GAMA-X

Uma Arquitectura Software para o Desenvolvimento Semi-Automático de Interfaces Utilizador-Sistema

José Francisco Creissac F. Campos
Fernando Mário J. Martins

Departamento de Informática/INESC-BRAGA
Universidade do Minho

Resumo

Apresenta-se neste artigo um arquitectura software destinada a fornecer o suporte tecnológico ao desenvolvimento de Interfaces Utilizador no enquadramento de uma metodologia rigorosa para o desenvolvimento de Sistemas Interactivos.

Assim, partindo da especificação formal da camada aplicativa, usando modelos matemáticos, a metodologia visa a sistematização do processo de construção do sistema interactivo, segundo princípios que têm por objectivo a criação de interfaces de qualidade dado o seu grau de assistência ao utilizador, dadas as suas caraterísticas preventivas de erros e finalmente dada a sua sensibilidade contextual.

O sistema apresentado, GAMA-X, oferece uma arquitectura de apoio ao desenvolvimento sistemático e semi-automático deste tipo de interfaces, incorporando ferramentas que apoiam os diferentes passos do desenvolvimento.

Apresenta-se neste artigo em maior detalhe a composição do GAMA-X, com relevo para os módulos mais ligados à IU, bem como a linguagem de especificação do controlador do diálogo.

Referem-se ainda questões relacionadas com a camada de apresentação e com a comunicação entre os diferentes módulos.

Introdução

A possibilidade de geração automática de Interfaces Utilizador-Sistema é largamente condicionada, por um lado, pela inexistência de modelos de interacção ajustados, por outro, pela inexistência de metodologias que considerem o problema do desenvolvimento de aplicações interactivas de uma forma global, isto é, considerando o desenho da camada computacional intrinsecamente relacionado com o desenho da camada interactiva.

Do trabalho teórico realizado sobre este problema no âmbito do projecto JNICT designado INTERLAB, resultou um *modelo de interacção* particular, designado *modo assistido*, e, principalmente, uma *metodologia* que permite relacionar, num mesmo contexto de desenho, a concepção da camada aplicativa e a concepção de uma sua possível Interface com o Utilizador.

Nesta comunicação apresenta-se principalmente a infraestrutura software que vem sendo desenvolvida como suporte à aplicação da metodologia, e que designámos **GAMA-X** (*Gerador Automático de Modo Assistido em X*).

Apresentam-se em seguida todos os andares que constituem este Sistema de Desenvolvimento de Interfaces, tendo em particular atenção a estrutura interna e a interligação dos módulos que implementam o *controlador do diálogo*, o *modelo da aplicação*, o *modelo da apresentação* e as suas *interfaces internas*.

Indica-se, relativamente a cada um destes componentes, a forma como a sua informação interna é criada partindo das especificações da aplicação e da interface com o utilizador, a introduzir através de editores interactivos que garantem a validação sintáctica das mesmas.

Mostrar-se-á não só o funcionamento do gerador mas também de que modo a metodologia seguida permite garantir a contrução de interfaces que, ainda que modulares relativamente à aplicação (cf. princípio de separação), apresentam um comportamento *sensível* e permanentemente *ajustado*, dada a sua grande ligação (via comunicação) à camada aplicativa.

Objectivo do GAMA-X

Com o projecto GAMA-X pretendemos, desenvolvendo as ideias inicialmente abordadas no projecto GAMA[1], complementar a linguagem de especificação CAMILA[2]* e o método de refinamento a ela associado, com um UIMS que permita especificar e gerar interfaces tanto para os

* CAMILA é uma linguagem de especificação formal modular e concorrente, por modelos, actualmente em desenvolvimento na UM sob financiamento da JNICT.

protótipos (os textos CAMILA são compilados para XMetoo[3] para posterior execução) como para as aplicações finais resultantes do processo de refinamento da especificação.

