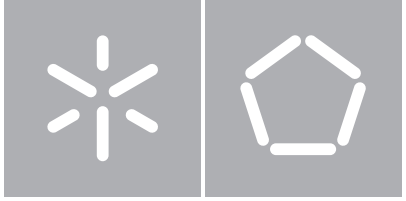
Universidade do Minho
Escola de Engenharia

Cláudia Sofia de Barros Gomes

Geração de Esqueletos Kettle a partir de Modelos BPMN usando Padrões ETL

Outubro de 2014



Universidade do Minho
Escola de Engenharia
Departamento Informática

Cláudia Sofia de Barros Gomes

Geração de Esqueletos Kettle a partir de Modelos BPMN usando Padrões ETL

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor Doutor Orlando Manuel de Oliveira Belo

Outubro de 2014

Anexo 3

DECLARAÇÃO

Nome

_____

Endereço electrónico: _____ Telefone: _____ / _____

Número do Bilhete de Identidade: _____

Título dissertação □/tese □

_____

_____

_____

Orientador(es):

_____

_____ Ano de conclusão: _____

Designação do Mestrado ou do Ramo de Conhecimento do Doutoramento:

_____

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE/TRABALHO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA TESE/TRABALHO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, , MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE;

3. DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA TESE/TRABALHO

Universidade do Minho, ___/___/_____

Assinatura: _____

Agradecimentos

Antes da presente dissertação começar, gostaria de demonstrar o meu apreço e agradecer a todos quantos, de uma forma ou de outra, me ajudaram nesta etapa da minha vida e me deram força durante todo o caminho para levar esta tarefa a bom porto:

- À minha família mais próxima – pais, avós e irmão a quem sou eternamente grata - que sempre me prestou apoio, sempre cuidou de mim, sempre me proporcionou todas as condições para que eu chegasse até aqui e que sempre teve paciência e afeto para todos os meus momentos de frustração quer durante esta fase quer durante todo o curso;
- Ao meu orientador, o Professor Doutor Orlando Belo, que me guiou, que sempre acreditou em mim mesmo quando eu nem sempre o fazia e que muitas vezes me sossegou e tranquilizou em relação ao que eu estava a fazer e a quem pude sempre recorrer quando necessitei de ajuda;
- Aos meus amigos Helena Soares e André Murta - por quem nutro enorme apreço e amizade - por toda a paciência, força, carinho, ajuda, preocupação, orientação e amizade, sem os quais me teria sido imensamente mais difícil e custoso terminar esta dissertação e que me deram muitas vezes a motivação necessária;
- À Yesika Reinolds e ao Pedro Castro que durante a minha estadia em Lisboa e durante um dos períodos mais difíceis da minha dissertação (e mesmo após), me prestaram auxílio e apoio e me deram força para continuar e seguir com o que tinha de ser feito;
- Ao Miguel Guimarães, pela paciência que teve comigo nas muitas questões que lhe pus e a ajuda que me deu em alturas críticas e sem a qual este trabalho não teria levado o rumo que levou;
- A todos os meus amigos e colegas que porventura me possam estar a escapar no momento mas que de uma forma ou de outra me deram palavras de incentivo e me guiaram durante todo este processo.

Um muito obrigado e bem-haja a todos.

Resumo

Geração de Esqueletos Kettle a partir de Modelos BPMN usando Padrões ETL

BPMN (*Business Process Modelling Notation*) foi uma linguagem implementada pela OMG para descrever de uma forma *standard* os processos de negócio de uma empresa. Esta oferece uma plataforma de fácil entendimento para os seus utilizadores, permitindo modelar os seus processos de negócio de uma forma simples e descritiva, independentemente da ferramenta utilizada. A BPMN apresenta-se como uma solução bastante adequada de modelação de sistemas de ETL pelo facto de disponibilizar uma formalização bastante completa para meta-modelos e, assim, ser capaz de acolher a modelação de tarefas ditas *standard* (padrões) em sistemas de ETL. Desta forma, consegue-se modelar um processo ETL de raiz utilizando modelos bastante simples, com um nível de abstração elevado, mas suficientemente descritivos relativamente aos seus conteúdos e tarefas. O objetivo deste trabalho de dissertação foi a modelação em BPMN de sistemas de ETL, recorrendo a padrões ETL, e a sua especificação através de uma linguagem formal de descrição de padrões, para que, a partir desse modelo, se pudesse gerar um "esqueleto" – a estrutura base de um sistema de ETL - capaz de ser executado e validado numa ferramenta comercial de ETL. Dessa forma, consegue-se reduzir substancialmente o tempo e os custos de implementação do próprio sistema de ETL.

Abstract

Creation of Kettle stubs from BPMN Models using ETL Patterns

Business Processing Modelling is a language developed by OMG to describe, in a standard approach, the business processes of a company. This language provides its users a framework of easy understanding, allowing them to model their business processes in a simple and descriptive manner independently of the tool used to model. The BPMN presents itself as a quite adequate solution for the modelling of ETL systems since it has a very complete formalization of meta-models and thus being able to represent the modelling of tasks said to be "standard" (patterns) in an ETL system. Through this, we are able to model a process of ETL from scratch using very simple models with a high level of abstraction but descriptive enough considering its content and tasks.

The purpose of this dissertation was the modelling of ETL systems in BPMN using ETL patterns and its specification through the use of a formal language specific for the description of patterns so that, with that model, a "stub" - the base structure of an ETL system - can be created in order to be validated and executed in a commercial ETL tool, thus substantially reducing the time and cost of the implementation of the ETL system itself.

Índice

Índice de Figuras

Lista de Siglas e Acrónimos

BPMN	- Business Process Modelling Notation
BPEL	- Business Process Execution Language
CDC	- Change Data Capture
DCI	- Data Conciliation and Integration
DM	- Data Mart
DQE	- Data Quality Enhancement
DW	- Data Warehouse
ETL	- Extract Transform Load
IDL	- Intensive Data Loading
RPC	- Redes de Petri Coloridas
SKP	- Surrogate Key Pipeline
SOA	- Service Oriented Architecture
XPDL	- XML Process Definition Language
XSD	- XML Schema

Capítulo 1

Introdução

1.1 A Importância da Modelação de Padrões em ETL

Desenvolver uma solução ETL é uma tarefa complexa e morosa, que consome recursos, quer humanos quer financeiros, em grande volume (cerca de 70% dos recursos dedicados à construção de um projeto de *Data Warehouse* - DW) (Belo e Oliveira, 2013). Mais ainda, a reutilização de elementos de outros projetos de sistemas de povoamento é praticamente impossível, já que cada sistema ETL é normalmente desenvolvido para um DW em específico e para determinados utilizadores em particular. Apesar de haver inúmeras ferramentas completamente dedicadas ao *design* de um sistema ETL, estas continuam a ser proprietárias tais como as linguagens/notações que utilizam. Como estas ferramentas impõem especificações e construtores muito específicos, próprios da empresa que a construiu e promove, dificultam um pouco o seu processo de aprendizagem e aplicação, bem como o diálogo no processo e compreensão dos esquemas produzidos com os clientes finais. Uma vez que este tipo de ferramentas exigem a utilização e aplicação da sua própria linguagem, isso inibe também a interoperabilidade entre ferramentas ETL, tornando a migração dos sistemas de ETL produzidos para outras plataformas mais difícil e consequentemente mais custosa (Akkaoui e Zimányi, 2009). Assim, é importante que se crie um modelo que permita reduzir a complexidade, erros e custos associados com o projeto e desenvolvimento de um sistema ETL e que, tanto quanto possível, seja em termos aplicacionais de âmbito universal (Trujillo e Luján-Mora, 2003).

Porém, uma nova abordagem tem estado a ser desenvolvida, promovendo uma modelação conceptual dos sistemas de ETL baseada em meta-modelos, ou seja, modelos que representem tarefas comummente utilizadas (standard) no desenvolvimento de sistemas ETL. Como referido em (Belo e Oliveira, 2013), um meta-modelo é "composto por um conjunto de atividades atómicas com um fluxo pré-estabelecido que, tendo por base um conjunto de parâmetros de entrada, produz o respetivo resultado". Desta forma, poder-se-ão criar modelos conceptuais mais simples e fáceis de compreender uma vez que o cliente é abstraído de pormenores de implementação e de especificação, minimizando assim ambiguidades e aumentando a eficiência da implementação de um sistema ETL (Oliveira e Belo, 2013a) (Belo e Oliveira, 2013). Durante os últimos anos, alguns autores propuseram metodologias e notações próprias, como foram os casos de Vassiliadis et al., (2002), por exemplo, que criaram a sua própria notação para modelação conceptual de ETL e de Trujillo e Luján-Mora (2003), que fizeram uma adaptação ao UML de forma a poder ser utilizado na modelação em ETL. No entanto e apesar do esforço efetuado, estes trabalhos não permitem ainda um processo de modelação de sistemas ETL fácil e intuitivo.

O BPMN – *Business Process Modelling Notation* – é um *standard* da OMG[1], muito utilizado na modelação de processos de negócio das empresas. A BPMN possui uma semântica bem definida que permite a criação de modelos descritivos, facilmente compreendidos pelos clientes, sendo uma notação independente da ferramenta utilizada. Sabendo que um processo de negócio "denota a coleção de atividades ou tarefas estruturadas e relacionadas que produzem um determinado serviço específico ou produto para um cliente em particular" (Akkaoui e Mazón, 2012) e considerando que um processo ou tarefa de desenvolvimento de ETL pode ser considerado um tipo particular de um processo negócio, a modelação de sistemas ETL em BPMN apresenta-se como uma boa alternativa às outras abordagens, principalmente devido às suas características relacionadas com o uso em larga escala, ser altamente descritiva, fácil e de compreender e de não depender da ferramenta utilizada. Mais importante ainda, a versão BPMN 2.0 trouxe um *standard* em termos de meta-modelos em BPMN e disponibiliza um mecanismo que permite estender os elementos da notação através da adição de novos elementos. Esta possibilidade permite formalizar conceitos, caracterizando os processos de ETL através da composição de elementos atuais com novos elementos, sem comprometer a integridade semântica dos modelos criados. Por último, esta última versão da BPMN

ii

[1] http://www.omg.org

providencia também os mecanismos necessários para produzir modelos mais detalhados a partir dos modelos conceptuais através da BPEL – *Business Process Execution Language* – permitindo assim a validação conceptual através da execução dos processos (Belo e Oliveira, 2013) (Oliveira e Belo, 2013a).

1.2 Motivação e Objetivos

A motivação adjacente a este trabalho esteve relacionada com o facto de ainda não existir uma clarificação no que toca à modelação de sistemas de ETL. Tal significa que, apesar de existirem várias propostas de notações e de classificações de processos ETL, ainda continua a falhar a existência de uma metodologia *de facto*, que permita uma modelação mais fácil de sistemas ETL. É neste aspeto, que consideramos essencial, que a definição dos meta-modelos que caracterizam um sistema ETL é importante, uma vez que permitirão um maior nível de abstração e, por consequência, serão mais fáceis de "ler" pelos clientes das aplicações em desenvolvimento, permitindo aumentar a eficiência dos próprios projetos de ETL. Daí que o principal desafio será definir o meta-modelo em si e caracterizá-lo para que possa ser facilmente utilizado.

A BPMN é uma linguagem de modelação padrão, bastante simples de aprender, de entender e de aplicar, e que, pelas vantagens que traz, tem sido utilizada em vários trabalhos, inclusive na modelação de sistemas de ETL (Belo e Oliveira, 2013) (Akkaoui e Mazón, 2012). Assim, o objetivo principal deste trabalho consistiu, em termos gerais, no estudo dos vários padrões *standard* que podem ser encontrados em sistemas de ETL e na definição dos seus correspondentes meta-modelos para serem utilizadas em especificações de sistemas ETL em BPMN. No entanto, apesar da utilidade da modelação conceptual desses sistemas, existe uma falta de aproveitamento de toda a especificação realizada até ao momento. Isto é, o esforço empenhado na concretização definição do modelo conceptual não teve até agora uma compensação visível no plano da implementação de um sistema deste género, sendo desperdiçadas, de certa forma, várias das potencialidades que um modelo conceptual pode oferecer. Assim, pretendeu-se apresentar uma solução para esse problema. Mais concretamente, pretendeu-se desenvolver uma linguagem de descrição comportamental de padrões de ETL (usados na modelação conceptual) que possibilitasse a incorporação de uma maior

semântica comportamental nos diversos elementos modelados, mais formal, capaz de fornecer mais detalhes sobre o padrão ETL e como este se comporta. Posteriormente, o objetivo seguinte foi satisfeito com o desenvolvimento de uma ferramenta de tradução que, a partir de um modelo ETL especificado em BPMN, desse origem a um esqueleto – um protótipo - "físico" de um sistema ETL gerado de forma (semi)automática e que pudesse ser executado numa das ferramentas de desenvolvimento de sistemas de ETL atualmente existentes no mercado, como, por exemplo, o *Oracle Warehouse Builder* da Oracle, *Data Services* da SAP *Business Objects*, *IBM Information Server (Datastage)* da IBM, *SAS Data Integration Studio* da SAS Institute Newest, ou o *PowerCenter Informatica* da Informatica, entre muitos outros.

1.3 Estrutura do Documento

Esta dissertação foi estruturada por forma a provir ao leitor uma pequena introdução ao se tema base (modelação conceptual de sistemas ETL), progredindo gradualmente para o contexto por detrás do tema (investigação na área da modelação conceptual de padrões de ETL), apresentando depois um exemplo de modelação conceptual de padrões de ETL numa linguagem específica, nomeadamente em BPMN, seguindo-se depois para um caso de estudo prático. Depois de termos feito neste capítulo a apresentação do tema e da problemática relacionada com a modelação de sistemas ETL, apresenta-se de seguida, no capítulo 2, outros trabalhos efetuados na área. Este capítulo está estruturado em 3 secções, cada uma apresentando uma linguagem de modelação diferente, nomeadamente as linguagens Reo, Redes de Petri Coloridas e a *Business Process Modelling Notation* (BPMN). Depois, no capítulo 3 o leitor poderá encontrar a linguagem específica a ser utilizada no âmbito do tema da dissertação, passando-se de seguida à apresentação de uma extensão a essa linguagem (necessária para a adaptar à modelação de sistemas ETL), bem como um exemplo específico da modelação de um padrão de tarefas de ETL (o padrão de CDC). Já no capítulo 4, são apresentadas algumas das desvantagens dos modelos conceptuais em BPMN e de como estas poderão ser ultrapassadas, sendo apresentada e detalhada uma linguagem de descrição de padrões.

