PROFILE EM UML PARA MODELAGEM SIMPLIFICADA DE INTERFACES GRÁFICAS EM APLICATIVOS

André Sandri

Prof. Me. Carlos Michel Betemps

UNILASALLE - www.unilasalle.com.br

30 de junho de 2006 – Curso de Ciências da Computação

Resumo

Este documento demonstra a especificação de um profile UML para a modelagem de interfaces gráficas com foco na apresentação de widgets sob o ponto de vista da usabilidade para ambientes de desenvolvimento de software que utilizam MDD, mais especificamente para MDA. Para avaliação do modelo, o profile foi implementado e testado em cima de duas ferramentas de modelagem, Borland Together Architect e Sparx Systems Enterprise Architect. Além disso, descreve os principais conceitos, problemas e justifica a solução apresentada.

Palavras-chaves: UML, Profile, GUI, Interface Gráfica, MDA, MDD, MDE, Usabilidade, OCL.

Abstract

This document demonstrates the specification of a UML profile for graphical interfaces modeling with focus in the presentation of widgets under the point of view of the usability for software development environments that use MDD, specifically for MDA. To validate the model, the profile was implemented and tested in top of two modeling tools, Borland Together Architect and Sparx Systems Enterprise Architect. Moreover, it describes the main concepts, problems and justifies the presented solution.

Keywords: UML, Profile, GUI, Graphic Interface, MDA, MDD, MDE, Usability, OCL.

1 INTRODUÇÃO

Durante uma pesquisa realizada na disciplina de Pesquisa em Ciência da Computação foi constatado que não existe um modelo padrão para a modelagem de interfaces gráficas. Por não existir um padrão, alguns fabricantes de ferramentas para MDD – *Model Driven Development* e MDA – *Model Driven Architecture* criaram suas próprias soluções para a especificação da camada de apresentação, enquanto que outros não disponibilizaram nenhum recurso para este fim.

É comum ver que a maioria dos sistemas modelados em UML não contempla a interface gráfica, somente utilizam esta linguagem para definir a camada de negócio, a camada de persistência, entre outros. A camada da interface, nestes casos, é modelada com técnicas proprietárias e finalmente construída em cima de uma ferramenta de programação.

Este trabalho de conclusão do curso de bacharelado em Ciência da Computação sugere um recurso para possibilitar a modelagem de interfaces gráficas de forma simplificada com ênfase em características de usabilidade para utilização em projetos de sistemas empresariais, podendo ser estendido para outras abordagens de desenvolvimento de *software*.

1.1 Motivação e Justificava

A relevância deste assunto afeta o futuro e o provável sucesso da arquitetura MDA, proposta pela OMG – *Object Management Group* (OMG, 2006), pois o modelo aqui proposto poderá servir como inspiração ou referência para a conscientização e a futura padronização de um diagrama específico para modelagem de interface gráfica para a camada PIM – *Platform-independent Model*, pois até o momento é um problema em aberto nesta área.

1.2 Objetivos Gerais

O objetivo principal deste trabalho é criar um *profile* UML para possibilitar a modelagem de interfaces gráficas com foco na apresentação de *widgets* sob o ponto de vista da qualidade e da usabilidade para ambientes de desenvolvimento de *software* que utilizam MDD.

1.3 Metodologia de Desenvolvimento

Como base teórica para o desenvolvimento deste trabalho foram utilizados livros e artigos científicos, além de toda a documentação publicada pela OMG (OMG, 2006) e por outras instituições que visam atingir excelência no desenvolvimento de *software* a partir de métodos e técnicas consagrados.

Este estudo contemplou o modelo do *profile* em UML, apresentando o problema e a solução, com suas justificativas, considerações e conclusões. Neste trabalho foram também apresentados os conceitos e termos pertinentes a este propósito.

