer característica definida no dicionário de tipos. Com base nessa raiz, o Monge obtém todas as formas gráficas correspondentes, de modo a poder obter as bases de flexão regular da raiz: são estas bases de flexão, juntamente com a raiz que darão origem à forma flexionada. A forma gráfica da raiz

é também usada para recolher informação acerca da sua categoria e sub-categoria gramaticais. Por fim, o Monge selecciona os paradigmas de flexão apropriados para as formas em causa e aplica as transformações neles descritas. No final do processo, as palavras flexionadas a partir das respectivas raízes são agrupadas pelas suas categorias gramaticais. Posteriormente, esta informação é transformada numa árvore XML, em que às palavras são associados os traços morfológicos correspondentes. Para efeitos de validação e de organização da estrutura, um esquema XML (XSD) é associado ao documento produzido. Actualmente, o Monge está implementado na linguagem Perl e utiliza os módulos DBI, para acesso à base de dados; utiliza a biblioteca Xerces-p para representar e gerar informação XML.

```
$ ./monge.pl -x ser number singular gender masculine
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
 <monge:lemma
    xmlns:monge="http://www.l2f.inesc-id.pt/~bfso/monge.xsd"
    value="ser">
   <morph cat="NOUN" subcat="COMMON">
     <form value="ser">
       <feature given="yes" name="number" value="SINGULAR"/>
       <feature given="yes" name="gender" value="MASCULINE"/>
     </form>
   </morph>
   <morph cat="VERB" subcat="MAIN">
     <form value="sido">
       <feature given="yes" name="number" value="SINGULAR"/>
       <feature given="no" name="mood" value="PARTICIPLE"/>
       <feature given="yes" name="gender" value="MASCULINE"/>
     </form>
   </morph>
 </monge:lemma>
```

Figura 3. Exemplo de execução do Monge para a raiz *ser* sem restrições à categorização. Note-se a produção de todas as formas para todas as categorias e subcategorias.

4 XISPA

O XISPA é um analisador morfológico baseado em tecnologia de estados finitos. Dada uma palavra, o XISPA verifica se ela existe e, caso tal aconteça, devolve o conjunto de pares traço/valor que lhe está associado. Assim, dada a palavra *anestesiando*, o resultado devolvido pelo XISPA seria como indicado abaixo.

lemma	**cat**	**subcat**	**mood**
anestesiar	VERB	MAIN	GERUND

Para realizar o processo de análise, o XISPA usa a informação gerada pelo Monge, na forma de uma máquina de estados finitos. A ideia subjacente a esta realização é a de facilitar a construção deste tipo de ferramentas a partir de dicionários de larga cobertura.

Actualmente, as ferramentas de análise morfológica implementam esse processo através autómatos ou transdutores de estados finitos, sendo uma das referências fundamentais o trabalho de Koskenniemi [8,9]. Tal acontece porque o tempo de análise é $O(n)$ (n é o comprimento da palavra a analisar), independentemente do número de palavras do dicionário. A desvantagem a sublinhar é a quantidade de memória ocupada. Quanto ao tipo de informação usada pelos sistemas de estados finitos, a escolha entre uma abordagem baseada em paradigmas e uma baseada em regras morfológicas está relacionada com a língua que se pretende tratar: para o Português é comum a abordagem paradigmática.

Os sistemas mais comuns são sistemas especializados, i.e., seguem o processo de desenvolvimento tradicional e apresentam uma complexidade de desenvolvimento consideravelmente superior à que aqui se descreve. Acrescem a estas dificuldades, a manutenção, quer do código desenvolvido, quer dos dados usados, sendo especialmente difícil a adaptação a contextos variados. Como exemplos de analisadores baseados em estados finitos, podemos citar o sistema SMORPH [5] e o utilizado no Xerox Research Center Europe (XRCE)[3].

Mokhtar [5] descreve um sistema de análise morfológica baseado num autómato de estados finitos determinista denominado SMORPH, independente da língua. O ficheiro de informação necessária para efectuar a análise do Português tem cerca de 1 MByte, cobrindo cerca de 900 mil formas. No XRCE, o sistema de processamento morfológico (disponível para várias línguas) é baseado em transdutores de estados finitos. Apesar de não estarem disponíveis dados concretos sobre o sistema, o XRCE salienta a rapidez e a (reduzida) dimensão dos dados usados.

4.1 Construção

O processo de construção do XISPA pode ser observado na figura 4: os dados XML produzidos pelo Monge são processados por forma a especificar um autómato finito contendo todas as formas conhecidas para uma dada língua. Note-se que este autómato não contém em si mais do que a selecção do átomo (*token*) a apresentar (um número inteiro): a transposição da saída do autómato para o conjunto de características correspondentes à forma de entrada é realizado separadamente. Note-se que o autómato não codifica nenhum tipo de ambiguidade relativamente às formas reconhecidas: a ambiguidade é codificada nos átomos apresentados na saída, o que produz resultados equivalentes. O facto de o autómato não codificar ambiguidade significa, no entanto, que a sua utilização é mais simples.

O processo de construção do XISPA ignora ainda outros aspectos tradicionalmente relacionados com a análise morfológica, nomeadamente, a segmentação da entrada em átomos a analisar e a captura de elementos regulares (e.g., números). Estes aspectos não se reflectem, no entanto, negativamente, uma vez que é possível delegar estas tarefas

[3] `http://www.xrce.xerox.com`

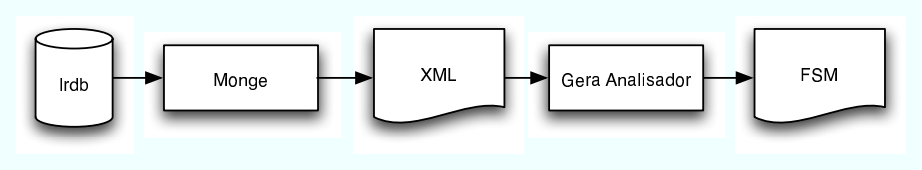


Figura 4. Processo de construção do XISPA (autómato finito).

noutras ferramentas e, assim, conseguir também distribuir as responsabilidades (ver figura 5).

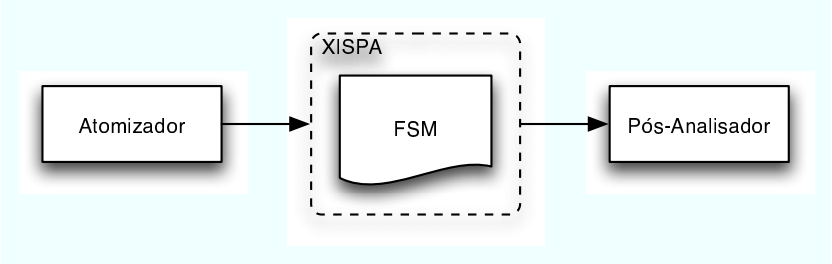


Figura 5. Processo de utilização do XISPA numa cadeia de análise morfológica: o atomizador segmenta a entrada em elementos a analisar; o módulo de pós-análise permite complementar a análise produzida.

Dada a simplicidade de construção, foram feitas várias implementações do XISPA, desde as mais ingénuas e limitadas (em dimensão, mas não funcionalidade), utilizando a ferramenta "flex",[4] até às "fsmtools" [10] da AT&T.[5]

Outras são possíveis, desde que suportem o conceito de autómato finito.

5 Conclusões

O processo de criação do XISPA beneficiou da estrutura clara e bem definida dos dados produzidos pelo Monge. O uso de XML garantiu a independência relativamente ao processo de gestão dos dados linguísticos, permitindo um elevado nível de flexibilidade na decisão de construção de ferramentas seguindo este processo. Outro aspecto positivo a relevar é o curto intervalo de tempo necessário à produção de um protótipo: o XISPA foi completamente concebido e implementado em menos de um dia.

A utilização da tecnologia de estados finitos garante tempos de análise semelhantes aos de ferramentas especializadas. O processo de construção flexibiliza a definição

[4] `http://www.gnu.org/software/flex/`
[5] `http://www.research.att.com/sw/tools/fsm/`

dos dados para a ferramenta, sendo um factor muito importante na consideração de utilizações em novos contextos.

Em §4.1 já se mencionaram algumas diferenças entre o XISPA e as ferramentas às quais pode ser equiparado. Um aspecto notório é a completa ausência de suporte ao tratamento de palavras desconhecidas pelo dicionário. Este aspecto, como já foi referido, pode ser delegado em ferramentas auxiliares especializadas, ficando a gestão do reconhecimento e classificação livre desta responsabilidade.

Como nota final, referimos a facilidade de aplicação do XISPA a outras línguas, uma característica que partilha com outras ferramentas baseadas em estados finitos.

Agradecimentos

Este trabalho foi parcialmente financiado pelo Projecto NLE-GRID (POSC/PLP/60663/2004).

Referências

1. de Matos, D.M., Ribeiro, R., Mamede, N.J.: Rethinking Reusable Resources. In: Proceedings of the Fourth Language Resources and Evaluation Conference – LREC 2004, Lisboa, Portugal, ELRA – European Language Resources Association (2004) 357–360 ISBN 2-9517408-1-6.
2. Ribeiro, R., de Matos, D.M., Mamede, N.J.: How to Integrate Data from Different Sources. In: A Registry of Linguistic Data Categories within an Integrated Language Resources Repository Area (LREC 2004), Lisboa, Portugal, ELRA – European Language Resources Association (2004)
3. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley Longman, Inc. (1999) ISBN 0-201-57168-4.
4. PAROLE: Preparatory Action for Linguistic Resources Organisation for Language Engineering – PAROLE (1998) O projecto teve início em Abril de 1996 e a duração de 24 meses. Ver `http://www.hltcentral.org/projects/detail.php?acronym=PAROLE` para um sumário do projecto.
5. Aït-Mokhtar, S.: L'analyse présyntaxique en une seule étape. Thèse de doctorat, Université Blaise Pascal, GRIL, Clermont-Ferrand (1998)
6. Wittmann, L., Ribeiro, R., Pêgo, T., Batista, F.: Some Language Resources and Tools for Computational Processing of Portuguese at INESC. In: Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000), Athens, Greece, ELRA – European Language Resources Association (2000) 347–350
7. de Oliveira, L.C.: EPLexIC – European Portuguese Pronunciation Lexicon of INESC-CLUL. documentation (n.d.)
8. Koskenniemi, K.: Two-level morphology: A general computational model for word-form recognition and production. Publications (11) (1983)
9. Koskenniemi, K.: Two-level model for morphological analysis. In Bundy, A., ed.: Proceedings of the Eighth International Joint Conference on Artificial Intelligence, EUA (1983) 683–685
10. Mohri, M.: Finite-State Transducers in Language and Speech Processing. Computational Linguistics **23**(2) (1997)

Geração de Formulários Web com base em *Templates InfoPath*

César Baptista, Nelson Branco, Luís Falcão, and Pedro Félix

Instituto Superior de Engenharia de Lisboa
Rua Conselheiro Emídio Navarro, 1, 1959-007 Lisboa, Portugal

Resumo A aplicação *InfoPath*, pertencente ao conjunto de ferramentas de escritório *Microsoft Office*, tem por função o desenho e preenchimento de formulários associados a documentos XML. Contudo, os cenários de utilização destes formulários são limitados porque a fase de preenchimento requer o uso da aplicação *InfoPath* e o acesso directo às fontes de dados.
O presente artigo descreve aspectos do desenho e implementação do sistema *WebInfoPath* (WIP), para a geração e suporte à execução de formulários web com base em *templates* gerados pela aplicação *InfoPath*.
Define-se uma arquitectura extensível para este objectivo e descrevem-se aspectos da sua implementação, nomeadamente: gerador de páginas e controlos ASP.NET; biblioteca de classes para suporte à execução; e forma de integração em aplicações web.

1 Introdução

A aplicação *InfoPath*, pertencente ao conjunto de ferramentas de escritório *Microsoft Office* [7], tem por função o desenho e preenchimento de formulários associados a documentos XML [18]. Nesse contexto, um formulário é definido por: o esquema de dados da informação, um conjunto de vistas gráficas que definem as formas de visualização e inserção da informação, e parâmetros de ligações a fontes de dados.

Na fase de desenho a aplicação *InfoPath* é usada na produção do *template* contendo a informação de definição do formulário. O *template* é constituído por um conjunto de documentos XML, donde se salientam o esquema de dados (documento XSD [16]) e as vistas (documentos XSL [14]). O desenho das vistas, e consequente produção dos documentos XSL, é realizado através duma ferramenta de edição gráfica. A definição do esquema de dados do formulário é realizada através dum editor gráfico, ou importada duma definição externa ou inferida duma fonte de dados.

Na fase de preenchimento, a aplicação *InfoPath* recebe um *template* e apresenta a interface gráfica, definida nas vistas, para a visualização e recolha da informação. Esta informação é posteriormente armazenada em ficheiros XML com o esquema do *template* ou submetida a fontes de dados externas.

A aplicação *InfoPath* simplifica o desenho de formulários associados a documentos XML, contudo apresenta um conjunto de limitações importantes:

- O preenchimento e visualização dos formulários requer o uso da aplicação *InfoPath*.
- As fontes de dados têm de estar acessíveis directamente ao computador onde o formulário é manipulado.
- O acesso a estas fontes de dados usa as credenciais do utilizador que edita o formulário.

Estes requisitos, plausíveis no contexto *intranet*, não se verificam em cenários de aplicação na *internet*, limitando o âmbito de utilização dos formulários *InfoPath*.

O sistema WIP tem por objectivo eliminar esta limitação, gerando de forma automática páginas ASP.NET [5] com base nos *templates* criados pelo *InfoPath*. Estas páginas são assim integráveis em aplicações web desenvolvidas sob esta plataforma. O sistema WIP usa o facto da informação do *template* estar definida em idiomas XML, nomeadamente em linguagens normalizadas como o XSD e o XSL.

Este artigo apresenta a arquitectura do sistema WIP, composta por: gerador de páginas e controlos ASP.NET; biblioteca de classes para suporte à execução; e forma de integração em aplicações web. Descreve-se o processo de geração de páginas e controlos gráficos, baseado na alteração dos documentos XSL com a definição das vistas. Apresenta-se a implementação da biblioteca de apoio à execução, enfatizando-se a componente de ligação às fontes de dados suportadas pelo *InfoPath* (sistemas de gestão de bases de dados, *web services* e ficheiros XML). Descreve-se a forma de integração dos formulários gerados em aplicações web.

Este artigo está organizado da seguinte forma: na secção 2 apresentam-se os aspectos da aplicação *InfoPath* necessários à compreensão do sistema WIP; na secção 3 define-se a arquitectura deste sistema; na secção 4 descreve-se o processo de geração de páginas ASP.NET; na secção 5 caracteriza-se a biblioteca de classes com os serviços de ligação de dados e de apresentação; na secção 6 apresenta-se a forma de integração dos *templates InfoPath* em aplicações web. Finalmente, na secção 7, estabelecem-se as conclusões deste trabalho.

2 InfoPath

A aplicação *InfoPath* é uma ferramenta de escritório, incluída no *Microsoft Office 2003*, para o desenho e preenchimento de formulários. A sua operação está dividida nas fases de *desenho* e de *preenchimento*. O *InfoPath* é usado na fase de desenho do formulário para: definir o esquema de dados; desenhar as páginas do formulário, designadas de vistas; e parametrizar as ligações às fontes de dados.

O esquema de dados é representado na linguagem XSD, existindo três formas de definição: importar o documento com o esquema definido na linguagem XSD (ficheiro XSD ou documento WSDL [15]); inferir o esquema duma base de dados ou instância XML; desenhar o esquema, usando o editor gráfico do *InfoPath*.

Cada vista representa uma página do formulário e é constituída por um conjunto de controlos gráficos. A definição da vista inclui a ligação entre estes controlos e elementos do esquema de dados. Cada vista é representada através dum

documento XSL. Estas transformações recebem como entrada o documento XML com os dados da instância do formulário e produzem o documento HTML [12] com a representação gráfica do formulário. Os elementos do documento HTML estão anotados, através de atributos, com informação adicional relacionada com o *InfoPath*. São exemplo destas anotações o atributo `binding` com a expressão *XPATH* [13] que define o valor do elemento e o atributo `xctname` com o tipo de controlo.

Na fase de preenchimento, o *InfoPath* suporta a escrita e leitura dos dados em ficheiros XML, usando o esquema de dados do formulário. Suporta também a ligação dos dados do formulários (obtenção e submissão) a outras fontes de dados: sistemas de gestão de bases de dados e *web services*.

O resultado da fase de desenho é um conjunto de ficheiros, designado por *template* com [6]:

- o manifesto contendo a constituição do *template*, representado num idioma proprietário;
- os esquemas de dados, representados em XSD;
- as vistas, definidas em XSL;
- os dados por omissão e de exemplo do esquema de dados definido;
- os parâmetros de ligação às fontes de dados.

Na fase de preenchimento, o *InfoPath* é usado para:

- criar, abrir e gravar instâncias de formulários;
- inserir, visualizar e editar a informação associada à instância do formulário;
- obter e submeter informação de e para a fonte de dados.

3 Arquitectura

A arquitectura do sistema WIP, ilustrada na figura 1, é constituída por:

- gerador de páginas ASP.NET com as vistas do formulário e controlos auxiliares;
- biblioteca de classes para suporte à execução das páginas;
- *handler* ASP.NET para o processamento de pedidos a *templates*.

O gerador de páginas recebe um *template* e produz um conjunto de páginas e *user controls* ASP.NET. Cada página gerada representa uma vista do formulário e é constituída por controlos ASP.NET, presentes na biblioteca de suporte, e código gerado *à medida*. Os *user controls* representam as linhas dos controlos de repetição.

A biblioteca de suporte à execução contém:

- controlos usados nas páginas geradas;
- a classe base para as páginas geradas;
- as classes com os serviços para o armazenamento persistente dos dados dos formulários;

– as classes para a ligação (obtenção e submissão) às fontes de dados.

A integração dos formulários *InfoPath* em aplicações Web é realizada através dum *handler* específico para este tipo de conteúdos. Os pedidos HTTP [11] para o *template* são direccionados para este *handler*, que realiza as seguintes acções:

– criação das páginas com as vistas do formulário requisitado;
– criação dos objectos com os serviços de suporte à execução;
– redireccionamento do pedido HTTP para as páginas geradas.

Desta forma, o *deployment* de formulários *InfoPath* em aplicações web resume-se à cópia do *template* para a pasta da aplicação.

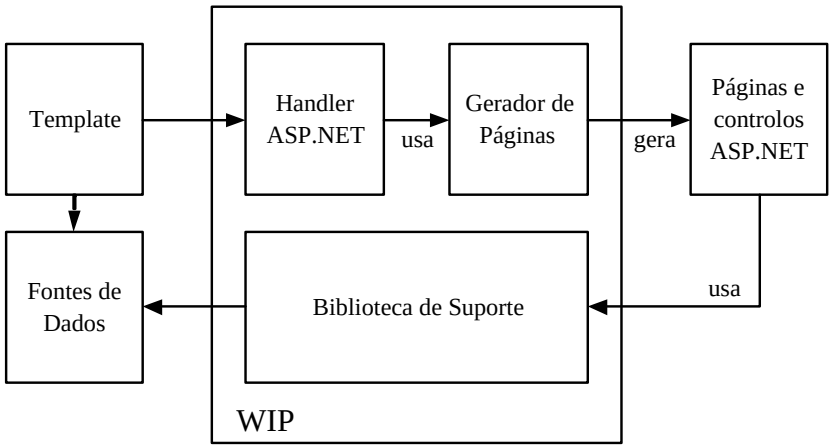


Figura 1. Arquitectura do sistema WIP.

4 Geração de formulários

Cada vista dum formulário *InfoPath* é representada por um documento XSL. Este documento descreve a transformação que produz a visualização HTML a partir do documento XML com os dados do formulário. É constituído por *templates* XSL associados a elementos do esquema de dados do formulário. Cada *template* define a visualização do elemento associado, sendo constituído por:

– Elementos literais HTML representando partes da vista, adicionados de atributos específicos do *InfoPath*.
– Elementos XSL com a selecção da informação do formulário a colocar nos elementos HTML.

O sistema WIP usa este documento XSL para gerar a página ASP.NET com a vista, através do processo descrito em seguida e ilustrado na figura 2.

Seja `view.xsl` o documento com a transformação associada à vista `view` e seja `view.aspx` a página que se pretende gerar. O processo de geração de `view.aspx` está divido em duas fases.

Na primeira fase é gerado o documento XSL `view.aspx.xsl` com a transformação para produzir a página `view.aspx` a partir dum documento XML com os dados do formulário. Esta nova transformação é baseada no documento `view.xsl`.

Na segunda fase, é usada o documento `view.aspx.xsl` para gerar o ficheiro `view.aspx` com base nos dados por omissão do formulário (`sampledata.xml`).

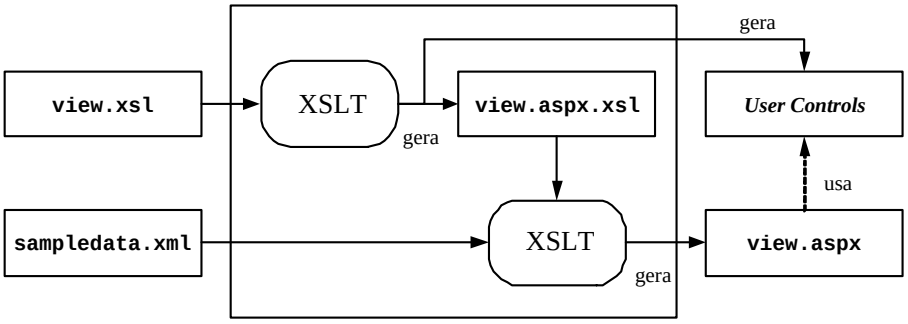


Figura 2. Geração de páginas e *user controls*.

O documento `view.aspx.xsl` é criado através da cópia de `view.xsl` com as seguintes alterações:

- É inserido texto literal com o cabeçalho da página ASP.NET (`<@Page ...>`).
- É inserido no elemento *<head>* o texto literal com o código C# específico para esta vista. Este código contém a redefinição de métodos abstractos da classe base das páginas e estabelece a ligação entre os controlos da página e o serviço de ligação às fontes de dados.
- É inserido o elemento *<form>* no elemento *<body>*.
- Transformam-se todos os elementos HTML correspondentes a controlos *InfoPath* em elementos representando controlos ASP.NET. A relação entre os elementos HTML e os controlos ASP.NET é definida por um ficheiro de configuração. A biblioteca de suporte do WIP possui controlos para um subconjunto dos controlos *InfoPath*.
  Os elementos com os controlos ASP.NET incluem os atributos de estilo dos elementos HTML, usados no HTML produzido na acção de *render*, e atributos adicionais usados na execução do controlo.

Neste processo é também criado um *user control* ASP.NET por cada elemento dum controlo de repetição. Este processo é idêntico ao descrito anteriormente,

com a diferença que o resultado é o próprio *user control* e não um documento XSL. Estes *user controls* são usados na criação, em tempo de execução, de elementos dos controlos de repetição.

Salienta-se que a transformação de `view.xsl` em `view.aspx.xsl` é também realizada através duma transformação XSLT.

5 Biblioteca de suporte

O WIP inclui uma biblioteca de suporte à execução das páginas ASP.NET geradas, que está dividida em dois subsistemas: **Ligação de Dados** e **WIP Controls**. Cada subsistema disponibiliza um conjunto de serviços.

O subsistema de **Ligação de Dados** inclui o Serviços de Persistência de Instâncias (SPI) e o Serviço de Ligação de Dados (SLD). O subsistema de **WIP Controls** inclui dois tipos de componentes gráficos (controlos ASP.NET): classes base das páginas e dos *user controls* criados na fase de geração; e uma hierarquia extensível de controlos gráficos, designados por Controlos WIP, que constituem as entidades existentes nos formulários web.

A biblioteca inclui implementações de referência para todos os aspectos dos subsistemas. No entanto, a sua arquitectura foi desenhada de modo a que seja extensível. Cada subsistema apresenta mecanismos próprios de extensibilidade, que serão referidos nas secções seguintes.

5.1 Ligação de Dados

No WIP, tal como no *InfoPath*, os controlos gráficos estão ligados a dados da instância corrente de formulário. Uma instância de formulário é um documento XML, instância do esquema de dados associado ao formulário.

A manipulação de formulários envolve quatro operações:

1. carregamento de dados (por omissão ou de uma instância seleccionada);
2. gravação de uma instância com os dados presentes no formulário;
3. pesquisa na fonte de dados;
4. submissão para a fonte de dados;

As operações 1 e 2 manipulam instâncias locais (tipicamente residentes no sistema de ficheiros). As operações 3 e 4 manipulam instâncias da fonte de dados principal.

Para suportar estas operações, o subsistema de **Ligação de Dados** inclui dois tipos de funcionalidades: acesso a dados locais, que suporta o carregamento e persistência de instâncias de formulários; acesso à fonte de dados principal (e secundárias, caso existam), que inclui mecanismos de interrogação e submissão de dados. Ambas as funcionalidades são disponibilizadas pelo SLD.

O carregamento e persistência de instâncias num repositório local é implementado num serviço autónomo, o SPI. Este é incluído pelo SLD que lhe delega a implementação das respectivas funcionalidades. Esta separação foi adoptada de modo a criar mais um ponto de extensibilidade do subsistema. Desta forma,

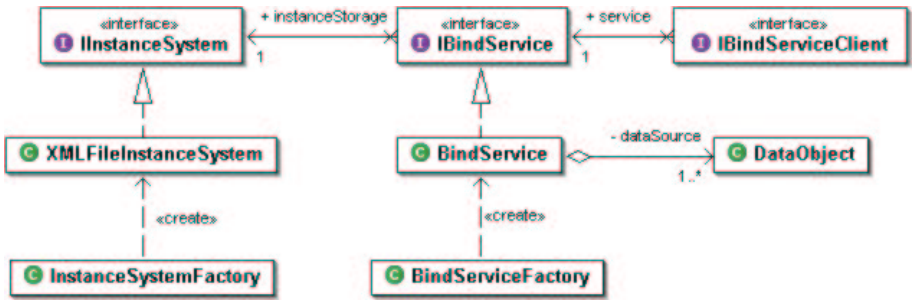


Figura 3. Diagrama de classes do subsistema de **Ligação de Dados**.

é possível mudar a implementação da persistência de instâncias sem alterar todo o mecanismo de ligação de dados.

Cada um dos serviços é representado por uma interface, `IBindService` para o SLD e `IInstanceSystem` para o SPI, conforme se ilustra na figura 3. A biblioteca inclui implementações de referência para cada um dos serviços, nas classes `BindService` e `XMLFileInstanceSystem`, respectivamente.

As instâncias dos serviços deste subsistema são obtidas através das fábricas: `InstanceSystemFactory` para instâncias de `IInstanceSystem` de e `BindServiceFactory` para instâncias de `IBindService`. Os tipos concretos criados pelas fábricas podem ser definidos no ficheiro de configuração da aplicação web. Em caso de omissão são criadas instâncias dos tipos referência.

Para que os controlos WIP possam ser associados aos dados de uma instância de formulário têm que implementar a interface `IBindServiceClient`. Através desta interface, o SLD sincroniza (em ambos os sentidos) os dados com os controlos. Os dados são transferidos entre os controlos e o serviço através do documento XML que contém os dados da instância corrente do formulário.

Serviço de Persistência de Instâncias - SPI

A classe `XMLFileInstanceSystem` é a implementação de referência do SPI. Esta carrega e guarda instâncias de formulários em ficheiros XML no sistema de ficheiros local.

Serviço de Ligação de Dados - SLD

O SLD é a entidade pública deste subsistema. Realiza a ligação entre os controlos gráficos WIP e a instância corrente de formulário. Suporta as quatro operações de manipulação dos formulários referidas anteriormente.

A implementação de referência deste serviço é realizada na classe `BindService`. Delega ao SPI a manipulação local de instâncias de formulários. Acede às fontes de dados principal e secundárias (caso existam) através de objectos de

dados (`DataObject`). Estes ligam-se através de adaptadores específicos para cada tipo de fonte de dados. Um `DataObject` suporta as funcionalidades de pesquisa, submissão, ou apenas uma destas, dependendo do tipo de adaptadores com que é construído. Um adaptador tem um papel semelhante aos adaptadores existentes no ADO.NET [10].

Este subsistema inclui implementações referência de adaptadores para as seguintes fontes de dados: ficheiros XML; sistemas gestores de bases de dados com *drivers* ADO; e *web wervices*.

Os objectos associados a este serviço são criados através de fábricas. A biblioteca inclui uma fábrica de referência (`BindServiceFactory`) que cria instâncias de `BindService` para a implementação do SLD e cria instâncias do SPI através da respectiva fábrica. Por omissão, cria instâncias dos adaptadores de referência para cada um dos tipos de fonte de dados suportados. No entanto, os tipos que implementam os adaptadores podem ser definidos no ficheiro de configuração da aplicação web.

De modo a suportar outro tipo de fontes de dados e/ou outra implementação deste serviço, a fábrica de referência pode ser substituída, definindo o tipo que a implementa no ficheiro de configuração da aplicação web.

5.2 WIP Controls

O subsistema de **WIP Controls** inclui controlos ASP.NET que dão suporte à execução das páginas e *user controls* criados na fase de geração.

Páginas

A biblioteca inclui a classe `WIPPage` que extende `System.Web.UI.Page` e é a base de todas as páginas geradas no WIP. Inclui funcionalidades e recursos comuns a todas as páginas no WIP, nomeadamente:

- botões para realizar as operações de manipulação de formulários (carregar, gravar, pesquisar e submeter);
- disponibilização da instância do Serviço de Ligação de Dados (5.1);
- iniciação e libertação de recursos;

Linhas de Controlos de Repetição

A biblioteca inclui controlos ASP.NET que representam os controlos de repetição do *InfoPath*. Estes, recebem como parâmetros de construção o SLD e os controlos que representam uma linha, na forma de um *user control*, criado na fase de geração de páginas. A geração de uma nova linha corresponde à criação duma instância desse *user control* e à sua adição à lista de controlos filhos. Como os *user controls* que representam linhas partilham funcionalidades e recursos comuns, é incluída a classe `WIPUserControl`, derivada de `System.Web.UI.UserControl`, que é a base de todos os *user controls* gerados no WIP.

