CAPÍTULO 9

A COMPONENTE ADAPTATIVA

ÍNDICE

9.1.- Introdução.

Rigor e método na concepção e desenvolvimento de sistemas interactivos são mais-valias qualitativas mais directamente dirigidas aos que concebem e progra-mam “software” interactivo. Ainda que o método sistemático proposto nesta tese tenha por objectivo, com base no MASS, o enriquecimento semântico das IU, e deste modo possua intrinsecamente uma mais-valia qualitativa dado potenciar um aumento da usabilidade, não é imediatamente inferível que “software” inter-activo mais bem concebido e mais robusto seja necessariamente de melhor qua-lidade do ponto de vista dos utilizadores finais.

Embora a qualidade intrínseca das aplicações interactivas tenha bastante a ver com o processo da sua concepção e desenvolvimento, factores extrínsecos, tais como a sua aceitação pelos utilizadores finais são, como se sabe e reconhe-ce, de importância primordial.

O objectivo ideal a atingir nesta área poderia sintetizar-se como sendo a construção metódica e rigorosa de sistemas interactivos, acrescentando às inter-faces com o utilizador rigorosamente construídas, propriedades e caracterísi-cas de funcionamento que as tornassem verdadeiras interfaces *amigáveis*. Este qualificativo, muito referido embora nem sempre com significado preciso, tem genericamente a ver com a facilidade com que os utilizadores alcançam os ob-jectivos para os quais a aplicação interactiva foi desenvolvida.

Não sendo possível no espaço temporal útil de realização desta tese desen-volver uma abordagem contemplando todas estas preocupações, pareceu no en-tanto importante ao autor realizar uma incursão no domínio das preocupações de *usabilidade*, em particular estudando a possibilidade de acrescentar caracte-rísticas de *adaptabilidade* às interfaces com o utilizador, sistemática e rigorosa-mente construídas segundo o método e as ferramentas que se apresentaram em capítulos anteriores.

Assim, e como premissa de trabalho, assumiu-se que uma característica fundamental de qualquer interface amigável seria possuir "alguma inteligência". Um aspecto particular e fundamental do emprego dessa "inteligência" neste con-texto, consistirá na capacidade da interface se "auto-adaptar" ao tipo particular de utilizador, facilitando-lhe o acesso às disponibilidades de um dado sistema. Estas interfaces com o utilizador, designadas *Interfaces Adaptativas*[1], são pois capazes de se ajustar, em tempo de execução, a determinadas características dos seus diversos utilizadores, tomando por base observações sobre o comporta-mento destes, observações

[1] Pretende-se com esta tradução da expressão inglesa *adaptive interfaces* deixar clara a distinção entre este tipo de adaptabilidade e a que pode ser encontrada nas *Interfaces Adaptáveis*, não "inteligentes".

que serão registadas e constituirão o conhecimento histórico e recente a utilizar pelo mecanismo de adaptação.

É importante relembrar a crescente importância da inclusão de capacidades adaptativas na IU de um qualquer sistema ou aplicação interactiva, procurando também estabelecer uma distinção entre Interfaces Adaptativas, Adaptáveis e Adaptadas. Dentro das limitações e complexidade inerentes ao estado actual da arte na área da adaptatividade [Kuhme et al. 92], e tendo em consideração não ser este o objectivo fundamental da tese, procurou-se ainda assim analisar uma possível melhoria qualitativa do sistema GAMA pela inclusão de características de *adaptatividade* na sua IU. Em resultado do estudo realizado, foi implementa-do o sistema GAIA (*Gerador Automático de Interfaces Adaptativas),* a ter em consi-deração como subsistema de qualquer arquitectura para interacção.

Apresenta-se neste capítulo a arquitectura funcional e os mecanismos de adaptatividade implementados no sistema GAIA [Martins e Ferreira 93].

9.2.- Interfaces Adaptativas.

9.2.1.- "Inteligência" e Interfaces.

Muitas dúvidas existem ainda quanto à caracterização do que se pode conside-rar uma interface "inteligente". Interfaces "bem sucedidas" são geralmente desi- gnadas por *amigáveis,* o que certamente é muito pouco ilustrativo das suas ca-racterísticas. Capacidade de "desculpa", flexibilidade, acessibilidade, etc., são propriedades em geral apontadas como indispensáveis para se atingir tal suces-so. Muitas destas características resultam da implementação de princípios tais como previsibilidade, coerência, observabilidade, etc., princípios estes menos subjectivos já que devidamente enunciados e até formalizados [Thimbleby 90] [Dix 91]. Em suma, para além de construída segundo bem definidos princípios formais, a interface deverá ser, de algum modo, *inteligente*.