Para avaliar o modelo proposto, foram utilizados os recursos oferecidos por duas ferramentas de modelagem que aceitam a criação e a utilização de *profiles* UML: o *Together Architect* 2006 (BORLAND, 2006) e o *Enterprise Architect* 6.1 (SPARX SYSTEMS, 2006).

2 PRINCIPAIS CONCEITOS

O modelo apresentado aqui poderá ser utilizado na camada PIM - *Platform-independent Model* da arquitetura MDA – *Model Driven Architecture*, publicada pela OMG, além de poder ser utilizada também em outras abordagens MDD – *Model Driven Development*.

2.1 MDA – *Model Driven Architecture*

O OMG é uma associação de várias empresas que se uniram para criar padrões visando facilitar a integração de recursos de TI, especialmente o UML - *Unified Modeling Language*, a notação padrão hoje utilizada por desenvolvedores de *software* no mundo todo (OMG, 2006).

Em 2001, após uma revisão do estado da arte das tecnologias, o OMG concluiu que a heterogeneidade de tecnologias existentes no mercado era permanente. Conforme o tempo passa, cada grande empresa é forçada a suportar novas linguagens de programação, outros sistemas operacionais, além de uma variedade de protocolos de rede. Então, decidiram que o principal objetivo, a interoperabilidade, poderia somente ser conseguido com um sistema padronizado e público de modelos e interfaces que são independentes de linguagem, sistema ou protocolo. MDA é este sistema, que engloba diversos padrões e conceitos.

2.2 *Profiles*

Apesar de a linguagem UML ter sido projetada para atingir qualquer plataforma ou tecnologia utilizando elementos genéricos, a OMG criou um mecanismo para possibilitar modelagem de artefatos utilizando terminologias de mais alto nível, permitindo utilizar UML de acordo com necessidades específicas em cima de tecnologias. Este mecanismo é chamado de *Profile*.

Profiles foram introduzidos no UML 1.3 como uma maneira de estender esta linguagem de modelagem. Antes do UML 2.0, a idéia de estender UML estava limitada a estereótipos (*stereotypes*) e *profiles*. A idéia de estender mudou ligeiramente no UML 2.0. Existe agora um novo mecanismo, o meta-modelo. Um meta-modelo é um mecanismo subjacente que define uma linguagem para expressar um modelo. Em outras palavras, o meta-modelo do UML é um modelo que é utilizado para definir o UML, e é possível adicionar novas regras ao meta-modelo de forma a estender o UML (KLEPPE, WARMER, BAST, 2003).

2.3 *Stereotypes* (estereótipos)

Estereótipo significa que um elemento do modelo tem uma utilização especial. É normalmente exibido nos diagramas com seu nome entre dois "*guillemots*", como por exemplo: <<DCOM>>

Caso o estereótipo tiver um ícone associado, é possível exibi-lo com seu ícone. Muitas ferramentas UML oferecem a possibilidade de escolher entre uma ou mais formas de visualizar um estereótipo (Figura 2.1).

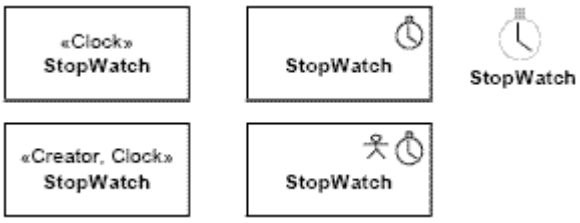

Figura 2.1 – Opções de apresentação de um estereótipo

Fonte: Unified Modeling Language: Superstructure (OMG, Versão 2.0, formal/05-07-04, p. 667, 2006)

2.4 *Tagged values* (propriedades)

Estereótipos podem ou não ter um ou mais valores associados. Este recurso oferece uma forma de possibilitar a colocação de informações pertinentes ao novo elemento.