Web Controls **WIP**

As páginas WIP são compostas por controlos WIP. Estes constituem-se numa hierarquia de `web controls` ASP.NET que têm como base `WIPControl`, que estende `System.Web.UI.Control` e que implementa a interface de cliente imposta pelo SLD (`IBindServiceClient`). A classe `WIPEditableControl` é a base dos controlos editáveis (que não são de repetição), dando suporte às suas funcionalidades comuns. Na fase de geração de páginas, os elementos HTML das vistas *InfoPath* são transformados em controlos correspondentes existentes nesta hierarquia. A hierarquia de controlos WIP é extensível, incluindo a biblioteca implementações referência para os seguintes controlos: `TextBox`, `ListBox`, `Calendar`, `RepeatingTable` e `ExpressionBox`.

6 Interface do WIP

A interface de acesso aos formulários WIP é realizada através de *Uniform Resource Locators* (URLs) [4]. O URL de acesso a um formulário inclui o nome do *template* gerado pelo *InfoPath*. A vista e instância de formulário são definidos, respectivamente, pelos parâmetros "`view`" e "`instance`" na *query string* do URL. O exemplo seguinte representa o URL de acesso à vista "`View1`" do formulário "`Form1.xsn`", com os dados da instância "`Instance1`", incluída na aplicação "WIPApp" residente no servidor "`someServer`":

```
http://someServer/WipApp/Form1.xsn?view=View1&instance=Instance1
```

Caso os parâmetros `view` e `instance` não constem no URL, é apresentada a vista por omissão com os dados por omissão.

6.1 Integração do WIP em aplicações Web

O processamento de pedidos para formulários WIP é realizada por um *handler* HTTP ASP.NET [9] que processa os pedidos para extensões `xsn`. A inclusão da infraestrutura WIP numa aplicação Web tem como único requisito a configuração deste *handler*.

O processo de publicação (*deploy*) de formulários consiste na cópia dos *templates* gerados pelo *InfoPath* para uma directoria da aplicação Web (configurada com o *handler* WIP).

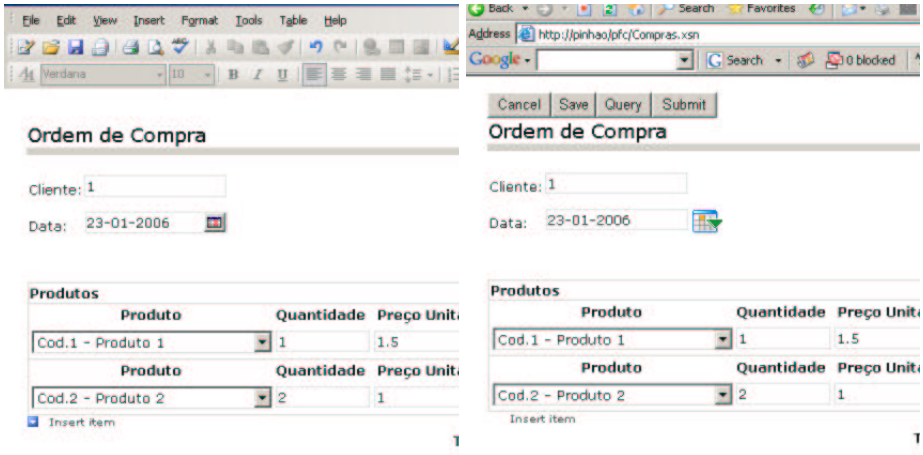
O *handler* é um módulo de software que consulta o *template InfoPath* identificado pelo URL e redirecciona o pedido para a página ASP.NET correspondente à vista pedida. O *handler*, no primeiro acesso a um *template*, cria uma estrutura de directorias (dentro de uma directoria com o nome do *template*), onde coloca as páginas ASP.NET geradas, correspondentes às vistas, e as instâncias de formulários gravadas pelos utilizadores. O processo de geração da página ASP.NET correspondente a uma vista, é realizado pelo *handler* no primeiro acesso a essa vista, antes de redireccionar o pedido.

Quando o *handler* detecta alterações no *template* de um formulário, invalida todas as páginas geradas correspondentes às vistas que lhe estão associadas, provocando uma nova geração destas nos acessos subsequentes.

7 Conclusões

Apresentaram-se neste artigo aspectos de desenho e implementação do sistema WIP, para a geração automática de formulários web baseados em *templates InfoPath*.

A figura 4 ilustra a utilização deste sistema, apresentando um formulário original, visualizado na aplicação *InfoPath*, e o formulário web gerado pelo sistema WIP, visualizado no *browser*.



a) b)

Figura 4. Exemplos de formulários: a) aplicação *InfoPath* ; b) *browser*

O sistema WIP inclui um gerador de páginas ASP.NET a partir das vistas dos formulários, uma biblioteca de classes para suportar execução destas páginas e um *handler* de atendimento de pedidos HTTP para ficheiros com os *templates* dos formulário (extensão `.xsn`).

A arquitectura do sistema WIP não está comprometida com a plataforma ASP.NET. É possível implementar este sistema noutras plataformas de suporte a aplicações web, desde que possuam as seguintes capacidades:

- Definição de *handlers* específicos para pedidos HTTP;
- Definição de controlos de servidor, preferencialmente num idioma XML;
- Mapeamento entre o modelo XML e o modelo relacional;
- Construção e processamento de mensagens SOAP [17];
- Seriação XML.

Destacam-se os seguintes aspectos do sistema WIP:

- São geradas páginas e código específico para cada vista do formulário.

- A tradução de controlos *InfoPath* em controlos ASP.NET é parametrizável pelo ficheiro de configuração na fase de geração. Esta solução possibilita a adequação dos controlos a cenários específicos, como sejam a geração de páginas para apresentação em dispositivos móveis, onde as capacidades gráficas são reduzidas.
- A componente de ligação às fontes de dados e de armazenamento persistente é extensível com novas classes fornecedoras destes serviços. É assim possível ligar a informação do formulário a qualquer fonte com interface de documento XML.
- O *deployment* de formulários requer apenas o registo do *handler* na plataforma ASP.NET e a cópia do ficheiro com o *template* para uma pasta da aplicação web.

As principais limitações deste trabalho são:

- Não suporta a associação de código *script* ao formulário. A implementação desta capacidade implica implementar todo o modelo de dados do *InfoPath* bem como interpretadores para as linguagens desejadas.
- As vistas geradas pelo *InfoPath* incluem mecanismos de formatação gráfica não normalizados, nomeadamente atributos de estilo, resultando em problemas de visualização em *browsers* diferentes do *Internet Explorer*.
- Não suporta todos os controlos do *InfoPath* e não suporta ligações a bases de dados como fontes secundárias. Prevê-se a resolução destas limitações em trabalho futuro.

O sistema WIP, ao definir um intermediário entre as páginas dos formulários e as fontes de dados, cria as condições para a realização de extensões à funcionalidade do *InfoPath*. Perspectivam-se, enquanto trabalho futuro, duas desses adições.

A primeira, consiste na adição de suporte para especificações de segurança dos *web services*, nomeadamente [8], [1] e [3]. São exemplos deste suporte a utilização da política dos *web services* no processamento das mensagens ou a utilização de *username tokens* [8] baseados nas credencias de autenticação do cliente na aplicação web.

A segunda adição perspectivada consiste no controlo do acesso às fontes de dados, realizado no serviço de ligação, usando modelos de autorização específicos para documentos XML [2].

O trabalho descrito neste artigo ilustra a utilização das tecnologias XML na geração automática de interfaces com o utilizador e na ligação destas a fontes de dados.

Referências

1. S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M. Hondo, C. Kaler, D. Langworthy, A. Malhotra, A. Nadalin, N. Nagaratnam, M. Nottingham, H. Prafullchandra, C. von Riegen, J. Schlimmer, C. Sharp, and J. Shewchuk. *Web Services Policy Framework*, Setembro 2004.

2. Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. A fine-grained access control system for xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002.
3. G. Della-Libera, M. Gudgin, P. Hallam-Baker, M. Hondo, H. Granqvist, C. Kaler, H. Maruyama, M. McIntosh, A. Nadalin, N. Nagaratnam, R. Philpott, H. Prafullchandra, J. Shewchuk, D. Walter, and R. Zolfonoon. *Web Services Security Policy Language*, Julho 2005.
4. L. Masinter, M. McCahill, and K. Raeburn. *RFC 1738: Uniform Resource Locators (URL).* Internet Engineering Task Force, Dezembro 1994.
5. Microsoft. Asp.net development center. http://msdn.microsoft.com/asp.net/.
6. Microsoft. Infopath developer portal. http://msdn.microsoft.com/office/understanding/infopath/default.aspx.
7. Microsoft. Microsoft office. http://office.microsoft.com/pt-pt/default.aspx.
8. OASIS. *Web Services Security: SOAP Message Security 1.0*, Março 2004.
9. Fritz Onion. *Essential ASP.NET with Examples in C#*, chapter 4. Addison Wesley, 2003.
10. Shawn Wildermuth. *Pragmatic ADO.NET: Data Access for the Internet World*, chapter 5. Addison Wesley, 2002.
11. World Wide Web Consortium. *Hypertext Transfer Protocol (HTTP).*
12. World Wide Web Consortium. *HyperText Markup Language HTML 4.01 Specification*, Dezembro 1999.
13. World Wide Web Consortium. *XML Path Language (XPath) Version 1.0 (Recommendation)*, Novembro 1999.
14. World Wide Web Consortium. *Extensible Stylesheet Language (XSL) Version 1.0 (Recommendation)*, Outubro 2001.
15. World Wide Web Consortium. *Web Services Description Language (WSDL) 1.1*, Março 2001.
16. World Wide Web Consortium. *XML Schema Language (Recommendation)*, Maio 2001.
17. World Wide Web Consortium. *SOAP Version 1.2 (Recommendation)*, Junho 2003.
18. World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Third Edition)(Recommendation)*, Fevereiro 2004.

Geração dinâmica de APIs Perl para criação de XML

José João Almeida and Alberto Manuel Simões

Departamento de Informática
Universidade do Minho
{jj|ambs}@di.uminho.pt

Resumo É consensual que o XML como linguagem para a estruturação de documentos tem vindo a tomar um lugar relevante. É também evidente a vantagem obtida no uso de XML como linguagem de intercâmbio. No entanto, a sua sintaxe é demasiado descritiva pelo que a geração de documentos de forma manual é dolorosa sendo útil dispor de módulos que simplifiquem essa tarefa.
Neste artigo propomos um módulo Perl (XML::Writer::Simple) configurável via DTD que simplifica a tarefa de gerar XML.

1 Introdução

Antes de iniciar a apresentação do módulo Perl XML::Writer::Simple, vamos apresentar as vantagens e possibilidades de usar uma API para a geração de documentos XML, analisando um módulo que gera documentos XHTML.

1.1 Vantagens do uso de uma API de geração

Desde que o Perl tem vindo a ser usado como uma das principais ferramentas para a geração de páginas Web, bem como de formulários e *sites* dinâmicos, que existe um módulo Perl denominado CGI[4]. Este módulo, para além de conter uma grande panóplia de funções para a manutenção de formulários e parâmetros de cgis, contém um conjunto de funções úteis para a geração de XHTML[5].

Basicamente, para cada elemento válido em XHTML passa a existir uma função no ambiente de programação Perl, pelo que a simples invocação da função p com uma string como argumento, gera o XHTML do respectivo parágrafo.

A tabela 1.1 mostra alguns exemplos de como as funções do módulo CGI são usadas, e qual o resultado prático em XHTML.

Desta tabela salienta-se a facilidade com que se criam elementos mistos (segunda e terceira linha da tabela), bem como o suporte natural para atributos (quarta linha). Por fim, é demonstrada a possibilidade de uso de mapeamentos de elementos a listas.

As vantagens do uso de funções para a construção de documentos XHTML são várias:

Código CGI	Código XHTML
`p("bar")`	`<p>bar</p>`
`p("foo", "bar")`	`<p>foo bar</p>`
`p("foo", b("zbr"), "bar")`	`<p>foo <b>zbr</b> bar</p>`
`a({href=>"http://www.sapo.pt"}, "sapo")`	`<a href="http://www.sapo.pt">sapo</a>`
`ul(li(["a","b","c"]))`	`<ul><li>a</li><li>b</li><li>c</li></ul>`

Tabela 1. Utilização básica do CGI

– **escrita compacta**

Como sabemos o XHTML por vezes é demasiado descritivo, o que torna o processo de escrita demorado e moroso. Uma das principais vantagens da sintaxe do módulo CGI é o facto de não nos termos de preocupar com o nome do elemento que estamos a fechar (e portanto, garantimos[1] que o XML é bem formado)
Considere-se o seguinte exemplo:

```
1     <ul>
2       <li><b>Bold</b></li>
3       <li><i>Italic</i></li>
4       <li><strong>Strong</strong></li>
5       <li><code>Code</code></li>
6     </ul>
```

que seria gerado facilmente com:

```
1     ul(li([
2         b("Bold"),
3         i("Italic"),
4         strong("Strong"),
5         code("Code")
6           ]))
```

É fácil notar que a versão funcional é mais compacta e legível.

– **possibilidade de definição de macros pelo utilizador**

Dada a fácil integração do módulo com a linguagem de programação, torna-se simples para o utilizador a definição de novas funções, ou *macros* para a geração de código.
Consideremos um exemplo típico, de geração de hiper-ligações cujo texto ligado é o próprio valor da ligação. A escrita habitual em CGI deveria ser:

```
1     a({href=>"http://www.sapo.pt/"}, "http://www.sapo.pt/");
```

No caso de este tipo de ligações ser comum no documento ou aplicação que está a ser desenvolvida, o utilizador pode definir uma macro como:

[1] Na verdade o utilizador pode sempre forçar um XML mal formado mas se as funções forem utilizadas da forma correcta a boa formação do documento é garantida.

```
1   sub alink { a({href=>$_[0]}, $_[0]) }

2   alink("http://www.sapo.pt/");
```

Da mesma forma, se ao desenvolver um determinado *site* se está a usar uma tabela para mostrar o conteúdo de uma notícia, torna-se simples definir uma macro que abrevie todo o trabalho:

```
1  sub new_entry {
2    table(Tr(th(class=>"new",$_[0])), Tr(td($_[1]])))
3  }

4  new_entry("500 mortos nas estradas", "No último ano...");
```

Repare-se também que se a determinado ponto o webmaster decidir mudar o XHTML gerado de forma a tornar mais acessível a página web (deixando de usar tabelas para design) bastará para isso alterar o conteúdo desta função.

– **ligação a uma Linguagem de Programação**

Pelo facto de dispormos de uma linguagem de programação, passa a ser possível calcular dinamicamente partes constituintes do documento XHTML com base em funcionalidade, bases de dados, bases de conhecimento, etc. a que a linguagem de programação possa aceder.
Como o módulo está desenhado para uma linguagem de programação específica, e as suas estruturas de dados, torna-se simples o uso das mesmas para a geração rápida de XHTML:

```
1   @amigos=("Rui", "Ana", "Eva");

2   print ul(li([@amigos]))
```

Por outro lado, temos toda a liberdade de uso da linguagem de programação em causa. Supondo que a base de dados `dicDeLinks` contém urls ligados a uma determinada palavra-chave (por exemplo ligada através de uma `DB_File`) considere-se o seguinte extracto:

```
1   sub keyword {
2     my $k=shift;
3     a({href=> $dicDeLinks{$k}}, b($k)) }

4   print p(keyword("xata"));
```

Notas:

linha 3: a base de dados `dicDeLinks` contém urls ligados a uma determinada keyword.

linha 4: imprime `<p><a href="http://www.xata.org"><b>xata</b></a></p>`

2 XML::Writer::Simple

O módulo que aqui apresentamos — XML::Writer::Simple — pretende facilitar a geração de XML ligado a um ambiente específico. Existem outros módulos com este objectivo como o XML::Writer[3] mas que acabam por não facilitar nem se adaptar ao DTD em uso.

O módulo CGI baseia-se num conjunto fechado de etiquetas pelo que a sua definição poderia ter sido feita criando uma função para cada uma das etiquetas existentes. No entanto, ao estarmos a manusear XML é imprescindível que estas funções sejam geradas em tempo real.

Assim, o utilizador do módulo XML::Writer::Simple deve indicar um DTD[2] sobre o qual vai trabalhar. O módulo, ao ser carregado, interpreta o DTD[1] e extrai o conjunto de etiquetas válidas assim como outras propriedades.

2.1 XML::Writer::Simple a partir dum DTD

Vamos supor que um determinado DTD contém as seguintes definições:

```
1 <!ELEMENT     a   (b | c )                  >
2 <!ELEMENT     b   (c | #PCDATA)             >
```

A análise do referido DTD por parte do XML::Writer::Simple cria uma conjunto de funções (a,b,c) que, quando usadas de acordo com o exemplo seguinte:

```
1 use XML::Writer::Simple dtd => "ex.dtd";
2 print a(b("p1",c("p2"),"p3"));
```

geram o seguinte XML

```
1 <a>
2    <b>p1<c>p2</c>p3</b>
3 </a>
```

Além da definição destas funções, o módulo também irá criar um conjunto de PowerTags, de que iremos falar na secção 2.3.

2.2 XML::Writer::Simple a partir dum exemplo XML

A utilização de um exemplo XML para a definição das funções de geração de XML passa pela inferência do respectivo DTD[1].

Deste modo, o módulo pode ser usado com:

```
1 use XML::Writter::Simple xml => "ex.xml";
2
2 print a(b("p1",c("p2"),"p3"));
```

[2] Na verdade, como veremos mais à frente, o utilizador poderá usar um XML exemplo em vez de um DTD, ou ainda indicar a lista de elementos válidos.

2.3 PowerTags

As PowerTags são funções que representam mais do que uma etiqueta. É habitual em XHTML e, genericamente, em XML, existir etiquetas que só podem ter um tipo de filhos. Em XHTML são exemplos deste tipo de etiquetas as `ul` e `ol` que só podem conter `li`. Este tipo de propriedades podem ser inferidas a partir do DTD de forma a criar funções que automatizem a criação deste tipo de estruturas.

A tabela 2.3 mostra como estas funções podem abreviar a escrita. No exemplo em causa usamos as etiquetas `ul` e `li` do XHTML apenas como exemplo.

Código XML::Writer::Simple	**Código XML**
`ul_li("a","b","c")`	`<ul><li>a</li><li>b</li><li>c</li></ul>`
`ul_li({attr=>"val"},"a","b")`	`<ul attr="val"><li>a</li><li>b</li></ul>`

Tabela 2. Utilização de PowerTags

Em certos casos estas PowerTags não podem ser inferidas a partir do DTD porque, na realidade, não são correctas. Por exemplo, os `tr` do XHTML permitem mais do que um tipo de filhos (`td`, `th`). Dada a utilidade real de uma PowerTag para compor `tr` com `td` (`tr_td`), existe um método do módulo para forçar a sua criação:

```
1  powertag("tr","td");

2  tr_td("a","b","c");
```

Este tipo de raciocínio pode ser generalizado definindo PowerTags com mais do que um nível. Por exemplo, para a definição directa de tabelas usando dois níveis de listas aninhadas podemos usar:

```
1  use XML::Writer::Simple;

2  powertag("table","tr","td");

3  table_tr_td( ["a","b","c"],["d","e","f"] );
```

Com o resultado de

a	b	c
d	e	f

```
1  <table>
2    <tr> <td>a</td><td>b</td><td>c</td> </tr>
3    <tr> <td>d</td><td>e</td><td>f</td> </tr>
4  </table>
```

O mesmo resultado poderia ser obtido usando o comando `powertags` directamente na importação do módulo, como se segue:

```
1  use XML::Writer::Simple powertags=>[qw/table_tr_td/];

2  table_tr_td( ["a","b","c"],["d","e","f"] );
```

2.4 Exemplo: árvore de directorias em XML

O seguinte exemplo académico demonstra o uso do módulo XML::Writer::Simple para a construção de uma árvore de directorias em XML, usando a etiqueta `file` para anotar ficheiros, e a etiqueta `dir` para delimitar os ficheiros contidos nessa directoria (usando o atributo `name` para armazenar o nome da directoria em causa):

```
1  #!/usr/bin/perl -w
2  use XML::Writter::Simple tags => [qw/dir file/];
3  my $dir = shift;
4  print process($dir);
5  sub process {
6    my $dir = shift;
7    my $xml;
8    chdir $dir;
9    for my $file (<*>) {
10     if (-d $file) { $xml .= process($file) }
11     else          { $xml .= file($file)    }
12   }
13   chdir "..";
14   return dir(name=>$dir,$xml)
15 }
```

Notas:

linha 2: carregar o módulo indicando-lhe as etiquetas válidas;
linha 4: processar a directoria base;
linha 8: mudar para a directoria a processar;
linha 9: para cada ficheiro existente...
linha 10: no caso de ser directoria, processar recursivamente;
linha 11: no caso de ser ficheiro, criar etiqueta respectiva;
linha 14: criar a etiqueta referente à directoria;

O output gerado é semelhante a:

```
1  <dir name="etc">
2     <file>passwd</file>
3     <file>group</file>
4     <dir name="httpd">
5        <dir name="conf">
6           <file>httpd.conf</file>
7        </dir>
8     </dir>
9  </dir>
```

3 Implementação

O uso de uma linguagem como o Perl para a implementação deste módulo facilita em grande parte a tarefa. Dada que a linguagem é reflexiva, torna-se possível definir funções em tempo de execução.

Por outro lado, dada a forma como internamente as invocações de funções de um módulo Perl são realizadas, é possível a escrita de uma função de ordem superior[2] que trate de as interceptar (AUTOLOAD). Assim, deixa de ser necessária a definição de um método por etiqueta existente, mas apenas validar a etiqueta, criar a função correspondente e executar em conformidade.

Em termos de eficiência, é possível que a função de AUTOLOAD sempre que é chamado, em vez de executar o código em causa para gerar o XML respectivo, gere em tempo real a função que gera esse código, de forma a que da próxima vez que essa mesma etiqueta seja usada, a função AUTOLOAD[3] já não seja usada.

```
1  sub AUTOLOAD {
2     my $attrs = {};
3     my $tag = our $AUTOLOAD;
4     $attrs = shift if ref($_[0]) eq "HASH";

5     if (exists($PTAGS{$tag})) {
6       my @tags = @{$PTAGS{$tag}};
7       my $toptag = shift @tags;
8       return _xml_from($toptag, $attrs,
9                        _go_down(\@tags, @_));
10    } else {
11      return _xml_from($tag,$attrs,@_);
12    }
13 }
```

Notas:

linha 3: Nome do método invocado;
linha 4: Extrair atributos da invocação;
linha 5: Validar se estamos perante uma PowerTag;
linha 6,7 e 8: Processar recursivamente a PowerTag;
linha 11: Gerar função respectiva a etiqueta simples;

No caso das PowerTags, há necessidade de análise do DTD do documento. Para não obrigar a que o DTD seja analisado sempre que determinado método é chamado e não reconhecido como etiqueta, as PowerTags são definidas na altura em que o módulo é invocado.

[3] O código aqui apresentado não é o código Perl real. Apenas pseudo-código para ilustrar a ideia.

4 Conclusões

A possibilidade de tornar a geração de XML embutida numa linguagem de programação, torna mais eficaz este processo. Não só se poupa espaço, como o código se torna mais legível e garante maior fiabilidade no XML gerado.

Se houver os cuidados necessários na definição das funções de geração de XML pode-se obter XML correctamente (ou quase) identado, e mais legível do que o habitual XML gerado por aplicações.

Com as PowerTags mostramos ainda que com uma análise pequena ao DTD se torna possível a definição de macros automáticas.

Referências

1. José João Almeida and Alberto Manuel Simões. Inferência de tipos em documentos xml. pages 144–154, February 2005.
2. Mark Jason Dominus. *Higher Order Perl*. Morgan Kaufman, 2005.
3. David Megginson. *XML::Writer — Perl extension for writing XML documents*, Perl man pages edition.
4. Lincoln D. Stein. *CGI — Simple Common Gateway Interface Class*, Perl man pages edition.
5. W3C HTML Working Group. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). Technical report, W3C - World Wide Web Consortium, 2002. `http://www.w3.org/TR/xhtml1/`.

Configuring Web Wizards in XML

Marcos Aurélio Domingues and José Paulo Leal

DCC-FC & LIACC, University of Porto
R. Campo Alegre, 823 – 4150-180 Porto, Portugal
{marcos,zp}@ncc.up.pt

Abstract. A "wizard" is a common pattern used in graphical interfaces when an application needs to collect a large number of parameters. Wizards use progressive disclosure to present windows with small sets of parameters, and parameters selected in the first windows control those presented in subsequent windows. This concept of wizard can also be used to develop web interfaces to existing monolithic systems that lack a GUI of some sort and would benefit from being accessible on the Internet. In this paper we propose a framework for developing web wizards whose extension points are XSLT transformations based on a XML description of the systems parameters. With this approach the system and its web interface are loosely coupled and thus parameters can be changed or mapped differently into the system just by reconfiguring XML files.
This framework has been tested in the development of web wizard for a system that generates mathematics exercises using constrained grammars. The system, developed within project AGILMAT, has a large number of parameters that difficult its use by novice users. This paper also describes a number of features that were developed for the AGILMAT's web wizard.

1 Introduction

Most systems are designed with an users interface in mind. Nevertheless, there are cases where complex systems are developed with a poor users interface and thus need to be re-engineered in order to made them available to a larger user base. This is often the case with systems that steam out of research projects since they were developed around concepts that were not completely defined in the design phase. Understandably, these systems have a large set of parameters that must be exposed in the users interface.

In order to develop web interfaces to existing systems that lack a GUI of some sort, we propose a framework for developing web wizards interfaces whose extension points are XSLT transformations based on a XML description of the systems parameters. With this approach the system and its interface are loosely coupled and thus parameters can be changed or mapped differently into the system just by reconfiguring XML files.

Decoupling the users interface from the systems core is consensually recognised as an important design principle. In our approach with take this principle a step further. Since parameters are defined in configuration files, they can be

easily added, changed or removed from the web interface. This feature is specially important in systems that are still being used in research. In these systems, changes in configuration parameters are frequent and user interfaces cannot be designed around rigid concepts.

Communication between the web server that manages the interface and the system is made trough input/output (I/O) streams. The parameters collected in the interface are converted into system commands using XSLT transformations and injected into its standard input stream. Data returned from the system's output stream is converted to XML and processed with XSLT transformation back into the web interface.

We claim that the approach we purpose is effective for developing a web interface to systems when certain condition are present:

- it was developed without an users interface in mind;
- it has a large number of configuration parameters;
- it has a small number of (ideally a single) actions;
- it can be fully controlled trough standard input/output.

All these conditions are met in AGILMAT. The goal of this research project is to produce a system to generate high-school and pre-university level exercises of mathematics. Exercises are generated using constrained grammars that are configurable by an extensive set of parameters. This system is implemented in Prolog and can be controlled by executing goals in the interpreters shell.

The ultimate goal of AGILMAT is to provide a tool for teachers available on the web. A first web interface was developed to AGILMAT and is currently being used for testing purposes in the development of the system [1]. This first web interface was tested with a focus group of high-school teachers. The results gathered from this experiment convinced us that the web interface would have to be profoundly changed, in order to reduce the actual number of parameters that a typical user would have to set.

This paper is organized as follows. We start by presenting the framework proposed for developing web wizards using XML (sect. 2). In Section 3 we present a case study where our framework has been tested in the development of web wizard for the AGILMAT. We also describe a number of features that were developed for the AGILMAT's web wizard. Then we conclude suggesting a few of the possible lines of future work (sect. 4).

2 A Framework for Developing Web Wizards

In this section, we present the framework proposed to make easier the development of web wizards. The key points of this framework consist in storing a XML description of the parameters of a system into a file and using XSLT transformations on this file to get the web wizards that will facilitate the configuration of the system parameters. The XSLT transformations act as extension points between the web wizard and the the system. The framework is illustrated in Fig. 1: browser, web wizard generator and system are represented as

strong squares, points of transformation are represented as strong T labeled circles, physical files with XML documents are represented as strong file icons and XML documents in memory are represented as dotted file icons.

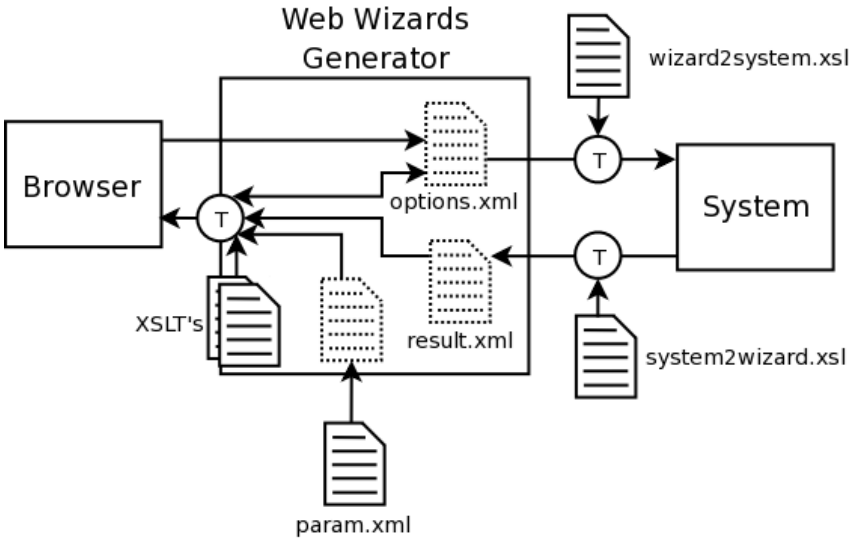


Fig. 1. Framework for Developing Web Wizards.

In Fig. 1, the file `param.xml` stores a XML description of the system parameters. This file is loaded to a DOM object and used with others document objects to make the XSLT transformations that generate the web wizards. The wizards generator has two other DOM objects in memory: `options.xml` and `result.xml`.

The DOM object `options.xml` manages and collects the parameters that are sent to the monolithic system. When the web wizards generator is initialized, the document `options.xml` is also initialized with the parameters from the file `param.xml`. This initializing process consists in applying XSLT stylesheets on the file `param.xml`. These generate the web wizards HTML and load the system parameters to the document `options.xml`. This document is never serialize into the file system. During interaction, the document `options.xml` is updated with the parameters received from the user browser.

XSLT transformations are also responsible for the appearance of the web wizards. In this way, to change the appearance of the web wizards, we only need to edit the appropriated XSLT stylesheets.

When the interaction requires executing an action in the system, the parameters in the document `options.xml` are convert to commands of the system. The conversion of formats is made using the document `wizard2system.xsl` to perform a XSLT transformation injected in the systems's standard input.