No capítulo 5 propõe-se um caso de estudo prático para comprovar a abordagem introduzida no capítulo anterior, sendo descrita a formulação de uma ferramenta de conversão para esquemas

conceptuais em BPMN. Finalmente, no capítulo 6, são retiradas algumas conclusões sobre o trabalho realizado e resultados obtidos. Para além disso são também apresentadas algumas sugestões para trabalho futuro, como algumas propostas de melhorias para a solução proposta nesta dissertação.

Capítulo 2

Modelação de Sistemas

Tal como referido no capítulo anterior, a modelação conceptual em ETL não é um assunto novo, tendo sido efetuados alguns avanços sérios nessa área nos últimos anos. Ao longo das próximas secções serão abordados alguns desses avanços (os mais relevantes), nomeadamente em termos de linguagens e métodos utilizados na modelação conceptual de sistemas, para que melhor se possa compreender o âmbito e o contexto do tema de dissertação exposto anteriormente.

2.1 O Uso da Reo

A Reo é uma a linguagem de coordenação, desenvolvida para modelar a coordenação e composição de serviços de uma aplicação. Mais concretamente, a Reo surgiu da necessidade de definir a estrutura composicional, e sua respetiva coordenação, dos *Web Services* que uma aplicação integra no seu funcionamento (Lazovik e Arbab, 2010) (Meng e Arbab, 2007). A Reo incorpora a noção particular de composição, que tanto permite composição coordenada de serviços individuais como permite também a composição de processos de negócio (Lazovik e Arbab, 2010). Com a Reo há todavia uma preocupação, não com o funcionamento ou composição interna das suas atividades ou elementos, mas com a coordenação de todo o processo e na interação entre cada um dos componentes e serviços (Oliveira e Belo, 2013b).

A Reo incorpora uma série de elementos na sua especificação. Os **nodos** (livremente traduzido do original "*connectors*") que representam os elementos coordenativos podem ser compostos de forma a elaborar outros nodos, mais complexos, através de uma operação de *join* (Lazovik e Arbab, 2010). Os nodos são compostos por elementos mais simples e permitem a ligação entre **canais**. Estes, por sua vez, permitem fazer a comunicação e sincronização entre os nodos, sendo a única forma de

transferir dados entre si. Os canais podem ser de dois tipos: *fonte* e *destino* (no original "*source*" e "*sink*" respectivamente). O primeiro aceita dados no seu canal, enquanto que o segundo permite que os dados sejam enviados a partir desse canal, sendo possível que ambas as extremidades do canal possam ser do mesmo tipo, isto é, ambas podem ser fonte ou destino. Os canais podem ser de vários tipos, permitindo, assim, diferentes tipos de comunicação: comunicação síncrona e assíncrona. Uma abordagem mais específica a este assunto não é relevante no âmbito da presente dissertação, mas mais detalhes sobre este assunto poderão ser encontrados em (Arbab, 2004).

Para além disso, a Reo disponibiliza ferramentas gráficas e mecanismos de validação dos modelos o que permite verificar os erros existentes. Isto é apenas possível devido ao formalismo da Reo. O seu formalismo pode ser representado pelo uso de autómatos de restrições que captam a semântica da linguagem permitindo, por isso, a sua validação. Através da construção deste tipo de autómatos, os modelos Reo podem ser vistos como um conjunto de transições, de nodo para nodo, e que representam os resultados das várias operações efetuadas nos nodos, permitindo assim a definição de fluxo de dados (Oliveira e Belo, 2013b). Assim sendo, considerando as principais características desta linguagem – baseada na coordenação e composição de elementos (sem se preocupar como é que internamente cada atividade ou componente é constituído), baseada em nodos e em canais, com um formalismo e semântica bem definidos – e considerando um sistema ETL, poder-se-á adaptar esta linguagem à modelação conceptual de um sistema ETL. Essa tentativa foi já efetuada em (Oliveira e Belo, 2013), tendo sido obtidos resultados interessantes. No entanto, a investigação efetuada neste sentido não encontrou mais nenhum resultado relevante, tendo-se verificado a aplicação de Reo mais a sistemas de larga escala baseados numa arquitetura SOA.

2.2 O Uso de Redes de Petri Coloridas

Antes de se explicar o conceito de Redes de Petri Coloridas, começar-se-á por explicar o conceito das Redes de Petri clássicas, sendo depois explicada a diferenças das Redes de Petri Coloridas, terminando depois com o paralelismo e aplicação a sistemas ETL.

As Redes de Petri são um conjunto de ferramentas de modelação com origem em Petri e que tem por base um forte formalismo matemático e uma notação gráfica fácil de entender (Žarnay e univerzita v Žiline, 2004) (Salimifard e Wright, 2001). Esta linguagem surgiu com o propósito de

modelar, descrever e simular o comportamento sistemas caracterizados por serem fortemente concorrentes, assíncronos, distribuídos, não-determinísticos e estocásticos (Silva et al., 2012).

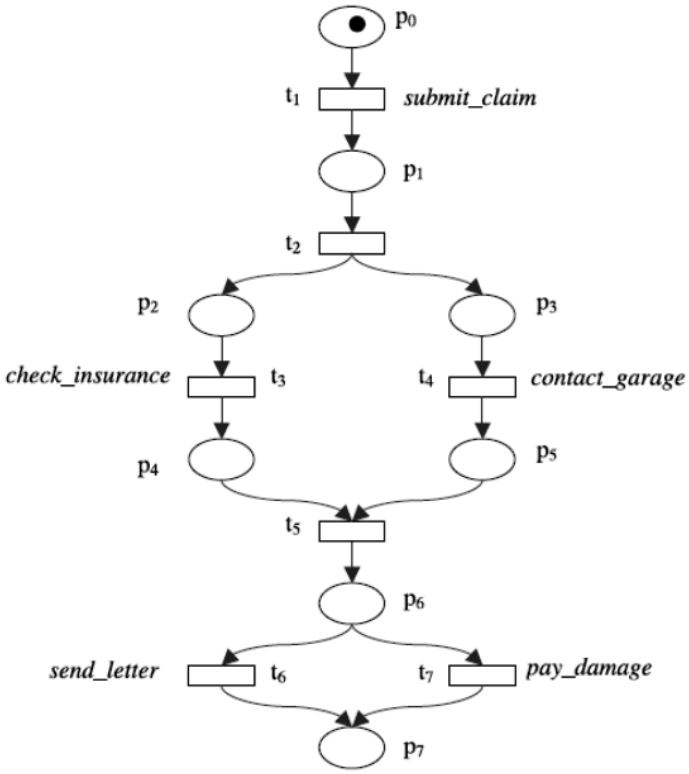


Figura 1: Exemplo de uma Rede de Petri sobre a modelação de uma queixa numa seguradora – figura extraída de (Salimifard e Wright, 2001).

Uma Rede de Petri é um grafo orientado constituído pelos seguintes elementos: locais, transições, arcos e *tokens*. Os locais são representados por círculos e são os possíveis estados ou condições do sistema. Já as transições são representadas por "caixas" ou "barras" e descrevem acontecimentos ou eventos que modificam o estado do sistema. Os arcos são a única ligação entre os locais (ou se se preferir, nodos) e os estados possíveis (transições), sendo que não é possível haver uma ligação entre nodos do mesmo tipo. Por fim, o comportamento dinâmico de um sistema pode ser modelado recorrendo ao uso de *tokens* e que são representados usando pontos pretos dentro dos nodos/locais (Salimifard e Wright, 2001). Um exemplo de uma Rede de Petri poderá ser visto na figura em baixo retirado de (Salimifard e Wright, 2001).

Apesar de ser uma linguagem bastante popular entre os investigadores académicos, as Redes de Petri são um pouco limitadas, nomeadamente em termos de modelação de dados e de tempo (Van der Aalst, 1999), tendo assim surgido algumas extensões ao longo dos anos para colmatar as referidas limitações. Umas dessas extensões são as *Redes de Petri Coloridas* (RPC), já mencionadas no início da desta secção. Nesta extensão, os *tokens* têm um tipo (que representam o tipo de dados que se quer modelar) e cujo valor é representado pela *cor do token.* Mais ainda, as transições descrevem as relações entre os valores do *tokens* dados como *input* com os valores do *output*, sendo também possível especificar pré-condições que tomam em consideração a cor dos *tokens* a ser consumidos (Van der Aalst, 1999).

Assim sendo, as RPC permitem adicionar a capacidade de modelação de dados e uma estrutura hierárquica às Redes de Petri clássicas. Consequentemente, torna-se possível criar modelos compostos por modelos mais pequenos (meta-modelos) que podem ser especificados e testados independentemente. Finalmente, as RPC são modelos executáveis, sendo possível simular o comportamento de um sistema, existindo também suporte em termos de ferramentas para simular (e também modelar) o comportamento de um sistema usando RPC (Silva et al., 2012).

Considerando que as redes de Petri permitem modelar sistemas concorrentes, assíncronos, não-determinísticos e estocásticos, e que as RPC permitem que os modelos tenham uma estrutura hierárquica e tipificada, o seu paralelismo com ETL torna-se evidente já que um sistema ETL tem muitas dessas características. Isso faz com que seja possível construir um modelo de um sistema ETL como sendo um conjunto de processos modelados em RPC (cada modelo é um processo mais pequeno), podendo cada um ser testado, validado e, posteriormente, reutilizado noutro projeto. Apesar de também não existir ainda uma investigação avançada no que à aplicação de RPC a sistemas ETL diz respeito, Silva et al. (2012) fizeram uma investigação nessa área e conseguiram efetivamente modelar e testar um padrão (*surrogate key pipeline - SKP*) de atividades em ETL com sucesso, sugerindo que mais investigação deverá ser efectuada. Na figura 2 podemos ver um exemplo de um modelo de RPC que modela uma situação muito frequente em sistemas de ETL – *Surrogate key Pipeline* (SKP).

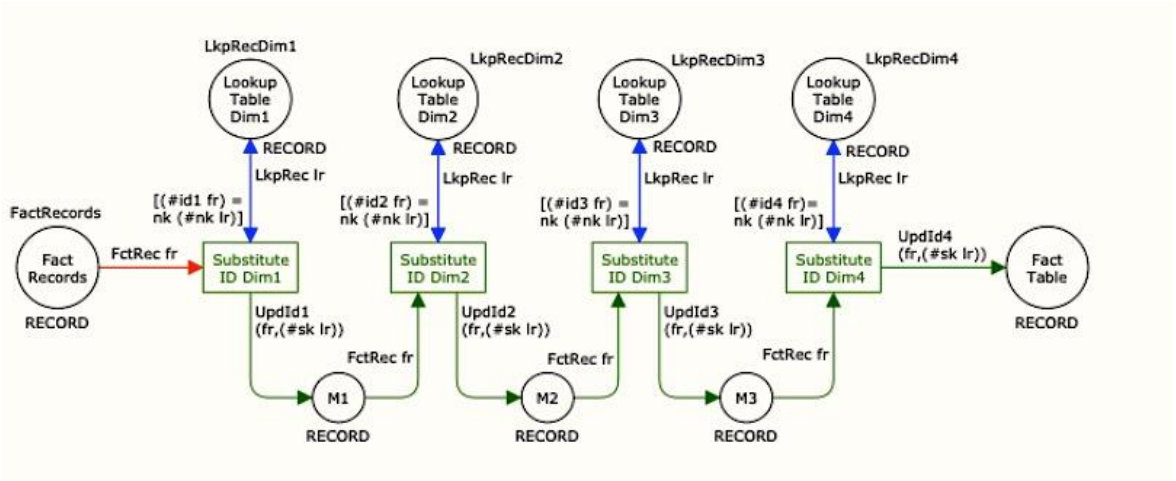


Figura 2: Exemplo de uma Rede de Petri colorida: o caso do SKP retirado de (Silva et al., 2012).

2.3 O Uso da *Business Process Modelling Notation*

Tal como referenciado na secção introdutória, a *Business Process Modelling Notation* é uma linguagem amplamente utilizada para a modelação de processos de negócio de uma empresa, sendo de fácil compreensão e utilização. Um processo de negócio no contexto empresarial é um conjunto ordenado de tarefas que descrevem um determinado processo de trabalho dentro de uma empresa, ou seja, como é uma empresa internamente trabalha.(Akkaoui e Zimányi, 2009) Estes processos de negócio eram normalmente representados recorrendo a linguagens de execução baseadas em XML. Apesar de haver várias ferramentas com suporte para a modelação, execução e análise de processos de negócio, cada tinha a sua notação e construtores próprios, sendo muitas vezes demasiado complexos e detalhados para serem usados na comunicação com diferentes organizações.

