UNIVERSIDADE da MADEIRA

MODELAÇÃO DE INTERFACES GRÁFICAS

NO ÂMBITO DO PROJECTIT

Carlos Alberto Rodrigues Martins

(Licenciado)

Tese Submetida à Universidade da Madeira para a

Obtenção do Grau de Mestre em Engenharia Informática

Funchal – Portugal

Agosto 2007

Orientador:

Professor Doutor Alberto Manuel Rodrigues da Silva
Professor Auxiliar no Departamento de Engenharia Informática
do Instituto Superior Técnico - Universidade Técnica de Lisboa

Resumo

Existe um consenso alargado a respeito da utilização do UML como linguagem de modelação standard no desenho de software e de sistemas de informação em geral. No entanto entre os *user interface designers*, o consenso não parece existir no que diz respeito às tarefas específicas da modelação de interfaces com o utilizador. Neste trabalho de investigação são analisadas diversas abordagens para o desenho e produção de interfaces gráficas, nomeadamente: (1) utilização de ferramentas para construção de interfaces gráficas (*UI builders tools approach*); (2) utilização de ferramentas para desenho e reconhecimento de esboços (*UI sketching tools approach*); (3) utilização de linguagens baseadas em XML (*XML-based languages approach*) e (4) utilização de linguagens de modelação (*MDE tools approach*). São também analisadas em maior detalhe, diversas iniciativas baseadas na abordagem que utiliza linguagens de modelação, para a produção de interfaces gráficas. Com base nesta análise, estendemos a versão 1 do perfil XIS com o objectivo de criar vistas específicas para modelar interfaces gráficas.

O perfil XIS é um conjunto coerente de extensões UML que permite modelar sistemas interactivos. Neste trabalho de investigação propomos a segunda versão do perfil XIS, que é um componente crucial do projecto de investigação ProjectIT. Esta nova versão do perfil XIS permite modelar sistemas interactivos, segundo três vistas principais: (1) a *Entities View* (*Domain View* e *BusinessEntities View*), (2) a *Use-Cases View* (*UseCases View* e *Actors View*) e (3) a *User-Interfaces View* (*NavigationSpace View* e *InteractionSpace View*), sendo esta última dedicada a modelar os aspectos relacionados com as interfaces com o utilizador.

A abordagem realizada segue a recomendação *Model Driven Architecture* (MDA) da OMG, segundo a qual os modelos descritos através do perfil XIS são especificados de forma independente da plataforma. Através de mecanismos de transformação de modelo-para–código, os sistemas podem ser transformados para diversas plataformas computacionais específicas. Neste trabalho de investigação, são apresentados aspectos relacionados com a criação de *templates* para geração de código, desenvolvidos para duas plataformas computacionais específicas: (1) a plataforma Microsoft Windows Forms.NET e (2) a plataforma Microsoft ASP.NET.

Palavras Chave

Engenharia de Software

Arquitectura Baseada em Modelos

Desenvolvimento Baseado em Modelos

Interacção Pessoa-Máquina

Desenho Interfaces Gráficas

Unified Modeling Language

Linguagens de Modelação

Perfil UML

Mecanismos de Geração

Abstract

The Unified Modeling Language is the industry standard language for modeling and designing software systems. It has been widely accepted by software developers, but not so much by user interface designers. In this investigation, we analise several approaches to design and produce graphical user interfaces, such as: (1) user interface builder tools approach; (2) user interface sketching tools approach; (3) XML-based languages approach and (4) model driven tools approach. We also analise in detail, several initiatives to produce graphical user interfaces, based on the model driven tools approach. This analysis influences our work and helps us to extend the first version of the XIS profile, creating specific views to model graphical user interfaces.

The XIS profile is a set of coherent UML extensions that allows a high-level, visual modeling way to design interactive systems. In this dissertation we propose the second version of the XIS profile, which is a profile for extreme modeling interactive systems, and a crucial component of the ProjectIT research project. The XIS profile proposes an integrated set of views, namely: (1) the Entities View (Domain View and BusinessEntities View); (2) the Use-Cases View (UseCases View and Actors View) and (3) the User-Interfaces View (NavigationSpace View and InteractionSpace View)

Our approach follows the OMG recommendation (Model Driven Architecture), so the XIS profile allows the design of interactive systems at a PIM level (platform independent model). Then, the systems can be targeted, through model-to-code specific transformations, to different platforms, such as Web, desktop or mobile specific platforms. This dissertation presents some aspects related to the development of model-to-code template features, to produce software and documentation artifacts from models, using generative programming techniques. We present templates to generate code for two specific platforms: (1) the Microsoft Windows Forms.NET platform and (2) the Microsoft Windows ASP.NET platform.

Keywords

Software Engineering

Model Driven Architecture

Model Driven Development

Human Computer Interaction

User Interfaces Design

Unified Modeling Language

Modeling Languages

UML Profile

Code Generation

Agradecimentos

Quero começar por agradecer ao meu orientador o Professor Doutor Alberto Rodrigues da Silva, pela confiança que depositou em mim ao aceitar orientar esta dissertação, bem como pelo nível de exigência e capacidade de orientação demonstrado ao longo deste trabalho de investigação. À toda a equipa do ProjectIT, principalmente ao Rui Silva com quem trabalhei mais de perto, ao João Saraiva e ao David Ferreira por todo o apoio dispensado para a minha integração no projecto.

Aos meus pais, porque sempre apostaram na minha formação ao longo dos anos, apoiando todas as minhas escolhas quer ao nível académico como profissional. Aos meus irmãos, cunhados e sobrinhos pela força e preocupação que manifestaram ao longo do mestrado.

À Raquel por me acompanhar nos bons e maus momentos da minha vida, aceitando às minhas ausências e por me compreender nos dias de maior cansaço, sem nunca colocar em causa a minha investigação.