The DOM object `result.xml` stores the results from the system, in XML format. System output in XML format is integrated in the web wizards generator using also a stylesheet. This requires a previous conversion of the system output into XML format. The conversion can be performed either in the system or in the generator.

In this framework, when the need to add or change a parameter to the system arises, it is only required to edit the files `param.xml`, `wizard2system.xsl` and/or `system2wizard.xsl`. The programmer does not need re-write and re-compile the web wizards generator.

In the next section, we present a case study where our framework has been tested in the development of web wizard for a system that generates mathematics exercises using constrained grammars.

3 Case Study

In this section we describe the application of our framework to the project AGILMAT - Automatic Generation of Interactive Drills for Mathematics Learning [1, 2].

The project AGILMAT [1, 2] aims at developing a system of Constraint Logic Programming [5] to automatic generation and explanation of mathematics exercises that is flexible enough to be easily customizable to different curricula and users. Its major guiding principles are: the abstraction and formal representation of the problems that may be actually solved by algebraic algorithms covered by the curricula, the customization of these models by adding further constraints, and designing flexible solvers that emulate the steps students usually take to solve the generated exercises.

3.1 Architecture of the AGILMAT System

To make the AGILMAT system available for students and teachers, we use our framework to develop a web wizard. In Fig. 2 we present the contextualization of the AGILMAT system inside the framework proposed to develop web wizards: web wizard generator, cache and AGILMAT system are represented as strong squares and rectangles, internal modules of the AGILMAT system are represented as dotted rectangles, points of transformation are represented as strong T labeled circles, switcher is represented by strong rhombus, physical files with XML documents are represented as strong file icons and XML documents in memory are represented as dotted file icons. We also have a small screenshot of the web wizard developed for the AGILMAT system.

In the AGILMAT system (Fig. 2) there are two main modules that are written in Prolog and act as filters: the **expression generator** processes the user constraints and produces an expressions and types file, the **exercise generator and solver** processes this file and produces exercises and theirs solutions. This last module is the control of the system and makes use of several libraries that

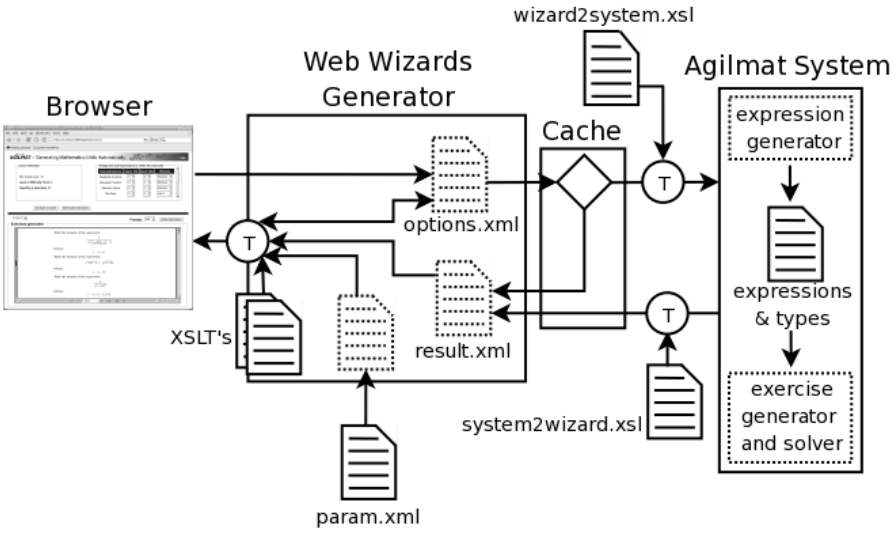


Fig. 2. Contextualization of the AGILMAT system in the framework for developing web wizards.

handle arithmetics, set operations and symbolic constraints (to solve inequations, disequations and equations).

The AGILMAT's web wizards uses the document `system2wizard.xsl` to convert the exercises and theirs solutions, represented by prolog predicates and clauses, to a XML representation. With a XML representation, we can further transform the exercises to the format XML - Question & Test Interoperability (XML - QTI) [3] with mathematical expressions represented in MathML.

In the current version, exercises and theirs solutions are converted to a LaTeX representation that is converted to different formats, such as: HyperText Markup Language (HTML), Portable Document Format (PDF) and PostScript (PS). The PDF file is embedded in the web interface. We are not yet using the document `system2wizard.xsl` to convert the exercises and theirs solutions to a XML representation. We hope to use this document in the next version of AGILMAT.

The web wizard of AGILMAT collects and manages a set of options to control the generation of mathematics exercises. As explained in Section 2, the web wizards generator is responsible for managing the user-system interaction. It receives data from the file `param.xml` and HTML forms, makes several XSLT transformations, produces the user constraints and runs the AGILMAT system to generate the exercises. The web wizard generator has been written in Java.

The current release of AGILMAT takes a considerable time to generate exercises. To improve the response time of the system we developed a cache system. When the cache is activated, the web wizard looks up a set of exercises previously generated with the same configuration values. We expect that most teachers will initialize using the default values, or changing only the parameters presented

in the first screen. The cache system will provide them with almost imediate feedback in these first atempts that will encourage them to explore AGILMAT and set parameters that will generate exercises more adapted to their goals.

3.2 Using AGILMAT Web Wizard

In Fig. 3 we present a screenshot of the first screen of the web wizard of AGILMAT. The most recent release of the system is available at `http://www.ncc.up.pt:8080/Agilmat`.

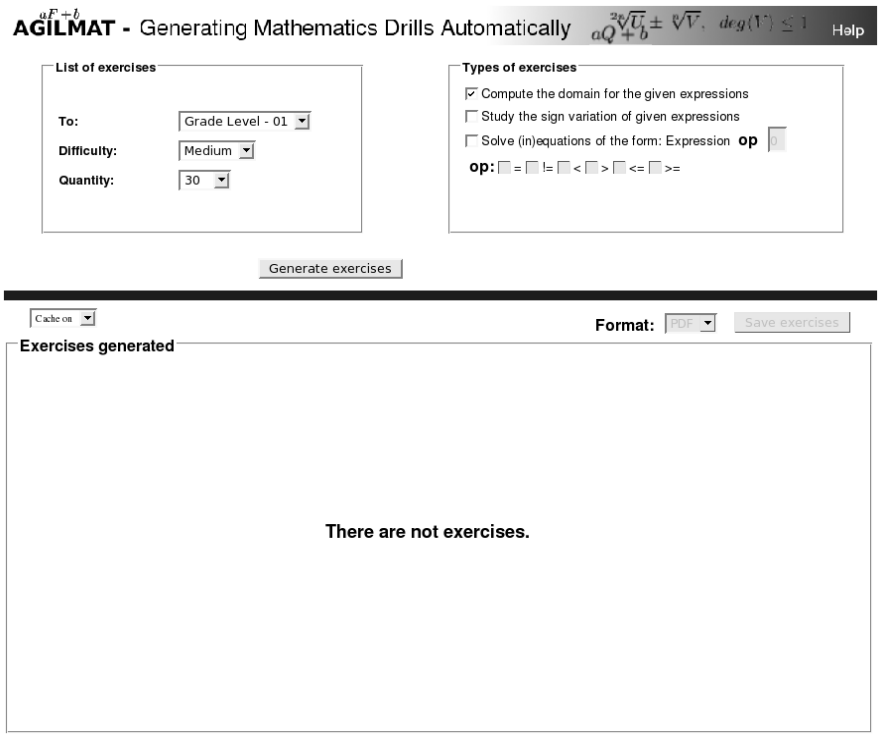


Fig. 3. Screenshot of the first screen of the AGILMAT wizard.

A novice user of AGILMAT starts generating exercises defining only a minimal set of options. In particular the user can define a **profile** of exercises that will be generated. This makes the system initialize the range of each parameter of AGILMAT system with suitable default values. These values may correspond not only to grade levels (Tenth grade, Eleventh grade, etc) but also to specific topics of the curricula. The advantage of this option is to allow novice users to generate exercises for their particular needs without having to set a large number of parameters.

The user can also define the quantity and types of exercises that will be generated. The available types are "Compute the domain", "Study the sign variation" and "Solve (in)equations". Exercise sheets may contain several different types of exercises. The user can generate exercises immediately after setting these parameters and expect them to be adequate to his or her needs.

Afterwards, the user can change the quantity and difficulty of each subexpression that can appear in exercises (second screen of the AGILMAT wizard, ilustrated in Fig. 4). These parameters give extra control over the generated exercises. Each parameter has a selection with certain number of options. The available parameters, their options and default values are dependent from the initial selection of exercise profile.

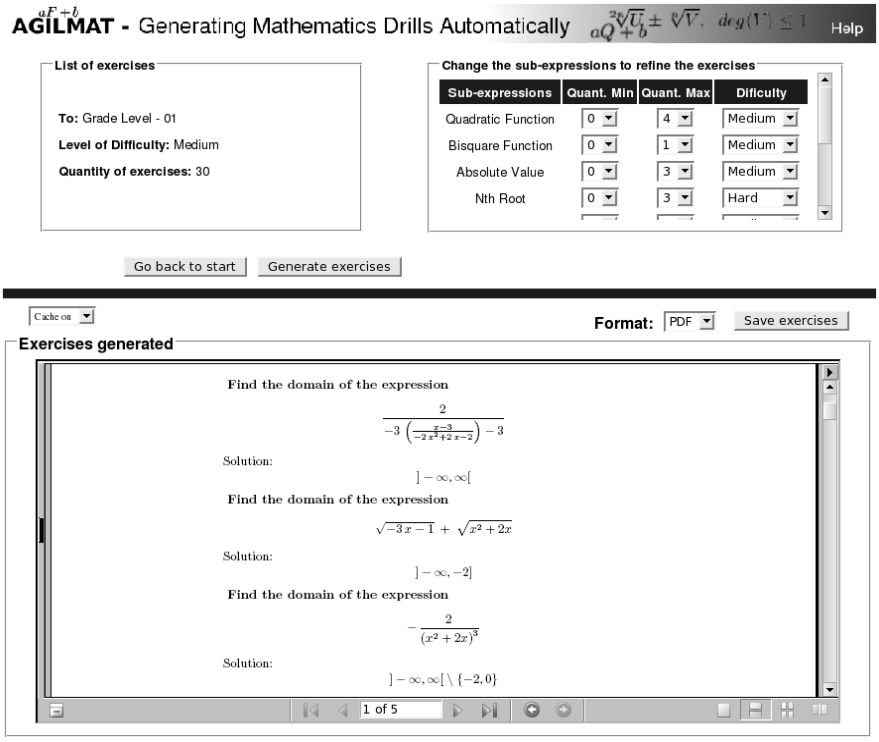


Fig. 4. Screenshot of the second screen of the AGILMAT wizard.

3.3 Configuring AGILMAT's Web Wizard

The first AGILMAT web interface was tested with a focus group of high-school teachers. The results gathered from this experiment convinced us that the web

interface would have to be profoundly changed, in order to reduce the number of parameters that a typical user would have to set.

Taking this fact in consideration, we decided to group the system parameter into profiles in such way that the choice of a profile would set the parameters default values and hide irrelevant parameters. In Fig. 6 we present fragments of the file `param.xml` that focus on a profile. The node `configs` is the root node of the file `param.xml`. The elements `profile` contain a set of parameters that represents a grade level or a specific topic of the curricula, as for example, the type of the exercises that will be generated. We can have simple parameters or composition of parameters, that can have one or more values. The element `option` represents a simple value and the element `options` represents an interval of values. Parameters can be hidden from the user by changing the attribute `isVisible`.

```
<!-- ... others parameters ... -->

<parameter isVisible="true" id="rdivisao" name="divisao">
  <option label="Hard" value="8"/>
  <option label="Medium" value="5" default="true"/>
  <option label="Easy" value="3"/>
</parameter>

<!-- ... others parameters ... -->

<parameter isVisible="false" id="rdivstype" name="divstype">
  <option value="" exprs="$rdivisao + 2" default="true"/>
</parameter>

<!-- ... others parameters ... -->
```

Fig. 5. Interdependency between two parameters.

In a profile, parameters value can be computed from the values of others parameters. In Fig 5 we show how the expression `$rdivisao + 2` sets the value of parameter `divstype` where `$rdivisao` references the value of another parameter.

Managing a large number of profiles with common parameters in a single `param.xml` is too canbersome. To deal with this problem, we implemented profile inheritance to reuse profile definitions, and made use of the XML inclusion mechanism to split the configuration document in several files. In Fig. 6 the element `profile` presents the attribute `extends` whose value is the name of the extended profile. The extension can be done between profiles in the same or in different files using `xinclude` [4]. An example of the inclusion mechanism `xinclude` is presented in Fig. 6.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- ... others elements ... -->

<configs>

  <profile name="p11" label="Grade Level - 11" extends="p01">

   <screen_wizard name="first">
      <parameter isVisible="true" id="qtyexe" name="quantity_exercise" >
        <options start="5" end="30" step="5" default="5"/>
      </parameter>

      <!-- ... others parameters ... -->

      <composite label="Types of exercises" isVisible="true">
        <parameter hasDependents="false" isVisible="true" id="edom" name="domain">
          <option label="Compute the domain" value="false" default="true"/>
        </parameter>
        <parameter hasDependents="true" isVisible="true" id="econ" name="constraint">
          <option label="Solve (in)equations: Expression" value="false" default="true"/>
        </parameter>
        <parameter depends="econ" isVisible="true" id="econccond" name="cond">
          <option value="0" default="true"/>
        </parameter>
        <parameter depends="econ" isVisible="true" id="econoeq" name="eq">
          <option label="=" value="false" default="true"/>
        </parameter>
      </composite>
   </screen_wizard>

   <screen_wizard name="second">
      <parameter isVisible="false" id="rabsquotbasic" name="absquotbasic">
        <option value="100" default="true"/>
      </parameter>

      <!-- ... others parameters ... -->

      <composite label="Quotient of Functions" isVisible="false">
        <parameter isVisible="false" id="fmidivbasic" limit="min" name="divbasic">
          <options start="0" end="2" step="1" default="1"/>
        </parameter>
        <parameter isVisible="false" id="fmadivbasic" limit="max" name="divbasic">
          <options start="0" end="2" step="1" default="1"/>
        </parameter>
        <parameter isVisible="false" id="rdivstype" name="divbasic">
          <options start="0" end="2" step="1" default="2"/>
        </parameter>
      </composite>

      <!-- ... others composites ... -->

   </screen_wizard>

</profile>

<!-- ... others profiles ... -->

<xi:include href="param01.xml" xpointer="p01" xmlns:xi="http://www.w3.org/2001/XInclude"/>

</configs>
```

Fig. 6. Fragments of the file `param.xml`.

4 Conclusions and Future Work

In this paper we propose a framework for developing web wizards whose extension points are XSLT transformations based on a XML description of the systems parameters. With this approach the system and its web interface are loosely coupled and thus parameters can be changed or mapped differently into the system just by reconfiguring XML files. We have successfully tested our framework in the development of web wizard for a system that generates mathematics exercises using constrained grammars. The system, developed within project AGILMAT, has a large number of parameters that difficult its use by novice users.

As future work, we plan to developt a document `system2wizard.xsl` to convert the exercises and theirs solutions from the AGILMAT system (represented by prolog predicates and clauses) to a XML representation, completing the proposed framework inside the project AGILMAT. With a XML representation, we will can make many others transformations, as for example, to transform the exercises to the format XML - Question & Test Interoperability (XML - QTI) [3]. We also plan to test our framework by developing a web wizards generator for an application with characteristics differents of the AGILMAT system. This will help us to better identify the features that are common to the framework and separate them from those specific to AGILMAT.

Acknowledgements

This work is partially supported by Fundação para a Ciência e Tecnologia (FCT), and Program POSI, co-financed by EC fund FEDER, under project AGILMAT (contract POSI/CHS/48565/2002).

References

1. A. P. Tomás, J. P. Leal. A CLP-Based Tool for Computer Aided Generation and Solving of Maths Exercises. In V. Dahl, P. Wadler (Eds), *Practical Aspects of Declarative Languages, 5th Int. Symposium PADL 2003*, Lecture Notes in Computer Science 2562, Springer-Verlag (2003) 223–240. ©Springer-Verlag.
2. A. P. Tomás, N. Moreira, N. Pereira. Designing a Solver for Arithmetic Constraints to Support Education in Mathematics. DCC-FC & LIACC, University of Porto, August 2005. *(working paper)* `www.ncc.up.pt/~apt/AGILMAT/PUBS/solver-Aug05.pdf`
3. IMS QTI Specifications. IMS Global Learning Consortium, Inc. `www.imsglobal.org/question/index.html`.
4. XML Inclusions (XInclude) Version 1.0. W3C Recommendation 20 December 2004. `www.w3.org/TR/xinclude/`.
5. Marriott, K., and Stuckey, P.: Programming with Constraints – An Introduction. The MIT Press (1998)

XML Annotation of Historic Documents for Automatic Indexing *

Cristina Ribeiro[1,2], Gabriel David[1,2], André Barbosa[1,2]

[1] FEUP—Faculdade de Engenharia da Universidade do Porto
[2] INESC — Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
{mcr,gtd,andre.barbosa}@fe.up.pt

Abstract. This paper describes the process and tools used in the development of a digital documentation center for historic documents. A team of archivists has handled document organization and description according to archival standards. The document repository and the access interface are supported on a prototype multimedia database. The focus here is on the treatment of the textual contents.
One of the goals of the project is to provide a search mechanism for the documents. Plain full-text search is of limited use in this kind of documents, where only archaic Portuguese or Latin versions of the text are available. The designations used for places, people and even common objects have diverse forms and are hard to recognize for non-specialized users.
The first step of this work has been the definition of a set of tags to mark relevant aspects of the documents, followed by the specification of a compact XML dialect. An XML editor and a set of associated tools were then developed. The editor helps archivists to enrich documents with content annotations that have high potential as search terms. This set of tools is currently in use and the annotated documents are indexed by a search tool sensitive to the annotations.

Keywords: XML Edition, XML dialects, Cultural Heritage Applications.

1 Introduction

Digital technologies are raising the public expectations with respect to access to cultural heritage. The traditional custodians of historic materials (archives, libraries, museums) are being prompted for providing access to various items in digital form. The diversity of available assets requires concern with their organization and search.

A researcher who deals with historic information sources needs access to the original documents or, at least, to good quality reproductions. That is one of the reasons for the extensive digitization programs going on. But images are

* Partially supported by FCT under project POSC/EIA/61109/2004 (DOMIR)

hard to search and the skills to read ancient documents are confined to a group of specialists who do not always agree on a given interpretation. A document image can be complemented by a transcription of its contents in text format. Depending on the language of the original document and the intended readers, a translation to a modern language may also exist. Summaries and comments by the historian may also condense or explain the document. All these forms are searchable and correspond to different reading levels.

Understanding a historic document requires knowledge of its content, of what it says, who is involved, persons or institutions, which are the places mentioned, and so on. But it also depends on who has created the document, when, for what purpose, aspects related to the context of the document, that explain its relevance. Archivists usually deal with this second aspect in a relatively structured way, following description rules according to international standards [1,2]. Storing such information is compatible with a relational database approach [3]. However, the work done by historians on the analysis of the document content is by nature a lot more dependent on the document itself and thus inherently semi-structured. It may contain the identification of certain words in the document as representing persons, places, events, etc. A summary or comments by the historian may be included in a more or less systematic way. This kind of information is better dealt with using an XML approach, using the document text and suitable annotations.

A second reason for using XML is its ability to express relationships between different elements through the use of XML attributes. Combined with authority files, this is an efficient way not only to mark, for instance, a certain expression as a person's name, but also to associate it to the person in the authority file, regardless of the specific name form used in the mention.

The kind of documents at stake is the textual document subject to digitization. But, as soon as the database is able to store images, why not storing sound or video? The whole world of multimedia databases becomes an issue. In this world content descriptions are a lot more varied than in the textual content. The relevant international standards [4] are based on XML and this constitutes the third reason for choosing XML in databases of historic documents.

Finally, to index the combination of database fields and XML documents under an approach of full text indexing but assigning different relevance factors to different tags or database fields, first a transformation of the latter into XML is performed. After that, it is possible to apply XML enabled indexing and querying tools. XML thus plays the role of a glue tying the document content and the several metadata components together.

The paper is organized as follows. Section 2 justifies the importance of having wide computer based access to historic documents. Then an architecture to improve their retrieval is presented in section 3. The component that is the subject of the paper is the annotation editor for historic documents presented in section 5, after the discussion of its schema in section 4. Some conclusions are put together in the final section 6.

2 Access to Cultural Heritage assets

Cultural heritage assets are complex objects: they may include for instance historic texts, photographs, songs, tales and performances, along with various texts, which convey the reflections that specialists have produced when analyzing them.

For a cultural heritage institution, it is more and more the case that its objects have been subject to some sort of treatment. The most common is description, and in museums, archives and libraries it is expected that objects only begin their life in some form of custody after they have been described. But other manipulations are also frequent. Digitization is one of them, performed either to safeguard especially valued items from frequent manipulations, to explore the digital content with specific automatic tools or simply to ease the access and expose the available collections. Another one is content analysis in several perspectives and to different depths.

Online public access to cultural heritage has grown as a consequence of the availability of digital versions of objects, the expansion of networking and generalization of systematic use of databases in the management of object collections. The huge numbers of items in these collections require some form of organization. The experience of archives points to a hierarchical organization with flexible number of levels and level designations but with a uniform set of description attributes.

Access to cultural heritage repositories must allow for a spectrum of different user profiles, ranging from the technical staff that describes objects to the manager of the objects or collections who chooses the structure of the collection, from the lay user that may engage in browsing digital representations to the scholar who can contribute valuable annotations on the history or features of the objects.

When digital versions of the objects are available, and specially if they already have specialized metadata attached, it becomes obvious that they might be made available for browsing and searching. Due to the diversity of objects, the different metadata standards and the different modes of use, there is currently no uniform solution for this task [1,4,5].

3 Searching Historic Documents

Historic documents are usually written records testifying facts considered important for historic studies. They are kept in public or private archives where they are preserved and put at the disposal of the users, according to specified access rules. Due to their uniqueness, the preservation concerns have always conflicted with the goal of providing access to this memory of the society. Reproduction has been the answer for it, in microfilm and, nowadays, by digitization and subsequent computer storage. Very high quality digitization is also seen as a way to ensure long time preservation as it can be subject to lossless reproduction, even if the original decays.

The approach implied in this paper has been developed in two previous projects, both the model and an implementation. In the first one, Archivum [3], a

hierarchical model for the storage of contextual metadata has been designed, thus addressing the problem of dealing with large sets of documents. In this model, the description of a document is a leaf in a tree whose root is the fonds, or structured collection. Implemented in a relational database, it supports searches on the description attributes, returning references to the place in the archive where the actual document can be found.

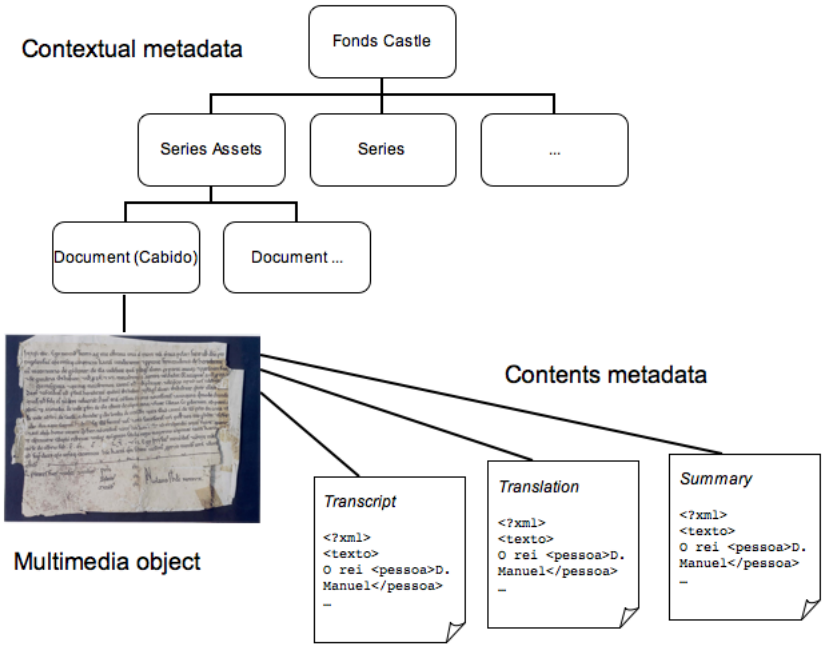


Fig. 1. Information architecture: contextual metadata, multimedia object and content metadata

In the second project, Metamedia, the goal has been to extend the ideas to deal with also storing the "documents", taken in an enlarged sense encompassing multimedia objects. For instance, a news program on a TV broadcaster could be seen as a document, as well as the digitization of a judicial process, or the digitally signed message of an e-commerce transaction. The goal has been to build a multimedia database able to display the object along with its description.

Soon after it was recognized that the richer nature of the objects called for a new part in the model. It is called the contents metadata and is primarily concerned with adding to the multimedia object descriptors related to its contents, like for instance the color distribution of a picture, the identification of scene cuts or the subtitles in a video or the melody in a music record. There is a

large number of such descriptors in international standards [4,5] and more could be devised. Due to the variability of their structure, it is not practical to force them into a relational model. Most of them are defined in XML and an arbitrary number can be associated to the object they describe. The search paradigm supported by these descriptors is no longer the satisfaction of a boolean expression, with no meaning in many cases, but some form of similarity search.

These ideas are fed back to the representation of historic documents, which are basically textual but where the image containing the scanned version plays such an important role, substituting for many purposes the actual document. In this setting (see example in Figure 1), the information relevant for a single document may be located in a few different places:

- contextual metadata: description at the level of Fonds (relational record; inherited by the document);
- contextual metadata: description at the level of Series (relational record; inherited by the document);
- contextual metadata: specific document description (relational record);
- multimedia object: high quality digitization (plays the role of information source; picture);
- multimedia object: low resolution digitization (for preview; picture);
- content metadata: transcription of the original into an annotated text format (XML descriptor);
- content metadata: translation to modern Portuguese (XML descriptor);
- content metadata: summary (XML descriptor).

The previous sections describe the query model for the end-user. However, to organize the production of metadata in particular content metadata consisting in text and markup., a different approach is more suitable: the documents are classified by fonds and a directory is created for each fonds; transcriptions and translations are then put together in the corresponding directories; the specialized XML Editor, before opening a single document, analyzes the directory in search for common definitions, relevant to the specific document.

4 A Dialect for Document Annotation

The project goals impose concern with both the technical description of documents and the exploration of the meaning of its textual content. The technical description is handled by archivists and uses the ISAD/ISAAR archival description standards [1,2], comprising items such as title, summary, creator, owner and dates. For the textual content, processing is required. The first step,transcription of the document text, is not an automatic task due to the nature of the documents but had already been produced for the documents in the collection. The result is in most cases a text in Latin or archaic Portuguese. The History experts have identified several annotations that can be associated to words or phrases in the text, clarifying their meaning. Examples of such annotations are those marking the name of a person or of an institution, a title of nobility or a place.

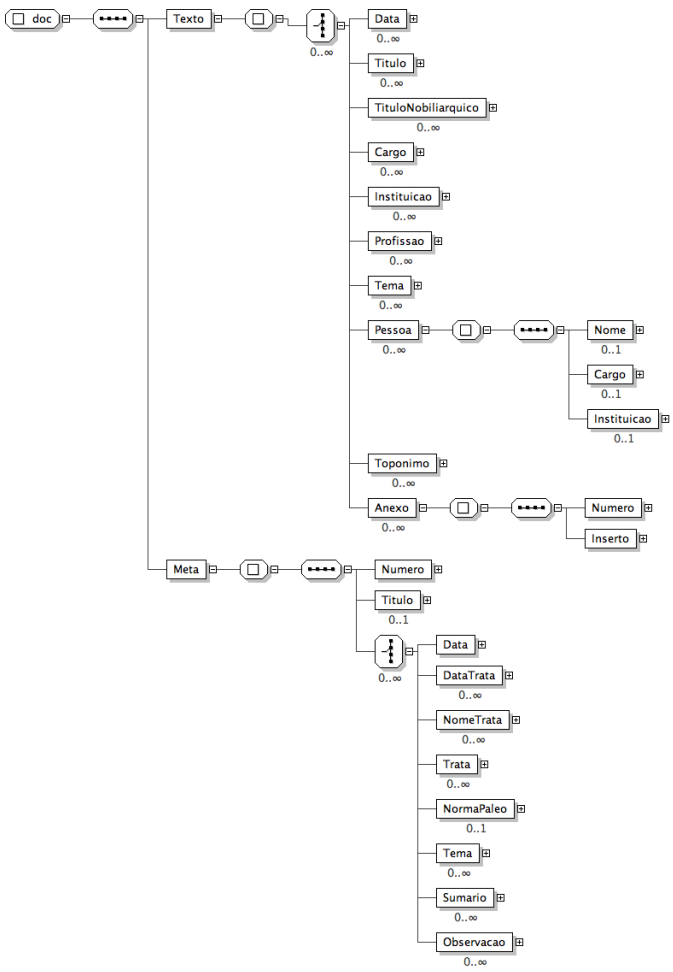


Fig. 2. XML Schema for the annotation dialect

The MetaMedia platform [6] has been proposed for supporting the descriptive metadata, providing the environment for archivists to edit and structure the documents in the collection and for regular users for browsing and searching. After collecting annotations on document contents it is possible to have search performed on a combination of descriptive metadata, full text and annotations. In the case of medieval documents, annotations are fundamental for taking advantage of the document content in search.

XML has been the obvious choice for the annotation of documents. The very specific nature of the documents excluded the use of a standardized dialect. The annotations are included in the documents by the domain experts in the process of analyzing the documents. A simple and intuitive interface was required for their task.

The first challenge of the project was to specify an appropriate dialect that could be applied to annotate all the documents of the documentation center. The issue has been discussed in the development team and the result is an XML dialect addressing the concepts identified for this kind of collection and with a focus on flexibility in order to encompass different kinds of documents.

The dialect is described with the XML Schema graphically represented in Figure 2. The structure of the annotated documents includes a set of metadata descriptors and annotations on parts of the text. The XML instances basically have a block of descriptors (document number and title, information on the scholars who treated the text, themes, a summary) followed by a block of text interspersed with annotations on expressions identifying people, places, nobility titles, among others. The Schema clearly distinguishes elements that qualify a piece of text from those that apply to the whole document. Annotations occur inside the *Texto* tag (text) and the so-called external annotations are inside the *Meta* tag (metadata). Annotations where translation occurs are used to specify the contemporary equivalent of some expression in the original text.

The root element is *Documento* (document) representing the document. Three elements are always present in the document: *Numero* (number), *Data* (date) and *Titulo* (title). The first is an identifier used in the documentation center, the second is a complex element containing information on meaningful dates related with the document that can also have a local (Toponimo) associated with the date and the third is the title of the document. Several other elements are defined such as *Nome* (name), *Cargo* (job), *Instituicao* (instituion), *Pessoa* (person), *Toponimo* (place), *Sumario* (summary), *Tema* (theme), *Observacao* (observation) and *Anexo* (annex). *Pessoa* is a complex element and contains optional information about name (*Nome*), job (*Cargo*) and institution (*Instituicao*). The other complex element is *Anexo*, containing informations about the identifier of an attached document (*Numero*) and its text, the content of a full document (*Documento*). Some of the elements are repeatable. Some can either be part of one complex element or just appear alone in the text.