Por essa razão, a OMG[2] criou a BPMN com o propósito de simplificar e de trazer um *standard* para a modelação de processos de negócio. Esta notação tem uma semântica e uma estrutura bem-definidas e providencia uma plataforma de fácil entendimento para os seus utilizadores (Akkaoui e Mazón, 2012; Akkaoui e Zimányi, 2009), havendo também a possibilidade de executar os modelos criados e de os validar recorrendo a linguagens de execução, nomeadamente BPEL.

xi
[2] http://www.omg.org

Tal como consta em (Akkaoui e Mazón, 2012), a BPMN é constituída por alguns elementos base, mais concretamente pelos seguintes (Figura 3):

- *Atividades*, que representam uma tarefa executada como parte de um determinado processo. Poderão ser atómicas ou então perfazer um conjunto de atividades mais pequenas a que se dá o nome de *Subprocessos*.
- *Ligações*, que ligam os objetos dentro de um modelo/diagrama.
- *Eventos*, que ocorrem durante a execução de uma determinada tarefa ou processo e que poderá despoletar outros processos (por exemplo, um determinado processo só ocorre quando uma determinação condição for verificada).
- *Swimlanes,* que permitem subdividir um determinado modelo, ou diagrama, em conjuntos de processos (*piscinas*) e em *faixas* que dividem as piscinas por vários tipos de processos.
- *Artefactos, que* permitem "estender" um determinado modelo, existindo para isso três tipos de artefactos: objetos de dados, grupos e anotações.

Atualmente, a BPMN encontra-se na sua versão 2.0, tendo sofrido uma evolução bastante significativa na medida em que disponibilizou mecanismos de extensão de modelos (os artefactos), permitindo agora adicionar novos elementos aos modelos sem no entanto comprometer a integração do modelo com as ferramentas atuais. Isto acontece porque os elementos originais não são alterados (Belo e Oliveira, 2013).

Sabendo que o BPMN é bem aceite e usado no meio e que consiste modelar atividades ou processos, poderemos estabelecer uma correspondência com um sistema ETL na medida em que um sistema ETL poderá ser visto como um processo de negócio um pouco particular. Em termos de modelação conceptual de sistemas ETL usando BPMN foram já efetuados. Mais concretamente, em 2009 (Akkaoui e Zimányi, 2009), estes autores começaram por adaptar os vários elementos da BPMN às propriedades ou tarefas de um sistema ETL, abordando também o mapeamento de BPMN para BPEL de forma a poder validar os modelos construídos com BPMN. Mais tarde os mesmos autores (Akkaoui e Mazón, 2012), com base em trabalhos anteriores, propuseram a classificação de ETL em fluxo de dados e fluxo de controlo apresentando também uma extensão de um meta-modelo baseado em BPMN.

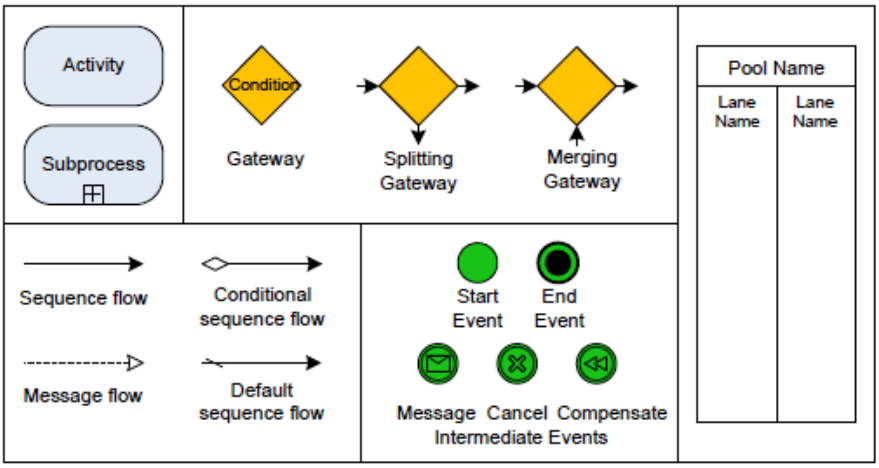


Figura 3: Elementos principais da BPMN – figura extraída de (Akkaoui e Mazón, 2012).

Mais recentemente (Belo e Oliveira, 2013; Oliveira e Belo, 2013a), com base nos trabalhos dos autores supra citados, fizeram progressos na modelação de sistemas ETL usando padrões (através do mecanismo de extensão da BPMN), tendo conseguido modelar e validar, por exemplo, um padrão de validação da qualidade de dados. Poderá ser visto esse exemplo na figura 4.

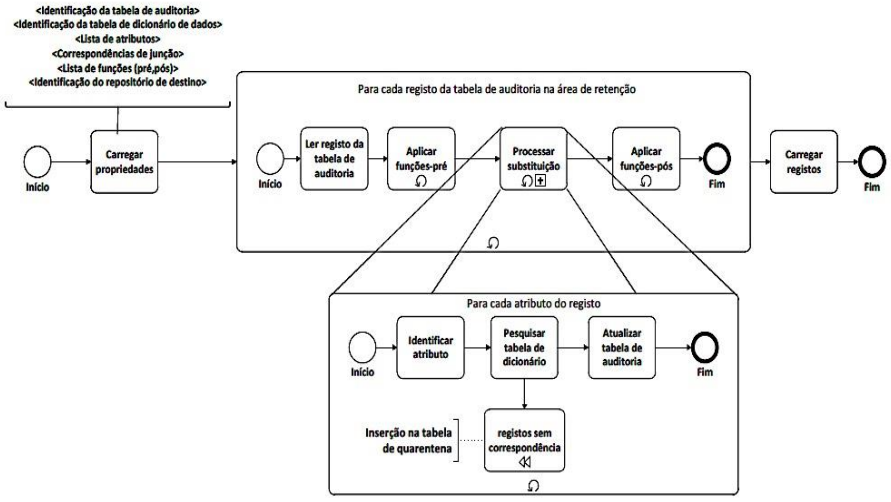


Figura 4: Exemplo de um padrão de ETL modelado em BPMN – figura extraída de (Belo e Oliveira, 2013)

2.4 Outros Trabalhos na Área da Modelação de Sistemas de ETL

Tal como pode ser concluído pela análise efetuada nas secções anteriores, a modelação de sistemas ETL não é um problema novo. Na verdade existem ainda alguns trabalhos interessantes nessa área e que não estão relacionados com as linguagens mencionadas mas que contribuíram de forma significativa para a evolução na solução do problema. Veja-se, por exemplo (Vassiliadis et al., 2003) no qual foi proposto uma *framework* com elementos próprios para modelar cenários de ETL, mais a nível lógico do que conceptual, apresentando também alguns modelos que representavam padrões de atividades em ETL. Ainda nesse ano, Simitsis (2003) apresentou uma proposta de linguagem própria de modelação de sistemas ETL, passando pelos níveis conceptual e lógico, havendo também espaço para a otimização de cenários (conjuntos de atividades) em sistemas ETL. Posteriormente, Vassiliadis et al. (2009) deram mais um contributo na modelação conceptual de sistemas ETL apresentando uma taxonomia para operações em ETL. Por último, Wilkinson et al. (2010) propuseram uma metodologia para adaptar o uso de BPMN à modelação conceptual de sistemas ETL tendo também em consideração os requisitos de um sistema como um sistema ETL.

Capítulo 3

BPMN em Sistemas de ETL

Considerando o exposto no capítulo anterior, no qual foram abordadas três linguagens diferentes para a modelação de sistemas, referindo-se as suas vantagens e desvantagens, selecionámos a BPMN como a nossa linguagem base de trabalho. A sua escolha deveu-se essencialmente a três fatores:

- Simplicidade de linguagem, de fácil compreensão por vários grupos de utilizadores, desde agentes de decisão (nomeadamente gestores, administradores entre outros) a técnicos e programadores, sendo uma linguagem *standard* na modelação de processos de negócio. Permite, ainda, criar modelos que apesar da sua simplicidade são bastante descritivos.

- A versão 2.0 da BPMN, tal como referido anteriormente, possui um mecanismo de extensão do seu próprio meta-modelo. Tal significa que novos elementos poderão ser construídos a partir de outros sem que a integração com as ferramentas existentes seja comprometida – uma característica muito vantajosa para o trabalho que pretendíamos realizar (Belo and Oliveira, 2013).

- Já existe uma história de casos de sucessos no mapeamento de BPMN a padrões ou processos de ETL (Akkaoui e Mazón, 2012; Akkaoui e Zimányi, 2009; Belo e Oliveira, 2013; Oliveira e Belo, 2013a), o que permite concluir que tal abordagem tem já casos por onde o tema da presente dissertação poderá ter como base.

Nas próximas secções será dado um exemplo mais concreto sobre a aplicação de BPMN a ETL, que foi apresentado em (Oliveira e Belo, 2013a).

3.1 Extensão ao BPMN para ETL

Para que se possa representar as operações ditas padrão na implementação de um sistema ETL, será necessário usar o mecanismo de extensão providenciado pela BPMN. Assim, poderão ser elaborados meta-modelos, cada um com o seu conjunto de atividades com um determinado fluxo, em que, tendo um determinado *input*, se obtém um determinado *output*.

Para tal, tal como explicado em (Belo e Oliveira, 2013) e (Oliveira e Belo, 2012), as extensões são feitas usando regras da notação BPMN+X que permite que seja feita a conversão para *XML Schema* (XSD) para que as ferramentas de suporte à BPMN possam interpretar os meta-modelos criados. Na Figura 5, podemos ver uma proposta a essa extensão do BPMN para se poder modelar os já referidos padrões comuns de operações em ETL. Ressalve-se que, neste caso, não estão representados todos os meta-modelos possíveis.

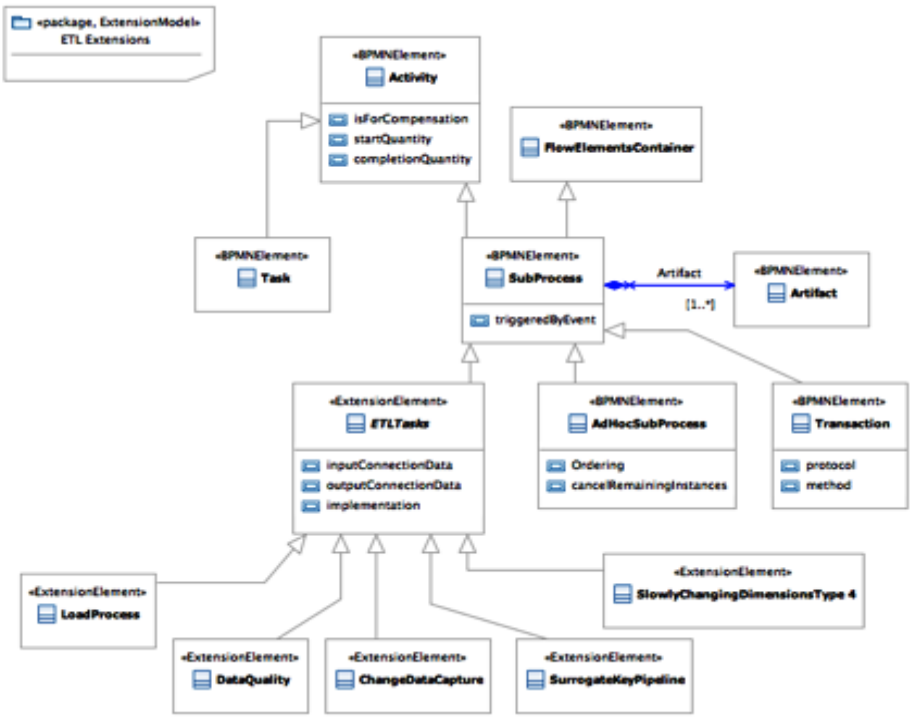


Figura 5 - Proposta de extensão ao BPMN (modelo BPMN+X) – figura extraída de (Oliveira e Belo, 2013a).

A proposta de extensão ao meta-modelo de BPMN é principalmente baseada na extensão da superclasse *Subprocess.* Para distinguir os elementos originais dos estendidos, são usados dois estereótipos: o *BPMNElement,* para os originais, e *ExtensionElement*, para extensão. Este último é usado na superclasse denominada *ETLTasks* e suas subclasses. Cada uma destas subclasses pretende representar um conjunto de tarefas típicas (padrão) em ETL que recebe um determinado parâmetro ou parâmetros de entrada e, aplicando um bloco de tarefas pré-definidas, produz um determinado resultado de acordo com a sua especificação (Oliveira e Belo, 2013a).

A grande vantagem deste meta-modelos é a de que, um arquiteto deste tipo de sistemas, poderá, num ambiente gráfico adequado, fazer *drag and drop* de instâncias dos vários elementos produzidos usando a notação proposta como se fossem tarefas normais de BPMN. Assim, o arquiteto do sistema ETL poderá ser capaz de criar um processo de ETL complexo mas de forma mais simples e clara, minimizando o tempo e erros de implementação do mesmo.

3.2 Um exemplo de um meta-modelo em BPMN

Para comprovar a utilidade e relevância dessa abordagem, na presente secção será apesentado um caso prático da aplicação de BPMN à modelação de padrões ETL. Neste caso será apresentado um processo simplificado de ETL, utilizado no povoamento de um esquema dimensional bastante simples (modelado de acordo com a metodologia Golfarelli e Rizzi (Golfarelli e Rizzi, 2009), apenas para efeitos de demonstração. Todos os detalhes do exemplo não serão aqui abordados, no entanto aconselha-se

ao leitor a consulta de (Belo e Oliveira, 2013; Oliveira e Belo, 2013a), publicações das quais foram retirados os exemplo aqui utilizados.

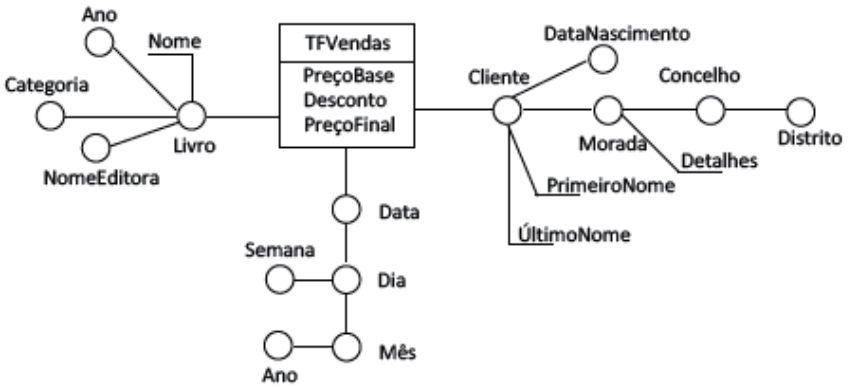


Figura 6 – Um esquema dimensional Vendas (esquema tipo "floco de neve") – figura extraída de (Belo e Oliveira, 2013).