Cada *tagged value* (valor etiquetado), ou propriedade, possui um nome, um tipo e opcionalmente um valor inicial. Por exemplo, a expressão abaixo se refere ao nome "Ano", com tipo "*Integer*", e valor inicial igual a "2006":

- *Ano*: *Integer* = 2006

2.5 PIM – *Platform-independent Model*

O padrão MDA define três etapas para a construção de um *software*. A primeira etapa consiste na criação de um modelo independente de plataforma chamado de PIM – *Platform-independent Model* (OMG, 2006).

Um PIM descreve um sistema completo para uma determinada necessidade de negócio. No PIM, o sistema é modelado a partir do ponto de vista que melhor representa o sistema: o negócio, e apenas o negócio. Neste modelo não existem indícios de que o sistema será implementado em um *mainframe* com um banco de dados relacional, ou se será implementado em uma aplicação J2EE sobre um sistema Linux (KLEPPE, WARMER, BAST, 2003).

2.6 PSM – *Platform-specific Model*

A segunda etapa definida pelo padrão MDA para a construção de um *software* consiste na criação automática de um ou mais modelos dependentes de plataforma, chamados de PSM – *Platform-specific Model*. Isto é feito utilizando transformações automatizadas do modelo representado pelo PIM (OMG, 2006).

Um PSM, inversamente ao PIM, deve refletir de forma muito próxima os conceitos e as construções utilizados na tecnologia correspondente a ser utilizada para a implementação do sistema. Em um PSM destinado a bases de dados, para citar um exemplo, os conceitos de tabela, coluna, e chaves estrangeiras devem estar claramente visíveis (KLEPPE, WARMER, BAST, 2003).

A terceira etapa é a geração do código-fonte ou da criação de uma base de dados em um SGBD a partir do PSM gerado. Como os PSMs são automaticamente gerados, cada PSM necessita ser compreendido apenas por ferramentas automatizadas de transformação e por peritos da tecnologia para a qual o PSM foi gerado. Para facilitar isso, atualmente já existem

profiles em UML para tecnologias específicas, como por exemplo, o *EJB Profile for UML* (KLEPPE, WARMER, BAST, 2003).

2.7 Usabilidade

O termo usabilidade foi definido na norma ISO/IEC 9126, que define características de qualidade para produtos de softwares, como "um conjunto de atributos de software relacionado ao esforço necessário para seu uso e para o julgamento individual de tal uso por determinado conjunto de usuários" (ISO/IEC 9126). Este conceito evoluiu e foi redefinido na norma ISO 9241-11, que define guias de utilização, como "a capacidade de um produto ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso" (ISO 9241).

Devido à importância da usabilidade em *softwares* modernos, é comum encontrar empresas que padronizaram a usabilidade em seus aplicativos ou até em suas arquiteturas de desenvolvimento. Como principal exemplo, temos a IBM e a Microsoft.

2.8 *Widgets*

Um *widget* (ou controle) é um componente da interface gráfica que um usuário de computador interage, como uma janela, uma imagem, um botão ou uma caixa de texto. *Widgets* são qualificados como objetos virtuais que imitam ou representam um objeto físico.

3 O PROBLEMA

Sem a existência de um recurso especializado que permita modelar interfaces gráficas durante a fase de análise e projeto, com um nível de abstração aproximado ao resultado final, não é possível modelar aspectos referentes à usabilidade, como posições dos elementos, informações textuais, especificar quais *widgets* serão utilizados, como serão utilizados pelo *software*, entre outros motivos.

Atualmente, para a modelagem de interfaces gráficas, existem abordagens que utilizam *profiles* UML entre outras técnicas, mas cada uma destas abordagens possuem problemas (estes problemas foram citados no documento da monografia).

Enfim, foi decidido oferecer um novo recurso para a modelagem de interfaces gráficas com a criação de um *profile* em UML.