Um sistema especificado em CAMILA consiste, do ponto de vista de quem o deve utilizar, num conjunto de operações e num conjunto de tipos de dados. Cada operação tem uma assinatura que deve ser respeitada e a sua utilização só é válida quando a pré-condição a ela associada se verifica. Por seu lado os tipos de dados podem ter a si associados invariantes, que mais não são do que condições que os valores do tipo devem verificar para serem considerados válidos**.

É com base, principalmente, nesta informação que partimos para a construção de um sistema que terá por missão guiar o utilizador do protótipo/aplicação na construção de valores válidos e na invocação válida das operações.

Arquitectura do Sistema GAMA-X

Como é claro do que atrás dissemos, o conceito de validade é aqui muito importante, deste modo, a arquitectura proposta para o Módulo de Interacção com o Utilizador (MIU), gerado pelo GAMA-X, não irá respeitar totalmente a separação rígida proposta pelo modelo estrutural de Seeheim[4] entre componente semântica (Modelo da Aplicação), componente sintáctica (Controlador de Diálogo) e componente léxica (Modelo da Apresentação); embora exista um Modelo da Aplicação (ModApl), um Controlador de Diálogo (CD) e um Modelo da Apresentação (ModApr) eles terão tarefas ligeiramente diferentes do proposto no modelo, estando a semântica da aplicação presente nos três.

Sendo óbvio que a representação dada aos dados na aplicação é inaceitável para ser manipulada por um utilizador e, já que à partida é desconhecida***, não pode ser utilizada no Controlador de Diálogo, um valor num sistema interactivo poderá ter até três representações distintas. A representação dos dados na aplicação, a que chamaremos representação semântica (*smdata*), é da responsabilidade da camada computacional; no Controlador de Diálogo utilizamos como representação dos dados os tipos do XMetoo, chamamos-lhe representação sintáctica (*stdata*) dos valores; a representação dos valores na apresentação, será especificada no Modelo da Apresentação, sendo uma de um conjunto de representações possíveis, trata-se da representação léxica (*lxdata*).

** O caso típico é o tipo Data, deve estar expresso no seu invariante que Fevereiro só tem 29 ou 28 dias, dependendo de o ano ser ou não bissexto.

*** A representação final dos dados na aplicação dependerá do processo de refinamento.

Torna-se então necessário efectuar a tradução entre as diversas representações, de modo a que um valor na aplicação possa ser passado para a apresentação (passando pelo Controlador de Diálogo) e vice-versa. A tradução smdata/stdata é muito naturalmente feita pelo ModApl, já que a representação semântica depende da aplicação em causa. As funções de tradução serão as funções de refinamento e retrive obtidas durante o processo de refinamento da especificação. Para a tradução lxdata/stdata, e já que as suas representações são fixas, é acrescentado um novo módulo, entre o ModApr e o CD, que será responsável por essa tarefa. Este novo módulo Controlador de Valores recorre ao ModApl para obter informação sobre os tipos.

Como o XMetoo é utilizado para a representação sintáctica dos valores, é necessário introduzir no sistema um servidor dessa linguagem para que as traduções e as manipulações de dados no Controlador de Diálogo possam ser efectuadas. Este novo componente servirá também para efectuar a execução dos protótipos das operações que ainda não tenham sido implementadas na aplicação. Deste modo, numa fase inicial todos os pedidos de execução são feitos ao servidor XMetoo, com o progressivo refinamento das operações as invocações vão sendo feitas à aplicação, até que no final todas as operações estão disponíveis na aplicação.