O esquema dimensional referido está apresentado na Figura 6. Nesse esquema poderão ser vistas 3 dimensões: "Livro", "Cliente" e "Data". Além dessas dimensões existe uma outra mais (um *outrigger*), numa linha mais secundária: "Morada". Por fim tem-se, também, o eixo principal de análise que contém todos os eventos ocorridos no contexto dos dados armazenados nas dimensões de análise: a tabela de factos "Vendas". Esta tabela de factos tem três medidas de análise: o "preço base" do livro, o "desconto" sobre a venda do produto e o "preço final" de venda do livro, permitindo que os dados de uma venda de um livro por um cliente num determinado dia possam ser agregados.

O esquema de povoamento deste *data mart* (DM) poderá ser visto na Figura 7. Este esquema foi modelado de acordo com as regras da BPMN e com a extensão ao meta-modelo da BPMN proposto anteriormente (Figura 5). O povoamento é feito começando pela dimensão Morada para que a

integridade referencial se mantenha, e só depois se poderá povoar as dimensões Livro e Cliente. A dimensão Data tem um método de povoamento específico e autónomo.

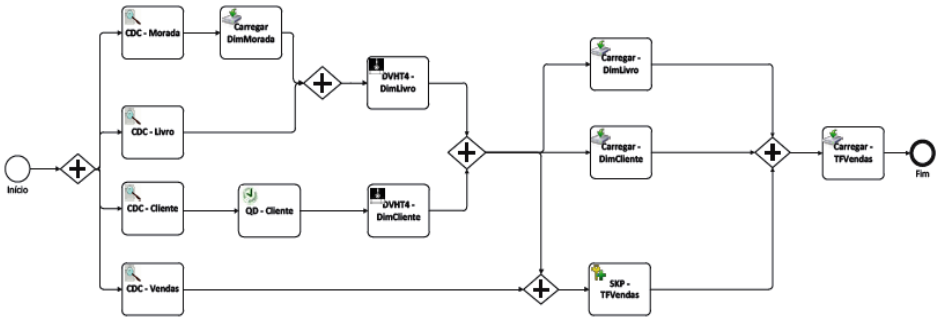


Figura 7 - Esquema de povoamento do DM *Vendas* em BPMN – figura extraída de (Belo e Oliveira, 2013).

Como poderá ser verificado pelo esquema anterior, existem quatro tarefas designadas por CDC (*Change Data Capture*) que representam um padrão de tarefas em ETL muito comum. Esta padrão de tarefas é executado sobre um ficheiro de *log* (que contém os registos das transações efectuadas e o seu tipo) de cada uma das tabelas sobre as quais o CDC é aplicado, com o intuito de capturar as mudanças efectuadas nas tabelas, isto é, extrair as operações realizadas (*insert, update* ou *delete*) com base num atributo de referência dos registos no ficheiro de *log*. Com este processo são identificados os novos registos de atualizações, desde o último povoamento, e posteriormente atualizados. Os registos são extraídos para uma tabela de auditoria.

Por fim, na Figura 8 poder-se-á ver o esquema de modelação do padrão de CDC (Oliveira e Belo, 2013a), que tem em consideração as regras de notação da BPMN. Neste esquema, pode ver-se que a primeira tarefa será de, para cada registo de uma transação presente no *log*, ler o registo. Em seguida passa-se para o subprocesso "processar transação" no qual se verifica o tipo da transação efetuada. Para cada tipo de transação são lidos todos os registos, sendo extraído o conteúdo necessário para a atividade de descodificação. No final de todas as transações serem processadas, o identificador do último registo do *log* é mantido para posteriores processos de CDC e os registos resultantes da operação do padrão CDC são enviados para a tabela de auditoria.

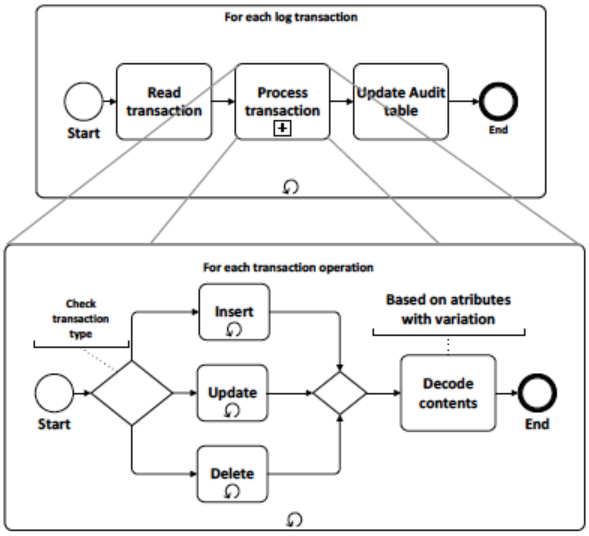


Figura 8 - Padrão de tarefas de ETL em BPMN - o caso do CDC – figura extraída de (Oliveira e Belo, 2013a).

Na Figura 7 também podemos observar outros padrões ETL. Porém, os detalhes desses padrões não serão abordados nesta secção. Apesar do esquema de povoamento apresentado (Figura 7) ser bastante simples (apenas contém alguns padrões de ETL e algumas atividades), pode-se verificar que este esquema providencia uma visão abstrata e simples de um processo de povoamento do DM, não envolvendo a descrição de imensas tarefas elementares, o que só tornaria o esquema mais confuso e de difícil compreensão e leitura (Belo e Oliveira, 2013).

Capítulo 4

Incorporação de Semântica em Padrões ETL

Apesar da utilização de padrões a nível conceptual simplificar muito os modelos criados e facilitar a comunicação entre todos os intervenientes de um projeto, permite também reduzir ambiguidades e erros desde uma fase muito inicial do processo. Porém esta abordagem conceptual carece de algo que permita a passagem para um nível com menor abstração e maior detalhe técnico como sejam o nível lógico ou mesmo o físico. Seria, pois, bastante útil que se conseguisse gerar modelos físicos a partir de modelos conceptuais criados, nos quais o programador apenas tivesse de intervir para complementar algumas partes do modelo gerado ou corrigir eventuais erros de geração.

Contudo, num modelo BPMN não existe a noção de validação ou execução, não de forma direta pelo menos, sem que para isso se tenha que recorrer à tradução do modelo para uma linguagem de execução como o BPEL ou a outras ferramentas proprietárias que forneçam esse mesmo tipo de mecanismos. Para o caso em questão, os padrões de ETL em BPMN são uma tarefa ou um subprocesso com um nome, por exemplo, "CDC" ou "Change Data Capture". Apenas isso. Uma tarefa descritiva que não possui nenhum outro adereço ou capacidade para poder dar algo mais.

Por essa razão, propõe-se neste capítulo uma solução para esse problema: a introdução de uma linguagem de descrição de padrões de ETL que forneça os pormenores necessários para a passagem de um modelo conceptual num modelo físico, isto é, uma espécie de esqueleto para um sistema ETL que possa ser mais tarde incorporado numa ferramenta comercial de ETL.

4.1 Propriedades dos Padrões

Para começar a idealizar a definição da linguagem, primeiro foi necessário analisar as tarefas mais comummente utilizadas na implementação de um sistema ETL e que tipo de parâmetros ou características as mesmas reuniam. Com essa análise, conseguiram definir-se algumas categorias de padrões e os correspondentes atributos para cada categoria. Mais concretamente, chegou-se à definição de três tipos genéricos de padrões de ETL: extratores, transformadores e carregadores. Cada categoria identificada e os padrões que dela fazem parte implicam a execução de várias tarefas e definição de propriedades, que dizem respeito a estados distintos do ciclo de desenvolvimento de um sistema ETL. Por essa razão, cada categoria tem propriedades diferentes e caracterizadoras dos diferentes estados de um processo de um sistema de povoamento.

A primeira destas categorias é responsável por identificar e extrair os dados que foram modificados nos sistemas operacionais de onde originam esses mesmos dados (inserções, atualizações ou remoções) para um determinado local de destino. Normalmente, esse destino são as tabelas de auditoria presentes na área de tratamento dos dados. Para que este padrão consiga executar as tarefas que o caracterizam, torna-se necessário fornecer alguns detalhes, nomeadamente a localização das fontes de dados bem como o caminho para o destino dos dados. Após a extração da informação é necessário, ainda, mais algum tipo de tratamento, como por exemplo conciliar dados ou mesmo resolver anomalias que os mesmos contenham. Para além disso, como hoje em dia as fontes de dados são bastante heterogéneas, é necessário ter em consideração, também, alguns detalhes sobre o tipo da fonte de dados, isto é, se é uma base de dados relacional, se é um ficheiro de texto, CSV ou XML, entre outros. Desta forma, poderemos adequar o padrão ao tipo de fonte de dados e executá-lo de forma mais eficiente.

Quanto à segunda categoria de dados, os transformadores, esta engloba todos os tipos de padrões que efetuam algum tipo de tratamento e garantia da qualidade dos dados que provêm das fontes operacionais. Algumas das suas tarefas mais frequentes são a eliminação ou tratamento de anomalias, quer estas sejam originadas valores de campos incorretamente preenchidos, que não tenham sequer sido preenchido, ou quando existirem registos duplicados. Para além disso, antes de serem carregados os dados no sistema de DW é frequente procederem-se a outras operações de transformação de dados como por exemplo *merge*, *split* ou a substituição das chaves operacionais por chaves de substituição. Tal como na primeira categoria, o padrão deverá conhecer qual a origem dos dados para que os possa transformar. Mais ainda, os dados já transformados devem ser guardados noutras estruturas de dados

presentes no sistema de povoamento, já que poderão ser depois utilizados por um outro padrão ou outra operação de transformação antes de serem carregados no DW. Alguns exemplos de padrões dentro desta categoria são o *Surrogate Key Pipeline* (SKP), que realiza uma substituição encadeada de chaves naturais, ou o *Error Handling Hub* (EHH), que suporta processos de monitorização e de tratamento de erros.

```
Etl:

'use' (elements+=Pattern)+  'on sources{' source+=Data '}'('and target{' target=Data '}')?
('with mapping source{' map+=Data '}')?
Pattern:
Gather | Transform | Load;


-- Parâmetros dos Extratores. Inclui a fonte dos dados e parâmetros de configuração para ler
as fontes de dados;
Gather:
'Gather'
('sort by'  sort = STRING ',')?
;

-- Campos para configurar a fonte de dados;
Data:
'BEGIN'
(Path | Source )','
'type=' type=STRING
'END'
;


Path:
'data=' src=STRING;

Source:
'name=' n=STRING ','
'server=' s=STRING ','
'database=' db=STRING ','
('table=' tb+=STRING ('{'(fields+=Fields)+'}')? ',')*
'technology=' tech=STRING','
'access=' a=STRING','
'user=' user=STRING ','
'password=' pwd=STRING(', port=' p=STRING)?
;

Fields:
('source' | 'target')?'fields{' (fd+=Field)+'}'
;

Field:
STRING
;

-- Parâmetros de configuração relativos aos Transformadores, incluindo de onde provêm os
-- dados,qual a operação a aplicar aos dados quais os campos que são necessários à operação
Transform:
'Transform'
```

```
'with' (action = Action)
;


Action:
'function' f=STRING (',')?
;

-- Corresponde aos Carregadores. Indica se o carregamento dos dados será feito em massa ou
por registo.
Load:
'Load from'
('each record')?
;
```

Figura 9 – A gramática da linguagem de descrição de padrões ETL

Finalmente, a terceira categoria integra os carregadores. Nesta categoria encontram-se todos os tipos de padrão e operações que são responsáveis por fazer o carregamento dos dados, já devidamente tratados, no DW. O carregamento dos dados pode ser efectuado de forma intensiva, utilizando operações do tipo BULK para o efeito, ou então registo a registo, sendo que a utilização de um em detrimento do outro depende do contexto aplicacional em questão. Exemplos que se enquadram nesta última categoria são padrões como o *Slowly Changing Dimension* (SCD), que faz o tratamento de dimensões de variação lenta, que pode também incluir a manutenção de um histórico (sendo aí referido como SCD-H), ou o *Intensive Data Loading* (IDL), um padrão que é responsável pelo carregamento intensivo dos dados, tal como referido anteriormente.

Tendo em consideração o exposto anteriormente, foi elaborada uma gramática para a linguagem que tivesse em conta todas as propriedades e características de cada uma das categorias de padrões identificadas e que pudesse, de uma forma genérica, descrevê-las suportando todas as definições específicas de cada uma das categorias. Na Figura 9 pode-se ver um excerto da gramática elaborada. Nesse excerto poder-se-á observar que, por exemplo, as fontes de dados, ou mesmo o destino dos dados, poderão ser configuradas através de uma lista extensiva de atributos (atributo *source*), caso seja uma fonte relacional, nomeadamente nome da ligação (*name*), servidor (*server*), base de dados (*database*) entre outros; caso não seja relacional, poderá ser definida (a fonte ou destino dos dados) através do caminho de um ficheiro (*data*) com os dados necessários, sendo que o tipo de fonte, isto é, se é relacional ou então outro tipo, é indicado pelo atributo *type*.

Esta linguagem é o meio que se crê que possa diminuir o "intervalo" existente entre o nível conceptual e o nível físico e conferir um aspeto mais formal à definição e caracterização dos padrões, isto é, que

permita aproveitar todas as potencialidades descritivas de um modelo conceptual, através da utilização de padrões ETL, e transportá-lo para um modelo físico posteriormente.