Com base na premissa anterior, o problema seguinte consistirá na determi-nação do tipo de "conhecimento" que a IU deverá possuir, por forma a que possa observar e participar no fenómeno de interacção com um ser humano de modo *compreensivo*, isto é, intervindo autonomamente e com coerência.

Adicionando a estes tipos de conhecimento o conhecimento que a IU deverá ter do utilizador para que a este se possa ajustar, podemos de imediato ter uma ideia do grau de complexidade requerida por uma modelação completa. Um es-tudo exaustivo sobre as diversas vertentes deste problema, com preocupações até de sistematização, é apresentado em [Kuhme et al. 92]. Em [Browne et al. 90] é mesmo considerado não existirem

limites especiais quanto às possibilida-des de adaptatividade numa IU, ainda que a adaptatividade seja dividida em duas grandes classes: de *comunicação* e de *funcionalidade.*

No sistema GAIA, assumiu-se, desde logo [Martins 87c], que a interface deve-ria possuir "inteligência", não sob a forma de capacidade para "entender" língua natural, mas sob a forma de capacidade de adaptação ao comportamento do utilizador, através da inclusão de um modelo explícito deste, sob a forma de um modelo comportamental, a partir do qual a interface pudesse inferir o "grau de conhecimento" do utilizador e, assim, adaptar a sua *apresentação* e *funcionalida-de* ao nível de conhecimentos daquele. Uma abordagem semelhante é apresenta-da em [Brooks e Thorburn 88] considerando diferentes estilos de interacção em função do "suposto" grau de conhecimento do utilizador, inferido ou calculado.

9.2.2.- Interfaces Adaptadas, Adaptáveis e Adaptativas.

Alguma confusão existe na literatura entre o que se deve entender por interfaces adaptadas, adaptáveis e adaptativas, que nesta subsecção se procura desde já esclarecer. Em primeiro lugar, apenas as *Interfaces Adaptativas*, por vezes desi-gnadas até *auto-adaptativas*[2] por questões de clareza, podem ser consideradas "inteligentes", dado modificarem o seu comportamento em “run time”. Todas as outras têm um comportamento imutável, ainda que determinado em circunstân-cias diferentes.

Assim, *Interfaces Adaptadas* são em geral concebidas para ambientes particulares, procurando-se “a priori” que sejam o mais adequadas possível às circunstâncias ambientais, isto é, de utilização. São em geral desenvolvidas por evolução gradual, através da utilização de protótipos com os quais os futuros utilizadores são previamente postos em contacto. Da apreciação *in loco* do com-portamento destes, particularmente dos erros mais frequentes, factores de de-sempenho, etc., o protótipo é ajustado e, por iteração, é atingido o desenho final da interface a implementar. A adaptação é portanto estática.

Interfaces Adaptáveis são interfaces passíveis de ajuste individual, em geral a cargo de cada utilizador. Nestas, vários níveis, estilos e objectos de diálogo são oferecidos, escolhendo cada utilizador o que, na sua opinião, mais se ajusta a si próprio, ou então a interface é programável e, partindo de um conjunto base, permite ao utilizador construir o seu próprio ambiente, em geral usando coman-dos pré-programados ou "macros". Em qualquer dos casos a adaptação é ainda a cargo do utilizador, sendo estática na primeira hipótese e semi-dinâmica na segunda (cf. um usual ficheiro editável de "preferências").

---

[2] de *Self-Adaptive User Interfaces.*

Interfaces Adaptativas são diferentes das anteriores porque nestas o utiliza-dor não controla a adaptação. Esta é realizada de forma automática e dinâmica, isto é, durante a utilização efectiva do sistema, pela IU. Sendo o objectivo fazer a interface aproximar-se o mais possível, em cada momento, das necessidades do utilizador, tal implica que a interface se deva basear num "conhecimento" que a cada momento deverá possuir sobre este, em particular sobre o tipo de utiliza-ção do sistema pelo mesmo feita. Assim, e contrariamente às anteriores, são "interfaces inteligentes", nas quais a adaptabilidade é determinada com base neste conhecimento e em regras que determinam a sua dinâmica, baseando-se particularmente no contexto actual e histórico da interacção. Por serem "inteli-gentes" e dinâmicas são, conforme se referiu atrás, por vezes referidas como *interfaces auto-adaptativas* realçando tais características.

Adaptatividade.