4 A SOLUÇÃO PROPOSTA

Para a criação deste meta-modelo para modelagem de interfaces gráficas foi escolhido a técnica de *profile*, ao invés de outras alternativas. Esta técnica permite compatibilidade com ferramentas existentes, além de possibilitar sua integração sem depender da interação do fabricante da ferramenta de modelagem. Teoricamente, qualquer ferramenta que oferecer o recurso de *profile* em conformidade com a especificação da superestrutura do UML 2.0 e seguir a especificação do *Diagram Interchange* permitindo obter a posição, a hierarquia e o tamanho dos elementos modelados, será capaz de utilizar com sucesso o modelo aqui apresentado.

O principal diferencial desta proposta é a especificação de um modelo que permita a representação de uma interface gráfica com aspectos e conceitos estudados pela área da usabilidade, ou seja, um meta-modelo que permita representar o aspecto final da interface

gráfica de forma mais semelhante possível, onde apenas as restrições impostas pela tecnologia de *profile* do UML alterarão o formato de como essa interface poderá ser modelada, ou seja, forçará diferenças visuais entre a interface modelada e a interface final construída durante a construção do *software*.

4.1 Requisitos do *profile*

Para contemplar os objetivos deste trabalho, foram definidos dez requisitos para que o *profile* seja fácil de utilizar, para ser utilizável em ferramentas de modelagem, para permitir a modelagem de interfaces gráficas quanto a utilização de *widgets*, entre outros. A seguir estão resumidos os requisitos que contemplam a solução apresentada neste trabalho:

1. Um diagrama deve modelar o *layout* (disposição) de uma interface gráfica;
2. O diagrama deverá ser fácil de aprender, utilizar e entender;
3. A criação de um digrama deverá ser uma tarefa de poucos minutos;
4. A criação de um diagrama não deve exigir muito trabalho adicional;
5. A simbologia deverá representar os elementos utilizados em interfaces gráficas;
6. O modelo deverá oferecer alguma forma de extensão;
7. O modelo deverá ser simples e compreensível;
8. O modelo deverá ser abstrato para possibilitar utilizá-lo na camada PIM;
9. Prever a utilização de OCL nos elementos (widgets)*;*
10. O modelo deverá ser criado para utilização em ferramentas de modelagem;

Note que não existem requisitos com a preocupação de possibilitar navegação a partir da interface gráfica modelada, interação entre outros diagramas, prototipação, pois são características não tratadas neste trabalho. Estas características poderão ser solucionadas em futuros trabalhos.

4.2 O projeto

A partir de um levantamento dos principais *widgets* utilizados por *softwares* e oferecidos por ferramentas de programação, foram selecionados os seguintes *estereótipos* para pertencer ao modelo: *Window*, *Panel*, *GroupBox*, *TabSheet*, *TextBox*, *MemoBox*, *RichBox*, *ComboBox*, *Media*, *Table*, *Gauge*, *TrackBar*, *RadioButton*, *CheckBox*, *Button*, *Label* e *LinkedLabel*. Além destes, foram incluídos estereótipos para extensão, ou seja, para suportar de forma genérica *widgets* não previstos neste modelo. Os estereótipos de extensão são *ExtendedContainer*, *ExtendedBox* e *ExtendedText*.

O modelo e os estereótipos foram definidos sempre tendo em vista que, futuramente, para possibilitar a transformação do modelo em *software* executável, existirá um *framework* de execução encarregado de instanciar cada estereótipo em seu respectivo *widget*, realizar a conversão entre as propriedades de cada elemento do modelo para as propriedades do *widget* (conforme a plataforma gráfica em execução) e prover a validação e a persistência dos conteúdos dos controles.

4.3 O modelo proposto

No texto da monografia são apresentados todos os estereótipos que representam os principais *widgets* utilizados por *softwares* e oferecidos por ferramentas de programação. Cada elemento possui uma descrição de seu uso, os atributos definidos para possibilitar sua integração com um *framework* de execução, as restrições referentes a outros elementos do modelo (quando for necessário), a notação visual (quando aplicável) e exemplos de utilização.