A todos os meus colegas de mestrado, principalmente aqueles que me acompanharam nos trabalhos de grupo e sessões de estudo ao longo da parte lectiva deste mestrado.

Aos meus chefes durante o período em que decorreu o mestrado, nomeadamente ao João Vila, ao Carlos Rebolo e ao Ricardo Manica por facilitarem a flexibilidade do meu horário nos períodos mais críticos da minha investigação. À equipa de Desenvolvimento de Sistemas de Informação do Serviço Regional de Saúde, por terem assegurado com responsabilidade e profissionalismo o normal funcionamento de todos os projectos durante às minhas ausências, com destaque para o Abreu, Irineu e Paulo que me acompanham há mais tempo.

A todos os meus amigos, que sempre se preocuparam e perguntaram pelo estado da investigação, apoiando-me para não desistir, em especial ao Sérgio, Salgado, Abreu, Armando, Carmo, Helena, Ricardo, a todo o grupo da GRIP pelos momentos de convívio e descompressão ao longo do ano, ao grupo da Espuma d'Ouro, ao grupo das viagens e ao fantástico grupo do 10º e 11º ano da Francisco Franco.

À UMa, pelo apoio financeiro dispensado para a realização de três viagens à Lisboa para reunir com o meu orientador. À UMa e ao INESC-ID por todo o apoio logístico.

Índice

RESUMO V
PALAVRAS CHAVE VII
ABSTRACT IX
KEYWORDS XI
AGRADECIMENTOS XIII
ÍNDICE XV
LISTA DE FIGURAS XVII
LISTA DE TABELAS XXI
ACRÓNIMOS XXIII
1. INTRODUÇÃO 1
1.1. ENQUADRAMENTO 2
1.2. PROBLEMA 4
1.3. PUBLICAÇÕES 5
1.4. CASO DE ESTUDO 6
1.5. ORGANIZAÇÃO DA DISSERTAÇÃO 7
2. ESTADO DA ARTE 9
2.1. INTRODUÇÃO 9
2.1.1. Conceitos Gerais 9
2.1.2. Toolkits, Controlos e Janelas 10
2.2. ABORDAGENS DE PRODUÇÃO DE INTERFACES GRÁFICAS 11
2.2.1. Ferramentas para Construção de Interfaces Gráficas (UI Builders) 12
2.2.2. Ferramentas de Desenho e Reconhecimento de Esboços (sketching) 13
2.2.3. Linguagens Baseadas em XML 14
2.2.4. Linguagens de Modelação 15
2.2.5. Análise Comparativa 17
2.3. PADRÕES DE DESENHO DE INTERFACES GRÁFICAS 18
2.3.1. Padrões de Escolha 18
2.3.2. Padrão Filtro Contínuo 20
2.3.3. Padrão Navegação por Menus 21
2.3.4. Padrão Apresentação em Grelha 22
2.3.5. Padrão Tabuladores 23
2.3.6. Padrão Tabela 24
2.3.7. Padrão Dupla Lista 25
2.4. CONCLUSÃO 27
3. PERFIL XIS 28
3.1. VISÃO GERAL DO PERFIL XIS 28
3.2. ENTITIES VIEW 30

3.2.1. Domain View *30*
3.2.2. BusinessEntities View *31*
3.3. USE-CASES VIEW 33
3.3.1. Actors View *33*
3.3.2. UseCases View *34*
3.4. USER-INTERFACES VIEW 35
3.4.1. NavigationSpace View *35*
3.4.2. InteractionSpace View *36*
4. DESENHO DE INTERFACES GRÁFICAS 39
4.1. METAMODELO 39
4.2. MODELAÇÃO DA NAVEGAÇÃO 41
4.3. MODELAÇÃO DO ESPAÇO DE INTERACÇÃO 42
4.3.1. XisInteractionSpace *42*
4.3.2. XisDataElement *43*
4.3.3. XisActionElement *47*
4.3.4. XisInteractionCompositeElement *51*
4.4. CONCLUSÃO 62
5. GERAÇÃO DE INTERFACES GRÁFICAS 63
5.1. FERRAMENTA PROJECT-IT-STUDIO 63
5.2. GERAÇÃO PARA WINDOWS FORMS.NET 67
5.2.1. A Plataforma Windows Forms.NET *67*
5.2.2. Apresentação para Windows Forms.NET *68*
5.2.3. Janelas e Tipos de Janelas *70*
5.2.4. Controlos e Tipos de Controlos *70*
5.3. GERAÇÃO PARA ASP.NET 74
5.3.1. A Plataforma ASP.NET *74*
5.3.2. Apresentação para ASP.NET *75*
5.3.3. Páginas Web *75*
5.3.4. Controlos e Tipos de Controlos *75*
5.4. CONCLUSÃO 79
6. TRABALHO RELACIONADO 81
6.1. USER-EXPERIENCE MODELING LANGUAGE 82
6.2. UNIFIED MODELING LANGUAGE FOR INTERACTIVE APPLICATIONS 85
6.3. UML-BASED WEB ENGINEERING APPROACH 89
6.4. WISDOM UML PROFILE E PROTÓTIPOS CANÓNICOS ABSTRACTOS 94
6.5. ANÁLISE COMPARATIVA 100
7. CONCLUSÃO E TRABALHO FUTURO 107
7.1. CONCLUSÃO 107
7.2. TRABALHO FUTURO 108
REFERENCES 111