Muito embora a adaptatividade possa ser realizada de múltiplas formas, em ge-ral apenas dois tipos de características do utilizador são tomadas em considera-ção como informação de base para a adaptatividade:

- A *experiência do utilizador* e o seu *conhecimento* relativamente aos princípios de utilização e funcionamento do sistema;
- As *preferências do utilizador,* tais como o estilo do diálogo, o tipo de apresentação, os objectos interactivos, etc.

Veremos na secção seguinte como estas características do utilizador foram consideradas no sistema GAIA como informação de base para a adaptatividade.

9.3.- GAIA: Gerador de Interfaces Adaptativas.

Características.

Numa fase inicial de desenvolvimento do projecto, e sendo conhecidas as dificul-dades que iriam ser encontradas, procurou-se definir claramente o seu grau de abrangência e um conjunto de premissas de trabalho relativamente a pontos tão relevantes como:

- *Estilo de diálogo, isto é, quais os objectos interactivos a serem utiliza-dos pela IU;*
- *Dispositivos de entrada de dados a admitir;*

- *Comportamento do utilizador a modelar;*
- *Formas de adaptatividade da IU.*

Tomadas as decisões quanto aos pontos anteriores, a implementação foi realizada procurando abstrair da aplicação concreta em causa ("abstracção semântica") mas adoptando um modelo de controlo de diálogo bem conhecido (“dialogue dominant control”), e no qual compete à IU estabelecer a sequência do diálogo, dado a aplicação ser vista como um conjunto de operações a serem invocadas. Esta decisão teve a ver com o próprio modelo de interacção do siste-ma GAMA, procurando-se assim facilitar uma posterior integração.

Comportamento do Utilizador.

Na concepção de uma IU existe uma componente de complexidade e não-deter-minismo que é sempre necessário tomar em consideração. Assim, à pergunta *"Que parâmetros do comportamento do utilizador podem ser qualificados e quanti-ficados numa interface?"*, a resposta de momento só pode ser heurística e intuiti-va.

Por um lado, e considerando que a aplicação é vista sob a forma de um con-junto de operações (*comandos*) e que as intenções do utilizador se podem repre-sentar por *sequências de acções*, originárias ou não de eventos, podemos consi-derar que cada tarefa pode ser dividida numa *sequência de eventos* que repre-senta os passos detectáveis pela IU do comportamento efectivo, operacional, do utilizador. Surgem deste modo dois elementos fundamentais (e atómicos) para a análise do comportamento interactivo do utilizador: *comandos* e *eventos*.

Por outro lado, segundo [Hammond e Barnard 85], o comportamento do uti-lizador deve ser estratificado, sendo de considerar diferentes níveis de perícia. A análise realiza-se aqui em dois contextos diferentes:

- *Dinâmico*:

  Trata-se da interacção propriamente dita. Analisam-se eventos e co-mandos durante a sessão de trabalho. Cada tarefa pode ser dividida em diferentes passos (*eventos*) e, sendo uma sessão de trabalho uma série de tarefas, o diálogo pode ser analisado como uma sequência de eventos.

- *Sintáctico:*

  A este nível situa-se o tratamento da sintaxe dos comandos bem como o tratamento léxico dos seus argumentos. O comportamento do utili-zador é analisado numa base temporal e estatística.

Adaptação ao Utilizador.

Qualquer utilizador de um sistema interactivo procura adaptar-se ao sistema. Se a IU realizar tal adaptação de forma eficaz, poderá evitar tal passo da parte deste. A adaptatividade deverá ser determinada a partir do "conhecimento" que a IU possui acerca do comportamento do utilizador. No sistema GAIA foram con-siderados três critérios de adaptatividade:

- *Via Análise*: Analisar o comportamento recente do utilizador, podendo daí serem deduzidas recomendações de adaptação;
- *Via Confirmação*: A partir do comportamento posterior do utilizador, confirmar as recomendações obtidas pela anterior análise;
- *Via Alteração*: Alterar a interface segundo as recomendações do módulo de inferência e conforme acordo do utilizador.

O nível de análise e adaptação pode igualmente ser abordado do ponto de vista dinâmico e estático (ou sintáctico), conforme a adaptação é feita sobre:

- *a sequência de comandos*, i.é., podendo-se destas extrapolar repetição de comportamento que pode sugerir a criação de "macros";
- *o modo de invocação dos comandos*, tendo em conta a forma de invo-cação destes (cf. menu, comando, botão,etc.), e o grau da sua utiliza-ção;
- *o nível de "help"*, i.é., a capacidade de adaptar o nível do texto de aju-da de um dado comando, ao nível e grau de conhecimento inferido para um dado utilizador;
- *o nível de perícia do utilizador*, ou seja, sobre a classe de níveis de comportamento definida, tendo em conta os parâmetros quantificá-veis, designadamente, taxa de erros, taxa de pedidos de "help", grau de utilização de comandos, etc.
- *o nível sintáctico*, ou seja, relativamente à correcção da utilização dos comandos, procurando aceitar "sinónimos" e até valores por defeito para os argumentos.