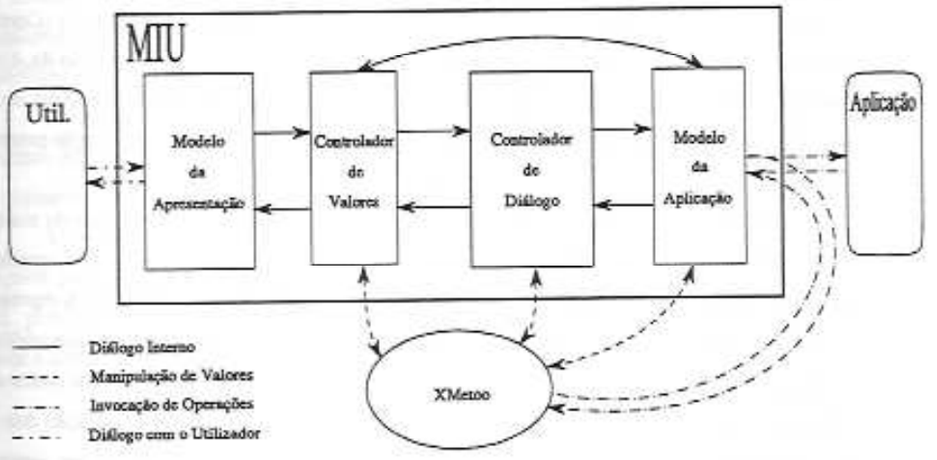

Fig. 1 - Arquitectura do MIU com utilização do XMetoo

A partir do momento em que na aplicação estão disponíveis todas as operações do sistema, o XMetoo é utilizado apenas para trabalhar sobre a stdata e efectuar as traduções. Durante o processo de refinamento é gerado um conjunto de funções equivalentes às existentes no XMetoo mas trabalhando sobre a representação de dados da aplicação. Se as restantes funções forem implementadas, podemos eliminar o XMetoo do sistema, passando todos os módulos do MIU a trabalhar sempre sobre a representação semântica com recurso ao ModApl.

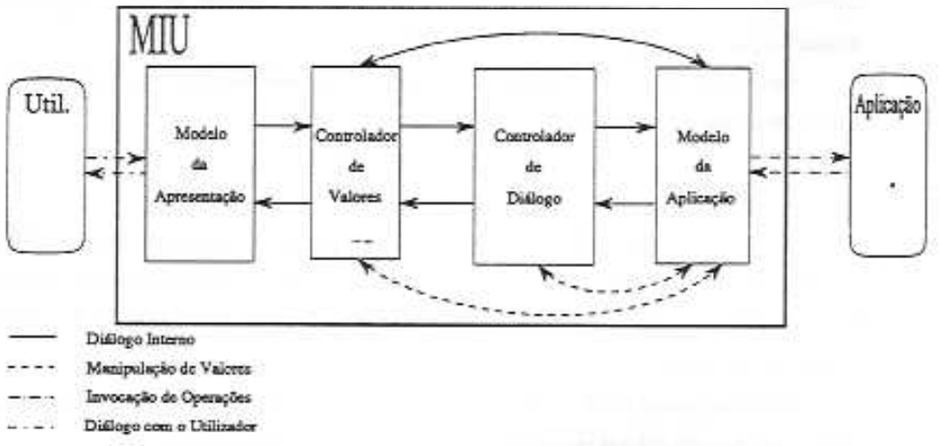

Fig. 2 - Arquitectura do MIU para uma aplicação

O Controlador de Diálogo

Como já referimos o MIU tem como tarefas principais guiar o utilizador na invocação das operações da aplicação e na introdução correcta de valores, uma outra tarefa importante (embora muitas vezes esquecida) é a apresentação dos resultados das operações. Podemos então separar a interacção em duas fases: primeiro a selecção da operação a invocar e de seguida a introdução dos valores dos argumentos dessa operação e posterior visualização do resultado da sua invocação.

Como também já dissemos as validações semânticas são muito importantes para uma boa interface Homem-Máquina em Modo Assistido. Ao nível do Controlador de Diálogo vamos apenas preocuparmo-nos com a verificação das pré-condições das operações, preocupação essa que se reflecte, como veremos, na metodologia utilizada para a sua especificação.