Lista de Figuras

Figura 1.1 Visão geral da abordagem ProjectIT (extraído de [Silva, Videira et al. 2006]). *3*
Figura 2.1: Abordagens de produção de interfaces gráficas *11*
Figura 2.2: Exemplo de um esboço e respectiva interface após mecanismo de rendering (extraído de [Coyette 2006]) *13*
Figura 2.3: Exemplo de um extracto da definição de uma interface em UIML e respectiva interface gerada após mecanismo de rendering (adaptado de [UIML.org 1997]) *15*
Figura 2.4: Transformação de modelos PIM em modelos PSM [Sparx_Systems 2007] *16*
Figura 2.5: Exemplo do Padrão Selecção Booleana *19*
Figura 2.6: Exemplo do Padrão Escolha *19*
Figura 2.7: Exemplo do Padrão Escolha a partir de um conjunto longo *20*
Figura 2.8: Exemplo do Padrão Filtro Contínuo *21*
Figura 2.9: Exemplo do Padrão Navegação por Menus *22*
Figura 2.10: Exemplo do Padrão Apresentação em Grelha *23*
Figura 2.11: Exemplo do Padrão Tabuladores *24*
Figura 2.12: Exemplo do Padrão Tabela *25*
Figura 2.13: Exemplo do Padrão Dupla Lista *26*
Figura 3.1: A organização em vistas do perfil XIS *29*
Figura 3.2: Exemplo da vista Domain View para o sistema DocPro *31*
Figura 3.3: Exemplo da definição da business entity OrganizationBE. *32*
Figura 3.4: Exemplo da vista Actors View para o sistema DocPro *34*
Figura 3.5: Exemplo de parte da vista UseCases View para o sistema DocPro *34*
Figura 3.6: Exemplo da vista NavigationSpace View para o sistema DocPro *36*
Figura 3.7: Exemplo da vista InteractionSpace View para o espaço de interacção OrganizationEditor . *38*
Figura 4.1: Metamodelo da NavigationSpace View. *39*
Figura 4.2: Metamodelo da InteractionSpace View. *40*
Figura 4.3: Exemplo da vista NavigationSpace View, parcial, para o sistema DocPro *41*
Figura 4.4: Metamodelo parcial (XisInteractionSpace) da InteractionSpace View *42*
Figura 4.5: Representação de um XisInteractionSpace cuja marca firstByDefault é true, cuja marca interactionSpaceType é Form e respectiva geração para a plataforma Windows Forms.NET. *43*
Figura 4.6: Metamodelo parcial (XisDataElement) da InteractionSpace View *44*
Figura 4.7: Representação de um XisDomainElement cuja marca controlType é Date e respectiva geração para a plataforma Windows Forms.NET. *45*
Figura 4.8: Metamodelo parcial (XisDomainElement) da InteractionSpace View *46*
Figura 4.9: Metamodelo parcial (XisOtherElement) da InteractionSpace View *46*
Figura 4.10: Metamodelo parcial (XisActionElement) da vista InteractionSpace View *47*
Figura 4.11: Representação de um XisActionElement cuja marca actionType é Button e respectiva geração para a plataforma Windows Forms.NET. *48*

Figura 4.12: Representação de um XisInteractionCompositeElement com dos itens de menu e respectiva geração para a plataforma Windows Forms.NET. *49*
Figura 4.13: Representação de um Menu com cinco itens, sendo um deles um separador e a respectiva geração para a plataforma Windows Forms.NET. *50*
Figura 4.14: Metamodelo parcial (XisInteractionCompositeElement) da InteractionSpace View *52*
Figura 4.15: Representação de um XisInteractionCompositeElement cuja marca compositeElementType é Menu e respectiva geração para a plataforma Windows Forms.NET. *53*
Figura 4.16: Representação de um XisInteractionCompositeElement cuja marca compositeElementType é GroupBox *54*
Figura 4.17: Geração do exemplo da Figura 4.16 para a plataforma Windows Forms.NET *54*
Figura 4.18: Representação de um XisInteractionCompositeElement cuja marca compositeElementType é Tab (Tabulador) *55*
Figura 4.19: Geração do exemplo da Figura 4.18 para a plataforma Windows Forms.NET *55*
Figura 4.20: Representação de um XisInteractionCompositeElement cuja marca compositeElementType é ListSelect e respectiva geração para a plataforma Windows Forms.NET *56*
Figura 4.21: Representação de um XisInteractionCompositeElement cuja marca compositeElementType é Table *57*
Figura 4.22: Geração do exemplo da Figura 4.18 para a plataforma Windows Forms.NET *57*
Figura 4.23: Representação de um XisInteractionCompositeElement (Pessoas_T) cuja marca compositeElementType é TableAssociate (padrão Dupla Lista, utilizando tabelas) *59*
Figura 4.24: Geração do exemplo da Figura 4.23 para a plataforma Windows Forms.NET *59*
Figura 4.25: Representação de um XisInteractionCompositeElement cuja marca compositeElementType é DialogQuestion *60*
Figura 4.26: Geração do exemplo da Figura 4.25 para a plataforma Windows Forms.NET *61*
Figura 5.1: Componentes do ProjectIT-Studio (extraído de [Silva, Videira et al. 2006]) *63*
Figura 5.2: Entidades da Arquitectura de Software (extraído de [Silva 2006]) *64*
Figura 5.3: Editor de Arquitecturas de Software *65*
Figura 5.4: Editor de Templates *65*
Figura 5.5: Entidades do Processo de Geração (extraído de [Silva, Videira et al. 2006]) *66*
Figura 5.6: Editor de Processos de Geração *67*
Figura 6.1: Exemplo de um Participants Diagram para o caso de estudo DocPro *83*
Figura 6.2: Exemplo de um diagrama de estados UX [Kozaczynski e Thario 2002] *84*
Figura 6.3: Modelo de apresentação, utilizando o diagrama de interface com o utilizador *86*
Figura 6.4: Modelação UML, representando o tipo de comportamento de selecção por ordem independente (extraído de [Silva e Paton 2000]) *86*
Figura 6.5: Modelação UML, representando o tipo de comportamento opcional (extraído de [Silva e Paton 2000]) *87*
Figura 6.6: Modelação UML, representando o tipo de comportamento de repetição (extraído de [Silva e Paton 2000]) *87*
Figura 6.7: Modelação UMLi, representando os tipos de comportamento: (a) de selecção por ordem independente; (b) opcional e (c) de repetição (extraído de [Silva e Paton 2000]) *88*