Arquitectura do Sistema.

Conforme é visível na figura 9.1, o sistema GAIA é constituído, no seu núcleo funcional, por dois módulos que, por terem missões distintas exigindo cada uma delas técnicas e ferramentas particulares, foram desenvolvidos e implementados em contextos completamente separados, pelo que se tornou necessário criar um protocolo específico para a sua comunicação. Os módulos *MRX* ("Módulo Run-time X-Windows") e *MIA* ("Módulo Interface Adaptativa") realizam as acções fun-damentais relativas à funcionalidade observável do sistema.

O módulo *MRX*, encarregado da gestão da camada de apresentação, ou seja do aspecto que a IU deve exibir em cada instante e à qual corresponde determi-nada funcionalidade da aplicação acessível, foi desenvolvido usando a lingua-gem C e OSF/MOTIF sobre X-WINDOWS.

Dado que a *apresentação* da IU é adaptativa em função de determinados pa-râmetros de comportamento do utilizador, e não sendo o *MRX* um módulo con-tendo conhecimento, é através de directivas que lhe são comunicadas pelo mó-dulo *MIA* que a apresentação é determinada, limitando-se o *MRX* a implementar tais directivas.

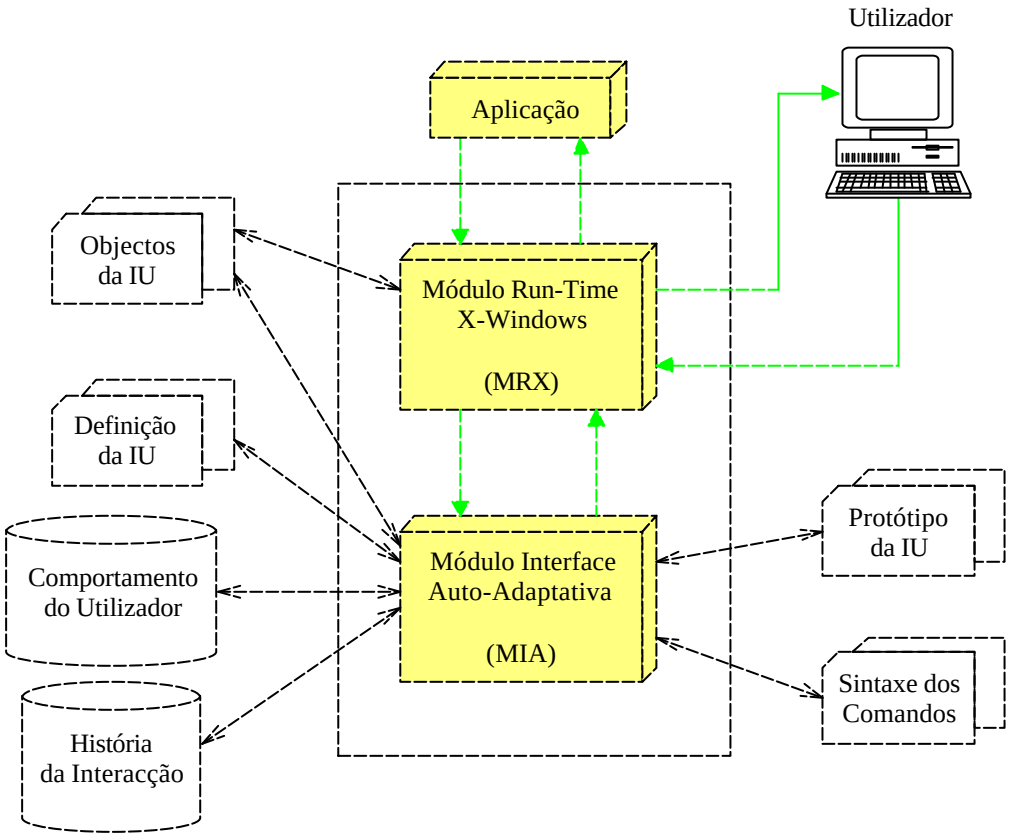

Fig. 9.1 - Arquitectura do GAIA.

O *MIA* é portanto o módulo "inteligente" que, através da análise do compor-tamento do utilizador e da utilização de inferência sobre um conjunto

de regras que constituem a sua base de conhecimento, determina a adaptação. Por ques-tões de facilidade de implementação, o *MIA* foi implementado em Prolog [Warren 77].