Podemos considerar a informação necessária para a especificação do Controlador de Diálogo a dois níveis. Por um lado qual a estrutura do diálogo (leia-se, qual a estrutura de menus da interface), esta informação dificilmente se poderá obter a partir da especificação do sistema, sendo mais natural que seja introduzida (de preferência interactivamente) por quem estiver a desenvolver a interface. Por outro lado a informação relativa às condições que se devem verificar para a correcta execução de uma dada operação. Estas podem, por sua vez, dividir-se em condições de contexto relativas ao estado interno da aplicação e condições relativas ao valores dos argumentos, este tipo de informação é deduzido a partir da especificação CAMILA do sistema.

Para a especificação do Controlador de Diálogo foi desenvolvido um formalismo a que foi dado o nome de Guiões de Interacção.

O Modelo da Aplicação

O Modelo da Aplicação é constituído pela descrição da assinatura das operações do sistema, pela definição dos tipos e seus invariantes, pelas funções de retrive e refinamento dos valores entre as representações sintáctica e semântica e pela implementação da funcionalidade do XMetoo sobre a representação semântica.

Toda a informação pode ser deduzida da especificação CAMILA, a implementação das funções do XMetoo obtem-se (na sua maior parte) durante o processo de refinamento.

O Modelo da Apresentação

O Modelo da Apresentação divide-se numa componente de descrição da interface (numa primeira fase apenas indica, para cada valor a ser lido/apresentado, qual a sua posição na janela de diálogo) e pelo Controlador da Apresentação que funciona com base na descrição mencionada e em colaboração com o Controlador de Valores.

Como o Modelo da Apresentação é independente do Controlador de Diálogo, podemos colocar mais do que um no MIU, possibilitando ao utilizador decidir (em runtime) qual o tipo de Apresentação que prefere.

O Controlador da Apresentação

O Controlador da Apresentação pode funcionar em dois modos: em modo de menus e janelas de diálogo (Modo Assistido) e em modo de linha de comando (Modo Comando).

Em modo comando os comandos/valores vão sendo validados automaticamente durante a sua introdução. A acção correspondente a seleccionar uma opção é a digitar do nome dessa opção, a introdução dos valores das variáveis faz-se escrevendo o nome da variável e o valor que lhe está associado.

Arquitectura geral do GAMA-X

No diagrama da figura três estão representados todos os componentes que permitem a geração de um MIU a partir de uma especificação CAMILA e quais as suas ligações.