Figura 6.8: Metamodelo do modelo de navegação (extraído de [Hennicker e Koch 2001]) *90*
Figura 6.9: Exemplo do modelo de navegação do caso de estudo DocPro..... *90*
Figura 6.10: Metamodelo do Abstract User Interface Model (extraído de [Hennicker e Koch 2001]) *92*
Figura 6.11: Exemplo de uma UI View - Abstract User Interface Model para o caso de estudo DocPro *92*
Figura 6.12: Cenário representado através de um Storyboard Model..... *93*
Figura 6.13: Exemplo de um modelo de apresentação Wisdom, representando parte do caso de estudo DocPro..... *95*
Figura 6.14: Notação Canónica Abstracta completa *97*
Figura 6.15: Exemplo de um Protótipo Canónico Abstracto para o espaço de interacção Organization Editor do caso de estudo DocPro *98*
Figura 6.16: Correspondência entre estereótipos Wisdom e os componentes Canónicos Abstractos *99*

Lista de Tabelas

Tabela 2.1: Tabela comparativa das diversas abordagens. *17*
Tabela 3.1: Estereótipos da vista Domain View *30*
Tabela 3.2: Estereótipos da vista BusinessEntities View *32*
Tabela 3.3: Estereótipo da vista Actors View *33*
Tabela 3.4: Estereótipos da vista UseCases View *34*
Tabela 3.5: Estereótipos da vista NavigationSpace View *35*
Tabela 3.6: Estereótipos da vista InteractionSpace View *37*
Tabela 4.1: Marcas para o estereótipo XisInteractionSpace *42*
Tabela 4.2: Marca para o estereótipo XisDataElement *44*
Tabela 4.3: Marca para o estereótipo XisOtherElement *46*
Tabela 4.4: Marcas para o estereótipo XisActionElement *47*
Tabela 4.5: Marca para o estereótipo XisInteractionCompositeElement *52*
Tabela 5.1: Tipos de mensagens das caixas de diálogo geradas *71*
Tabela 6.1: Estereótipos utilizados no perfil UX UML *82*
Tabela 6.2: Marcas definidas para o estereótipo Form *82*
Tabela 6.3: Estereótipos utilizados no diagrama de interface com o utilizador UMLi *85*
Tabela 6.4: Estereótipos utilizados no modelo de navegação *89*
Tabela 6.5: Estereótipos utilizados no modelo abstracto do espaço de interacção. *91*
Tabela 6.6: Estereótipos utilizados no modelo de apresentação Wisdom *94*
Tabela 6.7: Comparação dos conceitos de modelação, para as diversas iniciativas baseadas na abordagem de produção de interfaces gráficas, utilizando linguagens de modelação *101*
Tabela 6.8: Comparação das características das diversas iniciativas, baseadas na abordagem de produção de interfaces gráficas, utilizando linguagens de modelação *101*

Acrónimos

API - Application Programming Interface

ASP – Active Server Pages

AUIML - Abstract User Interface Markup Language

CASE – Computer Aided Software Design

CLI – Command Line Interface

CTT – ConcurTaskTrees

GUI – Graphical User Interface

HCI – Human Computer Interaction

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

IBM - International Business Machines

IDE – Integrated Development Envinonment

INESC-ID – Instituto de Engenharia de Sistemas e Computadores – Investigação e Desenvolvimento

JSP – Java Server Pages

MDA – Model Driven Architecture

MDD – Model Driven Development

MDE – Model Driven Engineering

MDIContainer – Multiple Document Interface Container

NFR – Non-Functional Requirements

OCL – Object Constraint Language

OMG – Object Management Group

PIM – Platform Independent Model

PIT – ProjectIT

PHP – Hypertext Preprocessor

PSM - Platform Specific Model

SILK – Sketching Interfaces Like Krazy

SQL – Structured Query Language

TI – Tecnologias de Informação

UI – User Interface

UIML – User Interface Markup Language

UML – Unified Modeling Language

UML*i* – Unified Modeling Language for Interactive Applications

URL - Uniform Resource Locator

UWE – UML-based Web Engineering Approach

UX – User-Experience Modeling Language

W3C – World Wide Web Consortium

WISDOM – Whitewater Interactive System Development with Object Models

XIML - Extensible Interface Markup Language

XIS – Extreme Modeling of Interactive Systems

XML – Extensible Markup Language

1. Introdução

Este trabalho de investigação incide em duas áreas principais: a área da interacção pessoa-máquina (HCI) e de desenho de interfaces com o utilizador e a área de desenvolvimento de software baseado em modelos.

A área de **Interacção Pessoa-Máquina** tem por objectivo estudar o modo como as pessoas interagem com os sistemas de hardware e software. Durante muito tempo a área de engenharia de software não se preocupou com a forma como as pessoas interagiam com os sistemas e não investiu na usabilidade dos sistemas desenvolvidos. Ainda hoje em dia, há muitos utilizadores que se queixam que os engenheiros de software não prestam atenção suficiente aos aspectos de usabilidade nas aplicações que desenvolvem e argumentam que estas não são "amigáveis". A variabilidade humana faz com que utilizadores diferentes interajam com os sistemas, aprendam os sistemas, memorizem as suas opções de forma diferente uns dos outros. As próprias preferências de determinado utilizador evoluem gradualmente desde que o utilizador começa a interagir com determinado sistema, até ao momento em que se torna um utilizador experiente no mesmo. Isto é um desafio para os engenheiros que desenvolvem interfaces com o utilizador.