Toda a comunicação entre estes módulos, a aplicação e o utilizador é feita através de *eventos*. As acções do utilizador são transformadas em eventos pelo *MRX* que os passa ao *MIA*. Este, trata-os e gera, se fôr caso disso, eventos de saída destinados ao *MRX*, para que este controle aspectos de apresentação, bem como eventos destinados à aplicação, que não são mais do que invocações de comandos desta. A aplicação, por sua vez, devolve igualmente eventos, quer como resultado da invocação de determinado comando quer como dados para visualização.

O Módulo MIA.

Este módulo contém o núcleo "inteligente" que realiza a inferência do conheci-mento do utilizador, bem como o gestor de comunicação do sistema, possuindo a arquitectura apresentada na Fig. 9.2.

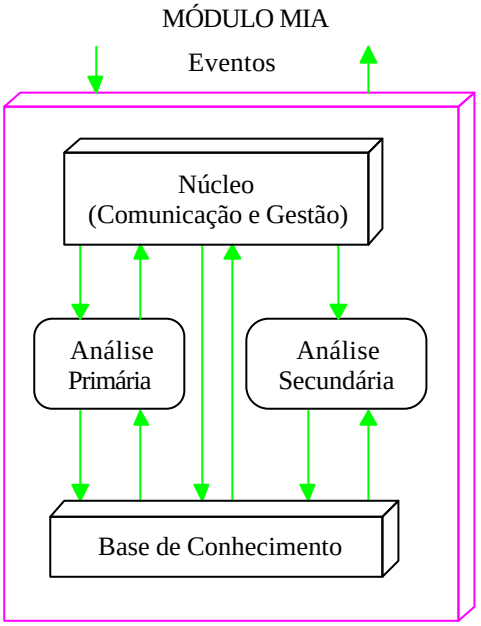

Fig. 9.2 - O Módulo MIA.

O *Núcleo* estabelece a comunicação com o *MRX* e, através deste, com a apli-cação. Gere ainda os objectos de interacção a serem apresentados ao utilizador, tais como menus, botões, etc.

A *Análise Primária*, executada no decurso de uma sessão de trabalho, anali-sa a última sequência de *n* eventos, que indiciam um "dado" comportamento do utilizador, tomando decisões a nível de ajuda imediata, sinónimos, etc.

A *Análise Secundária,* executada ao nível da sessão de trabalho, permite a tomada de decisões quanto ao nível de "help" dos comandos, nível do utilizador, modo de operação (menus ou linguagem de comandos) e outros.

O Módulo MRX.

Partindo de uma apresentação definida "por omissão", este módulo encarrega-se da "composição" do aspecto visual da IU bem como da recepção dos eventos do utilizador.

Cada comando disponibilizado pela aplicação tem a si associada uma *caixa de diálogo* composta pelos diferentes argumentos, tendo em conta o seu tipo. Para os textos de "help" e para "mensagens" do sistema estão igualmente dispo-níveis *caixas de mensagens.* Uma linguagem de especificação foi desenvolvida para definir estes objectos interactivos e associá-los aos respectivos objectos da camada computacional.

Interface e Linguagem de Especificação.

Para a construção da IU disponibilizaram-se as seguintes classes de objectos: menus, botões, comandos e argumentos.

A cada um destes objectos são associados *recursos* (definidos à custa dos recursos dos respectivos *widgets* do OSF/MOTIF) que permitem definir as carac-terísticas particulares, visuais e de comportamento de cada instância das diver-sas classes.

A interface pode ser considerada dividida em três componentes:

- *Objectos Interactivos* (apresentação);
- *Controlador do Diálogo* (comportamento);
- *Linguagem e sintaxe dos comandos*.

Cada aplicação define um protótipo inicial da sua interface, através de uma descrição dos três componentes atrás referidos, feita numa *Linguagem de Espe-cificação* simples, desenvolvida para tal efeito, que a seguir se descreve.

Linguagem de Especificação.

A linguagem de especificação prevê três secções correspondentes aos três com-ponentes da interface a serem descritos, designadamente: “objects”, “behaviour” e “syntax”.