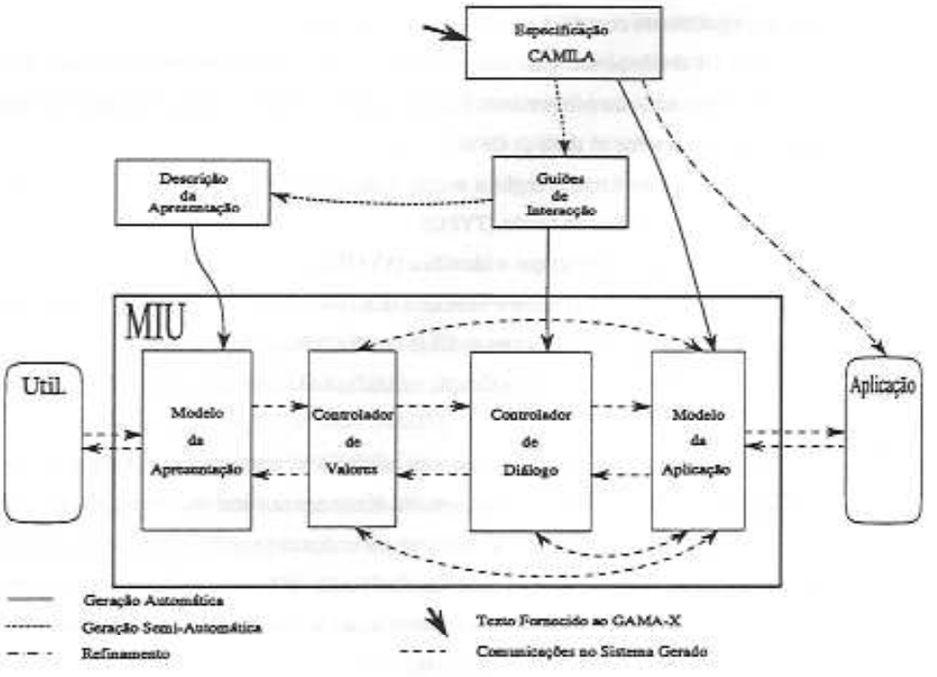

Fig. 3 - GAMA-X

Guiões de Interacção

Um Guião de Interacção (GI) descreve tipicamente um subdiálogo da interface. Foram considerados três tipos de guiões:

Menus - os GI´s do tipo MENU tem por finalidade estruturar o diálogo, tipicamente guiam o utilizador até uma dada operação que ele pretende invocar.

Contextos - os GI´s do tipo CONTEXT correspondem a contextos específicos de invocação de uma operação, especificando a leitura de valores e a apresentação do resultado produzido.

Acções - os GI´s do tipo ACTION especificam acções assíncronas que o utilizador pode realizar dentro de um dado contexto.

Embora o modo normal de activação de um dado contexto, seja percorrer o percurso especificado por GI´s MENU que leva até ele, o controlador estará preparado para receber directamente o nome de um contexto tendo então que verificar se a sua activação é válida (se o percurso especificado com GI´s MENU pode ser percorrido).

Um GI divide-se em duas componentes principais: a Componente Estática e a Componente Dinâmica. Cada uma das componentes é composta por diversas cláusulas. Vamos de seguida referir as que são comuns a todos os tipos de GI´s.

A Componente Estática engloba sempre a declaração:

- do tipo do Guião (TYPE);
- do símbolo que o identifica (SYMBOL);
- dos GI´s externos utilizados (EXTERNAL);
- de variáveis locais ao GI (STATE-CTRL);
- de variáveis da aplicação utilizadas no GI (STATE-APL).

A Componente Dinâmica engloba uma condição de contexto que determina se o GI pode ser activado (PRE-COND), temos assim informação semântica presente no Controlador de Diálogo.

Para a declaração de variáveis podem ser utilizados os tipos definidos na aplicação ou tipos pré-definidos dos GI´s. A declaração de variáveis em STATE-APL pode ser feita de dois modos:

- *var: tipo* - a variável da aplicação, de nome *var* e tipo *tipo*, é acedida directamente, apenas para consulta, pelo GI;
- *var1 <= var2* - a variável, local ao GI, *var1* toma o valor da variável da aplicação *var2*.

Os tipos pré-definidos nos GI´s são:

- inteiros (INT);
- reais (REAL);
- caracteres (CHAR);
- strings (STR);
- booleanos (BOOL).

Tipo MENU

A componente Dinâmica inclui ainda as cláusulas:

- GI - guião do tipo CONTEXT que será activado antes de serem apresentadas as opções;
- EVSEQ - especifica as opções que o GI apresenta ao utilizador (os GI´s presentes na expressão têm que ser dos tipos MENU ou CONTEXT).

Para a escrita da expressão estão disponíveis os operadores: + (ou exclusivo)-apenas uma das opções pode estar seleccionada num dado instante, | (concorrência) podemos ter mais do que uma opção seleccionada e ! (sair) depois de o GI a que o operador está associado ter sido reconhecido o GI actual termina (se tivermos !! termina o actual e o anterior, etc.). Note-se que a expressão EVSEQ é mantida activa até ser encontrado um !.

Tipo CONTEXT

A componente Estática engloba ainda a definição de:

- guiões locais utilizados (SUB-GI);
- argumentos da função a invocar (ARGS);
- variáveis da apresentação (STATE-UI).