```
<?xml version="1.0" encoding="UTF-8"?>

<Documento>
<Texto>
<Data>1284 AGOSTO 1</Data> - Inquirição no julgado de Fermedo.

Item <Nome>Fruitoso do Outeiro</Nome> a hi outra vinha que lhi deu o joiz a foro de sesta e de meya dereitura.
E <Nome>Domingos Paez</Nome> do <Toponimo>Campo</Toponimo> a hi outra vinha no <Toponimo>Campo</Toponimo> [fl.
2] de sesta e de meya direitura. E <Nome>Pedro Mauro</Nome> ha y outra vinha a par de sa casa e deu-lha o juiz
o foro de sesta e de meya dereytura. E <Nome>Pedro Periz de Paramoo</Nome> ha y hua vinha a foro de septima e de
meya dereytura. E <Nome>Paayo de Paramoo</Nome> ha y hua vinha e deu-lha o juiz a foro de septima e de meya
dereitura. E disserom que os juizes da terra derom estas vinhas de suso ditas a estes foros de suso ditos sen
mandado d'el Rey e que se acordam que os nom apergoavam ante. E todalas <Cargo translation="Testemuya">testemuya
</Cargo> disserom que essa <Instituicao translation="Igreja de Santa Maria de Formedo">igreja de Santa Maria de
Formedo</Instituicao> est d'el Rey e esta en posse de presentar a ela e est na posse dela.
</Texto>
<Numero>72A</Numero>
<Titulo>[Inquirição régia no Julgado de Fermedo]</Titulo>
<Tema>Vinho</Tema>
</Meta>
</Documento>
```

Fig. 3. Example of Document Annotations (XML source)

4.1 Example Documents and Annotations

Figures 3 and 4 contain two views of an instance document conforming to the developed XML Schema. The first is the textual source document where the separation between the enhanced full text (element *Texto*) and the document metadata section (element *Meta*) are visible.

The view in Figure 4 comes from the Annotation Editor interface described in what follows. Only the base text is visible here, and the parts where content annotations were introduced are highlighted (each kind of annotation corresponds to a different color). The annotations inside the *Meta* element are not visible—they are assigned by choosing their menu entries in the editor.

Item Domingos Paez da Botea a hi outra vinha no Conchouso e deu lha o joiz a foro d'oytava e de meya dereitura Item Fruitoso do Outeiro a hi outra vinha que lhi deu o joiz a foro de sesta e de meya dereitura. E Domingos Paez do Campo a hi outra vinha no Campo [fl. 2] de sesta e de meya direitura. E Pedro Mauro ha y outra vinha a par de sa casa e deu-lha o juiz o foro de sesta e de meya dereytura. E Pedro Periz de Paramoo ha y hua vinha a foro de septima e de meya dereytura. E Paayo de Paramoo ha y hua vinha□ e deu-lha o juiz a foro de septima e de meya dereitura. E disserom que os juizes da terra derom estas vinhas de suso ditas a estes foros de suso ditos sen mandado d'el Rey e que se acordam que os nom apergoavam ante. E todalas testemuya disserom que essa Igreia de Santa Maria de Formedo est d'el Rey e esta en posse de presentar a ela e est na posse dela.

Fig. 4. Example of Document Annotations (Editor view)

5 The Annotation Editor

The editor gives specialists an easy way to annotate the documents, generating files in the XML dialect specified in the XML-Schema. The editor has been designed with a focus on flexibility, so that changes in the underlying schema can be easily accommodated.

The editor is implemented as a Java applet, making it easy to execute in a web browser in any platform. Figure 5 shows the editor window, with a set of buttons at the top giving access to the generic file-manipulation functionalities and a side pane where the list of annotations for the main text are available.

For the collection of historic documents, content metadata was already available concerning aspects such as authorship or themes. The editor has been coupled with a set of tools to deal with the conversion of these metadata items into document annotations.

A controlled vocabulary has been adopted for describing document contents. The vocabulary includes two parts: a predefined thesaurus and the lists of terms created when editing a group of documents.

The features provided by the annotation editor therefore comprise file management, controlled vocabulary visualization and management, annotation visualization and management, and access to the conversion tools.

In the file management area we can find the general options like creating, opening, closing and saving a document. Management of the controlled vocabulary (*Vocabulario Controlado*) is performed according to the context of the

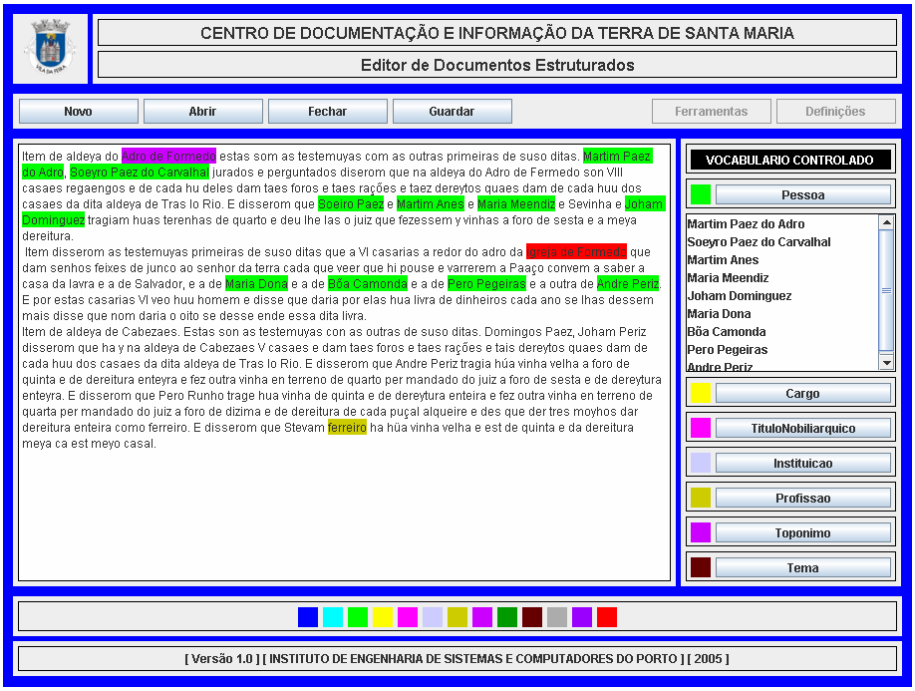


Fig. 5. Interface of the Annotation Editor

document being edited. The annotations from documents belonging to the current working directory are available in the controlled vocabulary for the current document. New terms created when annotating the current document will be added to this contextual vocabulary and become available for the edition of documents grouped under the same directory.

To add new annotations to the document the user can use either the controlled vocabulary menu or the contextual menu (right mouse button). It is possible to consult all the annotations made in the document using the legend area with the color signs at the bottom of the editor window.

At any time it is possible to run another tool or change the definitions of the editor. In this case it is necessary to close the document edition first. After this the options tools (*Ferramentas*) and definitions (*Definições*) are enabled.

5.1 Conversion Tools

The conversion tools have been created to incorporate in the XML documents information already created by the domain specialists and for allowing the use of an existing thesaurus for controlling themes to be associated to the documents.

The Documents Converter is a simple tool to convert previous annotations made by the documentation center and stored in Microsoft Excel format to XML

in the annotation dialect. This tool only requires the path of the original file and the path for the new file as shown in Figure 6.

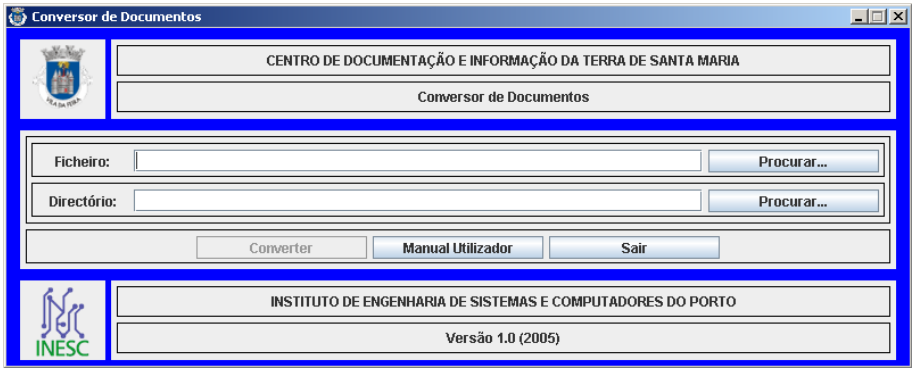


Fig. 6. Documents Converter

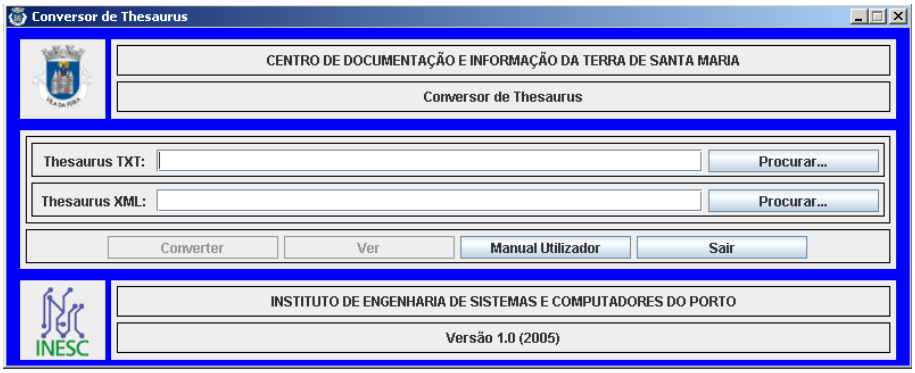


Fig. 7. Thesaurus Converter

The existing thesaurus is a hierarchic theme structure and has been received in Microsoft Excel format. The Thesaurus Converter creates a new representation used by the editor, converting the old format to XML. Just as in Documents Converter, this tool only requires the original and the destination paths. The interface is shown in Figure 7.

The Controlled Vocabulary Editor has been created to manage the annotations for the set of documents saved in a common directory. This vocabulary is saved as a binary file for efficiency. The tool is intended to read and edit it allowing the user to change this set of words and also to export the vocabulary

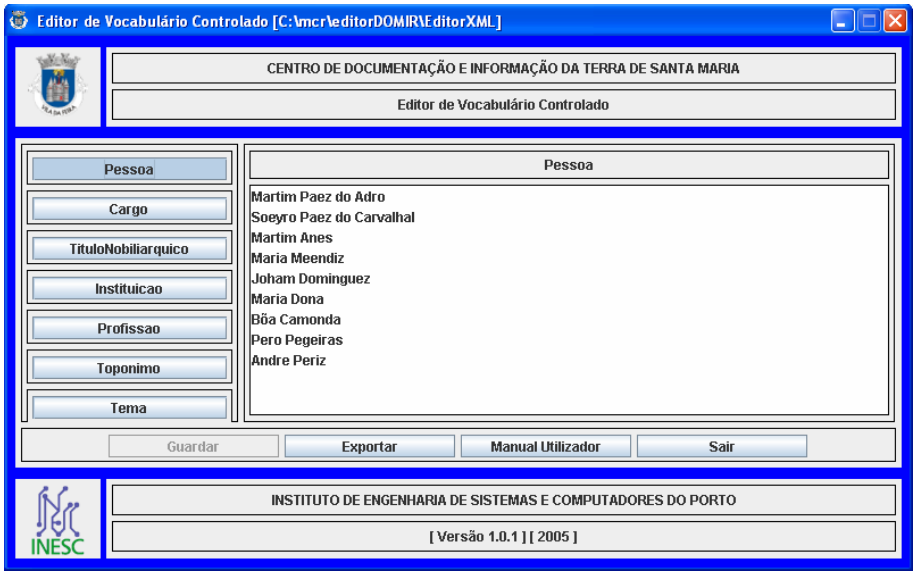


Fig. 8. Controlled Vocabulary Editor

in XML format for later use or just for presentation purposes. Figure 8 shows its interface.

6 Conclusions

Text content annotation is a natural use for XML, that can easily handle the need for a flexible set of annotations appearing at arbitrary spots in the text while keeping generic descriptors associated to the document as a whole.

This work has been motivated by the collaboration in a project for a digital documentation center managing a collection of historic documents. The original documents belong to several fonds on the National Archives, and the collection includes high-resolution digitizations and textual transcriptions. For some documents translations are also available. The purpose of the application is to make the documents available to an audience ranging from scholars to the general public, providing search on the textual document contents. The straightforward approach of using the full text of the documents is not an effective solution, as most documents are not available in current-day Portuguese.

The Annotation Editor is currently being used by the History and Archives specialists in the team to create the annotated versions of the documents. Existing controlled vocabularies and a thesaurus for themes are integrated in the editing facilities. The document model captured in XML Schema supports the definition of specialized indexes. Indexing on separate document descriptors allows more flexible querying and browsing. The final application will use both

the "enhanced full text" documents and their archival description records in the search interface.

The application supporting the work described here is a result of a line of research on multimedia databases and multimedia retrieval, seeking representations and methods for effective retrieval on structured multimedia objects. The current collection is being explored mainly on its textual content, but the availability of digitized versions of the documents will allow experimentation with image features as indexing dimensions and their use in the retrieval strategies.

References

1. ISAD(G): International Standard for Archival Description (1999) http://www.ica.org/biblio/isad_g_2e.pdf (1.12.2005).
2. ISAAR(CPF): International Standard Archival Authority Record for Corporate Bodies, Persons, and Families (1995) http://www.ica.org/biblio/isaar_eng.pdf (1.12.2005).
3. Archivum: System of Objects with Temporal Support for Archival Description. Technical report INESC (1998)
4. ISO/IEC: JTC1/SC29/WG11—MPEG-7 Overview (2004) http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm (1.12.2005).
5. TV-Anytime: TV-Anytime Forum Website (2005) http://www.tv-anytime.org/ (1.12.2005).
6. Ribeiro, C., David, G.: A Metadata Model for Multimedia Databases. In Bearman, D., Garzotto, F., eds.: Proceedings of ICHIM01: International Cultural Heritage Informatics Meeting- Cultural Heritage and Technologies in the Third Millennium. Volume 1., Politecnico di Milano and Archives & Museum Informatics (2001)

Poseidon: Uma aplicação de suporte ao desenho semi-automático de workflows

Jorge Cardoso

Departamento de Matemática e Engenharias
Universidade da Madeira
jcardoso@uma.pt

Resumo. Uma promissora solução de gestão, coordenação e orquestração de serviços electrónicos é o uso dos sistemas de gestão de workflows. Embora várias linguagens para modelar workflows, sistemas para a gestão de workflows e técnicas de análise formais tenham sido estudadas de um modo extensivo, a investigação de métodos para auxiliar o desenho de workflows é praticamente inexistente. O propósito deste trabalho é apresentar uma aplicação, denominada de Poseidon, para ajudar os analistas e designers de workflows no seu trabalho, permitindo a criação semi-automática de workflows com uma elevada qualidade. A aplicação Poseidon define um conjunto de passos que orientam o desenho de workflows com base em casos de negócio.

1 Introdução

A Web, o desenvolvimento do comércio electrónico (e-commerce), e novos conceitos arquitecturais tais como os serviços Web, criaram as bases para o aparecimento de uma nova economia interligada em rede (Sheth, Aalst et al. 1999). Com a maturidade das infra-estruturas que suportam o e-commerce, será possível às organizações incorporarem serviços Web como parte dos seus processos de negócio. Um vasto leque de modernos sistemas de gestão de workflows tem sido desenvolvido para suportar vários tipos de processos de negócio (Cardoso and Sheth 2003).

Os estudos e investigação realizados até a data na área da gestão de workflows centram-se em três domínios principais: arquitecturas de sistemas de workflow (Miller, Sheth et al. 1996), linguagem de especificação e modelação (Aalst and Hofstede 2003; BPML 2004; BPMN 2005; WS-BEPL 2005), e análise de workflows (Aalst 1998; Aalst 2000). Estas áreas de investigação são de reconhecida importância para a construção de robustos e sofisticados sistemas de gestão de workflows. Não obstante, uma área importante foi negligenciada, a investigação de metodologias para suportar o desenho de workflows. De facto, os estudos sobre o desenho de workflows são escassos. Em 1996, Sheth et al. (Sheth, Georgakopoulos et al. 1996) determinaram que a modelação e desenho de workflows e processos seria uma área de estudo importante que deveria ser investigada em mais profundidade. A utilização de metodologias adequadas para auxiliar o desenho de workflows é um elemento chave que

condiciona o sucesso de qualquer projecto de workflow e que requer a disponibilidade de aplicações específicas.

Devido a falta de aplicações apropriadas para assistir o desenho de workflows, desenvolvemos a aplicação Poseidon. A aplicação Poseidon é de grande utilidade tanto para os analistas de processos de negócio como para os desenhadores durante as duas primeiras fases do desenvolvimento do ciclo de vida dos workflows, i.e., a fase de identificação dos requisitos e a fase de desenho. Esta aplicação permite a criação de workflows de uma forma semi-automática. Embora existam várias ferramentas comerciais que permitem desenhar workflows, como por exemplo o ARIS Business Designer (ARIS 2005), TIBCO Staffware Process Suite (TIBCO 2002), COSA BPM (COSA 2005), MQSeries Workflow (IBM), METEOR designer (Kochut 1999), etc, estas aplicações não permitem o desenho semi-automático de workflows.

Este trabalho visa descrever a metodologia e funcionalidades da aplicação Poseidon. A aplicação pode ser usada durante a análise e a fase de desenho. Pode ser vista como uma metodologia que estrutura os vários passos que orientam o desenho de workflows baseada na compilação de requisitos obtidos da comunicação com as pessoas, gerentes e especialistas na área de modelação de processos de negócio.

Este trabalho está estruturado da seguinte forma. A secção 2 apresenta os requisitos e objectivos da aplicação Poseidon. Na secção 3 apresentamos e descrevemos a metodologia seguida pela aplicação Poseidon para suporta o desenho semi-automático de workflows. A secção 4 descreve alguns pormenores da implementação da aplicação. Finalmente, na secção 5 apresentamos as nossas conclusões.

2 Requisitos e objectivos da aplicação Poseidon

As equipas de desenvolvimento, os consultores e os académicos têm diferentes perspectivas acerca do desenvolvimento de processos de negócio e workflows. Algumas organizações vêem o desenvolvimento de workflows como uma actividade *ad-hoc* de desenho, outros vêem o desenvolvimento de workflows como um melhoramento ou o redesenho de processos de negócio isolados, e apenas uma minoria vêem-no como a reorganização de processos de forma compreensiva, utilizando metodologias e ferramentas de modelação para estruturar as actividades da organização num workflow bem definido.

Uma empresa pode escolher diferentes abordagens para desenhar diversos tipos de workflows. Não existe uma estrutura "certa" ou "errada" de desenvolvimento. Não obstante, algumas abordagens são mais apropriadas do que outras para o desenho de determinados tipos de workflows. Se uma abordagem inadequada for escolhida e usada então é muito provável que os workflows desenhados (Sheth, Aalst et al. 1999) tenham um valor e utilidade reduzidos. Tendo a noção que uma abordagem feita por medida pode não satisfazer as necessidades de diferentes empresas, desenvolvemos a aplicação Poseidon tendo por base os seguintes requisitos:

- **Simplicidade e facilidade de uso**: a aplicação e metodologia associada têm de ser facilmente entendidas pelos utilizadores finais;
- **Dimensão dos workflows**: a aplicação tem de suportar a modelação de workflows com uma dimensão pequena e mediana. Workflows pequenos são aqueles

que contêm aproximadamente 15 tarefas e workflows medianos são aqueles que têm aproximadamente 30 tarefas;

- **Estrutura dos workflows**: a aplicação tem de ser adequada para o desenho semi-automático de workflows de produção e de administração (McCready 1992), i.e., workflows mais estruturados, previsíveis e repetitivos;
- **Grau de automatização**: baseado numa das principais vantagens e objectivos dos sistemas de workflows, isto é a automatização, é requerido o desenvolvimento de uma aplicação e metodologia que permitam automatizar o maior número de etapas associadas ao desenho de workflows.

O ciclo de vida de um workflow é composto por várias fases, incluindo análise, desenho, implementação, testes e manutenção. Neste trabalho, estamos particularmente interessados na fase de desenho de workflows. O objectivo da aplicação Poseidon é fornecer uma metodologia implementada numa ferramenta que permita o desenho semi-automático de workflows. A aplicação permite auxiliar designers de workflows no seu trabalho e é independente da tecnologia utilizada na implementação ou programação do workflow. A aplicação fornece uma estrutura básica conceptual composta por passos, procedimentos e algoritmos que determinam como o desenho de workflows é empreendido.

3 Metodologia

Nesta secção iremos dar uma descrição geral sobre a metodologia seguida pela aplicação Poseidon para a construção de workflows. A metodologia envolve quatro fases principais e para cada uma apresentaremos mecanismos para o suporte e assistência a sua concretização.

A primeira fase (secção 3.1) da metodologia apoia-se fortemente na informação recolhida em entrevistas, sessões de grupos, "brainstorming" e reuniões entre as várias pessoas com conhecimento tácito sobre o funcionamento dos processos de negócio. Usaremos o termo 'entrevistas' para designar estes três métodos de obtenção de informação. As entrevistas deverão ser realizadas entre os analistas de workflows e as pessoas que conhecem bem a lógica dos processos de negócio. As pessoas que conhecem a lógica de negócio irão tipicamente incluir pessoal administrativo, gestores de departamentos, quadros gestores médios ou mesmo CEOs (Chief Executive Officer). Com base no conhecimento recolhido construímos uma tabela de casos de negócio (TCN). A propriedade básica de um processo é que este é baseado em casos. Isto significa que cada tarefa é executada com base num caso específico (Aalst 1998). A tabela captura os vários casos que um processo de negócios descreve.

Na segunda fase (secção 3.2) é extraído um conjunto de funções de escalonamento com base na informação presente na TCN. Como um workflow é composto por um conjunto de tarefas, para cada tarefa é extraída uma função de escalonamento. As funções são subsequentemente minimizadas. Uma função de escalonamento é uma função Booleana para a qual os parâmetros são as variáveis de negócios. Uma variável de negócio é uma variável do workflow presente nas condições que determinam o controlo do fluxo de tarefas. Tipicamente, essas variáveis estão associadas as transi-

ções que saem de das tarefas do tipo xor-split ou or-split (Cardoso 2005). Cada função modela o comportamento de uma tarefa ou tarefas em tempo de execução de um workflow, i.e., para um dado conjunto de variáveis de negócios e seus valores, a função de escalonamento indica se a tarefa ou conjunto de tarefas são executadas ou não em tempo de execução.

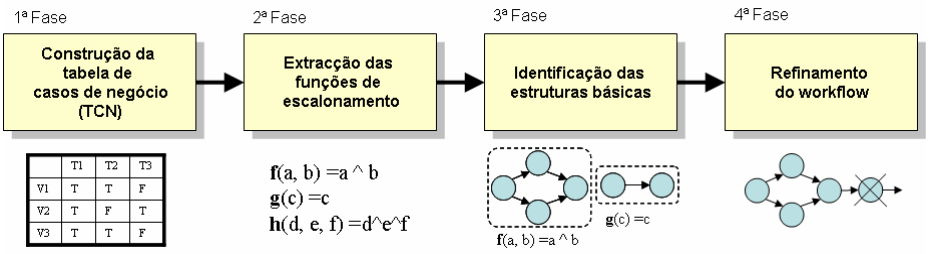


Fig. 1. Metodologia associada a aplicação Poseidon

A terceira fase (secção 3.3) consiste no uso das funções obtidas na fase anterior para identificar os blocos sequenciais e paralelos presentes num workflow. Um workflow pode ser decomposto em blocos básicos sequenciais, paralelos, e não determinísticos (Cardoso, Miller et al. 2004).

Na última fase (secção 3.4), integramos os blocos previamente identificados com os blocos não determinísticos existentes no workflow. O workflow é limpo de qualquer tarefa auxiliar introduzida sem valor para o workflow e, se necessário, o workflow pode ser ligeiramente reestruturado ou modificado por razões de claridade.

3.1 Tabela de casos de negócio

A propriedade básica de um processo é este ser baseado em casos de negócio (Aalst 1998). Isto significa que cada tarefa é executada no âmbito de um determinado caso de negócio. Os casos de negócio à muito que são adoptados no desenho de processos de negócio bem como no desenho de software. Para obtermos todos os casos de negócio representados num workflow, foi introduzido o conceito de tabela de casos de negócio (TCN). Esta tabela é uma ferramenta que permite capturar e descrever os casos de negócio de forma simples, mas poderosa.

Cada caso de negócio corresponde a uma entrada na TCN e estabelece as tarefas de workflow executadas com base no valor de variáveis de negócio. Variáveis de negócio são variáveis que influenciam o encaminhamento ou controlo do fluxo num workflow. As variáveis de negócio são identificadas durante a fase participativa em que são realizadas as entrevistas. Cada variável de negócio tem um domínio, também identificado durante a fase participativa. O domínio identifica os valores que a variável de negócio pode tomar. Por exemplo, num workflow que suporte os processos de negócio de um banco, a variável de negócio Valor de Empréstimo determina se o pedido de empréstimo será aceite ou rejeitado. Se a variável tiver um valor superior a 500 000 € então o empréstimo é rejeitado e a tarefa *'rejeitar'* é executada, caso contrário a tarefa *'aceitar'* é executada.

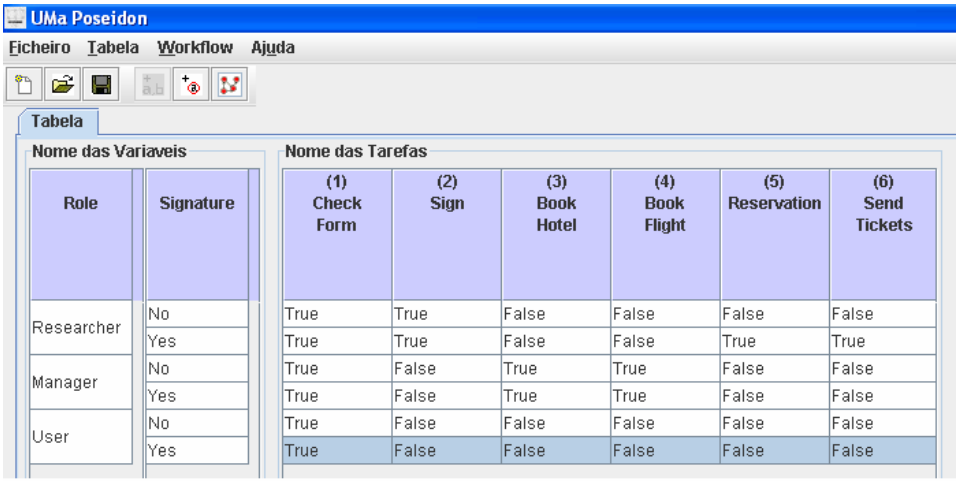
3.1.1 Esquema da tabela de casos de negócio

Uma tabela de casos de negócio é baseada numa tabela de duas dimensões. A estrutura da tabela é a seguinte. As colunas são divididas em duas classes. A primeira classe agrupa um conjunto de variáveis de negócio (vn_j), enquanto a segunda classe é constituída pelas tarefas (t_i) que fazem parte do workflow em questão. Cada célula na tabela relaciona o valor das variáveis de negócio com as tarefas. Em cada uma destas células é guardada a informação sobre se a tarefa será executada ou não (utilizamos os valores *true* e *false* para indicar se uma tarefa é executada ou não).

As primeiras células de cada linha, que pertencem às colunas da primeira classe, contêm os valores que podem ser atribuídos as variáveis de negócio. As células correspondentes às colunas da segunda classe contêm informação que indica se uma tarefa em particular deverá ser executada ou não em runtime. A ideia é estabelecer se uma dada tarefa será executada em runtime com base nos valores referentes a um determinado conjunto de variáveis. De maneira mais formal, o que pretendemos é obter para cada tarefa t_i, a seguinte função onde vn_i é uma variável de negócio:

$$\text{escalonamento}\ (t_i, vn_{1,} vn_{2,} \ldots, vn_j) \in \{\text{True, False}\} \qquad \textbf{(1)}$$

Uma célula pertencente às colunas da segunda classe poderá conter o símbolo de execução (i.e., True) ou o símbolo de não execução (i.e., False). A Figura 2 ilustra uma TCN preenchida.



Role	Signature	(1) Check Form	(2) Sign	(3) Book Hotel	(4) Book Flight	(5) Reservation	(6) Send Tickets
Researcher	No	True	True	False	False	False	False
	Yes	True	True	False	False	True	True
Manager	No	True	False	True	True	False	False
	Yes	True	False	True	True	False	False
User	No	True	False	False	False	False	False
	Yes	True	False	False	False	False	False

Fig. 2. Exemplo de uma TCN preenchida

3.1.2 Construção da tabela de casos de negócio

Perceber o esquema de uma TCN é relativamente simples, no entanto a sua construção é um pouco mais complexa. A metodologia (descrita em (Cardoso 2005)) para construir e preencher a tabela com casos de negócio é um processo iterativo e interactivo, baseado nas entrevistas realizadas.

No primeiro passo, é necessário aferir se as tarefas presentes na tabela são sempre executadas com base nos valores das variáveis de negócio. Se a tarefa é sempre exe-

cutada, esta recebe o símbolo *true*; se nunca é executada recebe o símbolo *false* (ver a figura 2). Caso contrário existe um conflito de símbolos. Os conflitos de símbolos indicam que a realização de um conjunto de tarefas depende de uma ou mais variáveis de negócio. O conflito verifica-se quando os dois símbolos possíveis (i.e., *true* e *false*) foram atribuídos a uma mesma célula da tabela de casos de negócio. Para resolver este conflito, o analista deverá identificar pelo menos uma variável de negócio que controla a realização da tarefa que se encontra em conflito. Depois dessa variável ter sido identificada é necessário realizar os seguintes passos:

1. Deverá ser adicionada uma coluna ao lado esquerdo da tabela de casos de negócio e deverão ser adicionadas as respectivas linhas para a coluna inserida.
2. A coluna deverá ter o nome da variável de negócio identificada.
3. Cada linha da tabela deverá ser duplicada *n-1* vezes, onde *n* é o domínio de valores possíveis para a nova variável de negócio introduzida.
4. Para cada célula, correspondente à nova coluna inserida, são atribuídos os valores do domínio da variável de negócio.

Depois da estrutura da tabela ser actualizada com a nova variável, as células referentes às tarefas deverão também ser actualizadas com os símbolos apropriados. Deverão ser realizadas novas entrevistas para que sejam novamente analisadas quais as tarefas que deverão ser realizadas em runtime com base nos valores referentes às variáveis de negócio presentes na tabela.

3.2 Extracção de funções de escalonamento

Na segunda fase, extraímos um conjunto de funções de escalonamento da TCN. Uma função de escalonamento é uma função lógica na qual os parâmetros são variáveis de negócio da TCN. Cada função modela a execução de tarefas em runtime, i.e. para um conjunto de variáveis de negócio, e respectivas asserções, a função indica se uma tarefa ou tarefas estão agendadas para execução ou não. Para extrair um conjunto de funções de escalonamento, é necessário em primeiro lugar mapear a TCN para uma tabela de verdade (truth table). O mapeamento poderá ser alcançado da seguinte forma:

- Para cada variável de negócio, determinar o número mínimo de bits *nmb* necessários para representar a variável. Representar cada bit com uma variável binária distinta (por exemplo, ‘a’, ‘b’, ‘c’, ...). Por exemplo, a variável Signature da figura 2 tem um domínio com apenas dois valores (“Yes” e “No”), assim apenas um bit é necessário para representar a variável. A variável Role tem um domínio com três valores distintos, (“Researcher”, “Manager” e “User”), e assim sendo são necessários dois bits para representar a variável de negocio;
- Criar um mapeamento entre o valor de cada variável e um número binário, começado por ‘0’. Cada valor da variável de negócio tem bits *nnb* e pode ser representado por uma sequência de variáveis binárias, por exemplo, ‘ab’ ou ‘¬ab’ (o símbolo ¬ indica a negação). Por exemplo, os valores do domínio da variável Signature, “Yes” e ”No”, podem ser ‘0’ e ‘1’, respectivamente. Os valores do domínio

da variável Role, “Researcher”, “Manager” e “User” podem ser ‘00’, ‘01’ e ‘10’, respectivamente;

- Mapear os símbolos *True* ou *False* para o domínio lógico {0, 1}. O símbolo *False* é mapeado para o ‘0’ e o *True* para o 1;
- Criar uma nova tabela utilizando os dois mapeamentos descritos anteriormente.

Uma vez realizado o mapeamento, é possível extrair as funções de escalonamento da tabela de verdade. As funções extraídas são disjunções lógicas de conjunções de variáveis de negócio. Dois métodos podem ser utilizados para gerar as funções: mapas Karnaugh (Karnaugh 1953) e o método Quine-McCluskey (McCluskey 1956).

Na nossa implementação usamos o método Quine-McClusKey por ser um método particularmente útil aquando da extracção de funções de escalonamento com um grande número de variáveis de negócio. Adicionalmente, programas informáticos que implementam este algoritmo foram desenvolvidos. A utilização desta técnica aumenta o grau de automação da metodologia, sendo este um dos objectivos iniciais definidos.

A Tabela 2 mostra uma tabela de funções de escalonamento construída com base na informação presente numa TCN. A tabela é composta de três variáveis binárias: ‘a’, ‘b’ e ‘c’.

Variáveis	Tarefas	Funções de escalonamento
a,b,c	Check Form	w(a, b, c) = true
	Sign	y(a, b, c) = (¬a∧¬b) ∨ (¬a∧b∧c)
	Sign(1)	y1(a, b, c) = ¬a∧¬b
	Sign(2)	y2(a, b, c) = ¬a∧b∧c
	Notify	z(a, b, c) = (¬a∧¬b) ∨ (¬a∧b∧c) ∨ (a∧¬b∧c)
	Notify_m	z1(a, b, c) = ¬a∧¬b
	Notify_u	z2(a, b, c) = ¬a∧b∧c
	Notify_c	z3(a, b, c) = a∧¬b∧c

Tabela 1. Tabela de escalonamento construída a partir de uma tabela de casos de negócio

Quando as funções de escalonamento são disjunções de conjunções, tarefas sinónimo precisam de ser criadas. As tarefas sinónimas têm exactamente o mesmo comportamento e semântica, apenas diferem no nome. O número de disjunções na função de execução indica o número de sinónimos a serem criados para cada tarefa. Cada uma das disjunções é associada a uma tarefa sinónimo. Por exemplo, a tarefa Notify tem a função de escalonamento (¬a ∧ ¬b) ∨ (¬a ∧ b ∧ c) ∨ (a ∧ ¬b ∧ c), por isso três tarefas sinónimo são criadas: Notify_m, Notify_u e Notify_c. As funções de escalonamento são decompostas nos termos “¬a ∧ ¬b”, “¬a ∧ b ∧ c”, e “a ∧ ¬b ∧ c”, e cada termo é associado a uma das tarefas sinónimo.

3.3 Identificar Estruturas de Blocos Básicas

Os sistemas de gestão de processos de negócio são centrados nos workflows, tendo uma atenção especial na gestão da lógica do fluxo. A grande maioria das linguagens de workflow são capazes de modelar blocos sequenciais, paralelos e condicionais que

são representados com estruturas standards tais como and-split, and-join, xor-split e xor-join (Aalst, Barros et al. 2000). As actividades associadas aos blocos sequenciais e paralelos são executados de uma maneira determinística, enquanto que os blocos condicionais são exemplos de uma orientação não-determinística. Os blocos condicionais indicam que o escalonamento das actividades depende da avaliação de uma condição Booleana.

A terceira fase da nossa metodologia consiste na utilização das funções de escalonamento da fase anterior para identificar os blocos sequenciais, paralelos e condicionais que irão constituir o workflow em desenvolvimento. Esta fase é composta por duas grandes etapas:

- Identificar os blocos paralelos e sequenciais associados a um workflow;
- Organizar estes blocos utilizando as estruturas condicionais.

3.3.1 Identificar Estruturas Sequenciais e Estruturas Paralelas

O objectivo da primeira etapa é identificar estruturas paralelas e estruturas sequenciais, e definir uma ordem parcial para as tarefas associadas com estas estruturas. Para completar este ponto, as seguintes passos são realizadas:

1. Criar um conjunto S de conjuntos s_i, onde cada conjunto s_i contém todas as tarefas com a mesma função de escalonamento,
2. Identificar e marcar cada conjunto s_i com a sua função de escalonamento,
3. Para cada conjunto s_i, estabelecer os blocos paralelos e sequenciais existentes, e estabelecer uma ordem parcial para as actividades.

No primeiro passo, produzimos um conjunto S de conjuntos de escalonamento s_i, onde cada conjunto s_i contém todas as tarefas com a mesma função de escalonamento. A ideia é criar um conjunto de tarefas com a seguinte propriedade: se uma actividade do conjunto s_i é escalonada em runtime, então todas as tarefas em s_i são também escalonadas. O segundo passo associa cada conjunto com um rótulo de função de escalonamento. Finalmente, o último passo estabelece os blocos sequenciais e os blocos paralelos e define uma ordem parcial para cada conjunto s_i. Cada conjunto s_i pode ser organizado utilizando blocos sequencias e/ou paralelos.

A Figura 2 mostra como a aplicação Poseidon estrutura os conjuntos s_i de tarefas. Cada conjunto está rotulado com uma função de escalonamento no canto superior esquerdo e cada conjunto contém tarefas com a mesma função de escalonamento. Para cada conjunto s_i as suas tarefas foram organizadas de forma a estabelecer os blocos paralelos e sequenciais existentes.

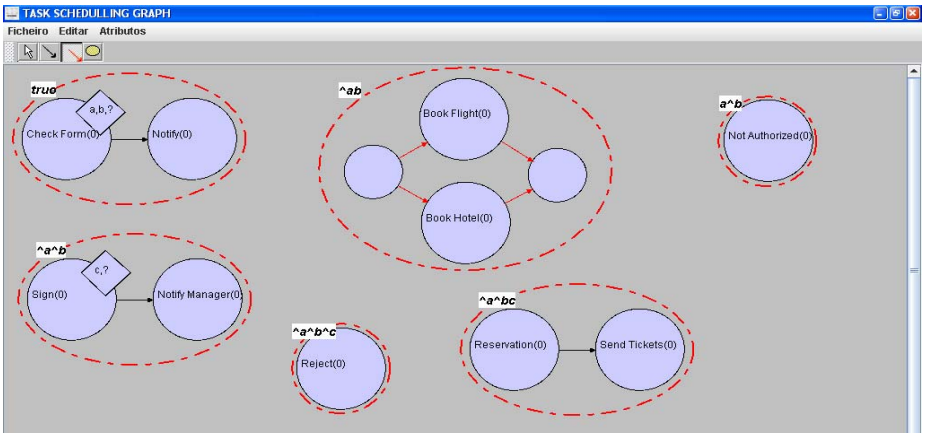


Fig. 3. Blocos paralelos e sequenciais definidos para os conjunto si criados a partir das funções de escalonamento

Estruturas condicionais não podem ocorrer para os conjuntos s_i porque o não-determinismo já foi capturado pelas funções de escalonamento.

A identificação dos blocos paralelos e sequenciais pode requerer o uso de actividades *null* (também conhecidas por actividades *dummy*). Uma actividade *null* não tem uma realização. Estas actividades podem ser utilizadas para modificar o workflow de forma a obter propriedades estruturais (e.g. bem estruturado) ou para possibilitar a modelação de procedimentos específicos de um processo de negócio.

Os dois primeiros passos podem ser automatizados, enquanto que o terceiro passo requer intervenção humana. Não obstante, acreditamos que o último passo pode ser parcialmente automatizado. Uma possível aproximação pode ser a análise das dependências dos dados e da informação entre tarefas. A dependência de dados existe entre duas tarefas se os dados de entrada de uma tarefa dependem da saída dos dados de outra tarefa. A dependência de informação existe entre duas tarefas se o conteúdo ou apresentação de uma tarefa segue o conteúdo lógico de outra tarefa. Por exemplo, vamos considerar que uma sequência de tarefas é utilizada para mostrar um contrato de negócio ao utilizador. Tendo sido identificado que diversas secções do documento necessitavam de ser aceites individualmente, foi decidido fragmentar o documento em partes. Cada parte foi associada a uma tarefa requerendo intervenção humana. Neste caso, existe uma dependência de informação entre as tarefas, desde que as tarefas porque necessitem de ser ordenadas de maneira a que o contrato seja lido em sequência lógica seguindo o documento original.

3.3.2 Identificar Estruturas Condicionais

No ponto anterior já foram identificados os blocos paralelos e os blocos sequenciais. O passo seguinte é identificar as blocos condicionais (formados por xor-splits) com base nos conjuntos s_i. O objectivo é identificar os blocos condicionais de um workflow e determinar como controlam e organizam os conjuntos anteriormente identificados (i.e. blocos paralelos e sequenciais).

Para identificar os blocos condicionais utilizamos o algoritmo de Identificação de Bloco Condicional (CBI – Conditional Block Identification) (Cardoso 2005). O algoritmo CBI pode ser visto como uma metodologia iterativs, com uma envolvente humana, para estruturar conjuntos s_i de um workflow. Este algoritmo recorre a um conjunto de assumpções e regras que são utilizados para estruturar os conjuntos de escalonamento num workflow. Depois de aplicar o algoritmo CBI, os conjuntos s_i apresentam dependências entre si representadas por transições como é mostrado na Figura 4.

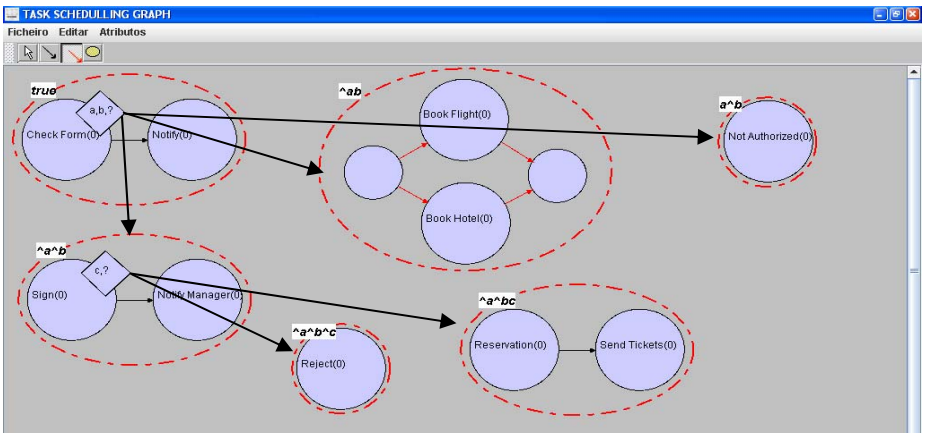


Fig. 4. Workflow com transições automaticamente criadas pelo algoritmo CBI

Contudo, vários elementos do workflow estão em falta. É aparente no exemplo que o workflow não inclui nenhuma junção dos xor-splits e o workfow tem vários pontos de saída. Ambos os problemas podem ser resolvidos pelo acerto dos xor-splits com xor-joins. Aalst (Aalst 2000) indicou a importância do equilíbrio do xor/and-splits e xor/and-joins para obter o que é chamado de "bom" workflow. Por exemplo, dois fluxos condicionais criados por um xor-split, não devem ser sincronizados por um and-join, mas por um xor-join. Equilibrar xor/and-splits pode requerer o uso de actividades *null* or *dummy*.

3.4 Limpeza, Análise e Implementação do Workflow

Na última fase, são excluídas as actividades *null* (*dummy*) e, caso necessário, o workflow poderá ser reestruturado ou modificado por motivos de clareza. Uma vez finalizadas as fases de limpeza e análise, o processo de desenho está pronto a ser implementado. O método WIDE proposto em (Casati, Fugini et al. 2002) poderá ser utilizado para esse fim. O método foca em aspectos mais técnicos e inclui a selecção do sistema de gestão de workflows alvo e o mapeamento dos workflows descrevendo um processo de negócio de alto nível na implementação do workflow. A Figura 5 ilustra a integração da metodologia Poseidon com a metodologia WISE.

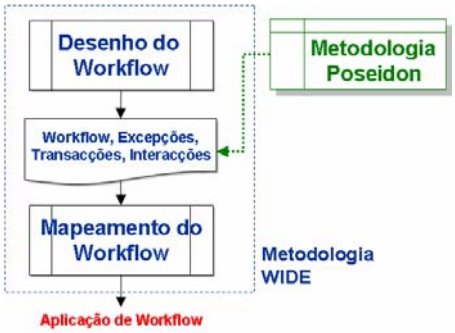


Fig. 5. Integração da metodologia Poseidon com a metodologia WISE

Na fase de desenho do workflow, a metodologia WIDE fornece conceitos que permitem implementar o workflow resultante em diferentes sistemas de gestão de workflow, tendo em conta as diferentes características dos sistemas. No fim da fase de desenho o seguinte resultado é obtido: um esquema do workflow, a especificação das excepções e transacções, e especificação das interacções com aplicações externas. A fase de desenho do workflow da metodologia WIDE pode ser substituída ou complementada pela metodologia Poseidon. A metodologia Poseidon é mais poderosa pois permite a o desenho semi-automático dos esquemas de workflows.

4 Implementação

Foram consideradas várias linguagens de programação durante a fase de análise da aplicação Poseidon. O MS Visual Basic, linguagem C e Java. Mas desde cedo tornou-se evidente que a implementação multiplataforma e a portabilidade do código eram aspectos que se revestiam de grande importância tendo influenciado a nossa escolha pela selecção da linguagem Java. Para o desenho gráfico dos workflows consideramos inicialmente usar os componentes Swing que a linguagem Java oferece ou usar uma ferramenta direccionada para a implementação rápida e prática de aplicações que manipulem grafos. A implementação da aplicação com o uso dos componentes Swing iria revelar-se trabalhosa e demorada, uma vez que todas funcionalidades associadas com o desenho de workflows teriam de ser programadas. Tendo em conta que os requisitos da aplicação Poseidon eram relativamente standards do ponto de vista da interface gráfica, optamos por analisar as ferramentas JGraph (www.jgraph.com) e JHotDraw (http://www.jhotdraw.org/). Finalmente, seleccionamos o JHotDraw pois a sua arquitectura foi desenvolvida com base num conjunto conhecido de *patterns* que forneciam soluções arquitectónicas para as funcionalidade que necessitávamos para a aplicação Poseidon.

4 Conclusões

Apesar da investigação realizada no sentido de evoluir os sistemas de gestão de workflow, o trabalho realizado sobre ferramentas, metodologias e métodos para suportar a fase de desenho dos workflows é praticamente inexistente.

O desenvolvimento de métodos e ferramentas adequadas é de grande importância para garantir que os workflows sejam construídos de acordo com as especificações iniciais. Infelizmente, é reconhecido que apesar da difusão dos sistemas de gestão de workflow, as ferramentas para suportar o desenho de workflows continuam a ser uma lacuna. Neste artigo, é descrito uma aplicação, chamada Poseidon, para assistir os analistas de processos durante as entrevistas com o pessoal administrativo, gestores e empregados em geral, de forma a possibilitar o desenho semi-automático de workflows.

A aplicação Poseidon apresentada foi utilizada com sucesso para desenhar workflows administrativos de média dimensão numa empresa do sector aeronáutico. Existe a convicção de que a aplicação é igualmente apropriada para desenhar workflows de grande escala, bem como o facto da aplicação representar um passo em frente na modelação de processos de negócio.

Referências

Aalst, W. M. P. v. d. (1998). "The Application of Petri Nets to Workflow Management." The Journal of Circuits, Systems and Computers **8**(1): 21-66.

Aalst, W. M. P. v. d. (2000). Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques. Business Process Management: Models, Techniques, and Empirical Studies. W. M. P. v. d. Aalst, J. Desel and A. Oberweis. Berlin, Springer-Verlag. **1806:** 161-183.

Aalst, W. M. P. v. d., A. P. Barros, et al. (2000). Advanced Workflow Patterns. Seventh IFCIS International Conference on Cooperative Information Systems, September 2000.Vol**:** pp. 18-29,

Aalst, W. M. P. v. d. and A. H. M. t. Hofstede (2003). YAWL: Yet Another Workflow Language (Revised Version). Brisbane, Queensland University of Technology2003.

ARIS (2005). ARIS Design Platform. http://www.ids-scheer.com/.

BPML (2004). Business Process Modeling Language. **2004**.

BPMN (2005). Business Process Modeling Notation - http://www.bpmn.org/.

Cardoso, J. (2005). "Poseidon: A framework to assist Web process design based on business cases." International Journal of Cooperative Information Systems (IJCIS)(accepted for publication).

Cardoso, J., J. Miller, et al. (2004). "Modeling Quality of Service for workflows and web service processes." Web Semantics: Science, Services and Agents on the World Wide Web Journal **1**(3): 281-308.

Cardoso, J. and A. Sheth (2003). "Semantic e-Workflow Composition." Journal of Intelligent Information Systems (JIIS). **21**(3): 191-225.

Casati, F., M. Fugini, et al. (2002). "WIRES: a Methodology for Designing Workflow Applications." Requirements Engineering Journal **7**(2): 73-106.

COSA (2005). COSA Workflow - http://www.cosa-bpm.com/.

Karnaugh, M. (1953). "The Map Method for Synthesis of Combinational Logic Circuits." Transaction IEEE **72**(9): 593-599.

Kochut, K. J. (1999). METEOR Model version 3. Athens, GA, Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia.

McCluskey, E. J. (1956). "Minimization of Boolean functions." Bell System Technical Journal **35**(5): 1417-1444.

McCready, S. (1992). There is more than one kind of workflow software. Computerworld. **November 2:** 86-90.

Miller, J., A. Sheth, et al. (1996). "CORBA-based Run Time Architectures for Workflow Management Systems." Journal of Database Management, Special Issue on Multidatases **7**(1): 16-27.

Sheth, A., D. Georgakopoulos, et al. (1996). Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Athens, GA, Deptartment of Computer Science, University of Georgia.

Sheth, A. P., W. v. d. Aalst, et al. (1999). "Processes Driving the Networked Economy." IEEE Concurrency **7**(3): 18-31.

TIBCO (2002). TIBCO InConcert, TIBCO.

WS-BEPL (2005). Business Process Execution Language for Web Services, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.

Solving Combinatorial Problems: An XML-based Software Development Infrastructure *

Rui Barbosa Martins[1,2], Maria Antónia Carravilla[1,2], Cristina Ribeiro[1,2]

[1] FEUP—Faculdade de Engenharia da Universidade do Porto
[2] INESC—Porto
Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
{ei01018,mac,mcr}@fe.up.pt

Abstract. The resolution of combinatorial problems typically requires the articulation of tools that range from modeling languages to dedicated solvers, including processing input data sets and visualizing results.
This work concerns the improvement of the software development environment for a research project using a custom-designed XML dialect. NestingXML has been designed to capture one kind of combinatorial problems in what concerns their input and output data. The dialect is used for storing problem and solution descriptions in a flexible way. In a project context, data formatted according to the dialect are imported into a Java API used for developing solvers and associated tools. The problem description is enriched with both preprocessing data and solution descriptions.
We describe the NestingXML dialect and the Java API used in the project and illustrate their use in the problem-solving process. The resulting environment demonstrates increased flexibility in data representations and will become an easy integration medium for new team members.

Keywords: XML, Cutting and Packing, Constraint Programming.

1 Introduction

The resolution of combinatorial problems typically requires the articulation of tools that range from modeling languages to dedicated solvers, and also include the processing of input data sets and the visualization of results.

A research team involved in developing solutions for a class of combinatorial problems faces the significant overhead of maintaining consistent representations for problem instances and solution descriptions that can be shared among researchers who are working on different approaches.

The GLOBALNest project (Global Constraints for Nesting Problems) is focused on developing constraint programming solutions for nesting problems,

* Supported by FCT under project POSI/SRI/45379/2002 (GLOBALNest)

a category of cutting and packing problems where polygon-shaped pieces are placed over a board and the goal is to minimize the length of the resulting layout. The development and test of new solution methods requires the use of known problem instances and the execution of test suits under different program configurations.

In the context of the project, data for the problem instances is available in the form of text files. The resolution methods make use of previously defined modules for geometric manipulations; *ad hoc* formats have been specified for their output. In the course of some of the project developments, extra information has been required for some of the modules and the data formats were modified accordingly. It soon became clear that extending and modifying the data format would be the rule in projects of this nature, and that a more flexible representation was needed.

An XML data format has been defined covering the expected complexity of the descriptions for this class of problems and their solutions. The existing tools for parsing input data and presenting results have been kept as "legacy data" transformers and new tools are being developed for manipulating this more expressive data format.

The project development platform has a Java API where the concepts of the problem domain are represented. Several tools that deal with the geometry and some output format generators were developed based on the API.

The paper describes an infrastructure for software development and solver evaluation comprising the XML data format (NestingXML), the Java API and the set of tools that support the preparation of inputs for the solvers and the evaluation of results.

The paper is organized as follows. Section 2 briefly describes the project development environment. The formats used for data input and output are detailed in Section 3. The Java API used in the project to capture the main concepts of the problem domain is presented in Section 4. The XML data format developed for this domain is the subject of Section 5, followed by the update of the API in Section 6. Section 7 illustrates the use of the dialect and the API and some conclusions follow.

2 Combinatorial Problems and Solutions

Combinatorial problems typically require modeling a problem domain, characterizing feasible solutions and using a search strategy to explore the solution space and find solutions that best satisfy a problem goal.

Research in combinatorial problems involves developing new representations for the problem domains, to ease the expression of the constraints that define feasibility, and devising new search strategies that lead to the exploration of either larger or more promising regions of the search space.

In the case of cutting and packing problems, the goal is to minimize waste when packing items in a container or cutting out small pieces from a surface of

raw material. Fig. 1illustrates a solution layout, with several polygonal pieces positioned on a plate.

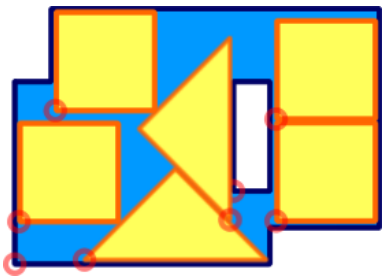


Fig. 1. A solution for a nesting problem

The solution of a nesting problem is the choice of a position for each one of a set of polygon-shaped pieces over a plate—a layout. It is an intrinsically 2D problem where the central constraints must express the fact that any 2 pieces may not overlap. The problem is being addressed in the Operations Research community with Mixed Integer Programming and heuristic methods.

To represent "non-overlappedness" in 2D requires some geometrical preprocessing of the input. Moreover, any development aimed at improving solution quality tends to require the computation of new values to characterize either the input or some intermediate solution configuration.

The testbed for the work described is the GLOBALNest project (Global Constraints for Nesting Problems), a research project where the main concern is the development of specialized constraints to handle the solution of these problems.

In a typical solution process workflow the problem description is read from a file into a memory representation used by the solver software. When modules in several languages are used, intermediate files may be required to pass-on information. If preprocessing of the input data is required, as is the case with geometric problems, then intermediate steps add to the characterization of the input data.

In a test setup, several input sets must be run through selected solvers in order to compare solutions. It is interesting in this case to be able to configure complete tests, assemble input data and collect results.

The problems tackled so far by the research team consider the board as a rectangular stripe whose length is to be minimized. The first versions of the solver were implemented using the SICStus-Prolog CLP-FD module [1]. In [2] and in [3], a CLP approach to the resolution of nesting problems is presented, using reified constraints and pieces represented by respectively convex and non-convex polygons. In the following step a special global constraint named "outside" [4] was developed to handle specifically the non-overlapping of a pair of general

polygons, leading to improvement of the domain reduction during search. During this period the Web application "CortaBem" [5] was used as a tool to draw the pieces, build the input files for the solver and draw the layouts.

In a more recent stage of the work, a solver using the global constraint "outside" has been implemented in ILOG with the purpose of testing another type of tool that would allow the use of a hybrid approach, combining CP techniques with Mixed Linear Programming.

3 Data Formats

The solution process for nesting problems is therefore a workflow with various intermediate files and solver modules. The way test examples are stored, can be manipulated, extended and put to some use may improve or strongly reduce researcher's productivity.

Initially the problem descriptions were stored in *ad hoc* file formats. Those formats were conceived with the purpose of making it easy for researchers to parse the data and feed them into their solver and there was no concern with the extensibility of the format. But today's ideas and data requirements for a solver may not be tomorrow's. During research different approaches are tried in order to get better solutions and the need to enrich previous formats arises. The lack of an easy way to extend format contents usually leads either to complicating even more an existing format by hacking in new required data, or to the creation of one more file format.

In our case study there are up to seven different formats taking an active role for each solver iteration:

1. *jfo* – contains a description of the pieces and their quantities
2. *brd* – contains board information
3. *prb* – contains problem information
4. *dat* – contains polygon descriptions
5. *nfp* – contains intermediate polygons built from the polygons on the *dat* format
6. *si* – contains input data to be directly read by the solver
7. *so* – contains nesting problem solutions

Some formats (*dat* and *nfp*) store intermediate results, are active for very short periods and do not get stored after the solution process is finished. All the other formats must be stored as each one contains unique information about the problem or the solution.

In Listing 1 we can see some of the content of an example *dat* file. The language used in some of the fields is portuguese, making it difficult to share the format among the research community. One thing common to all formats is the not so intuitive way data is stored. The file begins with the number of pieces to be described (7). It then proceeds enumerating each of the polygons assigning them a string identifier (e.g., *PECA 1*, defining the pieces quantity in the lot,

the number of vertices (e.g., 5) and their enumeration. Each vertex is described by its X and Y coordinates.

To understand the format there is a need for additional information, such as the one given above. Without that information, the meaning of the format will be lost and the format itself becomes useless.