? **Secção "Objects"**

Nesta secção declaram-se instâncias das Classes anteriormente referidas, segundo a forma sintáctica, *Objecto = Classe*, definindo-se de seguida os recur-sos do objecto seguindo a sintaxe *Objecto: Recurso := Valor*, podendo ser *Valor* uma constante, uma expressão literal ou um conjunto enumerado. Veja-se, o exemplo que se segue:

```
objects
    menu1 = menu.
    menu1:label := 'Ficheiro'.
    menu1:options := [cut, del, copy, paste, undo, com3].
    copy = command.
    copy:label := 'Copy'.
    copy:mnemonic := 'C'.
```

? **Secção "Behaviour"**

A parte da especificação destinada à descrição do comportamento é imple-mentada usando um conjunto de regras de transição de estados que descrevem o respectivo autómato. Cada *estado* pode ser visto como representando o estado interno do sistema interactivo num dado momento, podendo-lhe ser associados objectos dependentes da aplicação segundo a forma sintáctica:

Estado ? *Conjunto de Objectos*

Cada transição entre estados é descrita por uma regra da forma:

Estado1 : Conj. Comandos ? *Estado2*

Em complemento, é permitida a inclusão de *pré-condições* associadas a cada comando, assim se representando as regras semânticas. Apresenta-se a seguir um pequeno exemplo com o único objectivo de ilustrar o emprego da notação.

```
behaviour
    inicial <- [].
    comandoexec <- [].
```

```
                                .....
final <- [].
inicial: [com1, com2, com3, .., comk] -> comandoexec.
inicial: [cut, copy] -> bloco_guardado.
comandoexec: [undo] -> inicial.
bloco_guardado: [paste] -> bloco_guardado.
                                .....
pre_condition(cut) :-  .....
pre_condition(copy) :-  .....
```

? **Secção "Syntax".**

A representação sintáctica da linguagem de comandos, necessária à correcta avaliação da sua invocação, é feita segundo um formalismo que é uma extensão às DCGs ("Definite Clause Grammars") [Pereira e Warren 80], facilmente compatibilizado com a linguagem Prolog, tendo sido no entanto acrescentados alguns novos operadores, designadamente:

*****	(repete zero ou mais vezes)
+	(repete uma ou mais vezes)
?	(opcional)
#	(sem ordem explícita)
$value(arg)	(valor do argumento indicado)

Vejamos um pequeno exemplo onde se define a sintaxe de um comando *"savefile"* com argumentos *caminho*, *ficheiro* e *modo*.

```
syntax
    savefile <- ['savefile'] , #(arg1, arg2, arg3).
    arg1 <- ['-path'] , $value(caminho).
    arg2 <- ['-file'] , $value(file).
    arg3 <- $value(modo).
    modo('backup') <- ['-b'].
    modo('overwrite') <- [].
        ......
```

```
path <- (['/']? , pattern , (['/'] , pattern)*.
pattern <- ([X] , { X =/= '/' })+.
        ........
```

Apresentada de forma simples a linguagem de especificação, vejamos de se-guida algumas das características de funcionamento do sistema GAIA.

Comunicação entre Módulos.

Durante o diálogo, o utilizador apenas estabelece comunicação com o *MRX*. Este encarrega-se de interpretar todas as instruções do utilizador convertendo- -as em eventos que são comunicados ao *MIA*. Este módulo trata estes eventos gerando, se for caso disso, eventos de saída destinados ao *MRX* ou à aplicação (cf. Fig. 9.1). A aplicação responde aos eventos que correspondem à invocação de comandos gerando eventos que, por sua vez, correspondem à comunicação dos resultados, sendo estes tratados ao nível do *MIA*.

O facto de toda a comunicação entre o *MIA* e a aplicação se fazer através do *MRX* resulta de razões impostas pela implementação (o X-Windows não delega facilmente o controlo).

Protocolo.

Como para a implementação dos diversos módulos foram usados paradigmas e "ferramentas" diferentes, designadamente, o paradigma lógico, via Prolog, para o *MIA* e procedimental, via C e OSF/MOTIF/X, para o *MRX*, o protocolo de comu-nicação deixa de ser simétrico, devendo assim definir-se um protocolo no senti-do *MIA-MRX* e outro no sentido *MRX-MIA*. Assim, dois protocolos distintos de co-municação foram implementados.

O *MIA* funciona usando o interpretador de Prolog, pelo que o protocolo entre o *MRX* e o *MIA* se baseia em predicados Prolog da forma,

evento_entrada([Evento, Parâmetro1, Parâmetro2, ...])

Quanto à comunicação entre o *MIA* e o *MRX*, o protocolo consiste na repre-sentação dos eventos usando uma sequência estruturada de “bytes” cuja orga-nização nos dispensamos de, aqui, apresentar [Ferreira 92]. A comunicação que se realiza entre a aplicação e o módulo *MIA* obedece às regras impostas pelo *MRX*, baseando-se naturalmente no mecanismo de “callbacks” do X.

Gestão da IU.