A componente Dinâmica inclui as cláusulas:

- EVSEQ - descreve o diálogo aceite pelo guião;
- TRANS - descreve condições a verificar pelos valores lidos, acções a executar caso as condições se verifiquem e caso as condições não se verifiquem;
- STATES - acções a executar no início e no fim do guião;
- ACTIONS - guiões do tipo CONTEXT a activar e acções a executar em resposta às acções pré-definidas.

- Para a especificação de EVSEQ estão disponíveis os seguintes operadores:

  . sequenciação forte - **a.b** - a seguido de b;

  ; sequenciação fraca - **a;b** - a seguido de a e/ou b até a e b;

  , concorrência forte - **a,b** - a e/ou b até ou dois se verificarem;

  | concorrência fraca - **a|b** - a e/ou b até se verificar um deles;

  + paralelismo - **a+b** - a ou b (exclusivé);

  * repetição - **a*** - a indefinidamente repetido.

As Acções pré-definidas são:

- CANCEL - desactiva o GI;
- RESET - reinicializa o GI;
- OK - termina o GI (só disponível com EVSEQ totalmente reconhecido);
- APPLY - reinicializa o GI (só disponível com EVSEQ totalmente reconhecido).

Tipo ACTION

A componente Dinâmica inclui as cláusulas

- ACTION - acção a executar;
- GI - guião do tipo CONTEXT a activar antes da execução da acção.

A Máquina ATM

Apresentamos em anexo um extracto da especificação do Controlador de Diálogo de uma máquina ATM.

Conclusão

O GAMA-X é um sistema actualmente em desenvolvimento. A preocupação principal vai no sentido de proporcionar interfaces com elevado grau de assistência ao utilizador esensibilidade ao contexto, pelo que a componente semântica deixou de estar apenas no ModApl para estar presente em todos os componentes do sistema. A notação desenvolvida para especificar o Controlador de Diálogo procura captar esta necessidade pela introdução de condições de contexto. A informação necessária aos Modelo da Apresentação e da Aplicação já está definida, embora ainda não se tenha fixado uma notação para a sua especificação.

Referências

[1] Madeira F., Creissac F., Castro P., "GAMA - Geração Automática de Modo Assistido", Rel. Int., Depart. Informática, Univ. Minho, 1990.

[2] Barbosa L., Almeida J., "CAMILA by Example", Rel. Int., DI/INESC, Univ. Minho, Nov., 1991.

[3] The CAMILA Group, "XMETOO - User's Manual", Rel. Int., DI/INESC, Univ. Minho, Nov., 1990.

[4] Pfaff G. (Ed.), "User Interface Management Systems", Springer-Verlag, Berlin, 1985.

[5] Martins F., Oliveira J., "Archetype-Oriented User Interfaces", Computer & Graphics, Vol.14, N.1, 1990.

GI : ATM

TYPE
MENU
SYMBOL
{*IntCartao*}
EXTERNAL
LeCod, *L3*, *L5*, *L10*, *LN*, *VS*, *VM*
STATE − APL
s : *Saldo*

CONTEXT
s>0
GI
LeCod
EVSEQ
L3!|*L5*!|*L10*!|*LN*!|*VS*!|*VM*!
ACTIONS
CANCEL :→ *outCartao*()

GI : LeCod

TYPE
CONTEXT
SYMBOL
{*LeCod*}
ARGS
nc : *NumCartao*
cod : *Codigo*

EVSEQ
input(*nc*).*input*(*cod*)
TRANS
input(*cod*) : *confirma*(*nc*, *cod*) *EXCEP* *prende?*() → *getCartao*();
out("*Cartao Apreendido!*");
CANCEL
¬*prende?*() → *out*("*Codigo Errado!*")
ACTIONS
CANCEL :→ *outCartao*()

GI : LN

TYPE
CONTEXT
SYMBOL
{LN}
ARGS
q : Quantia

EVSEQ
input(q)
TRANS
input(q) : possivel?(q) → EXCEP out("Nao e possivel levantar essa quantia!")
ACTIONS
CANCEL :→ outCartao()
OK :→ levanta(q)