Por outro lado, atendendo à dimensão e complexidade da generalidade dos sistemas de informação que são produzidos hoje em dia, à enorme pressão na redução dos prazos, na alteração constante dos requisitos e na respectiva satisfação destes, uma das boas práticas emergentes é a construção de sistemas, parcial ou totalmente, segundo o paradigma de **desenvolvimento baseado em modelos** (*Model Driven Development* - MDD), seguindo ou não a recomendação *Model Driven Architecture* (MDA) [OMG ---a]. Todas as abordagens baseadas em modelos encontram-se enquadradas numa abordagem mais geral, a abordagem *Model Driven Engineering* (MDE) [Favre 2004; Schmidt 2006].

O MDD é um paradigma de desenvolvimento de software que dá primazia à utilização de modelos. Tem como objectivo descrever as funcionalidades do sistema, utilizando modelos, passando o foco do desenvolvimento do software, da codificação para a modelação. Os modelos são fundamentais, em projectos de média e grande dimensão, para partilhar a visão do projecto e facilitar a comunicação entre todos os interessados,

facilitando desta forma a gestão do projecto. No entanto, para além destas vantagens, o paradigma MDD pressupõe que os próprios modelos sejam refinados, de tal forma que em conjunto com arquitecturas de modelos e com arquitecturas de software, e através de técnicas de geração de código, seja possível a produção automática de parte significativa dos artefactos dos sistemas (e.g., código fonte, *scripts* SQL, *scripts* XML). Ou seja, aos modelos são aplicados mecanismos automáticos que permitam a geração dos diferentes artefactos que compõem o sistema, transformando-os num sistema real.

1.1. Enquadramento

Este trabalho de investigação enquadra-se no âmbito do ProjectIT. O ProjectIT [Silva 2004] é um programa de investigação cujo objectivo é a análise, integração e suporte das melhores práticas da gestão e implementação de projectos de TI, tendo em consideração as suas especificidades particulares e que serve de contexto para enquadrar e lançar vários projectos concretos de engenharia e investigação, teses de doutoramento, teses de mestrado e trabalhos finais de curso que se vão desenvolvendo a curto e médio prazo no contexto do Grupo de Sistemas de Informação do INESC-ID (*Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento*) [INESC-ID --]. Todos esses projectos estão ligados às problemáticas da concepção, desenvolvimento, gestão e operação de sistemas de informação. Apresenta uma razoável massa crítica e sinergia, atendendo ao facto de integrar o conhecimento adquirido pelo grupo em vários projectos ao longo dos últimos anos.

O ProjectIT pretende produzir vários resultados, nomeadamente (1) uma ferramenta colaborativa com interface *web* designada por "ProjectIT-Enterprise"; e (2) uma ferramenta com interface Windows, para realização de actividades de maior produtividade, designada genericamente por "ProjectIT-Studio". Estas duas ferramentas apresentarão fortes mecanismos de integração e de complementaridade. A versão *Studio* tem como principal objectivo, providenciar mecanismos de elevada produtividade ligados à gestão e especificação de requisitos, desenho de modelos, geração automática de código e desenvolvimento de software. Por outro lado, a versão *Enterprise* providencia mecanismos de colaboração para equipas de desenvolvimento de média e grande dimensão, privilegiando actividades ligadas à gestão do projecto e à gestão documental, tais como, controlo de versões, gestão do tempo, da qualidade, ou do risco.