O *MIA* está encarregue da gestão da interface. Toda a informação respeitante à IU de uma dada aplicação, para um determinado utilizador, está armazenada num conjunto de ficheiros. Um outro ficheiro, gerado a partir da especificação da interface inicial, representa o estado inicial da IU. Numa primeira verificação, o *MIA* testa a coerência de tal ficheiro, gerando de seguida os ficheiros que irão servir de base à construção da interface individual por utilizador.

Por aplicação, o *MIA* guarda um “default” destes ficheiros que serão forneci-dos a cada "novo" utilizador da aplicação. No início de cada sessão o *MIA* con-sulta, para dado utilizador, o respectivo ficheiro de forma a serem criados os ob-jectos interactivos constituintes da interface desse utilizador. No final da sessão de trabalho todos estes ficheiros são actualizados.

Note-se que o *MRX* apenas reflecte as respostas do *MIA* às acções do utiliza-dor, isto é, o *MRX* recebe eventos que indicam a reacção que a interface deverá ter em relação às acções deste. Assim, enquanto que o *MRX* apenas se encarrega de aceitar os eventos/intenções do utilizador, o *MIA* procura "gerir" racionalmen-te a interface interpretando "tudo" o que seja gerado pelo utilizador, de modo a que a comunicação com a aplicação seja feita de forma eficaz e adequada.

Mecanismo de Análise.

Uma sessão de trabalho pode ser vista, por um lado, como uma *sequência de comandos*. Porém, não é prático analisar toda a sequência até aí obtida. Assim, a análise funciona por "janelas", que correspondem a sequências finitas de *m* comandos, em particular os últimos *m* comandos introduzidos. A *análise* procu-ra, a partir de tal sequência, construir um grafo que permita derivar "frases" de comandos, interpretáveis até como *tarefas* (cf. [Hoppe 88]), compostas por, no máximo, *m* comandos. No início, e por utilizador, este grafo possui a configura-ção que se ilustra na figura 9. 3.

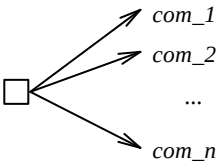

Fig. 9.3 - Grafo Inicial do Utilizador

À medida que as sequências válidas de comandos introduzidas pelo utiliza-dor vão evoluindo, também tal grafo evolui, possibilitando, a partir de certo ní-vel, a derivação de *macros* que são colocadas à disposição do

utilizador. A título de exemplo, se o utilizador introduzir a sequência válida de comandos

com_1 com_4 com_1 com_3 com_2

o grafo será actualizado passando a ter a configuração ilustrada na figura 9.4, admitindo-se uma "janela" temporal de 4 comandos.

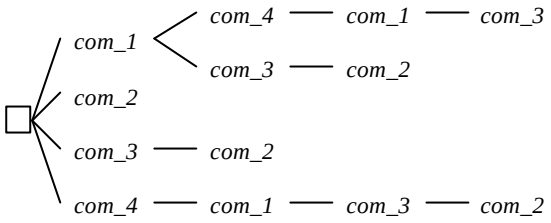

Fig. 9.4 - Evolução do Grafo de Interacção.

Este grafo, para além de conter a história da interacção via comandos, irá igualmente fornecer informação para a componente de adaptação. O grafo tem igualmente ligação com o autómato de comportamento da IU, especificado na secção “behaviour” da linguagem de especificação apresentada.

O autómato é construído por forma a possibilitar derivar (ou prever) o com-portamento do utilizador, associando *pesos* às transições. A partir de um con-junto inicial de pesos, podem atingir-se pesos máximos e pesos mínimos, neste último caso daí podendo resultar a inibição de tais transições. O autómato é dinâmicamente alterável nos seus estados e transições, em resultado das pró-prias directivas de adaptação.

Cada sessão de trabalho é também registada sob a forma de uma sequência de eventos, depois de se realizar uma análise e filtragem dos eventos mais rele-vantes para a *história do utilizador*. Esta *história* é agrupada em *zonas temporais* devidamente identificadas. Um “garbage collector” trata de, periodicamente, eliminar "fracções da história" de eventos consideradas caducas, isto é, perten-centes a zonas temporais antigas.

O registo dos eventos trocados em ambos os sentidos entre o *MIA* e o *MRX* é realizado usando o predicado

adp_trace(evento, [TipoEvento, Evento], Parametros, Time]

enquanto que a ocorrência de um comando é registada pelo predicado

adp_trace(comando, [ModoInvocação, Comando], Parametros, Time]

sendo a análise dos comandos e a alteração do grafo realizadas usando um con-junto de predicados que fazem parte da base de conhecimento do *MIA*.