```
NUMERO  DE       PECAS
        7

PECA             1  QUANTIDADE
        5
NUMERO  DE       VERTICES
        6
VERTICES(X,Y)
        0.0        0.0        2
        2.0        -1.0        2
        4.0        0.0        2
        4.0        3.0        2
        2.0        4.0        2
        0.0        3.0        2

(  ...  )
```

Listing 1: *dat* file content

Coping with all the scattered information of these files is not an easy task and it tends do get harder as new problem approaches are explored.

4 The Java API

To ease the manipulation of the formats described in the previous section, we chose to create a Java Application Programming Interface (API). The API provides an easy way to read each format into Java data structures, and to write the files back with added content.

These basic functionalities provided by the API allowed us to build a set of tools to manipulate the nesting problem's data. The first tools were built to compute some geometric properties of the problem's pieces. Some of these properties could then be used to generate sorted sequences of pieces to be positioned. An example is a sequence of pieces sorted by ascending convex hull area [6].

Graphic tools have also been created on top of the API. They provide an easy way to visualize both the problem's lot and its solutions. The renderings are then stored to jpeg files, allowing the representation of a problem and its solution to be easily rendered in a web browser.

5 NestingXML

The ideas for a XML format that could fully describe a nesting problem and its solutions first came about early 2005. In order to develop a format that would be used by all the researchers dealing with similar problems, the format was first developed in-house and then fully accepted by the European Research Group on Cutting and Packing Problems [7]. We currently have a stable first version of the XML format, for which an XML Schema is available [8]. NestingXML accomplishes the following objectives:

- Ability to represent both very simple and very complex problem instances in a common format.
- Ability to store both the nesting problem, its solutions and intermediate computations in a self contained file.
- A clear, human readable format.
- An easily extensible format (due to the use of XML).

The NestingXML format is structured in five sections:

1. Information about the author and a brief description of the problem.
2. Description of the problem instance.
3. Heuristics, intermediate computations and extra information.
4. Geometrical description of all the polygons used in the problem.
5. Solutions of the problem.

The first section includes some metadata for the problem described in the file, such as name and contacts of the author. A brief and informal description of the problem may also be added to this section. In this description, the author can point out interesting features of the problem that couldn't be easily spotted out by its formal description. A simple example of this first section can be seen in List. 2.