Para efeitos de referência a este modelo foi utilizado o nome *UGUI Profile*, que é um acrônimo para *Usability Graphical User Interface Profile*. O modelo completo da definição do *profile* pode ser consultado no Anexo A.

4.4 Exemplos de Utilização

Para ilustrar a utilização do UGUI *Profile*, aqui é apresentado um exemplo de utilização, onde foi utilizado como origem uma interface gráfica de um software existente, e o destino é o modelo da réplica da interface gráfica utilizando os elementos propostos neste trabalho. Foi utilizado como origem a interface gráfica de formatação de parágrafo do *WordPad* da Microsoft (Figura 4.1).

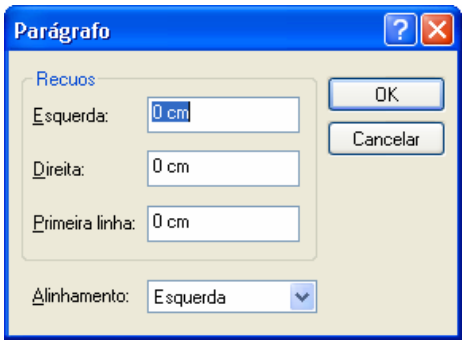

Figura 4.1 - GUI original

Fonte: Microsoft WordPad 5.1

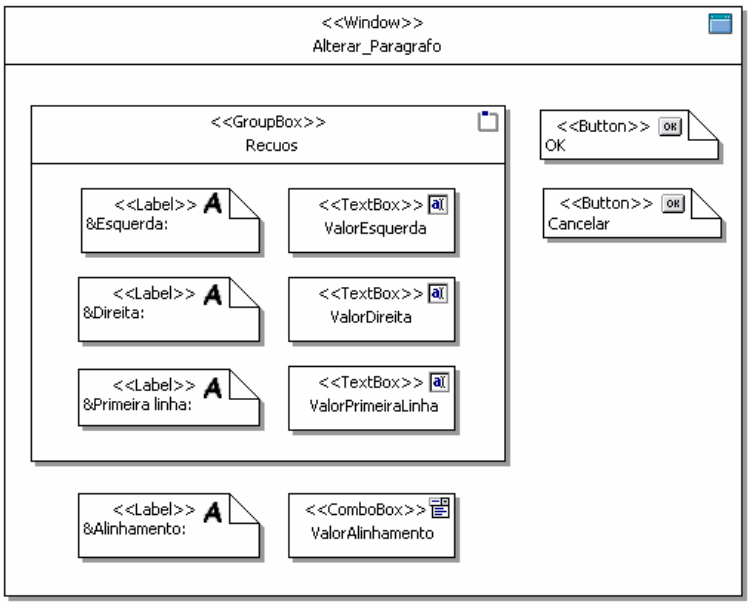

Figura 4.2 – Modelo do GUI de formatação de parágrafo

Fonte: Elaborado pelo autor

Esta interface gráfica basicamente possui uma janela modal, um *widget group box*, dois botões, vários rótulos de texto, vários *widgets text box* e um *widget combo box*. A réplica desta interface (o modelo) é apresentada na figura 4.2.

5 AVALIAÇÃO DO MODELO

Para avaliar e testar o modelo e a abordagem escolhida, foi utilizado duas ferramentas de modelagem comerciais que suportam a criação e a utilização de *profiles:* o *Together Architect* 2006 (BORLAND, 2006) e o *Enterprise Architect* 6.1 (SPARX SYSTEMS, 2006).

O *Together* tem suporte de especificação visual do *profile* de forma muito semelhante à forma proposta pela OMG. É possível configurar um estereótipo para ter sua própria representação gráfica utilizando um arquivo gráfico no formato SVG 1.1. Além disso, é possível associar um ícone para a representação do elemento na paleta de componentes.

Esta ferramenta também suporta a especificação de restrições em OCL com o auxílio de um editor específico para esta linguagem. Porém, a validação sistemática das expressões escritas nesta linguagem ainda não é suportada em tempo de definição do *profile*, somente são suportadas durante a utilização (instanciação) do *profile* em si.