4.2 Aplicação da Linguagem de Descrição

Para evidenciar a utilidade prática desta abordagem, considere-se um exemplo de modelação orientada a padrões ETL. Vejamos o *data mart* apresentado na secção 3.2 (Figura 6) em que temos uma tabela de factos "Vendas" relativa às vendas de livros efetuadas por um determinado cliente, numa determinada data. Associadas a essa tabela de facos podemos encontrar as tabelas de dimensão "Livro", "Cliente" e "Data", respetivamente. Considere-se ainda também o esquema do sistema de povoamento apresentado na Figura 7 da referida secção. Nesse esquema pode-se observar que o processo de povoamento para este *data mart* começa com quatro padrões de *Change Data Capture* (CDC), que podem ser executados em paralelo e que tratam de extrair as modificações ocorridas nos registos das fontes operacionais (inserções, atualizações ou remoções) relativamente à aos clientes, livros e registos de vendas de livros ocorridas. Seguidamente, na linha da dimensão da morada e mantendo a integridade referencial, é efetuado o carregamento dos dados na dimensão "Morada" utilizando-se um padrão *Intensive Data Loading* (IDL). Só após este processo se dá início ao restante processamento dos dados: executa-se um padrão de qualidade de dados (QD ou mais comummente a sua sigla em inglês DQE – *Data Quality Enhancement*) no eixo da dimensão "Cliente" que tratará de corrigir ou eliminar qualquer ruído que os dados contenham, e, depois, executa-se o padrão ETL *Slowly Changing Dimension with History Maintenance* (SCD-H ou como no diagrama a sigla em português DVHT – Dimensão com Variação Lenta com manutenção de Histórico) que trata de manter um histórico de qualquer alteração que tenha ocorrido sobre os atributos dessa dimensão. Em paralelo e no eixo de análise da dimensão "Livro" é também executado o padrão SCD-H.

Posteriormente, após a execução paralela dos padrões SCD-H ter acabado, é executado um padrão de carregamento dos dados (IDL) para as dimensões "Livro" e "Cliente". Em simultâneo, é executado um padrão de transformação – o *Surrogate Key Pipelining* (SKP) – que efetua a substituição das chaves naturais, presentes na estrutura da tabela que contém os dados a serem posteriormente carregados na tabela de factos. Finalmente, e após se terem carregado todas as dimensões, executa-se o padrão de carregamento dos dados para a tabela de factos e o processo de povoamento termina.

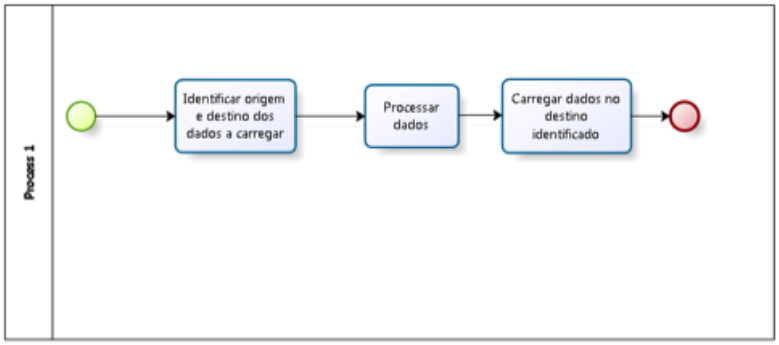


Figura 10 - Modelação do padrão de ETL *Intensive Data Loading* em BPMN.

```
-- Definição exemplo do padrão de carregamento dos dados da dimensão Cliente
--
use Load from
on source{
BEGIN
    name=teste,
    server=localhost,
    database=DSA,
    table=DimCliente,
    technology=MySQL,
    access=Native,
    user=root,
    password=abcdef,
    type=relational
END
}
and target{
BEGIN
    name=teste,
    server=localhost,
    database=Sales,
    table=DimCliente,
    technology=MySQL,
    access=Native,
    user=root,
    password=ghijklm,
    type=relational
END
}
```

Figura 11 - Exemplo da descrição de um padrão de carregamento de dados da dimensão "Cliente" incluída num esquema de povoamento.

Tal como referido na secção 3.2. o povoamento da dimensão "Data" é considerado como um processo autónomo e distinto das restantes dimensões do esquema não sendo por isso considerado no neste caso. Assim, considerando o acima exposto, tome-se como exemplo o padrão IDL para a dimensão

"Cliente" (carregamento intensivo de dados) que poderá ser descrito, em BPMN de acordo com o que se encontra na figura 10. Após a modelação do padrão, torna-se necessário, tal como já mencionado, descrever o comportamento deste padrão mais formalmente, ou seja, é necessário descrevê-lo usando a linguagem construída. Na figura 11 pode ver-se a descrição desse padrão.

Capítulo 5

Geração de Esqueletos Kettle

Ao longo do presente documento têm sido abordados vários aspetos ao nível conceptual, muitos deles relacionados com a relevância da utilização de padrões na modelação de padrões ETL, quais as falhas a nível conceptual, e uma proposta de solução (a linguagem formal de descrição de padrões ETL) para colmatar essas falhas e permitir que seja possível haver uma tradução quase automática de um modelo BPMN com o processo de povoamento um sistema ETL num esqueleto físico a ser importado numa ferramenta de ETL. Neste capítulo abordaremos o passo seguinte, isto é, demonstraremos como é que, a partir de um modelo BPMN e a sua descrição de padrões, é possível chegar efetivamente a um modelo físico, desde os passos relativos à modelação em BPMN até à ferramenta desenvolvida para realizar a conversão requerida. Esta ferramenta permitirá converter um esquema BPMN num ficheiro com um determinado formato compatível, com uma ou mais ferramentas disponibilizadas no mercado, já mencionadas anteriormente neste documento.

Para o caso de estudo prático que irá ser apresentado ao longo deste capítulo, a ferramenta escolhida para demonstrar a veracidade da abordagem seguida foi o *Kettle*[3] Isto significa que os esqueletos gerados serão orientados ao formato de ficheiro e estrutura aceites pelo *Kettle*. Este, de uma forma geral, aceita dois tipos de ficheiros: um que é o formato próprio da ferramenta e que se designa por *Kettle Transformation* (extensão. ktr) e outro em em *eXtensible Markup Language* (extensão. xml). Por ser um formato bastante utilizado e não ser uma notação proprietária, optou-se por ter como resultado da conversão dos esquemas BPMN um ficheiro no formato XML – mais detalhes sobre este processo serão dados mais à frente, na secção 5.3.

xxix
[3] http://community.pentaho.com/projects/data-integration/

A escolha por esta ferramenta teve como principais razões o facto de ser *open source* e de haver bastante documentação *online* disponível sobre os vários componentes, como e onde estes podem utilizados, e por a sua *interface* ser bastante intuitiva e de fácil aprendizagem.

5.1 Apresentação do Caso de Estudo

Para que se pudesse aplicar de forma prática a abordagem e a metodologia proposta, foi delineado um caso de estudo (fictício) que permitisse simular o processo de desenho e implementação de um sistema ETL. O caso em questão diz respeito a um esquema dimensional de dados relativo a um processo de tratamento de vendas relativas a produtos, nomeadamente calçado, com duas fontes de dados distintas e heterogéneas. O esquema dimensional referido encontra-se apresentado na Figura 12 e foi desenhado de acordo com as metodologias de Golfarelli e Rizzi (Golfarelli e Rizzi, 2009). Observando-se essa figura, poder-se-á verificar que existem três dimensões de análise:

1. "Produto" que contém a informação relevante acerca do produto, ou seja, o nome e detalhes do tipo de calçado que se vendeu;
2. "Funcionário" que, de forma similar à da dimensão "Produto", tem todas as informações relevantes sobre os funcionários que trabalham na empresa e que vendem os produtos;
3. "Data" que representa o eixo temporal do sistema.

Relativamente à tabela de factos que estará presente no sistema DW, a tabela de "Vendas", esta inclui quatro métricas de análise, nomeadamente:

1. "quantidade" a qual determina o número total de pares de sapatos vendidos;
2. "preço" que indica o preço base do par de sapatos;
3. "percent_promo" que faz referência à percentagem total de desconto efetuado sobre uma determinada venda;
4. "valor_final" que indica o valor final da compra efetuada por um cliente.

Quanto às métricas referidas será importante destacar que todas são agregáveis à exceção de "percent_promo" e "preço" as quais são apenas semi-agregáveis.

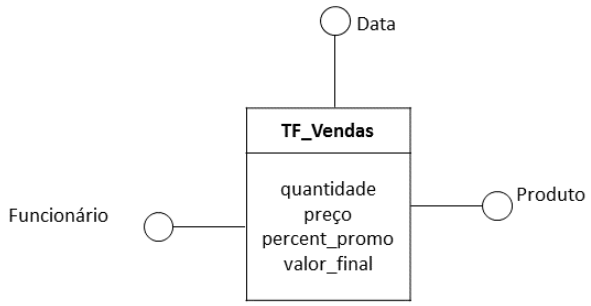


Figura 12 - Um esquema dimensional de dados do sistema de vendas de calçado.

Tendo por base o esquema dimensional definido, esquematizou-se então o processo de povoamento do sistema, recorrendo à utilização de padrões de ETL e a linguagem BPMN. O resultado foi o que se apresenta agora na Figura 13.

O processo de povoamento esquematizado encontra-se dividido em quatro faixas (*lanes*), uma por cada eixo lógico de análise identificado, nomeadamente os eixos que atuam, respetivamente, sobre as dimensões "Funcionário" e "Produto", sobre o ficheiro de dados (em CSV) com registos de vendas e, finalmente, sobre o sistema relacional, nomeadamente a fonte que contém os registos das vendas a nível do sistema operacional, sabendo-se que o fluxo de tarefas e padrões está organizado de acordo com cada um desses eixos de análise.

Assim sendo, pode verificar-se que o processo começa com a execução, em paralelo, de quatro padrões CDC (*Change Data Capture*) que capturam as mudanças ocorridas nos registos das fontes de dados, nomeadamente "CDC Vendas CSV", "CDC Funcionário", "CDC Produto" e "CDC Loja", estando cada um deles definido na faixa lógica correspondente. O tratamento da dimensão "Data" não está incluído no esquema, uma vez que se assumiu que o mesmo é feito de forma autónoma, suportado por um processo de povoamento independente. Como tal, não é necessário considerar esse caso no esquema apresentado na Figura 13.

Seguidamente o processo continua, sendo aplicadas algumas transformações aos dados de forma a garantir a sua qualidade, designadamente dois padrões de qualidade de dados "DQE Funcionário" e "DQE Produto". Estes dois padrões operam, respetivamente, sobre os dados das dimensões "Funcionário" e "Produto". Ainda sobre essas mesmas dimensões, não existindo nenhuma outra

transformação a operar sobre os dados, é executado o carregamento de dados no DW, de forma intensiva, através das tarefas "IDL DimFuncionário" e "IDL DimProduto" sobre as dimensões "Funcionário" e "Produto", respetivamente.

Considerando agora os eixos de análise referentes aos dados das vendas provenientes do sistema operacional e do ficheiro CSV, poderá observar-se que está definida uma operação de transformação após a captura de mudanças relativa ao sistema operacional das vendas. Essa transformação, "Function AddColumn", irá, tal como o nome poderá indicar, inserir uma coluna na tabela que contém os registos resultantes da tarefa "CDC Loja". Por fim, como se tem duas fontes heterogéneas de dados, ter-se-á que realizar um processo de conciliação dos registos de vendas, de ambas as fontes. Essa tarefa está representada no esquema conceptual através da tarefa "DCI" – *Data Conciliation and Integration*. Seguidamente, e atuando já sobre a estrutura de dados que contém os dados conciliados, é aplicado um padrão "SKP", já explicado anteriormente, que irá efetuar a substituição das chaves naturais da tabela de factos por correspondentes chaves de substituição. Após essa tarefa, e não havendo nenhuma outra transformação a realizar, restará fazer o carregamento dos dados no DW (("IDL TF_Vendas"), na tabela de factos respetiva.

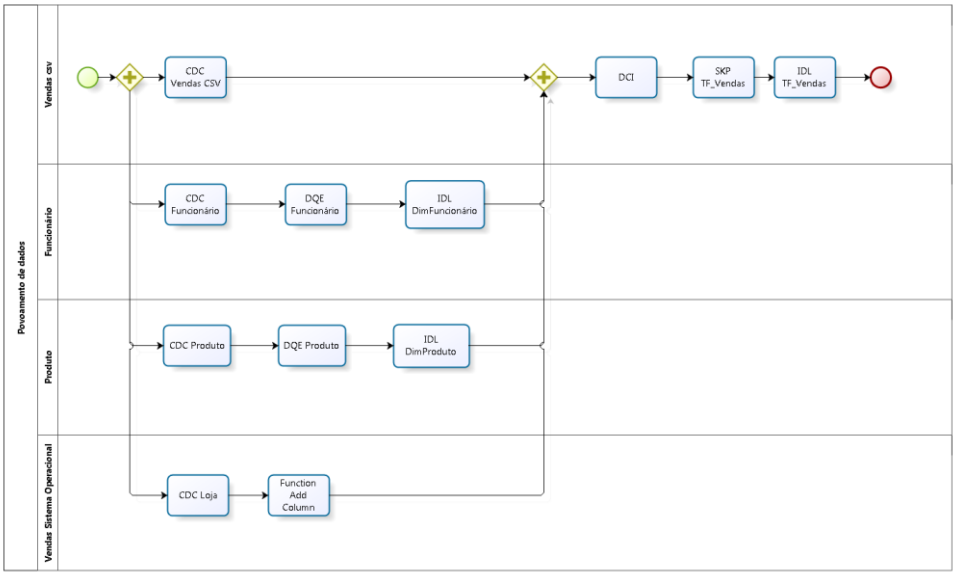


Figura 13: Esquema conceptual em BPMN do processo de povoamento do esquema dimensional de dados apresentados na figura 12.