```
<name>Nesting problem example</name>
<author>Rui Martins</author>
<date>2005/11/04</date>
<description>
    A brief informal description about the nesting problem ↩
should be placed here.
</description>
<verticesOrientation>clockwise</verticesOrientation>
<coordinatesOrigin>up-left</coordinatesOrigin>
```

Listing 2: Example of the first section of a NestingXML file

The formal description of the nesting problem is presented in the second section of the NestingXML format. The section begins with the description of

the boards, which represent simple or complex geometrical 2D figures where the pieces are positioned. A board can be described by several polygons, representing materials of different qualities or holes, e.g., a simple way of describing a window frame could be by defining two superimposed rectangles of different sizes, where the smaller one is a hole. The description of the pieces to be positioned follows. Pieces and boards use similar structures, since both represent geometrical shapes. A small example of this section is present in List. 3. Fig. 2 has the corresponding graphical representation.

```
    <problem>
        <boards>
            <piece id="board1" quantity="1">
                <component type="0" idPolygon="poly0" ↔
xOffset="0.0" yOffset="0.0" />
                <component type="-1" idPolygon="hole0" ↔
xOffset="6.0" yOffset="-3.0" />
            </piece>
        </boards>
        <lot>
            <piece id="piece1" quantity="1">
                ...
            </piece>
            ...
        </lot>
    </problem>
```

Listing 3: Example of the second section of a NestingXML file

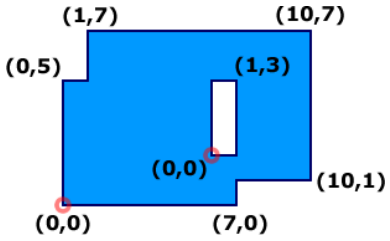


Fig. 2. Graphical representation of the board described in List. 3

The third section is meant for researchers to store extra information on the problem, such as time-consuming intermediate computations and information on heuristics. Listing 4 has a short example of the XML describing *Nofit Polygons*. Nofit polygons are a geometric transformation that makes the requirement

of "non-overlappedness" of two polygons easier to handle. Overlapping of two polygons can be decided based on the relative position of a polygon (the nofit) and a point. The visual representation of the computation of the nofit polygon can be seen in Fig. 3, where polygon R is the result of the orbit movement of polygon O around static polygon S.

```
<nfps>
    <nfp>
        <staticPolygon idPolygon="poly1" angle="0.0" ↩
mirror="none" />
        <orbitingPolygon  idPolygon="poly1" angle="0.0" ↩
mirror="none" />
        <resultingPolygon idPolygon="nfp_p1_p1" />
    </nfp>
     ...
</nfps>
```

Listing 4: Example of the third section of a NestingXML file

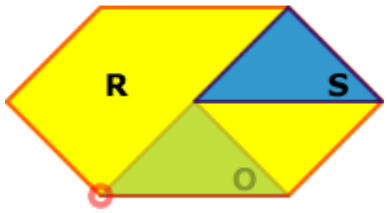


Fig. 3. Graphical representation of the No Fit Polygon information described in List. 4

Section four of the NestingXML format takes advantage of the fact that nesting problems use several geometrical data. To reduce the verbosity and avoid duplication of information in the file, this section describes polygon line segments and other relevant geometrical information. Each polygon represented here has a unique id, enabling it to be used as components of the boards, pieces and other geometrical structures. Consider for example a piece to be a square of 2×2 and a board with a 2×2, both of them could use a 2×2 square polygon described in this section avoiding the duplication of information. A small example containing the description of a triangle can be seen in List. 5. Going back to Fig. 3 one can see two triangles which correspond to the polygon described here.

The fifth and final section of the NestingXML format is reserved to describe the solutions of the problem, which are represented by the X and Y coordinates of

```
<polygons>
    <polygon id="poly1" nVertices="3">
        <lines>
            <segment n="1" x0="0.0" y0="0.0" x1="1.0" ↩
y1="1.0" />
            <segment n="2" x0="1.0" y0="1.0" x1="2.0" ↩
y1="0.0" />
            <segment n="3" x0="2.0" y0="0.0" x1="0.0" ↩
y1="0.0" />
        </lines>
        <xMin>0.0</xMin>
        <xMax>2.0</xMax>
        <yMin>0.0</yMin>
        <yMax>1.0</yMax>
        <perimeter>4.8284</perimeter>
        <area>1</area>
    </polygon>
</polygons>
```

Listing 5: Example of the fourth section of a NestingXML file

the positioning points (placement vertex) of the pieces. An example of a solution is presented in List. 6.

For the time being we chose not to create a specific element to store a numeric value representing the "solution's quality" because there is no clear way of generalizing this. Different approaches tend to use different measures of quality. There is room here for future extensions of NestingXML.

```
<solutions>
    <solution>
        <placement idBoard="board1" idPiece="piece1" x="0.0" ↩
y="0.0" angle="0.0" mirror="none" />
        <placement idBoard="board1" idPiece="piece2" x="1.0" ↩
y="1.0" angle="0.0" mirror="none" />
        <placement idBoard="board1" idPiece="piece2" x="4.0" ↩
y="0.0" angle="0.0" mirror="none" />
        <usagePercentage>15</usagePercentage>
    </solution>
    ...
</solutions>
```

Listing 6: Example of the fifth section of a NestingXML file

6 An extended API

As with the older file formats described in section 3, we have also developed a Java API for the NestingXML format. Because we intend to maintain backward compatibility, some code from the previous API has been integrated to enable an easy transformation between formats. The tools developed for the previous API have been enhanced for the new one.

In the future, it may be necessary to add new features to the XML format. To use the new features in tools and solvers, they will have to be available in the API. We expect that such extensions to the API can be smoothly added, as the main concepts of the application domain are already captured in the current API. New features will account for parsing and handling new elements or attributes. This allows a steady evolution of the development environment.

As already stated, a NestingXML file may contain complex information regarding the problem, such as rotation of the pieces and multi-holed boards. While some research teams might be using such information, we are focusing our research on simpler problems. Therefore our API does not account for features we are not currently handling.

7 The Software Development Infrastructure

We are now going to take a look at an example solving process and the way the NestingXML file is transformed. Fig. 4 should be used along the description, as a way of getting a visual picture of the whole process.

In a first step we create a new NestingXML file where the problem is stored, including metadata for author and problem in the *nesting* root tag. To turn our file into valid XML we should define the header describing the version and namespace used for the document.

After creating the file's skeleton, we need to insert the formal description of the nesting problem. This information is inserted right after the information added in the previous step. Because both boards and pieces are sets of polygons, we should also add those to a specific section near the bottom of the NestingXML file. In Fig. 4 we can see arrows coming off this step's preview and pointing to NestingXML sections 2 and 4. Section 2 represents the place where the boards and the pieces are declared in the NestingXML file. Section 4 refers to the polygon description section.

In the third step of the process, we do some intermediate computations which will be used by some of our solvers. These data should also be stored for later usage. The intermediate computations section in the NestingXML file is represented by number 3 in Fig. 4. Note also that section 4 is also pointed by this process because some of the computation generate new polygon information.

The fourth and final step of our example process is the resolution of the generated problem instance. The solver retrieves the problem's data and intermediate computations from the proper location in the input NestingXML file. It appends the computed solution to the last section of the file where other solutions may already be present.

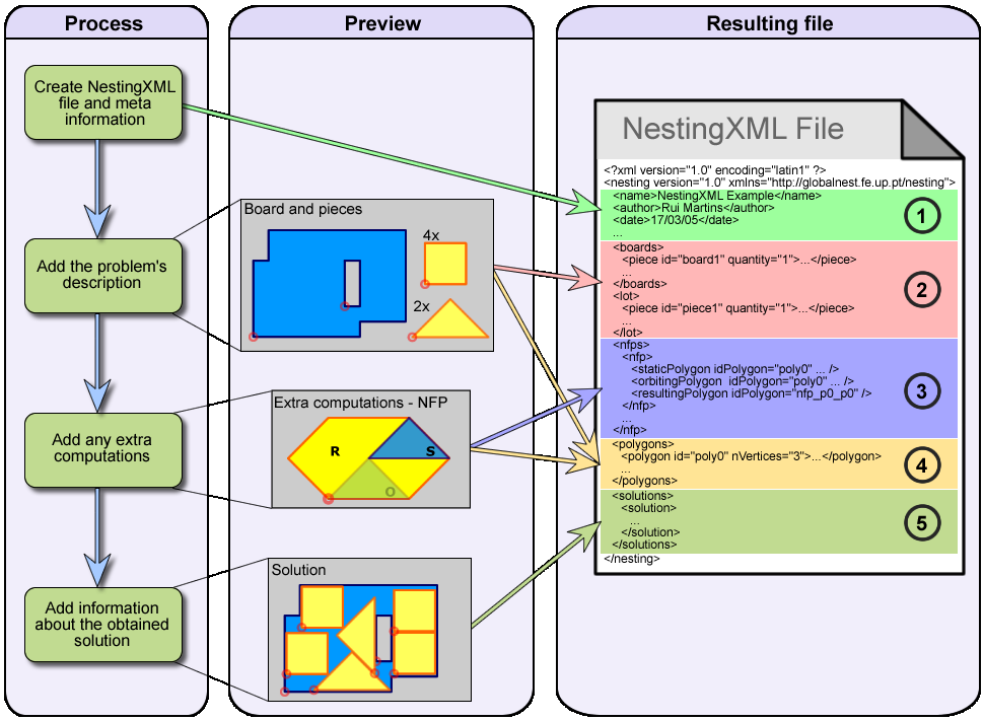


Fig. 4. Process example

8 Conclusions and Ongoing Work

The work reported here concerns part of the infrastructure put together to support a research project on Constraint Programming methods and solutions for a class of combinatorial problems. Lines of research such as this one extend for significant periods of time—the current project builds on results from work done since 1997. This kind of work typically involves the collaboration of several people, the consolidation of project results for building new approaches and the inclusion of new collaborators at a regular pace. In this context a dependable development environment is an important issue.

Researchers working on cutting and packing problems have collected a significant number of benchmark problems, and new ones are proposed when new techniques require illustration. Our project environment has shown that developing new approaches leads to the production of intermediate results that we must organize to be used in the problem-solving process. The CortaBem application [5] has served to test a resolution environment and provide an easy interface for colleagues who would like to get acquainted with our work, but no easy way to interchange problem instances and solutions was available.

The flexible representation of the NestingXML dialect has proven essential for exploring the structure of the problem domain and for avoiding the custom parsing and systematic migration that would otherwise be required.

The process infrastructure illustrated in Section 7 is being further developed in the sense of automating the repeated execution of sequences of operations on the input data to solve a problem. Data representation with NestingXML makes some other useful products simple to define. We are currently exploring the simplification of complex input shapes and the transformation of the output representations into standard graphic formats that can be used for rendering purposes [9].

9 Acknowledgments

We thank our colleagues José Fernando Oliveira and António Miguel Gomes for contributing with their expertise on the problem domain for defining the NestingXML dialect and for many useful discussions. João Paulo Mendes and Hugo Almeida, members of the GLOBALNest team, have contributed the first Java API and were also strongly engaged in the definition of NestingXML.

We thank Helmut Simonis, consultant for the GLOBALNest project, for the emphasis given to the project development environment and for his insights on the goals of the whole line of research.

References

1. Carlsson, M., Ottosson, G., Carlson, B.: An Open-Ended Finite Domain Constraint Solver. In Glaser, H., Hartel, P., Kucken, H., eds.: Programming Languages: Implementations, Logics, and Programming. Volume 1292 of Lecture Notes in Computer Science., Southampton, Springer-Verlag (1997) 191–206
2. Ribeiro, C., Carravilla, M.A., Oliveira, J.F.: Applying constraint logic programming to the resolution of nesting problems. Pesquisa Operacional **19** (1999) 239–247
3. Carravilla, M.A., Ribeiro, C., Oliveira, J.F.: Solving nesting problems with non-convex polygons by constraint logic programming. International Transactions in Operational Research **10** (2003) 651–663
4. Ribeiro, C., Carravilla, M.A.: A global constraint for nesting problems. In Régin, J.C., Rueher, M., eds.: CPAIOR. Volume 3011 of Lecture Notes in Computer Science., Springer (2004) 256–270
5. GLOBALNest: GlobalNest Website (2005) `http://192.168.102.63/~globalnest/`
6. Mendes, J.P., Almeida, H., Ribeiro, C., Carravilla, M.A.: An Evaluation Infrastructure for Nesting Problems. Technical report, INESC-Porto (2005)
7. ESICUP: EURO Special Interest Group on Cutting and Packing (2005) `http://www.apdio.pt/sicup/`
8. GLOBALNest Project Team: NestingXML Schema (2005) Disponível em `http://globalnest.fe.up.pt/nesting/nesting.xsd`
9. Gonçalves, B., Baldaia, N., Cerqueira, N., Martins, R.: NestingXML Visualization and Geometric Manipulation Tools. Technical report, INESC—Porto (2006)

O Documento Único Automóvel

Marta H. Jacinto[1], Jorge Nuno Pereira[2]

[1]ITIJ — Instituto das Tecnologias de Informação na Justiça
Ministério da Justiça
1049-068 Lisboa
marta.jacinto@itij.mj.pt
[2]INCM — Imprensa Nacional Casa da Moeda, S.A.
1000-042 Lisboa
jn@incm.pt

Resumo Em Portugal, cada veículo estava, até 31 de Outubro de 2005, associado a dois documentos: o Livrete e o Registo de Propriedade. Os dados associados a cada um destes documentos estavam ao cuidado de uma entidade diferente que era responsável pela sua emissão, a DGV e o ITIJ (por parte da DGRN).
A legislação comunitária veio trazer o conceito de Documento Único Automóvel.
Para transpor tal legislação para o cenário português, foi necessário o intercâmbio de informação entre as entidades já referidas, para congregar a informação do novo documento utilizando BD em IMS e, posteriormente, a definição do formato para enviar tal informação para uma terceira entidade - a INCM - que procede à emissão dos documentos. O formato escolhido para o envio dos dados para a INCM foi XML pelas vantagens já conhecidas e ainda pela experiência da INCM em trabalhar com esta tecnologia.
Com este artigo pretende-se abordar o processo de produção e tratamento dos dados a disponibilizar em XML, focando as dificuldades encontradas. Aborda-se ainda a disponibilização de parte da informação em XML através de QR-Code directamente impresso no documento.
Palavras-chave:
XML, XML-Schema, XSL, DUA, QR-code

1 Introdução

Como é do conhecimento geral, até há muito pouco tempo atrás, cada veículo necessitava de dois documentos para a circulação: o Livrete e o Registo de Propriedade. Os dados associados ao Livrete, propriedade da Direcção-Geral de Viação (DGV), eram recolhidos e processados por esta entidade em sede própria. Já os associados ao Registo de Propriedade, da responsabilidade das Conservatórias do Registo Automóvel, eram recolhidos por estas conservatórias e processados pelo Instituto das Tecnologias de Informação na Justiça (ITIJ). No caso dos reboques toda a documentação era produzida pela DGV.

Em 1 de Junho de 1999, foi publicada a Directiva Comunitária 1999/37/CE do conselho de 29 de Abril [1], que estabelece que "Os Estados-Membros devem emitir um certificado de matrícula para os veículos sujeitos a matrícula nos termos da legislação nacional" e que "o certificado de matrícula emitido por um Estado-Membro deve ser reconhecido pelos demais Estados-Membros quer para identificação do veículo em circulação internacional quer para nova matrícula noutro Estado-Membro".

Nesta directiva são indicadas as dimensões máximas do certificado, as protecções que devem ser utilizadas contra a falsificação e os vários campos e respectivos códigos comunitários harmonizados (estes últimos permitem que haja independência relativamente à língua utilizada e dispensam a tradução dos descritivos).

A directiva comunitária 2003/127/CE de 23 de Dezembro [2], veio dar nova redacção à enunciada acima.

A transposição para a legislação portuguesa foi feita através do Decreto-Lei n.º 178-A/2005 de 28 de Outubro [3], que "aprova o projecto ≪Documento único automóvel≫, criando o certificado de matrícula, que agrega a informação anteriormente constante do título de registo de propriedade e do livrete do veículo."

Para colocar em prática o DUA, foi necessário um trabalho conjunto de vários ministérios e entidades, nomeadamente Ministério da Justiça, Ministério das Finanças, Ministério da Administração Interna, Unidade de Coordenação da Modernização Administrativa e Ministério da Ciência e do Ensino Superior.

Neste artigo apresenta-se a forma como a informação da DGV e das conservatórias foi reunida e depois descrevem-se as várias questões levantadas no intercâmbio de informação entre o ITIJ e a Imprensa Nacional Casa da Moeda (INCM) e as soluções encontradas.

Na secção 2 expõe-se a recolha dos dados necessários para a produção do DUA. Na secção 3 aborda-se o Documento Único Automóvel na perspectiva técnica. Na secção 4 explica-se o envio dos dados para impressão e na secção 5 a recepção e produção do DUA. Conclui-se o artigo com a secção 6 (Conclusão) e 7 (Agradecimentos).

2 Recolha de dados

Os dados relativos aos veículos estavam compartimentados de forma estanque em duas instituições diferentes: a DGV e as Conservatórias do Registo Automóvel (informação processada pelo ITIJ).

O conceito de Documento Único Automóvel, que exigia congregar informação, trouxe dificuldades significativas de intercâmbio de informação entre sistemas e aplicações completamente díspares como os da DGV e do ITIJ.

O ITIJ foi escolhido como entidade agregadora dos dados. Desta forma foi necessário criar uma base de dados para o DUA, da qual constam os dados de todos os DUAs criados, conforme se verá na secção 4.

Para o exposto acima, foi necessário receber os dados já guardados na DGV relativos a veículos. Essa recepção foi feita por FTP através de *flat files*. A in-

formação constante nestes ficheiros foi depois colocada numa base de dados IMS [4] criada especificamente para o efeito, onde são guardadas as características dos veículos. No caso dos reboques, nenhuma informação é guardada em Base de Dados.

Com o conceito de balcão único também estabelecido pelo Dec-Lei n.º 178-A/2005, o pedido da documentação automóvel passou a poder ser feito só na DGV ou só nas Conservatórias do Registo Automóvel, implicando uma sucessiva actualização das duas bases de dados sediadas no ITIJ (a de características dos veículos agora criada e a de registo automóvel já existente anteriormente no ITIJ) referidas acima com informação da DGV e conservatórias. A informação recolhida nas Conservatórias é introduzida automaticamente na base de dados do Registo Automóvel, como já acontecia, e agora também na Base de Dados de características. A informação resultante da recolha na DGV chega ao ITIJ pela forma já indicada. A figura 1 apresenta esquematicamente o intercâmbio de informação entre estas três entidades.

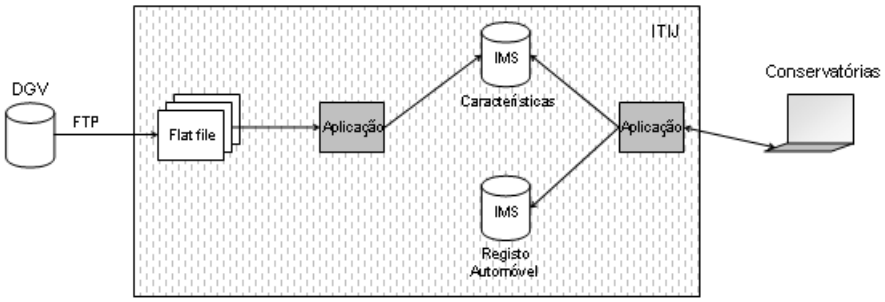


Figura 1. Intercâmbio de informação entre o ITIJ e a DGV e as Conservatórias.

O carregamento da base de dados do Documento Único Automóvel é feito tendo como dados de entrada os dados provenientes da DGV e das conservatórias guardados naquelas duas bases de dados. Simultaneamente é gerado o documento a ser enviado à Imprensa Nacional Casa da Moeda (INCM), como se verá na secção 4.

3 O Documento Único Automóvel

Como se referiu, o ITIJ agrega a informação para a produção do Documento Único Automóvel. No entanto a entidade que imprime o DUA é a Imprensa Nacional Casa da Moeda (INCM). O formato escolhido para o intercâmbio de informação entre o ITIJ e a INCM foi o XML, em parte tendo em conta a experiência da INCM a esse nível.

Assim, o ITIJ, em parceria com a INCM, definiu a gramática para o documento XML a ser transferido entre as duas instituições, o XML-Schema [5,6].

Em virtude de o número de DUAs a gerar por dia ser relativamente elevado – estimado em 6000 DUAs por dia – foi acordado que cada ficheiro a ser enviado conteria dados relativos a vários Documentos Únicos, ou seja, vários "pedidos". Assim, o elemento raiz é <dua> que tem pelo menos um elemento <pedido> como filho:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <xs:element name="dua" type="tdua"/>
   <xs:complexType name="tdua">
       <xs:sequence maxOccurs="unbounded">
           <xs:element name="pedido" type="tpedido"/>
       </xs:sequence>
   </xs:complexType>
   <xs:complexType name="tpedido">
       ...
       </xs:sequence>
       <xs:attribute name="cons" type="num7_c" use="required"/>
       <xs:attribute name="priority" type="tpriority" use="required"/>
       <xs:attribute name="nipc_lote" type="num9" use="optional"/>
       ...
   </xs:complexType>
</xs:schema>
```

Repare-se que cada elemento <pedido> tem um conjunto de atributos. Este conjunto de atributos permite identificar os pedidos *standard* (de particulares — o atributo @nipc_lote não é instanciado), os *lote* (de importadores — o atributo @nipc_lote é instanciado) e os *urgentes* (o atributo @priority vale "0"), para além da identificação do serviço emissor.

O elemento <pedido> tem, por sua vez, tantos filhos quantos os campos do documento impresso. No sentido de simplificar a compreensão visual dos documentos XML, a geração do ficheiro XML e o seu paralelismo com o documento final, admitiu-se a seguinte nomenclatura em que cada elemento tem como nome o código comunitário harmonizado (igual em todos os documentos automóveis da União Europeia) para o campo do documento que representa:

```
<xs:element name="A" type="tA"/>
<xs:element name="B" type="tB"/>
<xs:element name="B1" type="tB1"/>
```

O tipo de cada elemento foi definido tendo em conta a informação que o campo pode albergar — como extensão de um tipo simples e admitindo um atributo @nome que vale a descrição reduzida do campo do documento impresso, para facilitar a leitura no caso português.

A título de exemplo, veja-se a descrição de alguns tipos e respectivos tipos simples.

No caso da matrícula, elemento <A>, o atributo @nome vale *Matricula* e o conteúdo, obrigatório, é texto, com comprimento entre 1 e 10:

```
<xs:complexType name="tA">
  <xs:simpleContent>
    <xs:extension base="texto10_o">
       <xs:attribute name="nome" type="xs:string" use="required"
       fixed="Matricula"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="texto10_o">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:minLength value="1"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
```

No caso do elemento <M>, distância entre eixos, o atributo @nome vale *dist_eixos* e o conteúdo tem comprimento máximo 4, com o máximo de 2 casas decimais:

```
<xs:complexType name="tM">
  <xs:simpleContent>
    <xs:extension base="num4">
      <xs:attribute name="nome" type="xs:string" use="required"
      fixed="dist_eixos"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="num4">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:pattern value="[0-9]{0,4}|[0-9]{0,2}[\.][0-9]{0,1}|..."/>
  </xs:restriction>
</xs:simpleType>
```

No caso do usufruto, da locação e do aluguer de longa duração, elementos <C43>, <C441> e <C442>, os atributos @nome têm o valor correspondente e o conteúdo a existir é texto que vale "Sim", ou "De *data* a *data*":

```
<xs:complexType name="tC43">
  <xs:simpleContent>
    <xs:extension base="dura">
      <xs:attribute name="nome" type="xs:string" use="required"
      fixed="usufruto"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

```
<xs:simpleType name="dura">
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:pattern value="[S][i][m]|[D][e][ ][0-9]{4}-[0-9]{2}-[0-9]{2}
    [ ][a][ ][0-9]{4}-[0-9]{2}-[0-9]{2}|"/>
  </xs:restriction>
</xs:simpleType>
```

No caso do elemento <C33>, morada do utilizador do veículo, o atributo @nome vale *morada_utilizador* e tem no máximo dois elementos <linha> filhos que, por sua vez, têm como conteúdo texto com o máximo de 60 caracteres.

De seguida apresenta-se um documento XML exemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dua xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dua.xsd">
   <pedido cons="5945954" priority="1">
     <A nome="Matricula">exemplo</A>
     <B nome="data_1_matricula"/>
     <B1 nome="matricula_anterior"/>
     <C11 nome="apelido_titular">Galhardete</C11>
     <C12 nome="nomes_titular">
       <linha><![CDATA[Neopoldina Conceição Pafúncio]]></linha>
     </C12>
     <C13 nome="morada_titular">
       <linha><![CDATA[Rua Pedras Empenadas, 15]]></linha>
       <linha>Xerimbó das couves</linha>
     </C13>
     <C21 nome="apelido_proprietario">Galhardete</C21>
     <C22 nome="nomes_proprietario">
       <linha><![CDATA[Neopoldina Conceição Pafúncio]]></linha>
     </C22>
     <C23 nome="morada_proprietario">
       <linha><![CDATA[Rua Pedras Empenadas, 15]]></linha>
       <linha>Xerimbó das couves</linha>
     </C23>
     <C24 nome="quota_parte"/>
     <C25 nome="num_comproprietarios"/>
     ...
     <C43 nome="usufruto"/>
     <C441 nome="locacao"/>
     <C442 nome="aluguer"/>
     ...
     <D1 nome="marca">Carro de Bois</D1>
     <D2 nome="modelo_variante_versao">
       <linha>duas cabeças</linha>
     </D2>
     ...
     <Data nome="data_certificado">2005-12-01</Data>
```

```
      ...
      <I nome="data_matricula">1845-10-12</I>
      <J nome="codigo_CE"/>
      <J1 nome="categoria">LIGEIRO       </J1>
      <J2 nome="tipo_veiculo">MERCADORIAS              </J2>
      ...
      <Num nome="num">00000000 0</Num>
      ...
      <Z3 nome="anot_veiculo"/>
    </pedido>
    <pedido cons="9699696" priority="0">
      ...
    </pedido>
    ...
 </dua>
```

A informação descrita anteriormente é aquela que é enviada pelo ITIJ à INCM. Esta informação permite preencher completamente os campos do DUA legíveis pelo olho humano.

Adicionalmente, para que este documento no qual não ia ser inserido um chip, fosse legível por máquinas, foi reservada uma zona para a colocação de informação sobre o veículo num código de barras. Dado que a quantidade de informação a incluir nesse código se estimava da ordem dos 3000 caracteres, a utilização de códigos de barras convencionais (1D) era insuficiente. Sendo assim, foi necessário recorrer a códigos de barras de uma geração superior (2D). De entre os formatos actualmente disponíveis para códigos 2D, destacam-se o PDF417 [7], o DataMatrix [8], o DataCode e o QR-code [9]. Destes quatro, o mais estudado e utilizado é o último.

Nas especificações do DUA relativamente à informação a incluir no código de barras, para além da necessidade de codificação estimada, havia a limitação do espaço disponível a um quadrado de 5cm x 5cm. Adicionalmente havia a possibilidade de destruição de zonas do código (por exemplo por risco de caneta ou dobragem) por se tratar de documento muito transportado e sem prazo de validade. Optou-se então pelo formato de código de barras QR-Code por ser o que melhor se adaptava às restrições indicadas.

A informação disponibilizada neste QR-Code é um subconjunto da informação impressa no documento (sendo obtida pela INCM a partir do documento enviado pelo ITIJ) e segue a seguinte gramática:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <xs:element name="dua" type="tdua"/>
    <xs:complexType name="tdua">
        <xs:sequence>
            <xs:element name="A" type="tA"/>
            <xs:element name="C21" type="tC21"/>
```

```
            <xs:element name="C22" type="tC22"/>
            <xs:element name="C23" type="tC23"/>
            ...
            <xs:element name="D1" type="tD1"/>
            <xs:element name="D2" type="tD2"/>
            ...
            <xs:element name="J" type="tJ"/>
            <xs:element name="J1" type="tJ1"/>
            <xs:element name="J2" type="tJ2"/>
            <xs:element name="J3" type="tJ3"/>
            <xs:element name="Num" type="tNum"/>
            ...
        </xs:sequence>
    </xs:complexType>
    ...
    <xs:complexType name="tC22">
        <xs:simpleContent>
            <xs:extension base="texto160">
                <xs:attribute name="nome" type="xs:string" use="required"
                fixed="nomes_proprietario"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
 ...
</xs:schema>
```

Para o caso do primeiro pedido do documento apresentado anteriormente, o QR-Code fica como segue:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dua xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dua.xsd">
  <A nome="Matricula">exemplo</A>
  <C21 nome="apelido_proprietario">Galhardete</C21>
  <C22 nome="nomes_proprietario">
     <![CDATA[Neopoldina Conceição Pafúncio]]>
  </C22>
  <C23 nome="morada_proprietario">
     <![CDATA[Rua Pedras Empenadas, 15]]> Xerimbó das couves
  </C23>
  ...
  <D1 nome="marca">Carro de Bois</D1>
  <D2 nome="modelo_variante_versao">duas cabeças</D2>
  ...
  <J nome="codigo_CE"/>
  <J1 nome="categoria">LIGEIRO       </J1>
  <J2 nome="tipo_veiculo">MERCADORIAS              </J2>
  <Num nome="num">00000000 0</Num>
  ...
```

```
</dua>
```

As figuras 2 e 3 mostram o aspecto final de um DUA impresso.

Figura 2. Frente do Documento Único Automóvel.