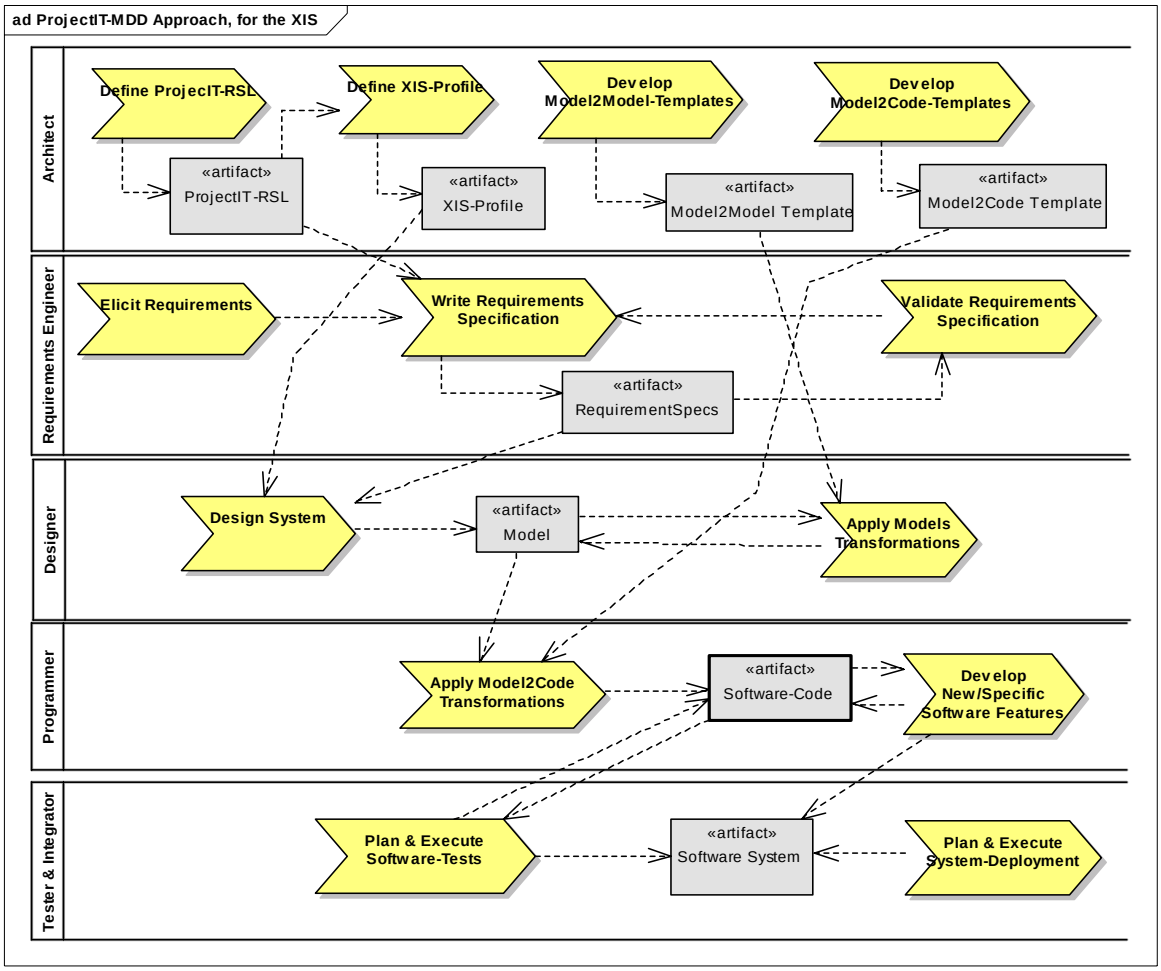


Figura 1.1 Visão geral da abordagem ProjectIT (extraído de [Silva, Videira et al. 2006]).

A abordagem ProjectIT [Silva, Videira et al. 2006; Silva, Saraiva et al. 2007a] assenta nos princípios do desenvolvimento de software baseado em modelos. Pretende privilegiar as actividades de projecto (como sejam a engenharia de requisitos, a análise e o desenho), em detrimento das actividades de produção (como sejam a programação, os testes e integração de componentes e de sistema), de modo que estas sejam realizadas tanto quanto possível, de forma automática. Genericamente, a abordagem ProjectIT recebe os requisitos da aplicação (e.g., requisitos funcionais, não funcionais e de desenvolvimento) e produz os artefactos digitais da aplicação (e.g., ficheiros de código fonte, scripts de configuração). A Figura 1.1 ilustra as principais actividades, artefactos e intervenientes da abordagem ProjectIT.

1.2. Problema

Tendo em conta às grandes pressões com que se deparam os projectos de engenharia de software, tais como o aumento da dificuldade dos projectos, prazos e orçamentos reduzidos, alteração constante dos requisitos, tarefas de manutenção constante, é de crucial importância que se investiguem mecanismos, que tenham como objectivo, o aumento de produtividade dos projectos de engenharia de software. Através do paradigma de desenvolvimento baseado em modelos, é possível a construção automática de sistemas, de forma parcial ou total. Para tal, é necessário estender o UML através da criação de perfis, que nos permitam representar diversos aspectos dos sistemas de software.

No âmbito do ProjectIT, já tinha sido estudada e desenvolvida uma primeira versão do perfil XIS [Silva 2003b; Silva 2003a; Silva, Lemos et al. 2003], com o objectivo de modelar e representar sistemas interactivos. No entanto, esta primeira versão apresentou algumas limitações, nomeadamente, não permitia modelar explicitamente as interfaces com o utilizador.

O desenho de interfaces com o utilizador é uma tarefa muito importante no trabalho de desenvolvimento de software. A usabilidade do sistema deve ser uma preocupação nos projectos de engenharia de software. Devido à grande variabilidade humana torna-se difícil criar interfaces que sejam adequadas para todos os utilizadores.

Existem uma série de questões que orientam esta dissertação e que resumem o problema a solucionar nesta investigação:

1. Como podemos desenhar e especificar interfaces com o utilizador de uma forma independente da plataforma?
2. Que padrões de desenho de interfaces podem ser aproveitadas no desenho, especificação e produção de interfaces com o utilizador?
3. Como podemos, a partir dos desenhos e especificações das interfaces com o utilizador, gerar interfaces para diversas plataformas computacionais específicas?

Ainda não existe consenso sobre a forma como devem ser representados as interfaces com o utilizador. O objectivo deste trabalho de investigação é analisar o estado da arte,

no que diz respeito às diferentes abordagens de desenho e produção de interfaces com o utilizador e analisar diversas iniciativas baseadas no paradigma MDD com impacto na modelação de interfaces com o utilizador.

A solução a encontrar para desenhar e especificar interfaces com o utilizador deverá ser formal e independente da plataforma, de modo a permitir criar mecanismos de geração para diversas plataformas computacionais específicas e deverá seguir a recomendação MDA (*Model Driven Architecture*) da OMG.

1.3. Publicações

No âmbito deste trabalho de investigação foram produzidos os seguintes artigos:

XIS – UML Profile for eXtreme Modeling Interactive Systems

Este artigo apresenta a segunda versão do perfil XIS, descreve o seu objectivo, os princípios que orientam a sua definição e os principais elementos que o compõem.
in 4th International Workshop on Model-based Methodologies for Pervasive and Embedded Software (MOMPES 2007), (Braga, Portugal, Março 2007) [Silva, Saraiva et al. 2007b].

Modeling User Interfaces with the XIS UML Profile

Este artigo discute diversas iniciativas para a produção de interfaces gráficas com o utilizador, baseadas numa abordagem MDA / MDD e apresenta as vistas do perfil XIS que permitem a modelação de interfaces com o utilizador.
in 9th International Conference on Enterprise Information Systems (ICEIS 2007), (Funchal, Portugal, Junho 2007) [Martins e Silva 2007].

1.4. Caso de Estudo

Ao longo da dissertação exemplificam-se diversos aspectos de modelação e geração de interfaces com o utilizador. Para facilitar a leitura e entendimento deste trabalho, optamos por utilizar exemplos baseados num caso de estudo que nos acompanhará ao longo de toda a dissertação e cujo enunciado transcrevemos de seguida, de forma sucinta.