Adaptação.

Decorrente da análise realizada e do conhecimento que possui sob a forma de regras, o *MIA* determina a adaptatividade seguindo as etapas que atrás foram descritas, encontrando-se implementadas todas as capacidades adaptativas que anteriormente foram referidas.

A adaptatividade baseia-se na permanente consulta e alteração da base de regras e factos, usando predicados específicos, e na utilização do mecanismo de inferência. A capacidade de criação de "sinónimos" dos comandos e de correcção automática de erros sintácticos comuns, baseia-se na utilização da base de co-nhecimento pelo "parser" incluído no *MIA*.

Da análise das sequências de comandos vai-se obtendo um grafo das várias sequências possíveis em função da dimensão da "janela" em vigor. Quando uma nova sequência de dois ou mais comandos é detectada, é acrescentada ao grafo e, na base de conhecimento registado tal facto através da cláusula

```
pattern(Grafo, Seq, Num, Time).
```

Quando a sequência se repete um número de vezes pré-determinado no sistema, o mecanismo de adaptação vai sugerir a transformação de tal sequência numa macro, criando na base de conhecimento tal associação através da cláusula

```
macro(Name, SeqCommands).
```

A criação de abreviaturas dos comandos e a consideração de sinónimos[3] é tam-bém contemplada, estando criadas cláusulas semelhantes às anteriores para tal efeito. É importante que se saliente que todas estas decisões de adaptação to-madas pelo sistema são propostas ao utilizador. Caso o utilizador confirme as adaptações sugeridas, uma nova "janela temporal" é iniciada, correspondente à fixação das características da nova IU.

Um elevado conjunto de regras destina-se a inferir o grau de perícia do uti-lizador (cf. iniciado, intermédio, avançado e especialista) e o respectivo nível de auxílio. No entanto, este nível de adaptação, dado o grande conjunto de regras e o grande volume de informação que deve ser analisado, é apenas realizado no fi-nal de cada sessão de trabalho. Mesmo estas possíveis

[3] *aliases*.

modificações de nível do utilizador só são de facto fixadas com a confirmação deste.

9.4.- Sumário.

O sistema GAIA neste capítulo apresentado é, técnica e estruturalmente, um sistema com algum grau de complexidade, dada a abrangência de áreas distin-tas de tecnologia e conhecimento envolvidas na sua concepção e construção.

Devido à sua relativa imaturidade, não foi ainda possível submetê-lo a reais "provas de esforço" que permitam objectivamente "medir" o seu grau de sucesso e efectividade, o que o torna um protótipo. No entanto, dadas as suas caracterís-ticas, é indiscutivelmente, no mínimo, uma ferramenta de análise e concepção iterativa de IU de grande utilidade, apresentando-se como um óptimo auxiliar de construção de IU através do desenvolvimento rápido de protótipos e experimen-tação prévia junto dos futuros utilizadores.

De notar que, ainda que tal não fosse um objectivo essencial, o GAIA incor-pora muitas das ideias atrás apresentadas para o desenvolvimento de sistemas interactivos.

São no entanto reconhecidas algumas limitações da actual implementação, designadamente as seguintes:

- *modelo simplificado do utilizador;*
- *heurística simples para a adaptação;*

As duas limitações apontadas resultam principalmente, conforme se afirmou atrás, da imaturidade experimental do sistema, sendo de acreditar que a sua utilização em diferentes contextos[4] possa resultar na aquisição de mais e melho-res heurísticas, que poderão ser posteriormente incorporadas enriquecendo quer o modelo do utilizador quer o algoritmo de adaptatividade.

Em contrapartida, a ligação à aplicação é, de momento, baseada na defini-ção sintáctica da linguagem da aplicação sob a forma de comandos e argumen-tos. A abordagem tem a vantagem de ser abstracta relativamente à linguagem da aplicação, ainda que assuma que esta possui uma API acessível segundo tal estrutura. No protótipo actual do GAIA a aplicação é até constituída por um con-junto de funções de simulação.

Porém, algum trabalho adicional deve ser ainda realizado, quer no sentido de tornar o sistema GAIA uma ferramenta autónoma e mais

[4] No âmbito de um contrato de investigação estabelecido entre o DI/INESC e a Olivetti Ricerca, o sistema GAIA vai ser disponibilizado a título experimental, esperando-se que daí possam advir resultados de utilização interessantes.

sofisticada de análi-se, visando quer a geração de interfaces adaptativas quer a criação iterativa de interfaces adaptadas, quer no sentido de a transformar num módulo adicional, activável ou não, dentro do sistema GAMA.