5.2 Modelação dos Padrões em BPMN

O primeiro passo para traduzir os esquemas BPMN num esqueleto *Kettle* passa por garantir que, a nível de modelação conceptual com BPMN, todos os padrões estão bem definidos e caracterizados. Considerando o caso de estudo apresentado, isso significa que cada tipo de padrão que esteja presente no esquema de povoamento da figura 13 terá que ser modelado conceptualmente, recorrendo-se, também, ao BPMN, uma vez que a tradução do esquema geral de povoamento depende da definição conceptual individual de cada padrão, de forma a garantir que o esqueleto físico convertido seja o mais correto e adequado possível.

Observando com maior atenção o esquema de figura 13, pode-se ver que existem os seguintes tipos de padrões:

- "CDC" (*Change Data Capture*).
- "DQE" (*Data Quality Enhancement*).
- "Function" (uma operação genérica dentro da categoria dos transformadores).
- "DCI" (Data *Conciliation and Integration*).
- "SKP" (*Surrogate Key Pipeline*).
- "IDL" (*Intensive Data Loading*).

Aplicando o princípio em cima enunciado, cada um destes padrões terá de ser caracterizado através do seu correspondente esquema conceptual. Por exemplo, considerando apenas como exemplo o padrão SKP, este foi esquematizado, em BPMN da forma como a figura 14 apresenta. Todas as tarefas que estão definidas nesse esquema irão ser transformadas posteriormente em componentes da ferramenta *Kettle*, de acordo com o nome ou a descrição das mesmas (e também de acordo com a descrição formal do padrão). Os restantes padrões terão, como se poderá deduzir, o seu próprio esquema em BPMN.

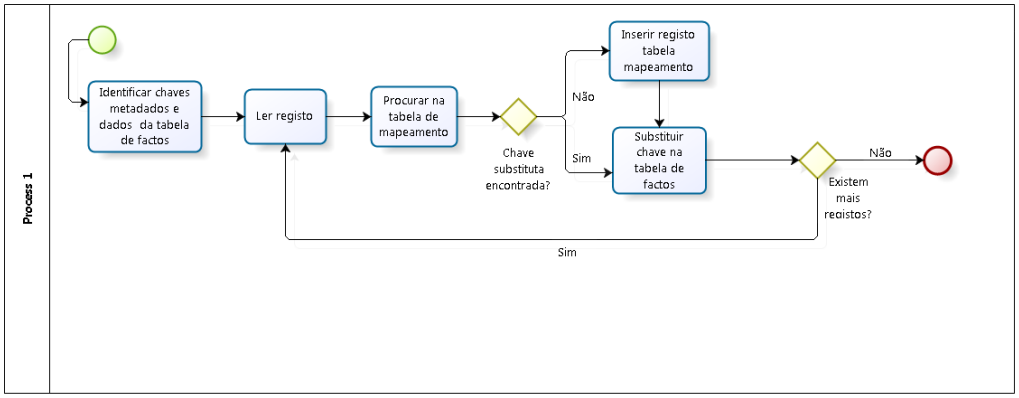


Figura 14: Esquema conceptual em BPMN do padrão *Surrogate Key Pipeline.*

Após a esquematização das tarefas em BPMN, é necessário garantir que todos os padrões do esquema sejam munidos de uma descrição de carácter mais formal. É, pois, necessário conferir aos esquemas BPMN alguma semântica comportamental, já que de outra forma não é possível traduzir de forma fiável os esquemas produzidos. Por essa razão foi introduzida uma linguagem de descrição de padrões de ETL e é através dessa linguagem que os padrões da figura 13 terão de ser descritos. Tal descrição poderá ser incorporada no esquema BPMN de acordo com as funcionalidades que as ferramentas de modelação BPMN disponibilizam, sendo que a grande maioria delas dispõe de uma caixa de diálogo (normalmente designada por "propriedades") que permite efetuar todas as configurações de uma determinada tarefa, incluindo, por exemplo, uma pequena descrição em texto. Recorrendo a essa funcionalidade e à linguagem de descrição, os padrões do esquema da figura 13 poderão ser descritos da forma como as figuras 15 e 16 apresentam.

```
-- Descrição do padrão DCI com a linguagem de descrição.
--
use Transform
with
-- Descrição do tipo de conciliação a ser feito.
function Union
on sources{
-- Descrição das fontes de dados a conciliar.
BEGIN
    name=teste2,
    server=localhost,
    database=DSA_Source1,
    table=Sale,
    technology=MySQL,
    access=Native,
    user=root,
    password=ghijkl,
```

```
    type=relational
END
BEGIN
    name=teste,
    server=localhost,
    database=DSA_Source2,
    table=Sale,
    technology=MySQL,
    access=Native,
    user=root,
    password=abcdef,
    type=relational
END

}
and target{
-- Descrição do destino dos dados após a conciliação ter sido executada.
BEGIN
    name=teste,
    server=localhost,
    database=DSA_Source2,
    table=Sale,
    technology=MySQL,
    access=Native,
    user=root,
    password=abcdef,
    type=relational
END
}
```

Figura 15 - Descrição do padrão DCI.

```
-- Descrição do padrão DQE (Funcionário) com a linguagem de descrição.
--
use Transform
with
-- Descrição do tipo de conciliação a ser feito.
function Decomposition
on sources{
-- Descrição da fonte de dado a sobre o qual se executará o padrão.
BEGIN
    name=teste,
    server=localhost,
    database=DSA_Source2,
    table=Funcionário{
            fields{
                fullName
            }
    }
    technology=MySQL,
    access=Native,
    user=root,
    password=abcdef,
    type=relational
END
}
and target{
-- Descrição do destino dos dados após a execução da função de decomposição ter sido
executada.
BEGIN
```

```
    name=teste,
    server=localhost,
    database=DSA Source2,
    table=Funcionário{
            fields{
                firstName,
                lastName
            }
    }
    technology=MySQL,
    access=Native,
    user=root,
    password=abcdef,
    type=relational
END
}
```

Figura 16 - Descrição do padrão "DQE Funcionário".

Com a realização da esquematização e da especificação dos padrões, toda a parte de modelação fica concluída. Contudo, para que o processo de conversão possa ser levado a cabo, é necessário que os modelos sejam exportados (ou guardados) num determinado formato que não o BPMN. Isto porque a extensão do ficheiro e formato no qual cada esquema de BPMN é guardado varia um pouco de ferramenta para ferramenta, apesar de a linguagem ser a mesma. Isso significa que um ficheiro guardado numa determinada ferramenta de modelação BPMN não significa que possa ser editado por outra ferramenta de BPMN diferente. Desta forma, os esquemas terão de estar num formato que possa ser melhor trabalhado e que seja um *standard*. O formato disponível para exportação é usualmente o XML ou o XPDL (*XML Process Definition Language*). Ambos são, de facto, bastante utilizados. Porém, devido a algumas restrições a nível da ferramenta de modelação escolhida, o formato XPDL foi aquele que foi escolhido para suportar o processo de exportação do modelo para o ambiente de trabalho da ferramenta de conversão. Consequentemente, cada um dos tipos de padrões, e correspondente esquema geral de povoamento, deverão ser exportados para XPDL, de forma a que ferramenta possa fazer a tradução e conversão para *Kettle* do modelo em questão. Contudo, deve-se destacar que, neste processo, apenas é necessário exportar cada tipo de padrão uma única vez, uma vez que, depois da definição desses padrões, esta poderá ser reutilizada noutros esquemas ou em outros projetos. Uma vez definidos, não é provável nem natural que a definição dos padrões mude. Aliás, um dos propósitos dos padrões ETL é precisamente o de aumentar a reutilização de componentes entre projetos ETL, pelo que não faz sentido estar constantemente a definir esquematicamente cada tipo de padrão.

5.3 Conversão dos Modelos BPMN

Antes de começar a delinear a ferramenta de conversão, isto é, saber quais as estruturas de dados e estratégias a aplicar, tivemos a necessidade de analisar a estrutura dos ficheiros exportados e que tipo de informação continham. Assim, observando o ficheiro exportado a partir do esquema geral (Figura 17), verificou-se que, por cada tipo principal de componente, por exemplo piscina (*pool*), faixa (*lane*), tarefa entre outros, existe um elemento que o representa no XPDL exportado pela ferramenta de modelação. Observando com maior atenção, pode verificar-se que todas tarefas ou atividades atómicas estão especificadas dentro do elemento "Activities", no qual cada tarefa é um subelemento "Activity". Nesse subelemento constam todas as informações importantes acerca da tarefa, como sejam o identificador único de cada tarefa, nome, o tipo de tarefa e, mais importante, a descrição da tarefa que contém, no caso do esquema da Figura 13, a descrição comportamental de cada padrão e a informação necessária para a conversão do esquema. No entanto, para além da informação sobre as tarefas, existe ainda um outro elemento pertinente presente, as ligações entre as tarefas, já que será através delas que as ligações entre os vários componentes do *Kettle* serão criadas. Essas ligações estão representadas também num elemento próprio no ficheiro, nomeadamente o elemento "Transitions", em que cada subelemento ("Transition") representa uma ligação entre duas tarefas no esquema. As principais informações que esse subelemento contém são os identificadores das tarefas que compõem a ligação e que estão nos campos "From" e "To", que representam, respetivamente, a origem e o destino da ligação.

Pode-se então concluir, através da análise do ficheiro, que o mais importante a retirar, e que irá realmente importar para a conversão do esquema, são os elementos "Activity", cuja informação irá guiar a criação de componentes, e os elementos "Transition" que irão guiar a criação das ligações entre os futuros componentes *Kettle*. Estes são os elementos que interessará tratar, devendo-se assim guardar os seus dados mais relevantes em estruturas de dados auxiliares.

Numa segunda parte, teve de ser efetuada uma análise dos vários componentes *Kettle* e como estes seriam guardados em XML (um dos formatos no qual se pode guardar uma transformação) para que se pudesse adequar aquilo que estava no esquema (Figura 13) àquilo que a ferramenta de ETL permitia utilizar e executar. Porém, não se verificou exaustivamente como é que todos os componentes *Kettle* seriam, mas verificaram-se, sim, quais os padrões, quais as tarefas de cada padrão e qual poderia ser a sua correspondente versão em *Kettle*, isto é, que tipo de componentes poderiam ser utilizados para

cada padrão. Consequentemente, sabendo quais os componentes que seriam utilizados, verificou-se um a um qual a sua estrutura no ficheiro XML gerado pelo *Kettle*, isto é, por que tipos de elementos era constituído e quais os elementos que teriam de estar obrigatoriamente preenchidos. Como exemplo, mostra-se na Figura 18 a configuração em XML de um componente *Kettle* que virá a ser muito utilizado: "TableInput" e que permite indicar uma fonte de dados do qual se irão extrair dados para outras transformações. Na figura referida apenas se mostram os atributos sobre os quais se pretende trabalhar, não estando estes preenchidos com nenhuma informação em especial.

```
<Pools>
    <Pool Id="a2f99c94-5e06-40d0-957e-ed356469d857" Name="Povoamento de dados"
Process="a62c14b2-4201-4d28-a508-5509c7a09e75" BoundaryVisible="true">
      <Lanes>
        <Lane Id="123c6d7f-921e-4c88-aa22-b6c9498910a6" Name="Vendas csv"
ParentPool="a2f99c94-5e06-40d0-957e-ed356469d857">
          <NodeGraphicsInfos>
            <NodeGraphicsInfo ToolId="BizAgi Process Modeler" Height="235" Width="1351"
BorderColor="-11513776" FillColor="-1">
              <Coordinates XCoordinate="50" YCoordinate="0" />
            </NodeGraphicsInfo>
          </NodeGraphicsInfos>
          <ExtendedAttributes />
        </Lane>
            (…)
      </Lanes>
      <NodeGraphicsInfos>
        <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="835" Width="1401"
BorderColor="-16777216" FillColor="-1">
          <Coordinates XCoordinate="30" YCoordinate="30" />
        </NodeGraphicsInfo>
      </NodeGraphicsInfos>
    </Pool>
  </Pools>

<Activities>
            (…)
        <Activity Id="2730af6b-28e0-43ab-974d-ff2b51d08b73" Name="">
          <Description />
          <Route GatewayType="Parallel" />
          <Documentation />
          <NodeGraphicsInfos>
            <NodeGraphicsInfo ToolId="BizAgi_Process_Modeler" Height="40" Width="40"
BorderColor="-5855715" FillColor="-52">
              <Coordinates XCoordinate="196" YCoordinate="117" />
            </NodeGraphicsInfo>
          </NodeGraphicsInfos>
          <ExtendedAttributes />
        </Activity>

        <Activity Id="0c45ab72-8273-43c8-82d4-c533621846aa" Name="CDC Vendas CSV">
          <Description>&lt;p style="text-align:left;text-indent:0pt;margin:0pt 0pt 0pt
0pt;"&gt;&lt;span style="color:#000000;background-color:transparent;font-family:Segoe
UI;font-size:8pt;font-weight:normal;font-style:normal;"&gt;use
Gather&lt;/span&gt;&lt;/p&gt;(…) "&gt; &lt;/span&gt;&lt;/p&gt;</Description>
          <Implementation>
            <Task />
          </Implementation>
          <Performers />
          <Documentation>&lt;p style="text-align:left;text-indent:0pt;margin:0pt 0pt 0pt
0pt;(…) /p&gt;</Documentation>
            (…)
        </Activity>
            (…)
      <Transitions>
        <Transition Id="f6c152e1-8425-47e3-9cd3-802ba74bb632" From="f4b74888-8f3d-41c5-b7b6-
f4bba3cbbdb2" To="2730af6b-28e0-43ab-974d-ff2b51d08b73">
          <Condition />
          <Description />
          <ConnectorGraphicsInfos>
            <ConnectorGraphicsInfo ToolId="BizAgi_Process_Modeler" BorderColor="-16777216">
```

```
              <Coordinates XCoordinate="169" YCoordinate="137" />
              <Coordinates XCoordinate="196" YCoordinate="137" />
            </ConnectorGraphicsInfo>
          </ConnectorGraphicsInfos>
          <ExtendedAttributes />
        </Transition>
      </Transitions>
(…)
```

Figura 17 - Excerto do ficheiro XPDL gerado a partir do esquema da figura 13.

```
<step>
    <name>Table input</name>
    <type>TableInput</type>
    <description/>
    <distribute>Y</distribute>
    <custom distribution/>
    <copies>1</copies>
         <partitioning>
           <method>none</method>
           <schema_name/>
         </partitioning>
    <connection/>
    <sql>SELECT &#x3c;values&#x3e; FROM &#x3c;table name&#x3e; WHERE
&#x3c;conditions&#x3e;</sql>
    <limit>0</limit>
    <lookup/>
    <execute each row>N</execute each row>
    <variables active>N</variables active>
    <lazy conversion active>N</lazy conversion active>
     <cluster_schema/>
     <remotesteps>   <input>   </input>   <output>   </output> </remotesteps>

      <GUI>
          <xloc>237</xloc>
          <yloc>372</yloc>
          <draw>Y</draw>
      </GUI>
</step>
```

Figura 18 - Estrutura de configuração do componente "TableInput" em XML na ferramenta *Kettle.*

Tendo sido analisada a informação mais pertinente, começámos então a delinear a estratégia para a conversão do esquema. Logo à partida, verificou-se que a conversão teria de ser, também, dividida em vários passos e que, eventualmente, poderia ser possível seguir um ou mais padrões de desenvolvimento de *software*, isto é, adaptar uma determinada solução genérica de um determinado problema a uma questão específica. Após algum planeamento, conclui-se que toda a parte de conversão propriamente dita estaria dividida em três fases:

- o *parsing* do ficheiro XPDL e criação das entidades que guardariam a informação necessária das tarefas e das transições;
- a construção intermédia de uma estrutura auxiliar que englobasse já as informações devidamente trabalhadas e mais próxima da estrutura XML do *Kettle*;
- a construção do XML a ser importado no *Kettle*.