Durante a avaliação com esta ferramenta, foram encontrados problemas referentes à utilização de imagens e ícones utilizados para especializar a notação visual dos elementos. Um dos problemas é que, ao utilizar um estereótipo que estende a meta-classe *Component* (para prover o comportamento do tipo *container*) que contenha uma imagem associada, não é possível visualizar os elementos aninhados de forma gráfica, apesar de que é possível verificar que os elementos continuam obedecendo a relação hierárquica aos quais foram adicionados. Este aspecto particular de comportamento não é explicitamente previsto na Superestrutura e nem na Infra-estrutura das atuais versões do UML (OMG, 2006), o qual poderia ser previsto em futuras revisões destes documentos.

Outro problema é que, ao utilizar um estereótipo que não seja do tipo *container*, e que contenha uma imagem associada, não é apresentado o nome do elemento, comportamento que é previsto na Superestrutura e na Infra-estrutura das atuais versões do UML (OMG, 2006) ao utilizar imagens em um estereótipo. Inclusive, para estereótipos com notação textual, a utilização de uma imagem impede que a informação textual seja exibida.

Uma forma de ignorar estes problemas é não utilizar imagens nos estereótipos, fato que prejudica a utilização do *profile*, visto que a notação visual especificada tem como objetivo facilitar a visualização e a compreensão dos *widgets* modelados.

A ferramenta de modelagem *Enterprise Architect* apresentou problemas semelhantes.

6 CONCLUSÃO

O modelo apresentado neste trabalho de conclusão oferece um recurso para possibilitar a modelagem de interfaces gráficas com aspectos de usabilidade para ambientes MDD, mais precisamente para a camada PIM do MDA, pois é independente de tecnologia, possibilita modelar o comportamento básico dos *widgets* ao serem executados, prevê propriedades para serem utilizadas para a validação, persistência e apresentação dos dados, e permite construir o diagrama de forma aproximada aos elementos que compõem as atuais interfaces gráficas. Todos os requisitos propostos definidos foram satisfeitos e justificados no texto da monografia do trabalho.

Com a avaliação do modelo em cima de ferramentas de modelagem foram encontrados problemas referentes à utilização de imagens nos estereótipos e da utilização de restrições OCL. Portanto, conclui-se que existe uma necessidade de amadurecimento das ferramentas quanto a estes aspectos, além de que é recomendável reforçar a padronização da OMG para evitar estes tipos de problemas, já que estas especificações não previram explicitamente estes tipos de necessidades. Mesmo assim, é possível utilizar o UGUI *Profile* nestas ferramentas de modelagem sem a utilização da notação gráfica (ícones), pois estas ferramentas exibem o nome do estereótipo em cada elemento.

ANEXO A – DIAGRAMA DO UGUI PROFILE

REFERÊNCIAS

OMG 2006. **OMG Web Site**, Object Management Group. Disponível em <http://www.omg.org>. Acesso em 17 jun. 2006.

SHNEIDERMAN, B., PLAISANT, C. **Designing the user interface**: strategies for effective human computer interaction. Addison-Wesley: 4 ed., 652p, 2004.

BLANKENHORN, K. **A UML Profile for GUI Layout**. Master's Thesis. University of Applied Sciences. Furtwangen, 2004.

KLEPPE, A., WARMER, J., BAST, W. **MDA Explained: The Model Driven Architecture™**: Practice and Promise. Addison Wesley, 2003.

BORLAND, **Together Architect 2006**. Cupertino, EUA. Disponível em <http://www.borland.com/br/products/together/index.html >. Acesso em 17 jun. 2006.

SPARX SYSTEMS, **Enterprise Architect 6.1.** Creswick, Austrália. Disponível em <http://www.sparxsystems.com/products/ea.html>. Acesso em 17 jun. 2006.