Caso de Estudo: "DocPro - Sistema de Gestão Documental e de Projectos"

O DocPro mantém informação sobre um conjunto relevante de objectos, nomeadamente sobre entidades e sobre projectos / estudos. Na perspectiva da gestão de entidades, refere-se que qualquer entidade pode ter associada várias moradas e contactos (e.g., *url*, email, telefone). Por outro lado, existem dois tipos principais de entidades: pessoas e organizações. Caso seja uma organização, importa determinar qual o tipo de organização envolvida. Caso seja uma pessoa, importa determinar o título profissional, a função e eventualmente a organização à qual a pessoa está associada. Na perspectiva da gestão de projectos / estudos, importa referir que um "projecto geral" (é a designação dada genericamente a um estudo e um projecto) tem uma entidade responsável e um número não determinado de entidades participantes, uma descrição, uma data de início e uma data de fim. Adicionalmente, para os projectos e estudos, podem-se associar um ou mais documentos electrónicos. O DocPro suporta as seguintes funcionalidades: gestão de projectos / estudos; associação de documentos a estudos; gestão de entidades organizacionais, que podem ser organizações ou pessoas; associação de entidades a projectos/estudos.

1.5. Organização da Dissertação

Esta dissertação encontra-se organizada em sete capítulos. O capítulo 1 introduz o contexto, o enquadramento e princípios que norteiam este trabalho de investigação e apresenta o caso de estudo DocPro que nos acompanhará ao longo do mesmo. No capítulo 2 é apresentado o estado da arte das abordagens de desenho e especificação de interfaces com o utilizador e são descritos padrões de desenho de interfaces gráficas. No capítulo 3 introduz-se a visão geral do perfil de modelação XIS. No capítulo 4 são descritos de forma mais detalhada os aspectos de modelação de interfaces gráficas, utilizando o perfil XIS. No capítulo 5 são descritos os aspectos de geração de artefactos para as plataformas Windows Forms.NET e ASP.NET. No capítulo 6 são comparados trabalhos relacionados e outras iniciativas no âmbito da modelação de interfaces com o utilizador, e discutidos de forma comparativa alguns dos seus aspectos. Por fim, no capítulo 7 apresentam-se as conclusões, considerações finais e aspectos em aberto para trabalho futuro. Apresenta-se em Apêndice A os artefactos de modelação XIS para o caso de estudo DocPro e em Apêndice B a descrição técnica do perfil XIS.

2. Estado da Arte

Neste capítulo definimos e clarificamos conceitos básicos que estão na base desta investigação, nomeadamente conceitos da área de HCI (Interacção Pessoa-Máquina). Apresentamos também diversas abordagens para o desenho e produção de interfaces gráficas, nomeadamente: (1) utilização de ferramentas para construção de interfaces; (2) utilização de ferramentas para desenho e reconhecimento de esboços; (3) utilização de linguagens baseadas em XML e (4) utilização de linguagens de modelação, seguido de uma breve análise comparativa. Introduzimos e apresentamos um conjunto de iniciativas com impacto no desenho de interfaces gráficas, no âmbito da abordagem MDA / MDD (*Model Driven Architecture / Model Driven Development*). Estas iniciativas são descritas em detalhe no capítulo 6 - Trabalho Relacionado. Finalmente descrevemos padrões de desenho de interfaces gráficas, discutindo as suas vantagens em termos de usabilidade.

2.1. Introdução

Antes de iniciar a apresentação do estado da arte, vamos clarificar alguns conceitos que estão na base desta investigação, nomeadamente os conceitos de interface, utilizador, interface com o utilizador, interface gráfico com o utilizador, plataforma computacional, *GUI toolkit*, controlo e janela.

2.1.1. Conceitos Gerais

Uma **interface** é uma fronteira comum, ou uma conexão, na qual ou através da qual sistemas independentes podem interagir. Uma **interface** é um ponto de interacção ou comunicação entre um computador e uma outra entidade, e.g. impressora, operador humano ou outro computador.

Neste contexto, um **utilizador** (*user*) é uma pessoa, organização ou outra entidade que utiliza os serviços prestados por determinado sistema. No âmbito desta investigação consideramos que o utilizador é sempre um ser humano que utiliza o sistema.

Uma **interface com o utilizador** (UI - *User Interface*) é tudo o que é desenhado num sistema de hardware e ou de software e através do qual o utilizador consegue interagir. Ou seja, os aspectos de um sistema que podem ser vistos, ouvidos e percebidos pelo utilizador e os comandos e mecanismos que este utiliza para controlar e introduzir informação no sistema, fazem parte da interface com o utilizador e servem de ligação entre este e o sistema.

Uma **interface gráfica com o utilizador** (GUI – *Graphical User Interface*) é uma interface com o utilizador dotado de capacidades gráficas, tirando partido destas para facilitar a sua utilização.

Para além das interfaces gráficas existem outros tipos de interfaces. Existem sistemas com os quais o utilizador interage através da utilização de comandos de texto (CLI – *Command Line Interface*). É frequente a utilização de sons nas interfaces com o utilizador (e.g. emissão de sons em situações de erro). As melhorias recentes nos mecanismos de reconhecimento de voz permitem a criação de interfaces com o utilizador que reconhecem comandos falados. Outra possibilidade comum, e que permite uma interacção mais directa com o sistema, consiste na utilização de interfaces tácteis. Existem também interfaces baseados em realidade virtual, que permitem a interacção com o utilizador através da simulação da realidade tais como simulação através de imagens, sons, vibrações, temperatura ou odores.

As interfaces gráficas com o utilizador podem ser criadas para diferentes plataformas computacionais, como por exemplo *Desktop*, *Mobile* ou *Web-based*. Uma **plataforma computacional** define o hardware e software que permitem que determinado software seja executado.