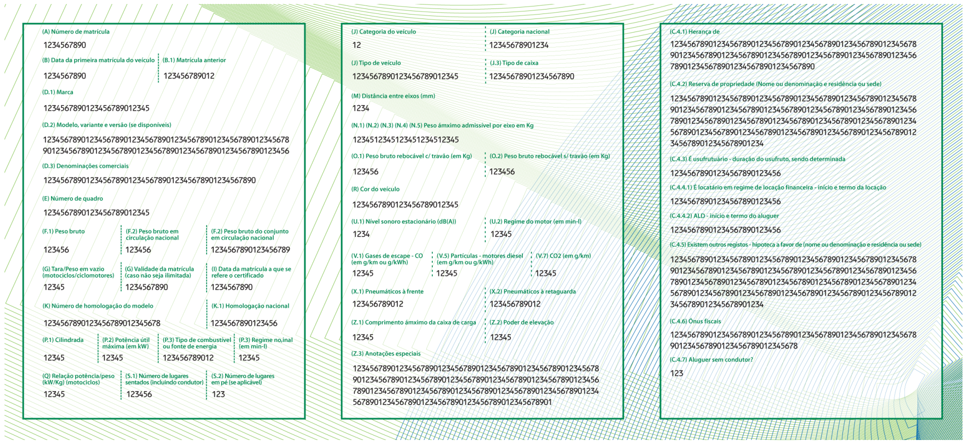
4 Envio para a INCM

Uma vez que os dados estavam, como foi dito anteriormente, todos em base de dados IMS (com excepção dos reboques, aos quais não é feito nenhum tratamento prévio), havia que gerar os documentos XML utilizando a linguagem de programação disponível no mainframe - o Cobol.

Foi então feito um programa que analisa as criações de determinado período temporal e executa três acções:

- Atribui um número de DUA a cada pedido efectuado nesse período;
- Coloca tal informação na base de dados do Documento Único Automóvel;
- Coloca a informação no documento XML a ser enviado para a INCM.

Tendo em conta que a versão de Cobol disponível ainda não permitia a exportação para XML e havia urgência em colocar todo o sistema em produção, foi necessário criar diversas variáveis com as etiquetas de abertura e fecho dos vários elementos. Seguidamente, para cada elemento, são movidos a etiqueta de início, o conteúdo do elemento (quando este não existe, nada é escrito) e a etiqueta de fecho. A excepção a este procedimento é no caso dos elementos cuja informação é dividida em várias linhas (por exemplo <C22> — nomes do proprietário) e colocada em elementos <linha> uma vez que, nesse caso, apenas os elementos <linha> para os quais há informação devem ser escritos, exigindo programação adicional.



(A) Número de matrícula
1234567890
(B) Data da primeira matrícula do veículo
1234567890
(B.1) Matrícula anterior
123456789012
(D.1) Marca
1234567890123456789012345
(D.2) Modelo, variante e versão (se disponíveis)
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
(D.3) Denominações comerciais
12345678901234567890123456789012345678901234567890
(E) Número de quadro
1234567890123456789012345
(F.1) Peso bruto
123456
(F.2) Peso bruto em circulação nacional
123456
(F.2) Peso bruto do conjunto em circulação nacional
1234567890123456789
(G) Tara/Peso em vazio (motociclos/ciclomotores)
12345
(G) Validade da matrícula (caso não seja ilimitada)
1234567890
(I) Data da matrícula a que se refere o certificado
1234567890
(K) Número de homologação do modelo
1234567890123456789012345678
(K.1) Homologação nacional
1234567890123456
(P.1) Cilindrada
12345
(P.2) Potência útil máxima (em kW)
12345
(P.3) Tipo de combustível ou fonte de energia
123456789012
(P.3) Regime no,inal (em min-l)
12345
(Q) Relação potência/peso (kW/Kg) (motociclos)
12345
(S.1) Número de lugares sentados (incluindo condutor)
123456
(S.2) Número de lugares em pé (se aplicável)
123

(J) Categoria do veículo
12
(J) Categoria nacional
12345678901234
(J) Tipo de veículo
1234567890123456789012345
(J.3) Tipo de caixa
12345678901234567890
(M) Distância entre eixos (mm)
1234
(N.1) (N.2) (N.3) (N.4) (N.5) Peso ámximo admissível por eixo em Kg
1234512345123451234512345
(O.1) Peso bruto rebocável c/ travão (em Kg)
123456
(O.2) Peso bruto rebocável s/ travão (em Kg)
123456
(R) Cor do veículo
1234567890123456789012345
(U.1) Nível sonoro estacionário (dB(A))
1234
(U.2) Regime do motor (em min-l)
12345
(V.1) Gases de escape - CO (em g/km ou g/kWh)
12345
(V.5) Partículas - motores diesel (em g/km ou g/kWh)
12345
(V.7) CO2 (em g/km)
12345
(X.1) Pneumáticos à frente
123456789012
(X.2) Pneumáticos à retaguarda
123456789012
(Z.1) Comprimento ámximo da caixa de carga
12345
(Z.2) Poder de elevação
12345
(Z.3) Anotações especiais
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
78901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901

(C.4.1) Herança de
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
7890123456789012345678901234567890
(C.4.2) Reserva de propriedade (Nome ou denominação e residência ou sede)
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
7890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234
(C.4.3) É usufrutuário - duração do usufruto, sendo determinada
12345678901234567890123456
(C.4.4.1) É locatário em regime de locação financeira - início e termo da locação
12345678901234567890123456
(C.4.4.2) ALD - início e termo do aluguer
12345678901234567890123456
(C.4.5) Existem outros registos - hipoteca a favor de (nome ou denominação e residência ou sede)
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456
7890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
3456789012345678901234
(C.4.6) Ónus fiscais
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
(C.4.7) Aluguer sem condutor?
123

Figura 3. Costas do Documento Único Automóvel.

Uma outra dificuldade encontrada na geração dos ficheiros foi a existência de "&" em certos nomes (por exemplo de empresas). Substituir "&" por "&" consistia num trabalho adicional para aumentar progressivamente os comprimentos das áreas e estava sujeito a erros no caso de o número de caracteres destes ser superior ao máximo previsto. Assim, optou-se por inserir essa informação numa secção marcada de texto (CDATA) [10].

Portanto, para a geração de cada documento XML são movidos a declaração XML; a etiqueta de abertura do elemento raiz — <dua> —; para cada pedido a etiqueta de abertura, os vários elementos e a etiqueta de fecho; e a etiqueta de fecho de <dua>.

O envio dos ficheiros XML correspondentes aos pedidos de impressão de DUAs para a INCM é feito através de uma ligação segura estabelecida entre as duas entidades, garantindo assim a confidencialidade dos dados.

5 Recepção/Produção

O servidor da INCM direcciona os ficheiros recebidos para uma directoria criada exclusivamente para essa tarefa. Os vários ficheiros recebidos e colocados nessa directoria ficam a aguardar processamento.

Periodicamente (actualmente 30 minutos) e de forma automática, é executado um serviço que inspecciona a directoria referida e processa todos os ficheiros que lá se encontram. Primeiro cada ficheiro XML processado é validado contra o XML-Schema apresentado na secção 3. Quando o documento XML não é válido, é gerado um log indicando a impossibilidade do processamento do ficheiro. Quando o ficheiro XML é válido, o seu conteúdo é submetido a uma segunda operação — a submissão a um WebService, ou seja, cada pedido de DUA constante no ficheiro XML é transformado num post XML a um WebService

interno que, ao receber esse pedido e após um conjunto de validações adicionais, o insere na base de dados.

Todo este processo pode ser visto esquematicamente na figura 4.

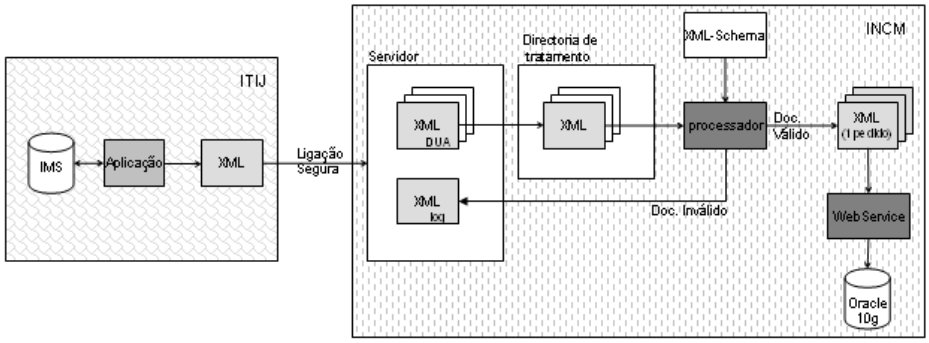


Figura 4. Intercâmbio de informação entre o ITIJ e a INCM e processamento do lado da INCM.

Uma vez na base de dados, os registos são controlados por uma máquina de estados com os seguintes estados: recebido; em produção; entregue (para *urgente* e *lote*); devolvido (para *standard*); e não reclamado (para *lote*), seguindo o processo de produção.

Por cada mudança de estado é gerado um log para informação ao ITIJ do estado dos pedidos, colocado na directoria do servidor da INCM ao qual o ITIJ tem acesso.

Paralelamente aos ficheiros de log, a INCM disponibiliza, num site de acesso restrito, informação sobre a actividade de produção dos DUAs. Neste site é disponibilizada informação sobre os totais produzidos e pendentes para produção, conforme a figura 5.

Neste site, pode-se ainda pesquisar a base de dados por matrícula, número de DUA e número de registo dos Correios de Portugal SA (CTT). Os DUAs que respeitem o critério de pesquisa são listados com as respectivas datas de mudança na máquina de estados. No caso dos DUAs *standard*, o número de registo dos CTT é apresentado sob a forma de link para o site dos CTT por forma a facilitar o acesso ao sistema "track and trace" daquela entidade.

6 Conclusão

O Documento Único Automóvel, veio trazer um novo conceito de documentação automóvel para Portugal, reduzindo o custo associado a este tipo de documentação e a quantidade de papel a transportar para o utilizador final, aumentando a segurança e a simplicidade.

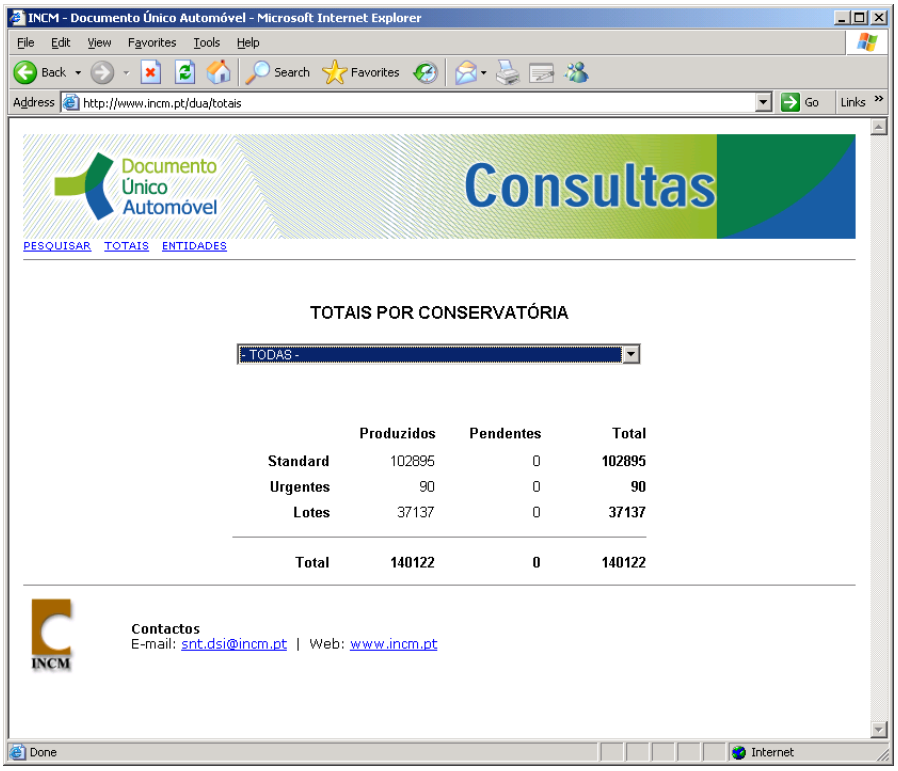


Figura 5. Site com informação sobre a produção de DUAs.

No plano técnico, representou uma oportunidade de intercâmbio de informação entre sistemas primeiro para congregar os dados antes utilizados para fins diferentes em instituições diferentes e depois para transferir a informação relevante para impressão entre instituições.

Este artigo apresenta um processo original e criado de raiz pelo ITIJ e pela INCM, incluindo a definição completa do XML-Schema a usar para envio da informação respeitante aos DUAs e os processos de extracção de informação da BD do ITIJ, geração do documento XML, e recepção e processamento na INCM para impressão dos vários documentos.

Mas nem tudo está feito. Há que melhorar a forma como a informação é transferida entre o ITIJ e a INCM, por exemplo. Espera-se, num futuro próximo, entregar a informação directamente ao WebService disponibilizado pela INCM.

7 Agradecimentos

Os autores agradecem a colaboração de Óscar Veiga e Anabela Baptista, responsáveis pelas aplicações do lado do mainframe no ITIJ.

Referências

1. Directiva Comunitária 1999/37/CE do conselho de 29 de Abril, http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:31999L0037:PT:HTML, último acesso em Dezembro de 2005.
2. Directiva Comunitária 2003/127/CE do conselho de 23 de Dezembro, http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:32003L0127:PT:HTML, último acesso em Dezembro de 2005.
3. Decreto-Lel nº 178-A/2005 de 28 de Outubro, http://www.dgsi.pt/gdep.nsf/0/c89691b0b62acc5e802570c20032f726?OpenDocument, último acesso em Novembro de 2005.
4. IMS Family, http://www-306.ibm.com/software/data/ims/, último acesso em Dezembro de 2005.
5. Duckett, J., Griffin, O., Mohr, S., Norton, F., Ozu, N., Stokes-Rees, I., Tennison, J., Williams, K., Cagle, K.: Professional XML Schemas. Wrox Press (2001)
6. Eric van der Vlist: XML Schema. O'Reilly & Associates, Inc. (2002)
7. PDF417, http://www.symbol.com/, último acesso em Dezembro de 2005.
8. DataMatrix, http://www.gavitec.com/Advantages_of_Data_Matrix.601.0.html, último acesso em Dezembro de 2005.
9. QR-Code, http://www.qrcode.org, último acesso em Dezembro de 2005.
10. Ramalho, J. C., Henriques, P.: XML e XSL da Teoria à Prática. FCA. ISBN 972-722-347-8 (2002)

Posters

Navegante: Um Proxy de Ordem Superior para Navegação Intrusiva

José João Almeida — DI/UM
Alberto Simões — DI/UM

Resumo

Um proxy é um serviço disponível na Internet que permite que clientes façam ligações indirectas a outros serviços. Os casos mais típicos são as proxy de instituições que fazem *caching* de documentos, sem de qualquer forma os alterar. Outros proxies limitam-se a barrar determinadas páginas.

No entanto, existe um conjunto de ferramentas que podem ser vistas como proxies que processam e **alteram** estruturalmente documentos HTML. Exemplos destas ferramentas são um "corrector ortográfico on-line,", "detector de língua" ou mesmo um "melhorador de acessibilidade" de páginas Web.

Todas estas ferramentas podem ser vistas como um processador genérico que, mediante uma função de processamento, analisa os documentos HTML. Este processador (*Navegante*) funciona como um proxy que aplica o processador às páginas HTML. Na verdade, o *Navegante* não está implementado (actualmente) como um proxy habitual, mas como uma CGI com um comportamento semelhante ao de um proxy.

Dado um URL, o *Navegante* processa-o e mostra o resultado ao utilizador. Ao seguir ligações para outros documentos HTML o *Navegante* intromete-se de forma a que continue a ser utilizado nos documentos seguintes.

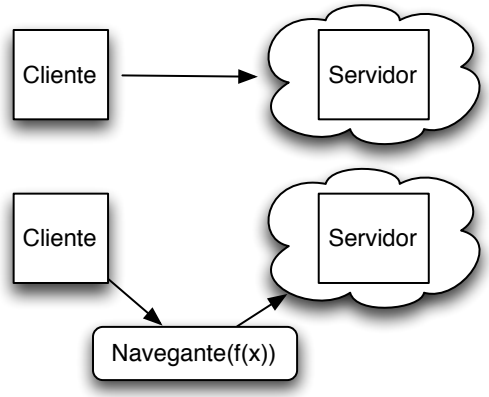
Actualmente estamos a usar o *Navegante* para um corrector ortográfico online. A função de processamento é chamada para cada elemento PCDATA existente no documento. A função analisa o texto e retorna um HTML em que o texto aparece com os erros ortográficos marcados.

Um proxy de ordem superior para navegação intrusiva

Navegante(f(x))

- CGI simula um proxy
- Intercepta ligações a HTMLs
- Torna referências absolutas
- Preserva elementos especiais
- Transforma conteúdo de acordo com f(x)

Proxy de Ordem Superior



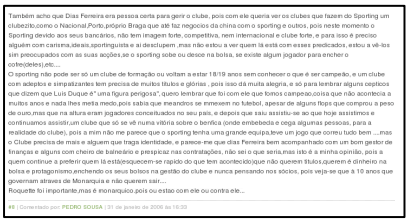
Navegante (f)

$f : pcdata \longrightarrow HTML$

Exemplo: ProxySpell

$$f(x) = \quad \textbf{para toda } palavra \in x$$

$$\begin{cases} palavra \in \mathcal{PT} & \Rightarrow & palavra \\ palavra \in \mathcal{EN} & \Rightarrow & verde(palavra) \\ \textbf{senão} & \Rightarrow & vermelho(palavra) \end{cases}$$



Navegante Monádico (f , g)

$f : \quad pcdata \times estado \longrightarrow HTML \times estado$
$g : \quad estado \longrightarrow HTML$

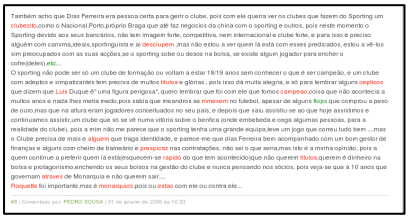
Exemplo: ProxySpell Monádico

$$f(x) = \quad \textbf{para toda } palavra \in x$$

$$\begin{cases} palavra \in \mathcal{PT} & \Rightarrow & palavra \\ palavra \in \mathcal{EN} & \Rightarrow & verde(palavra) \\ \textbf{senão} & \Rightarrow & \begin{cases} estado[palavra] + + \\ vermelho(palavra) \end{cases} \end{cases}$$

$$g(x) = \quad \textbf{para toda } palavra \in Dom(estado)$$

$$li(palavra : estado[palavra])$$



Outras Utilizações

- Marcar entidades mencionadas
- Tornar páginas acessíveis

Bugs ...

Falta suporte para JavaScript;

Pouco robusto (solucionável com implementação de um proxy...)

José João Almeida - jj@di.uminho.pt Alberto Simões - ambs@di.uminho.pt
Departamento de Informática - Universidade do Minho - Campus de Gualtar - 4710-057 Braga

Projecto de Arquitectura em XML para Publicação de Dados Académicos e Científicos

Carlos Caldeira — Universidade de Évora

Resumo

O conceito fundamental é o de criar uma infra-estrutura que permita a um grupo de docentes pertencentes a um mesmo sector ou departamento publicarem os seus dados pessoais, académicos e científicos de uma forma uniforme e visualmente atraente.

Esta plataforma, com suporte em XML, e derivada do projecto Apache Cocoon, transforma inputs de diversas fontes num documento normalizado. A arquitectura é modular e facilmente extensível e os documentos gerados podem ter um conteúdo estático ou dinâmico.

Projecto de Arquitectura em XML para Publicação de Dados Académicos e Científicos

Carlos P. Caldeira
Rua Romão Ramalho, 59 7000 Évora

Introducão

O conceito fundamental é o de criar uma infra-estrutura que permita a um grupo de docentes pertencentes a um mesmo sector ou departamento publicarem os seus dados pessoais, académicos e científicos de uma forma uniforme e visualmente atraente.

Esta plataforma, com suporte em XML, e derivada do projecto Apache Cocoon, transforma inputs de diversas fontes num documento normalizado.

A arquitectura é modular e facilmente extensível e os documentos gerados podem ter um conteúdo estático ou dinâmico.

Objectivos

O objectivo fundamental deste projecto é o de definir uma estrutura normalizada para transformação de dados provenientes de fontes diversas num documento unificado, de forma a que a informação de um sector ou departamento seja percebida através de uma interface única e normalizada.

Método

- Análise das normas XML do dialecto Apache Cocoon
- Estudo de outros projectos na área
- Definição da arquitectura inicial
- Desenvolvimento do protótipo

Resultados

Figura 1: Exemplo de uma página com publicação das notícias de uma disciplina

Os primeiros resultados obtidos com o a construção do protótipo permitem concluir o seguinte:

- a grande capacidade do sistema para agregar dados provenientes de várias fontes e com formatos distintos numa interface uniforme
- a facilidade com que os produtores de informação conseguem publicar os seus dados sem estarem preocupados com aspectos técnicos específicos

Figura 2: Exemplo de uma página com publicação do plano semestral de uma disciplina

Figura 3: Exemplo de aplicação da arquitectura a um conjunto de publicações

Contacto

Para mais informações
Departamento de Informática
Universidade de Évora

Contacto:
Carlos Caldeira
Email: ccaldeira@di.uevora.pt
Web: http://www.di.uevora.pt/~ccaldeira

Agradecimentos

O autor agradece à Universidade de Évora todo o apoio concedido.

Voice Over M2L

Daniel Silva — Faculdade de Engenharia da Universidade do Porto
Pedro Abreu — Faculdade de Engenharia da Universidade do Porto
Pedro Mendes — Faculdade de Engenharia da Universidade do Porto
Vasco Moreira — Faculdade de Engenharia da Universidade do Porto

Resumo

Apesar da facilidade de acesso à informação e simbiótica disseminação, subsistem nichos com barreiras de acessibilidade. Existem igualmente áreas de conhecimento cujo léxico próprio tem necessidades expressivas actualmente sem resposta aceitável. Associando-se-lhes os nichos mencionados, assumem carácter particularmente relevante.

A interpretação da informação matemática por invisuais ilustra esta problemática, emergindo a necessidade de ferramentas compatibilizadoras de apresentação e percepção. A sua realização passa pela vocalização da tradução do formato MathML para linguagem natural. Domínio privilegiado para a aplicação desta abordagem é a utilização dos audio-browsers, visto permitirem acesso a conteúdos disponibilizados na Web. Assim, usufruindo das normas existentes, optou-se pela codificação da linguagem natural em SSML.

A versatilidade da tecnologia torna-a adaptável a outras áreas, destacando-se ensino e investigação, com aplicações como leitura automática de enunciados, respectivos processos inversos e aprendizagem da simbologia matemática. Isto possibilita a generalização para sistemas de interacção por voz, reformulando paradigmas de interacção pessoa-computador.

VoiceOverM²L

"Talk the Math"

Introdução

Pretendem-se tornar perceptíveis as representações convencionais da simbologia matemática para indivíduos com necessidades especiais. Assim, partindo da representação de uma expressão matemática numa linguagem de anotação (*MathML*), converte-se esta para dois formatos distintos: um destinado à vocalização através de motores TTS (*text to speech*) e outro contendo meta informação sobre a expressividade do texto produzido, em formato anotado (*SSML*).

Implementação

MathML

root index three of fraction with numerator beggining of power three to the power of five x end of power and denominator summation with lower bound i equals one and upper bound nine of six i end of summation end of fraction end of root

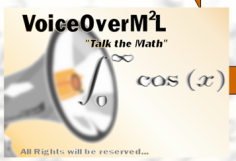
SSML

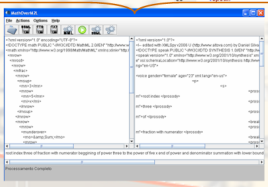
```
<?xml version="1.0"?>
<!-- edited with XMLSpy v2006 U (http://www.altova.com) by Daniel Silva (FEUP) -->
<!DOCTYPE speak PUBLIC "-//W3C//DTD SYNTHESIS 1.0//EN" "http://www.w3.org/TR/speech-synthesis/synthesis.dtd">
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
http://www.w3.org/2001/10/synthesis http://www.w3.org/TR/speech-synthesis/synthesis.xsd" xml:lang="en-US">
<voice gender="female" age="23" xml:lang="en-us">
<p>
<s>
<prosody pitch="medium" rate="medium" volume="medium">root index </prosody>
<prosody pitch="medium" rate="medium" volume="medium">three </prosody>
<prosody pitch="medium" rate="medium" volume="medium">of </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">fraction with numerator </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">begging of power </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">three </prosody>
<prosody pitch="medium" rate="medium" volume="medium">to the power of </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">five </prosody>
<prosody pitch="medium" rate="medium" volume="medium">x </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">end of power </prosody>
<break/>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">and denominator </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">summation </prosody>
<prosody pitch="medium" rate="medium" volume="medium">with lower bound </prosody>
<prosody pitch="medium" rate="medium" volume="medium">i </prosody>
<prosody pitch="medium" rate="medium" volume="medium">equals </prosody>
<prosody pitch="medium" rate="medium" volume="medium">one </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">and upper bound </prosody>
<prosody pitch="medium" rate="medium" volume="medium">nine </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">of </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">six </prosody>
<prosody pitch="medium" rate="medium" volume="medium">i </prosody>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">end of summation </prosody>
<break/>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">end of fraction </prosody>
<break/>
<break/>
<prosody pitch="medium" rate="medium" volume="medium">end of root </prosody>
<break/>
</s>
</p>
</voice>
</speak>
```

ALTOVA xmlspy 2006

NC1.6.0

Free TTS





Conclusão

O projecto contempla a conversão de um ficheiro em *MathML* (validado na aplicação) para dois tipos de ficheiros. É já suportada uma gama considerável do domínio da matemática (aritmética, trigonometria, operadores complexos entre outros), bem como a tradução para linguagem natural em dois idiomas distintos. A exportação em formato anotado de voz segue algumas considerações a nível linguístico.

A arquitectura modular usada no projecto permite evoluções diversas, a nível dos domínios da matemática, suporte a um maior número de idiomas, criação de novas interfaces que usem paradigmas diversos de interacção pessoa-computador.

Daniel Silva **ei01083@fe.up.pt**
Pedro Abreu **ei01023@fe.up.pt**
Pedro Mendes **ei01022@fe.up.pt**
Vasco Vinhas **ei01053@fe.up.pt**

Web-based Knowledge Portal for Educational Purposes

Joaquim Silva — Faculdade de Engenharia da Universidade do Porto
Francisco Restivo — Faculdade de Engenharia da Universidade do Porto

Resumo

Our focus is on a Web-based Knowledge Portal for educational purposes.

We consider a distributed system for information retrieval and document collection, which will enable different forms of knowledge construction. Also accessing information from multiple information systems and integrating them into a knowledge portal are key issues in developing our system.

Some tools are considered to be used namely ontologies, concept maps and software agents to deal with semantic issues. Previous related initiatives are described and tools for dealing with semantics are also present. A comprehensive section on syntactic and semantic interoperability is described in detail. A very high-level architecture is drafted along with ideas to deal with its implementation.

XATA 2006

Web-based Knowledge PORTAL

for educational purposes

Joaquim Fernando Silva, joaquim.silva@fe.up.pt
Francisco José Restivo, fjr@fe.up.pt
Faculty of Engineering of University of Porto

Aims:

- Building a distributed system for dynamic information search, organization and classification. Integration of information from multiple information systems.
- Interaction of different members (learners, tutors, teachers, content producers).
- Orient learners, within the subject domain, to build up their own understanding and conceptual association with value-adding activities.
- Achieve a new multidimensional of understanding.

Integrate knowledge and have information visualization

Methodology:

Technology

Software agents.
Ontologies.
XML,RDF,OWL.
Concept maps.

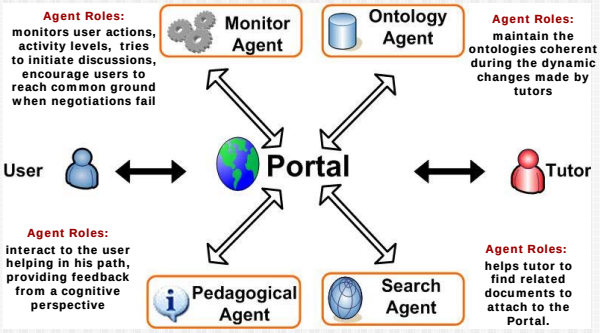
Semantic Web interoperability

Web search for information retrieval.
Reuse learning contents available in autonomous and heterogeneous information systems.

Cognitive

Based on social constructivist theory.

High-level Architecture:



Future work:

- Building the system using rapid prototyping
- Organize information in a user and knowledge perspective
- Evaluating its effectiveness, usefulness and value proposition
- Deep analysis and bed tests
  - Reach critical mass

Additional information:
http://paginas.fe.up.pt/~prodei/comic06/p10.pdf

Universidade do Porto Faculdade de Engenharia FEUP

ESTG, PORTALEGRE, 9 E 10 DE FEVEREIRO DE 2006

Modelos XML para Documentação de Requisitos

Ricardo Pinto — Faculdade de Engenharia da Universidade do Porto

Resumo

Durante um projecto de desenvolvimento genérico de software, a documentação dos requisitos é frequentemente dificultada, uma vez que as funcionalidades podem variar de cliente para cliente.

Uma abordagem a este problema consiste em criar um modelo que permita a reutilização de uma estrutura genérica, para garantir a rápida implementação e evolução dos sistemas.

Partindo da definição de um conjunto dos diversos elementos constituintes de um documento de requisitos, pretende-se utilizar uma plataforma baseada nas tecnologias Wiki, o XSDoc, para servir de suporte. Neste sentido, é essencial construir as estruturas de dados em XML, as folhas de estilo em XSL e os esquemas em XSD, no sentido de materializar estes conceitos.

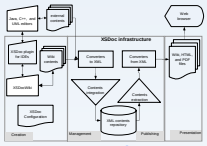
Modelos XML para Documentação de Requisitos

Ricardo Pinto
Faculdade de Engenharia da Universidade do Porto

Resumo

Uma abordagem para reutilizar documentação de requisitos consiste em aliar um modelo a uma aplicação, garantindo a rápida implementação e evolução dos sistemas.

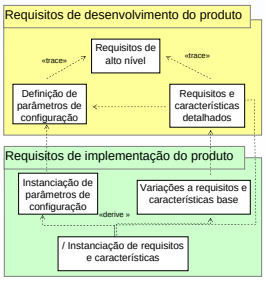
Definindo os elementos constituintes (requisitos de alto nível, parâmetros, restrições, casos de uso e entidades) pretende-se utilizar uma plataforma do tipo Wiki, o XSDoc, para servir de suporte.



Neste sentido, é essencial construir as diversas estruturas em XML que irão materializar estes conceitos.