5.3.1 Parsing do Ficheiro XPDL

Tal como explicado na secção anterior, os principais elementos que podem ser tratados são as "Activity" e as "Transition". Sendo um ficheiro XPDL baseado no *standard* XML, procurou-se uma forma prática e eficaz de extrair a informação necessária. Nesse sentido, tomou-se a opção de utilizar *XPath*, uma linguagem de consulta que, um pouco à semelhança do que SQL é para as bases de dados, permite construir, sob a forma de expressões que se aproximam muito de expressões regulares, interrogações a ficheiros XML que retornam resultados também na forma de XML. Através dessas expressões foi possível conseguir a lista de atividades (que foram guardadas em entidades denominadas "Task") e a lista de ligações entre as mesmas. Com essas listas foram construídas duas estruturas de dados auxiliares: uma que guardava as tarefas através do seu identificador (campo que faz parte dos atributos do elemento "Activity") e um grafo dirigido no qual os vértices são as tarefas que foram identificadas e tratadas (as referidas entidades "Task") e as arestas eram as ligações identificadas através do referido elemento "Transition".

Relativamente à descrição comportamental de cada padrão, a mesma pode ser capturada através de um subelemento presente no elemento do "Activity" – o subelemento "Description". Depois de ter acesso a essa descrição, temos que fazer algum tratamento, pois, neste caso específico, a ferramenta de modelação, aquando da exportação do texto da descrição comportamental dos padrões, guarda esse texto num formato que carece de ser transformado para texto puro para assim ser possível tratá-lo. A outra parte do tratamento da descrição é relativa à extração da informação relevante e criação todas entidades necessárias que permitam albergar a os dados que forem sendo recolhidos. Esta extração é realizada muito à custa da verificação da presença de certas palavras-chave na descrição comportamental e que, caso estejam presentes, são criadas determinadas entidades. Dando um exemplo concreto, se ocorrerem as palavras-chave "on sources" é criada uma entidade (ou mais, dependendo de quantas forem definidas nesse espaço dedicado) que contém todos os pormenores

relativos ao acesso a fontes dados - caminho para um ficheiro de dados (um. txt, um. xml, um. csv entre outros) ou então o utilizador, palavra-passe e localização do servidor (entre outros) da base de dados que se pretender consultar, caso a fonte seja relacional. Para além disso, é também criada uma estrutura que, caso seja necessário aceder a tabelas e a determinados atributos da mesma, guarda as várias tabelas e correspondentes atributos a que será necessário aceder ou mesmo nos quais poderá ser necessário colocar determinados dados. No final de todo o processo de *parsing* da descrição comportamental do padrão, toda a informação gerada é guardada numa entidade em separado designada por "Task Behaviour", cuja utilização será explicada um pouco mais à frente.

Será importante destacar que, este mesmo processo de tratamento do ficheiro XPDL se aplica não só para o ficheiro XPDL correspondente ao esquema conceptual do processo de povoamento bem como a cada ficheiro XPDL de cada tipo de padrão presente nesse mesmo esquema conceptual. A única diferença reside no facto de não existir tratamento do subelemento "Description" em cada ficheiro XPDL de cada tipo de padrão, pois o mesmo só é necessário a nível da tarefa que representa o padrão no esquema geral e não das atividades atómicas que compõem cada padrão (em que aliás nem está preenchido tal subelemento).

5.3.2 Tratamento dos Dados e Construção de Estrutura Intermédia

Após o tratamento de todos os ficheiros XPDL, passa-se então à fase de tratamento dos dados recolhidos. Nesta fase da ferramenta, foi avaliada a possibilidade de introduzir nesta etapa um padrão desenvolvimento de *software*, sendo que tal possibilidade se deveu a três fatores:

- No ficheiro de XML em *Kettle*, cada componente é apenas identificado unicamente pelo seu nome que será posteriormente o nome da atividade que lhe deu origem (a partir do ficheiro XPDL) e as ligações entre componentes são guardadas em XML através dos nomes dos componentes num elemento separado chamado "order". Para além disso, as ligações entre tarefas no XPDL são criadas recorrendo unicamente ao identificador de cada atividade e não possui um atributo com o nome da tarefa.

- Devido a esse fator, o tratamento das ligações e das atividades em si terá de ser feito de forma (quase) independente.

- Cada padrão iria dar origem a diferentes transformações com diferentes componentes, ou seja, a inclusão ou não de um determinado componente dependia do padrão que lhe tivesse dado origem assim como as ligações, que poderiam ser ligeiramente diferentes das ligações de origem do esquema conceptual pois as mesmas dependem também da descrição comportamental do padrão.

Analisando estes três fatores, pôde-se verificar uma similaridade muito grande com um padrão de *design* de *software,* denominado "Builder", cujo propósito é abstrair a criação complexa de objetos através de um processo de similar construção (ou seja, as mesmas funções de construção) mas que dá origem a diferentes objetos mediante determinadas condições, isto é, cada tipo de objeto é uma instância filho da classe pai que tem o processo geral de construção. Consequentemente, cada instância filho (a que normalmente se designa por subclasse) implementa as suas próprias funções de construção, dando por isso origem a diferentes objetos. Esta proximidade com o problema em mãos e por se mostrar como uma solução que facilitaria muito a sua resolução, optou-se por adotar esta estratégia para assim se poder criar diferentes objetos – estruturas de dados com a informação pronta a ser diretamente traduzida para XML puro - mediante os tipos de padrões que fossem sendo identificados no esquema conceptual geral do processo de povoamento do caso de estudo.

Assim, utilizando essa estratégia, analisando cada padrão e suas correspondentes tarefas e ligações, foi criada uma entidade filha por cada tipo diferente de padrão em que foram implementados dois métodos: um que, mediante as tarefas (entidades "Task"), elaborava diferentes entidades designadas de "Transformation", contendo a informação para construir o componente *Kettle* em XML, e outro que, mediante o grafo de ligações entre tarefas no ficheiro XPDL, construía entidades chamadas "Transition", contendo a informação relevante para também poder construir as ligações em XML. Esses dois métodos produzem dois resultados que são depois aglomerados numa única entidade, sendo esse o produto final da construção intermédia: a entidade "Pattern". Esta entidade inclui também a entidade "Task Behaviour", que foi previamente tratada na fase de *parsing* XPDL (secção 5.3.1) uma vez que contém dados mais específicos necessários para certas configurações dos componentes em XML.

5.3.3 Construção do XML Final

Finalmente, na terceira fase, e após ter já uma representação mais próxima da estrutura XML alvo, foi realizada a tradução direta dessa representação para XML. Essa tradução foi também executada em duas partes:

- uma construção do XML geral correspondente ao esquema geral de *Kettle* onde cada padrão ETL seria representado por um componente específico do *Kettle* que permite aglomerar vários componentes num só;
- a construção do XML correspondente às tarefas de cada padrão presente no esquema conceptual.

Assim sendo, por cada padrão que fosse identificado eram lançados dois tipos de processos: um por cada tipo de padrão que construía o ficheiro XML, com os componentes a que o mesmo originava e outro para construir o XML geral. No primeiro processo, acedia-se ao resultado produzido pela "conversão intermédia", a entidade "Pattern". Nesta entidade, através da lista de "Transformation" (representação próxima do que é um componente *Kettle*), foram criados em XML os vários componentes Kettle, sendo que para cada componente Kettle que era necessário foi criado um método que fizesse a tradução para XML das estruturas específicas. Para além disso, com base na entidade "TaskBehaviour", que como contém os detalhes de comportamento de cada padrão, certos elementos da estrutura XML dos componentes são preenchidos. Por exemplo, considerando o caso de a fonte de dados a ser consumida ser relacional, o componente normalmente mais utilizado é o "TableInput" (Figura 18).

Com os detalhes fornecidos com a "TaskBehaviour", os elementos "connection" e "SQL" serão preenchidos, com os detalhes de acesso à base de dados e com a expressão SQL que permitirá aceder aos dados das tabelas e atributos que forem necessários, respetivamente. Por outro lado, através da lista de "Transition", presente na entidade "Pattern" (entidade próxima do que é uma conexão/ligação em *Kettle*), a tradução para o XML necessário é quase direta sendo apenas preciso, por vezes, alguns ajustes devido a especificidades que provêm da descrição semântica de cada padrão.

Num segundo processo, é construído o esquema geral de XML, que será o esquema final importado pela ferramenta ETL. Tal como aconteceu com cada padrão individual, nesta parte acede-se igualmente à informação extraída do XDPL do esquema de povoamento do DW. Contrariamente ao que acontece

com a tradução de cada padrão, neste caso não é necessária uma conversão intermédia pois o componente a usar neste esquema será sempre o mesmo ("TransExecutor"), que é o que permite agregar um conjunto de componentes dentro de um só, além do que as ligações terão de ser as mesmas que estão presentes no esquema BPMN (excluindo claro componentes como eventos de início/fim ou *gateways*).

Assim sendo, por cada tarefa que represente um padrão é construído em XML o mesmo componente, apenas mudando o nome e a localização do ficheiro que contém a tradução em XML das tarefas específicas de cada padrão. Para construir as ligações em *Kettle*, a estratégia passa por percorrer o grafo que contém as ligações do esquema geral e construir as transições diretamente.

5.4 Resultado Final e Importação em *Kettle*

Ao se chegar ao final do processo de conversão, e tendo como resultados finais um ficheiro XML geral e um conjunto de outros ficheiros XML correspondentes a cada padrão que tenha sido identificado, pode-se finalmente testar-se o trabalho realizado a partir da ferramenta de conversão. Para isso bastará que, através da ferramenta Kettle, se selecione a opção de importação a partir de um ficheiro XML e escolher o ficheiro XML correspondente ao esquema geral do processo de povoamento. Dessa forma, obtém-se o aquilo que está apresentado na figura 19. Logo de seguida, na figura 20, está mostrado um exemplo de como foi configurado um dos muitos padrões presentes na figura 19.

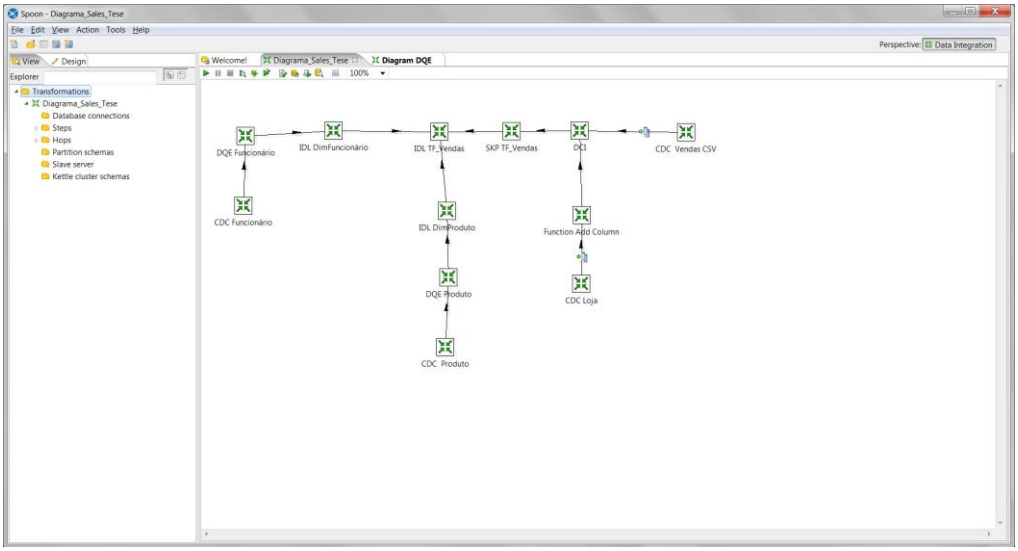


Figura 19 - Esquema geral em *Kettle* resultante do esquema de povoamento inicial em BPMN.

Com este sistema – um esqueleto - disponível, caberá agora à equipa de desenvolvimento do sistema de ETL, complementar esse esqueleto com o que for necessário e corrigir eventuais erros que possam surgir na execução deste esquema em *Kettle*.

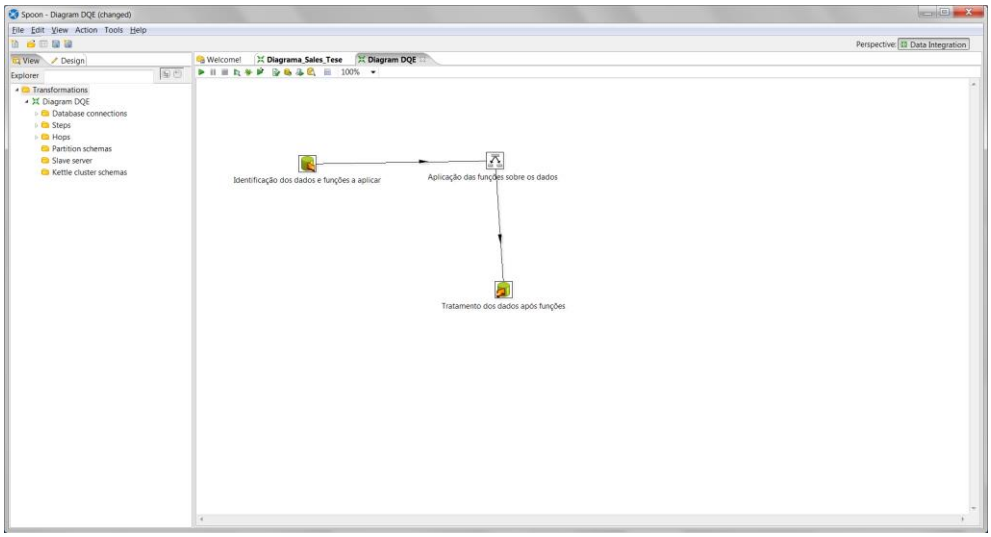


Figura 20 – Configuração de um dos padrões – *Data Quality Enhancement* – em *Kettle* (para os dados relativos aos funcionários).

5.5 Aspetos Específicos de Implementação

O sistema de conversão de modelos BPMN em modelos *Kettle* foi desenvolvido através da utilização de várias ferramentas. Dessas, devem salientar-se algumas que se mostraram particularmente relevantes no desenvolvimento da ferramenta. Na fase de modelação e especificação dos padrões foram utilizadas duas ferramentas: a ferramenta de modelação em BPMN Bizagi (*Bizagi Modeler*, 2014) e o IDE (*Integrated Development Environment*) Eclipse, com a adição de uma *framework* chamada Xtext (*Eclipse+Xtext*, 2014).

Relativamente ao Bizagi, a escolha de entre várias ferramentas com o mesmo propósito deveu-se a vários fatores mas principalmente porque reuniu algumas das condições essenciais para o trabalho: facilidade de *interface*, ampla utilização e formatos de exportação de ficheiros BPMN no âmbito do XML, só assim sendo possível realmente começar a pôr em prática o caso de estudo. No entanto, relativamente à parte de especificação do comportamento de padrões e do desenvolvimento da linguagem, foi feita uma pesquisa alargada sobre ferramentas ou programas que pudessem facilitar ou diminuir o tempo de execução dessa tarefa. Após algumas investigação e análise, encontrou-se a *framework* XText, um *plugin* do IDE Eclipse que não só permitia desenvolver a gramática de uma forma simples, providenciando igualmente toda a documentação essencial para dar os primeiros passos, como também disponibilizava mecanismos de correção e validação da mesma, podendo testar-se exemplos da linguagem e receber imediato *feedback* por parte do IDE.

Em relação à parte de conversão dos modelos, devem-se destacar três ferramentas no desenvolvimento do sistema de conversão: o IDE NetBeans (*Netbeans*, 2014) e as bibliotecas JAVA *JGraphT* (*JgraphT*, 2014) e *JSoup* (*Jsoup*, 2014). Relativamente à escolha do IDE, foi por ser já largamente utlizado e fácil de utilizar, sendo particularmente fácil fazer o *debugging* do código e ter acesso à documentação de vários métodos enquanto se está a escrever uma aplicação. Relativamente às bibliotecas, a sua escolha deveu-se a motivos um pouco distintos. A *JgraphT* é uma biblioteca que permite criar e visualizar grafos de todo o tipo, providenciando inclusive todo o género de algoritmos sobre os mesmos desde algoritmos de travessia, a caminhos mais curtos ou árvores de custo mínimo até a algoritmos dos chamados problemas NP-completos. Esta biblioteca foi particularmente relevante pois foi a estrutura de dados responsável pela gestão das ligações entre componentes no *Kettle* já que, na fase de *parsing* do ficheiro

XPDL, as ligações entre as atividades são guardadas num grafo dirigido (as ligações entre vértices são unidirecionais) e os algoritmos que a biblioteca providencia foram essenciais para, partir desse grafo, gerar as ligações entre os componentes. No entanto, a utilização da *JSoup* deveu-se a um caso de necessidade. Tal como referido anteriormente, na secção 5.3.1, o Bizagi, na exportação do modelo BPMN, que inclui a descrição comportamental dos padrões, exporta esse texto descritivo com elementos de caracterização, ou seja, inclui elementos típicos de XML e HTML como por exemplo a cor do texto, o espaçamento entre outros, pelo que foi mesmo necessário transformá-lo em texto puro para ser possível tratá-lo da forma mais eficiente e menos complexa possível. Assim, através de uma pequena pesquisa descobriu-se a *Jsoup*, uma biblioteca muito semelhante à biblioteca nativa de JAVA que inclui o *XPath* (já mencionado antes), que permite extrair, selecionar e manipular a informação contida em elementos HTML, possibilitando assim a extração apenas do texto e retirando todos os restantes elementos supérfluos do elemento "Descritption" (o elemento no XPDL que contém a descrição formal do padrão).

Capítulo 6

Conclusões e Trabalho Futuro

6.1 A Metodologia Abordada e Resultados

A modelação conceptual de sistemas de ETL é um tema "quente". Mutos trabalhos têm sido realizados nesta área. A preocupação com a redução dos erros e os elevados custos de implementação de um sistema ETL assim o estimulam. É uma representação de alto nível e, por consequência, a abstração de pormenores mais técnicos e mais complexos que tal metodologia possibilita, permite que se visualizem apenas os principais elementos envolvidos e o comportamento dos mesmos, facilitando a comunicação entre as pessoas envolvidas no projeto sobre o qual se está a modelar, principalmente gestores e administradores cuja visão é naturalmente diferente, prevenindo assim ambiguidades ou erros numa fase inicial do projeto.

Contudo, apesar de, ao longo do tempo, terem sido propostos vários trabalhos na área de modelação conceptual de ETL (alguns mais relevantes no campo das RPC e BPMN) ainda nenhum proporcionou uma alternativa que visasse a redução do tempo de implementação deste tipo de sistemas, apenas através do seu desenho conceptual. Isto porque os modelos conceptuais são apenas uma visão genérica descritiva do processo não possuindo nenhum outro tipo de valência que não a mera descrição visual. Tal acontece porque existe a carência de mecanismos, notações e linguagens de ETL que possibilitem a utilização de todas as especificações realizadas ao nível da modelação conceptual ETL para o mapeamento dessa descrição inicial em primitivas executáveis e, dessa forma, realizar a tradução do modelo conceptual num modelo físico passível de ser executado numa ferramenta própria de desenvolvimento de sistemas ETL.

Na presente dissertação foi abordada uma solução que tenta colmatar essa falha existente no processo de desenvolvimento de um sistema ETL, entre a parte da conceção e da concretização física do mesmo. Através da modelação baseada em padrões de ETL, e recorrendo à linguagem de modelação BPMN, foi introduzida uma nova linguagem de descrição de padrões que visa conferir alguma semântica ao modelo conceptual, no sentido de fornecer informação mais detalhada sobre o comportamento de cada padrão e, consequentemente, do modelo como, que contribuiu para que o esforço na construção dos modelos conceptuais seja mais proveitoso.

De forma a poder comprovar a abordagem proposta, foi elaborado um caso de estudo (fictício) que desse a oportunidade de poder passar por todas as fases: modelação, especificação, tradução e visualização do resultado final. Apesar de se encontrarem algumas dificuldades, principalmente a nível de limitações impostas pela ferramenta de modelação utilizada, conseguiu-se produzir uma ferramenta que, a partir da especificação conceptual de um modelo em BPMN com a utilização de padrões ETL, consegue criar um esqueleto físico para um sistema ETL correspondente, que pode ser importado numa ferramenta comercial de ETL, como o *Kettle.* O êxito da resolução do cenário proposto comprova que, de facto, é possível potenciar de forma considerável a utilização dos modelos conceptuais para um plano mais físico e, dessa forma, conseguir poupar tempo e recursos na implementação de um sistema ETL. Todavia, a investigação nesta área deverá continuar, no sentido de se melhorar a metodologia agora adotada e torná-la, quem sabe, num *standard de facto* na indústria ETL.

6.2 Trabalho Futuro

Apesar de todo o esforço despendido, será importante referir que o trabalho desenvolvido na presente dissertação não está isenta de melhorias, sendo que, num trabalho futuro, algumas funcionalidades deverão ser implementadas, para que dessa forma se possa melhorar a qualidade dos esqueletos produzidos. Desta forma, sugere-se que, num futuro próximo, sejam abordados os seguintes temas para melhoria:

- Tratamento de erros e exceções. A ferramenta de conversão, caso encontre algum problema de execução ou não encontre algo de que se está à espera deverá abortar o processo, devendo assim ser implementado um mecanismo que indique, no esquema em *Kettle*, onde se concentra o erro.

- Melhorias a nível da construção intermédia e nas estruturas de dados. Podemos retirar alguma complexidade à ferramenta agora produzida, se se proceder a alterações na arquitetura aplicacional e optar por diferentes estruturas de dados em algumas partes que tornem, posteriormente, os algoritmos de construção de objetos um pouco mais simples.

- Refinar a gramática da linguagem de descrição de padrões. Nesta área poderão surgir novas necessidades, o que implicaria uma revisão da gramática desenhada e implementada. Além disso, uma análise mais atenta poderá identificar possíveis de pontos de melhoria.

- Melhoria do algoritmo de visualização. A configuração de cada componente em *Kettle* inclui a sua posição no espaço de visualização da ferramenta, sendo as suas posições geradas na ferramenta de conversão com base na relação de pais e filhos aliado a um pequeno fator de aleatoriedade. Assim, para que melhores resultados se possam obter a nível de visualização no *Kettle,* será necessário trabalhar também essa parte.

Bibliografia

Akkaoui, Z. El, Mazón, J., 2012. Bpmn-based conceptual modeling of ETL processes. Data Warehousing and … 1–14.

Akkaoui, Z. El, Zimányi, E., 2009. Defining ETL worfklows using BPMN and BPEL. Proceedings of the ACM twelfth international … 41–48.

Arbab, F., 2004. Reo: a channel-based coordination model for component composition. Mathematical Structures in Computer Science 14, 329–366.

Belo, O., Oliveira, B., 2013. Modelação de Processos de ETL com Meta-Modelos em BPMN.

Bizagi Modeler, 2014. Disponível em http://www.bizagi.com/en/bpm-suite/bpm-products/modeler. Acedido em Abril de 2014.

Eclipse+Xtext, 2014. Disponível em https://www.eclipse.org/downloads/packages/eclipse-ide-java-and-dsl-developers/lunasr1. Acedido em Junho de 2014.

JgraphT, 2014. Disponível em http://jgrapht.org/. Acedido em Julho de 2014.

Jsoup, 2014. Disponível em http://jsoup.org/. Acedido em Setembro de 2014.

Lazovik, A., Arbab, F., 2010. Using Reo for service coordination. Procs. of the 5th Int'l Conf. on Service-Oriented Computing 398–403.

Matteo Golfarelli, Stefano Rizzi, 2009. Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill.

Meng, S., Arbab, F., 2007. Web services choreography and orchestration in Reo and constraint automata, in: Proceedings of the 2007 ACM Symposium on Applied Computing. ACM, pp. 346 –353.

Netbeans, 2014. Disponível em https://netbeans.org/features/index.html. Acedido em Maio de 2014.

Oliveira, B., Belo, O., 2013a. ETL Standard Processes Modelling - A Novel BPMN Approach. Presented at the 5th International Conference on Enterprise Information Systems, Angers, France.

Oliveira, B., Belo, O., 2013b. Using REO on ETL conceptual modelling: a first approach. Proceedings of the sixteenth international workshop … 1–6.

Salimifard, K., Wright, M., 2001. Petri net-based modelling of workflow systems: An overview. European Journal of Operational Research 134, 664–676.

Silva, D., Belo, O., Fernandes, J.M., 2012. Colored Petri nets in the simulation of ETL standard tasks: the surrogate key pipelining case.

Simitsis, A., 2003. Modeling and managing ETL processes., in: VLDB PhD Workshop.

Trujillo, J., Luján-Mora, S., 2003. A UML based approach for modeling ETL processes in data warehouses. Conceptual Modeling-ER 2003 2813, 307–320. doi:ETL processes, Data warehouses, conceptual modeling, UML

Van der Aalst, W.M.P., 1999. On the automatic generation of workflow processes based on product structures. Computers in Industry 39, 97–111.

Vassiliadis, P., Simitsis, A., Baikousi, E., 2009. A taxonomy of ETL activities, in: Proceedings of the ACM Twelfth International Workshop on Data Warehousing and OLAP. ACM, pp. 25–32.

Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., 2003. A Framework for the Design of ETL Scenarios, in: Advanced Information Systems Engineering. Springer, pp. 520–535.

Vassiliadis, P., Simitsis, A., Skiadopoulos, S., 2002. Conceptual modeling for ETL processes, in: Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP. ACM, pp. 14–21.

Wilkinson, K., Simitsis, A., Castellanos, M., Dayal, U., 2010. Leveraging business process models for ETL design, in: Conceptual Modeling–ER 2010. Springer, pp. 15–30.

Žarnay, M., univerzita v Žiline, Ž., 2004. Use of Petri Net for Modelling of Traffic in Railway Stations, in: Proceedings of International Conference Infotrans.