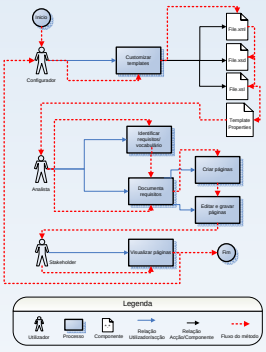
Objectivos

O objectivo principal deste projecto consistiu em definir os diferentes ficheiros XML (bem como XSL e XSD), para os vários elementos acima identificados :



Método

Para atingir os objectivos definidos, adoptou-se um protocolo de acção, de acordo com o seguinte diagrama:



O fluxo descrito a vermelho representa o conjunto de passos tomados e componentes criados ou modificados durante a aplicação deste método.

Após a conclusão da fase de configuração de modelos, foram obtidos vários *templates* XML, XSD e XSL, como os seguintes:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <use-case xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="default C:\tomcat5\webapps\sadeo04-crm\WEB-
  INF\wikiwebs\webs\main\templates\UseCase.xsd">
  <identifier />
  <name>Nome do caso de uso. Exemplo: Create Marketing Campaign</name>
  <brief_description>Breve descrição tornando evidente o objectivo principal
  do caso de uso. Exemplo: This use case allows the definition of a marketing
  campaign and its target segment or set of customers.</brief_description>
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" />
- <xsl:template match="use-case">
  - <html>
    - <body>
      <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>
  <xsl:template match="identifier" />
- <xsl:template match="name">
  - <h1>
    <xsl:value-of select="." />
    </h1>
  </xsl:template>
- <xsl:template match="brief_description">
    <h2>Descrição Sumária</h2>
    <xsl:apply-templates />
  </xsl:template>
```

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" id="UseCase">
  <xs:element name="use-case">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="identifier" type="xs:string" />
        <xs:element name="name" type="xs:string" />
        <xs:element name="brief_description" type="xs:string" />
```

Resultados

Os resultados obtidos são bastante positivos: foram criados e integrados com sucesso todos os modelos identificados.

A imagem seguinte ilustra uma página da plataforma XSDoc, para um caso de uso:

Como é possível observar, a informação relativa a este caso de uso apresenta uma estrutura bem definida, que é importada do respectivo modelo XML. Quanto à formatação é efectuada de acordo com o XSL e inclui *headers*, *links* e listagens.

Conclusões

Após o término desta fase do projecto, é possível concluir que os objectivos propostos na fase de planeamento foram cumpridos.

Deste modo, e tendo em conta o modelo de documentação definido, foi possível conceber o suporte estruturado em XML, XSD e XSL.

Quanto à integração com a plataforma XSDoc regista-se também que foi efectuada de acordo com o planeado, estando já em funcionamento.

No futuro, haverá novos desafios: a integração de documentos *delta* e de um editor de XML no XSDoc.

Referências

Artigos

Wiki Based Requirements Documentation of Generic Software Products - Clara Silveira, João Pascoal Faria, Ademar Aguiar, Raul Vidal, 2005

Páginas Web

http://www.w3schools.com
http://www.xml.org
http://www.topxml.com

FEUP

Um Sistema de Pesquisa de Conteúdos de Aprendizagem para a Web Semântica

Vitor Gonçalves — Escola Superior de Educação do Instituto Politécnico de Bragança
Eurico Carrapatoso — Faculdade de Engenharia da Universidade do Porto

Resumo

A Sociedade da Informação (SI) em que vivemos exige-nos uma contínua e rápida actualização dos conhecimentos para que possamos adaptar-nos e reagir às constantes mudanças profissionais e sociais.

A interiorização do conceito de educação ao longo da vida tem vindo a impulsionar novas formas de auto-aprendizagem, tais como: a aprendizagem através de conteúdos Web dispersos localizados por motores de busca e a aprendizagem através de conteúdos disponibilizados por sistemas de e-Learning. No entanto, o crescimento da Web tornou cada vez mais problemática a descoberta e a recuperação desses objectos de aprendizagem.

Neste sentido, propõe-se, por um lado, o recurso a metadados e ontologias para a pesquisa de objectos de aprendizagem dispersos e, por outro, o uso de mapas de conceitos, descritos através da tecnologia XTM (XML Topic Maps), para apoiar a recuperação de objectos de aprendizagem organizados em cursos de e-Learning e, consequente, geração de planos de formação personalizados.

Assim, este artigo sugere uma arquitectura para um sistema de pesquisa de conteúdos de aprendizagem baseado em tecnologias de e-Learning, tecnologias da Web Semântica e tecnologias de Agentes.

XATA2006

XML: Aplicações e Tecnologias Associadas

Um Sistema de Pesquisa de Conteúdos de Aprendizagem para a Web Semântica

Vitor Gonçalves & Eurico Carrapatoso

Problemática:

A interiorização do conceito de educação ao longo da vida tem vindo a impulsionar novas formas de auto-aprendizagem: aprendizagem através de conteúdos Web dispersos localizados por motores de busca e aprendizagem através de conteúdos organizados em sistemas de e-Learning. No entanto, o crescimento da Web tornou cada vez mais problemática a sua descoberta e a recuperação.

Proposta:

Uma arquitectura para um sistema de pesquisa de conteúdos de aprendizagem baseado em tecnologias de e-Learning, tecnologias da Web Semântica e tecnologias de Agentes.

Objectivos:

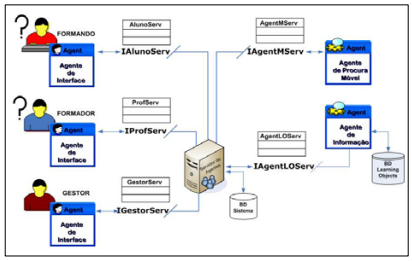
• Explorar as vantagens que a visão da Web Semântica pode provocar nos sistemas de e-Learning;
• Integrar as tecnologias da Web Semântica no desenvolvimento de conteúdos de e-Learning baseados no modelo ADL SCORM (*Sharable Content Object Resource Model*) e nas normas IMS/LOM (*Instructional Management Systems / Learning Objects Metadata*);
• Conceber um sistema baseado em agentes de software (orientados por metadados, mapas de conceitos e ontologias) que possibilite a localização, recuperação e reutilização de objectos de aprendizagem de sistemas remotos;
• Recorrer a metadados e ontologias para a pesquisa de objectos de aprendizagem dispersos;
• Usar mapas de conceitos, descritos através da tecnologia XTM (XML *Topic Maps*), para apoiar a recuperação de objectos de aprendizagem organizados em cursos de e-Learning e, consequente, geração de planos de formação personalizados.

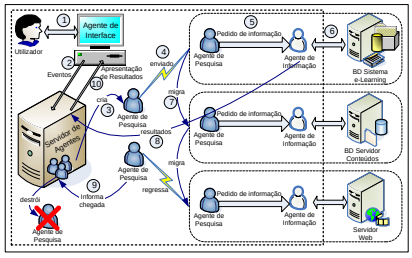
Tecnologias envolvidas:

• **Tecnologias de e-Learning:**
  - Plataformas de e-Learning: LMS Moodle e LCMS Atutor;
  - Modelo SCORM e normas LOM e IMS *Content Packing* (Reload Editor);
  - Mapas de conceitos e XTM (XML *Topic Maps*).

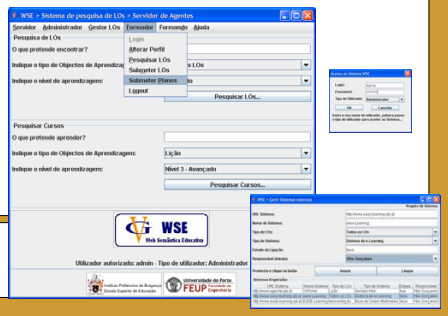
• **Tecnologias da Web Semântica:**
  - XML e tecnologias derivadas XOM / XTM;
  - Tecnologia de metadados RDF (*Resource Description Framework*);
  - Tecnologia de Ontologias OWL *Web Ontology Language*);;
  - Tecnologia de inferência SWRL (*Semantic Web Rule Language*).

• **Tecnologias de Agentes:**
  - Agentes de interface, de tarefas e de informação;
  - Agentes de pesquisa: móveis e inteligentes;
  - Linguagem de programação Java;
  - Plataforma Voyager (ORB Java).





Realização da pesquisa de objectos de aprendizagem:
1) Os utilizadores submetem uma pesquisa através dos agentes de interface com base em ontologias OWL da área do conhecimento;
2) O Agente de Interface informa o Servidor de Agentes da necessidade de realizar uma tarefa: procurar informação (objectos de aprendizagem);
3) O Servidor de Agentes cria e envia para a rede um agente orientado à realização dessa tarefa (Agente de Pesquisa);
4) Com base num itinerário previamente estabelecido, o Agente de Pesquisa migra para o servidor de objectos de aprendizagem (LOs);
5) Não sendo especialista na comunicação com as BDs, o Agente de Pesquisa móvel requisita essa função ao Agente de Informação remoto;
6) O Agente de Informação localiza e recupera os LOs solicitados, recorrendo a metadados (XML/RDF/LOM/MPEG7), às páginas Web anotadas e às ontologias disponíveis de acordo com as regras de inferência previamente estabelecidas;
7) Enquanto o Agente de Informação realiza a sua tarefa, o Agente de Pesquisa segue o seu itinerário, migrando para outro servidor Web;
8) Caso existam LOs, o Agente de Informação devolve directamente o resultado ao Servidor de Agentes (lista de recursos e respectivos links);
9) Após cumprido o itinerário, o Agente de Pesquisa informa o Servidor de Agentes do seu regresso, para que este proceda à sua destruição.
10) Caso os objectos de aprendizagem se enquadrem em pelo menos um dos mapas de conceitos existentes no sistema, os mesmos são incluídos como links dos conceitos ou tópicos descritos em XTM.



Contactos:
Vitor Gonçalves
E-mail: vg@ipb.pt
URL: http://www.vgportal.ipb.pt

Universidade do Porto
FEUP Faculdade de Engenharia
Instituto Politécnico de Bragança
Escola Superior de Educação

Conversão de Oracle Forms para Microsoft .NET Usando Dialecto XAML

António Sernadas — Faculdade de Engenharia da Universidade do Porto
Ricardo Pinto — Faculdade de Engenharia da Universidade do Porto

Resumo

O XAML (eXtensible Application Markup Language) apresenta-se como um dialecto XML com inúmeras potencialidades, ao nível do desenvolvimento gráfico e integra-se na nova família de tecnologias de última geração criadas pela Microsoft.

Tendo em conta a necessidade de criar uma arquitectura .NET, partindo de interfaces Oracle, surge a ideia de criar uma aplicação capaz de converter um formulário num ficheiro descritivo do tipo XML e convertê-lo para XAML.

Assim, será gerado código C# facilmente integrável na plataforma .NET. Este documento descreve os principais conceitos envolvidos no projecto, centrando-se na aplicação a desenvolver.

XATA2006

Ferramenta de conversão de interfaces Oracle Forms para a plataforma .NET, usando dialecto XAML

```
<?xml
version="1.0"
encoding="ISO
-8859-1" ?>

<xsl:stylesheet
version="1.0"
xmlns:xsl="http
://www.w3.org/
1999/XSL/Tran
sform">

<grid
xmlns="http://s
chemas.micros
oft.com/winfx/a
valon/2005"
xmlns:x="http:/
/schemas.micr
osoft.com/winf
x/xaml/2005">
```

XSL

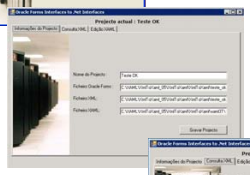
Fevereiro de 2006

XAML

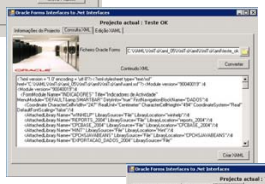
```
<?xml
version="1.0"
encoding="utf-
8"?>

<Canvas
Margin="10">
<Canvas
Height="100"
Width="100"
Top="0"
Left="0">
<Rectangle
Width="500"
Height="150"
Fill="#BEF">
</Rectangle>
</Canvas>
```

Aplicação orientada a projectos de conversão: criar um novo ou editar um já existente

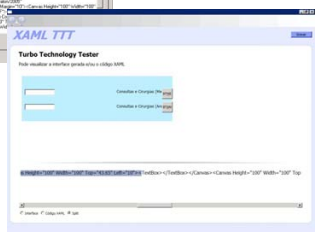


Definição da informação do projecto: nome, form, ficheiros XML e XAML

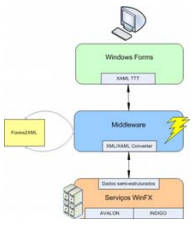


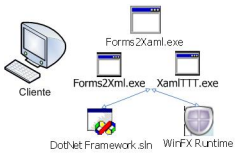
Selecção de um ficheiro Oracle Form e obtenção do respectivo XML

Criação do código XAML a partir do XML original, utilizando *stylesheets*



Aplicação de edição/visualização de XAML, criada ela própria em Visual Studio .NET 2005 e XAML





Aplicação Windows Form, com arquitectura *three-tier* e integração de dois componentes externos: Form2Xml e XamlTTT

António Sernadas

Ricardo Pinto

SPEAK: A Generic Framework to Search in Metadata

Paulo Gomes — Polytechnic Institute of Portalegre
Jorge Machado — Polytechnic Institute of Portalegre

Resumo

This poster presents SPEAK framework. SPEAK will be defined as an architecture and a protocol acting as a service under a web-server.

SPEAK mainly task is to promote a distributed middleware where applications of several kind can submit searches in Metadata that can be XML or known file types. Textual information will be another objective of SPEAK server and clients. SPEAK Server act as an inverted index like Google but only index contents in local machines or dedicated applications.

Any one can use SPEAK to register his contents, textual or normalized, including firms that use local schemas to represent their Contents and Metadata. SPEAK index Metadata is based on MITRA general way to index XML and Text. SPEAK Client applications can submit queries specifying schema and the values to find in specific fields of those schemas. SPEAK will promote results with calculated values like average, maximum or minimum values for each searched field. This is particular interesting to research projects like GPRM — Global Project for Research Management.

This kind of solutions needs to get statistical information from community that will help them to find a way. SPEAK will use a XML schema, SUSI (SPEAK User Standard Interface) to promote automatic communication with client applications that use SPEAK to transport their queries over community.

Basically SPEAK will be a Web application to index local XML and other contents and where local applications can submit advanced searches in any fields of any schema known in SPEAK community.

Using SPEAK (Search Process of Engineering for Assimilation of Knowledge) in GPRM (Global Project for Research Management)

Paulo Gomes, Ph.D * Jorge Machado

Abstract

A generic framework to search in Metadata

This poster presents the SPEAK framework working with the GPRM solution. SPEAK is defined as an architecture acting as a service under a web-server. SPEAK mainly task is to promote a distributed middleware where applications of several kind (like GPRM) can submit searches in Metadata like XML or other known file types, retrieving digest and statistic information.

SPEAK Server act as an inverted index like Google but only for dedicated applications to get objective answers.

Any one can use SPEAK to search and to register his contents, textual or normalized, including local schemas to represent their contents and metadata. SPEAK index metadata is based on MITRA general way to index XML and Text.

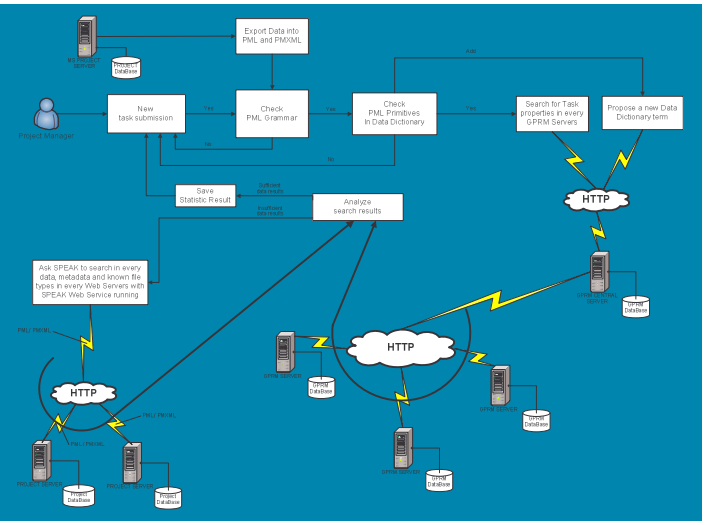
SPEAK Client applications (like GPRM) can submit queries specifying a schema and the values to find in specific fields of those schemas. SPEAK will promote statistic information results like average, maximum or minimum values for each searched field.

In this case, GPRM, normally wants to know the less or the average duration for a specific task, also the average cost, etc. But SPEAK can be used by other applications like e-commerce or e-business. In that case SPEAK can give the cheaper price of a product or even the fastest way to do something. SPEAK results are not link pages, but digest information based on statistic and numeric information.

Functional Diagram

GPRM it's a project management solution for distributed environments. The goal of this system is to give Project Researchers and Companies a way to manage their projects in particular more easy ways to find information about similar projects, like resources availability, tasks duration, tasks order in some kind of project, etc. GPRM uses an XML schema, named as PML (Project Markup Language) and PMXML to put their nodes communicating each others. That schema works like a language to define an entire project.

SPEAK goal is to promote the platform to put all **GPRM** nodes in communication and provide ways to find information in that schema. Any one who installs **GPRM** will have a bridge developed by **GPRM** team that use **SPEAK** as transport and search protocol. an be

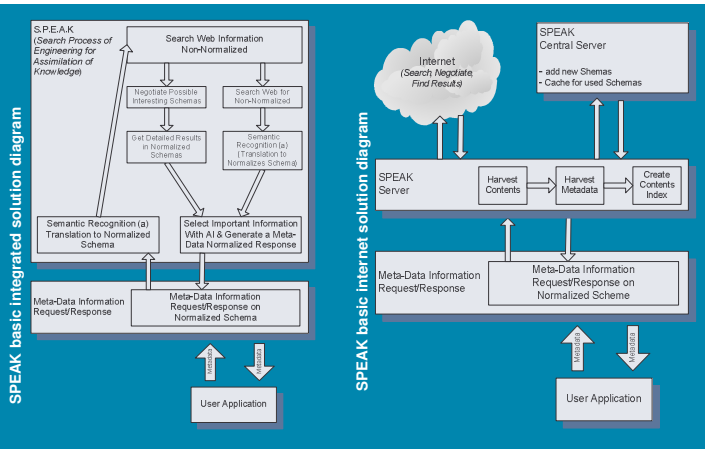


Basic-Class Diagram

SPEAK Server Web Services will have conversation over the world with **SPEAK Client Applications** (like **GPRM**). The **SPEAK Server** goal is to know where the contents are and create indexes to provide a quick answer to the request information by the **SPEAK Client Applications** and indicate where and how the search must be done, to have objective and mathematical responses.

SPEAK have a set of possible ways to declare contents. Those ways are sub-divided in: Contents Registration (with or with out metadata), Metadata.

At the end server will harvest all of them to create is own representation of information.



INSTITUTO POLITÉCNICO DE PORTALEGRE
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
Luís Mourinho, Marco Mestre, Duarte Caldeira

MITRA: A Metadata Aware Web Search Engine for Digital Libraries

Jorge Machado — Escola Superior de Tecnologia e Gestão de Portalegre
José Borbinha — Instituto Superior Técnico

Resumo

MITRA is an open-source advanced search engine developed by the BND, the National Digital Library initiative. It supports several services, namely the generic search service for the National Library of Portugal, BUSCA (http://busca.bn.pt).

MITRA can harvest and index on-line contents in multiple formats, as also structural and descriptive information in XML. If each site to index provides structural and descriptive information coded in XML, MITRA can be easily configured to index it accordingly, and use that information to enrich the searching functions or the presentation of the results.

In the case of the collection of the BND (one of the collections available for searching in the service BUSCA), all the resources (both digitized and born-digital) have descriptive records in UNIMARC or Dublin Core, coded in XML. Also, all the digitized resources have a structural description in METS, which is an XML schema too, that makes it possible to describe the images according their location in the work (page number, chapter, part, etc.). In these cases, when the images are processed by OCR, the resulting text for each page can be also referenced by the METS structure, for convenient content indexing (even if the OCR has errors, it is useful for this purpose). All these schemas of descriptive and structural descriptions can be parameterized in MITRA, so it can take it all in consideration for the searching functions and presentation of the results (especially for ranking and clustering).

MITRA is based on the LUCENE basic search engine, but it uses also MySQL for internal databases. Besides the human search interface, the MITRA package provides also service for Z39.50, SRW/SRU, and OAI-PMH (from which all the descriptive metadata is available in Dublin Core).

Finally, MITRA has also a fully functional management interface available as web-services, making it possible to be controlled by external services. This is very convenient for applications using MITRA as their own indexing and public search service, like for example in DEPTAL, another open-source solution of the BND for institutional repositories.

MITRA: A Metadata Aware Web Search Engine for Digital Libraries

Jorge Machado, José Borbinha

Biblioteca Nacional Digital
http://bnd.bn.pt

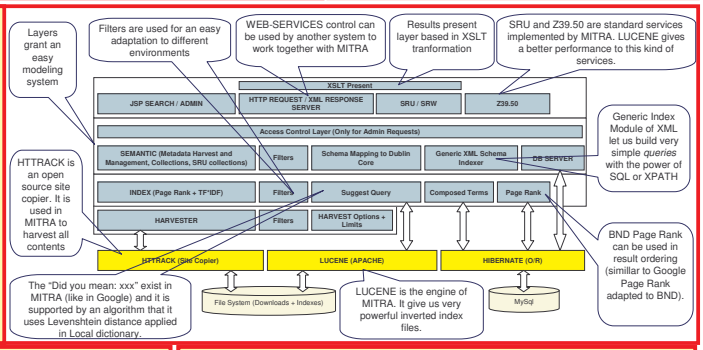
MITRA is an open-source advanced search engine developed by the BND, the National Digital Library initiative. It supports several services, namely the generic search service for the National Library of Portugal, BUSCA (http://busca.bn.pt).

MITRA can harvest and index on-line contents in multiple formats, as also structural and descriptive information in XML. If each site to index provides structural and descriptive information coded in XML, MITRA can be easily configured to index it accordingly, and use that information to enrich the searching functions or the presentation of the results.

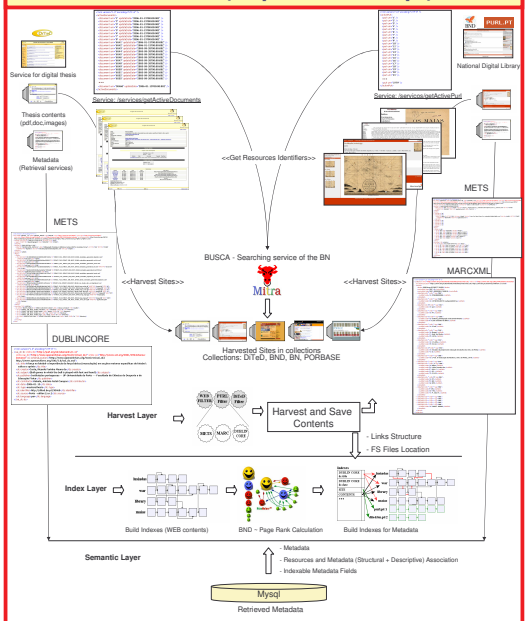
In the case of the collection of the BND (one of the collections available for searching in the service BUSCA), all the resources (both digitized and born-digital) have descriptive records in UNIMARC or Dublin Core, coded in XML. Also, all the digitized resources have a structural description in METS, which is an XML schema too, that makes it possible to describe the images according their location in the work (page number, chapter, part, etc.). In these cases, when the images are processed by OCR, the resulting text for each page can be also referenced by the METS structure, for convenient content indexing (even if the OCR has errors, it is useful for this purpose). All these schemas of descriptive and structural descriptions can be parameterized in MITRA, so it can take it all in consideration for the searching functions and presentation of the results (especially for ranking and clustering).

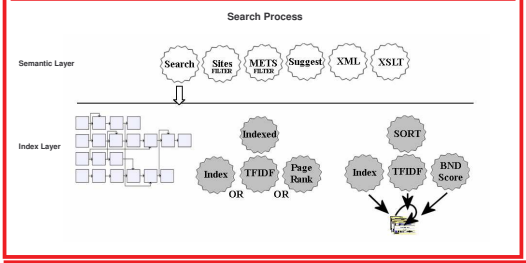
MITRA is based on the LUCENE basic search engine, but it uses also MySQL for internal databases. Besides the human search interface, the MITRA package provides also service for Z39.50, SRW/SRU, and OAI-PMH (from which all the descriptive metadata is available in Dublin Core).

Finally, MITRA has also a fully functional management interface available as web-services, making it possible to be controlled by external services. This is very convenient for applications using MITRA as their own indexing and public search service, like for example in DEPTAL, another open-source solution of the BND for institutional repositories.

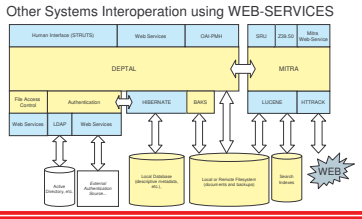


MITRA na BN (http://busca.bn.pt)…





MITRA em acção…



BIBLIOTECA NACIONAL
inesc id lisboa
INSTITUTO SUPERIOR TÉCNICO

The Unimarc Metadata Registry

José Borbinha — INESC-ID
Hugo Manguinhas — INESC-ID

Resumo

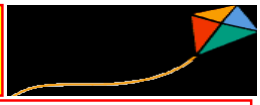
This poster describes the first steps in creating a metadata registry for the UNIMARC formats.

This registry aims to hold formal descriptions of the structure of the formats, keeping track of their versions, as also the register of the textual descriptions in multiple languages. These structural representations are recorded in XML.

This poster gives a special focus to the results already available for the bibliographic format: its on-line textual publication and the automatic validation of bibliographic records.

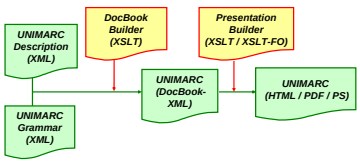
The UNIMARC Metadata Registry

José Borbinha, Hugo Manguinhas

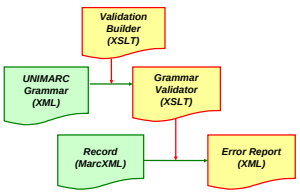


- **UNIMARC** is a family of metadata schemas with formats for descriptive information, classification, authorities and holdings. The UNIMARC activities are an **IFLA Core Activity**, the ICABS: IFLA-CDNL Alliance for Bibliographic Standards.
- In 2004 the **National Library of Portugal** (BN) started an activity to develop a UNIMARC Registry. The purpose is to support the processes around the evolution of the **UNIMARC formats** and provide a reference point for the professionals and organizations using it.
- The first target was on the bibliographic format, especially the development of a formal description of its **structure and grammar in XML**. The application of those results was tested and validated in two practical cases:
  - The on-line textual publication of the format, in this moment in Portuguese and English languages
  - The automatic validation of **bibliographic records**.
- The system for the on-line publication provides **persistent URN identifiers** for the most recent version of the elements of the format. In this moment those URN are resolved only for HTML pages, but in the future it'll be possible to return also the same information in a structured XML reply.
- A standalone tool, **MANGAS**, is available for the automatic validation of bibliographic records. This tool validates the integrity of a **UNIMARC** record and it's information, supporting also features to provide detailed reports and correct records. The engine is also used in an equivalent on-line service.
- The next steps of this activity will be:
  - To revise and finish the formal UNIMARC Registry according to the **ISO/IEC 11179** (formal description of terms, revision of the URN space, etc.)
  - To develop a stable web-service for machine access
  - Add more UNIMAR formats, versions and languages, and formalize the descriptions of changes between versions.

UNIMARC Publishing Workflow



UNIMARC Validation Workflow



The UNIMARC Registry

Purpose

- **Publishing of Human Readable Versions of UNIMARC**
  - **Multilingual**
  - **Multi-version**
  - **For all the formats**
- **Publishing of Machine Readable Versions of UNIMARC**
  - **Grammar (structured rules)**
  - **Descriptions (structured text)**

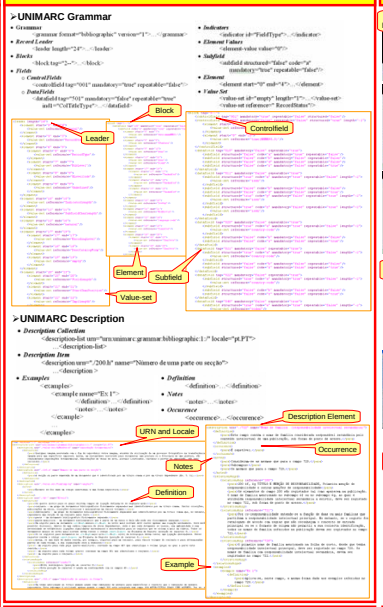
Main Concepts

- **Each format has its own :**
  - **XML grammar file**
  - **URN space**
  - **Version control**
  - **Multilingual, having for each language:**
    - **Its own XML description file**
    - **Its own URN sub-space**

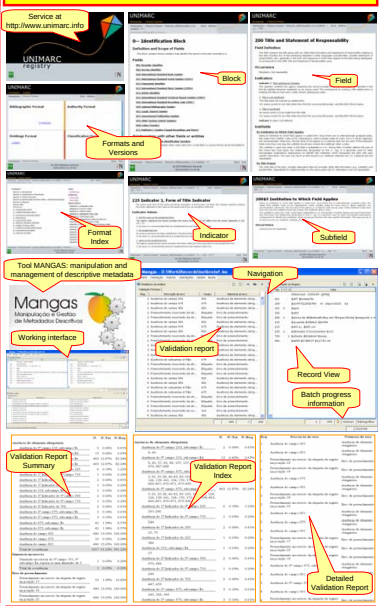
Applications

- **Online Publishing**
  - **For Humans (access to a readable description any version of the format in any language)**
  - **For Systems and Machines (access to structured descriptions of the format for building user interface applications)**
- **Record Validation**
  - **Quality Control Services (validation of the consistency of records according to any version of the format)**
  - **Editing and Cataloguing Services (automatic configuration of editing UNIMARC interfaces for professionals)**

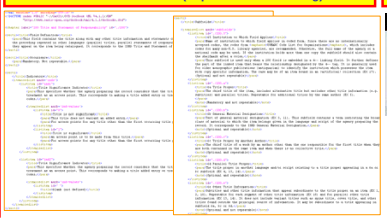
UNIMARC in XML



Applications



UNIMARC in DocBook (http://www.docbook.org)



Mangas On-line (Diagnostic Service)

IFLA · Biblioteca Nacional · Porbase · Biblioteca Nacional Digital http://bnd.bn.pt · inesc id lisboa · Instituto Superior Técnico

Índice de Autores