Este trabalho de investigação está focado no desenho e especificação de interfaces gráficas com o utilizador. Por motivos de simplicidade, a partir de agora faremos apenas referência a "interface com o utilizador" em vez de "interface gráfica com o utilizador", ou ainda, simplesmente "interface gráfica".

2.1.2. Toolkits, Controlos e Janelas

Um ***GUI toolkit*** é um conjunto básico de unidades que permitem construir interfaces gráficas. Normalmente encontram-se implementados através de uma biblioteca de componentes ou incluídos num *framework*. Um ***GUI toolkit*** inclui um conjunto de

elementos da interface com o utilizador, controlos (*widgets*) que podem ser adicionados de forma a construir rapidamente interfaces gráficas. Por exemplo o Windows Forms [Microsoft ---b] é uma API incluída no Microsoft .NET Framework que contém um ***GUI toolkit*** para construir interfaces gráficas em ambiente Windows.

Um **controlo** (*widget*) é um elemento da interface gráfica que exibe informação ou que fornece mecanismos para que o utilizador possa interagir com o sistema. Qualquer elemento da interface gráfica é um controlo, por exemplo: janelas, botões, menus e itens de menus ou listas.

Uma **janela** é uma área de visualização no ecrã, geralmente de forma rectangular que pode ter capacidades de *scroll* e que apresenta o seu conteúdo de forma independente do resto do conteúdo do ecrã. A janela é um controlo particular, sendo o principal elemento da interface gráfica. Esta por sua vez pode conter outros controlos.

2.2. Abordagens de Produção de Interfaces Gráficas

Um tema que interessa às comunidades de investigação ligadas à área de HCI e de Engenharia de Software é a especificação, o desenho e o desenvolvimento de sistemas interactivos, em particular no que diz respeito ao desenho de interfaces gráficas.

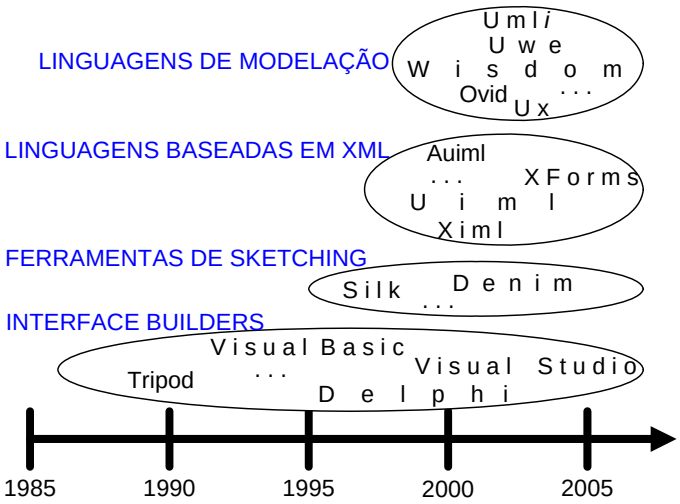


Figura 2.1: Abordagens de produção de interfaces gráficas

A Figura 2.1 sugere quatro das principais abordagens para desenho, especificação e produção de interfaces gráficas:

- Utilização de ferramentas para construção de interfaces gráficas (*UI builders tools approach*);
- Utilização de ferramentas para desenho e reconhecimento de esboços (*UI sketching tools approach*);
- Utilização de linguagens baseadas em XML (*XML-based languages approach*) e
- Utilização de linguagens de modelação (*MDE tools approach*).

2.2.1. Ferramentas para Construção de Interfaces Gráficas (UI Builders)

A primeira abordagem para o desenho e especificação de interfaces gráficas é uma abordagem muito produtiva e popular e consiste na utilização de **ferramentas para construção de interfaces gráficas** (*user interface builders tools*). Estas ferramentas permitem de uma forma visual a criação de formulários, janelas, a colocação de controlos e componentes bem como a definição de comportamento. Esta abordagem é suportada por diversas ferramentas IDE (*Integrated Development Environment*) tais como: o Visual Basic [Microsoft ---c], o Delphi [Borland --], o Visual Studio.NET [Microsoft ---e] ou o Eclipse [The_Eclipse_Foundation --].

Uma razão importante para o sucesso desta abordagem deve-se à utilização de meios gráficos e visuais (caixas de ferramentas, colocação de controlos, localização e dimensão de controlos) para expressar conceitos gráficos (e.g., *interface layout*). Desta forma o foco do desenho e construção de interfaces gráficas passa da codificação convencional, para uma especificação do layout do sistema de forma gráfica e interactiva. Os programadores beneficiaram desta abordagem visto que viram drasticamente reduzido o tempo para a construção de interfaces gráficas. Em contrapartida esta abordagem é considerada de demasiado baixo nível, permitindo apenas produzir interfaces gráficas para uma determinada *framework* (e.g. ASP.NET, JSP, Struts) ou para uma plataforma computacional específica (e.g., *Desktop*, *Mobile*, *Web-based*), não permitindo a produção de interfaces gráficas de forma independente da plataforma.

2.2.2. Ferramentas de Desenho e Reconhecimento de Esboços (*sketching*)

Esta abordagem consiste na criação de protótipos de interfaces gráficas, utilizando **ferramentas de desenho e reconhecimento de esboços** (*sketching*). Existem diversas ferramentas que utilizam esta abordagem, entre as quais se destacam: (1) a ferramenta SILK [Landay e Mayers 2001]; (2) a ferramenta DENIM [Newman, Lin et al. 2003]; (3) a ferramenta SketchiXML [Coyette e Vanderdonckt 2005]; (4) a ferramenta JavaSketchIt [Caetano, Goulart et al. 2002; Fonseca e Jorge 2002] e (5) a ferramenta Freeform [Plimmer e Grundy 2005]. Estas ferramentas procuram reconhecer os esboços (*sketchs*) desenhados através de ferramentas de reconhecimento, de forma a produzir interfaces gráficas específicas.

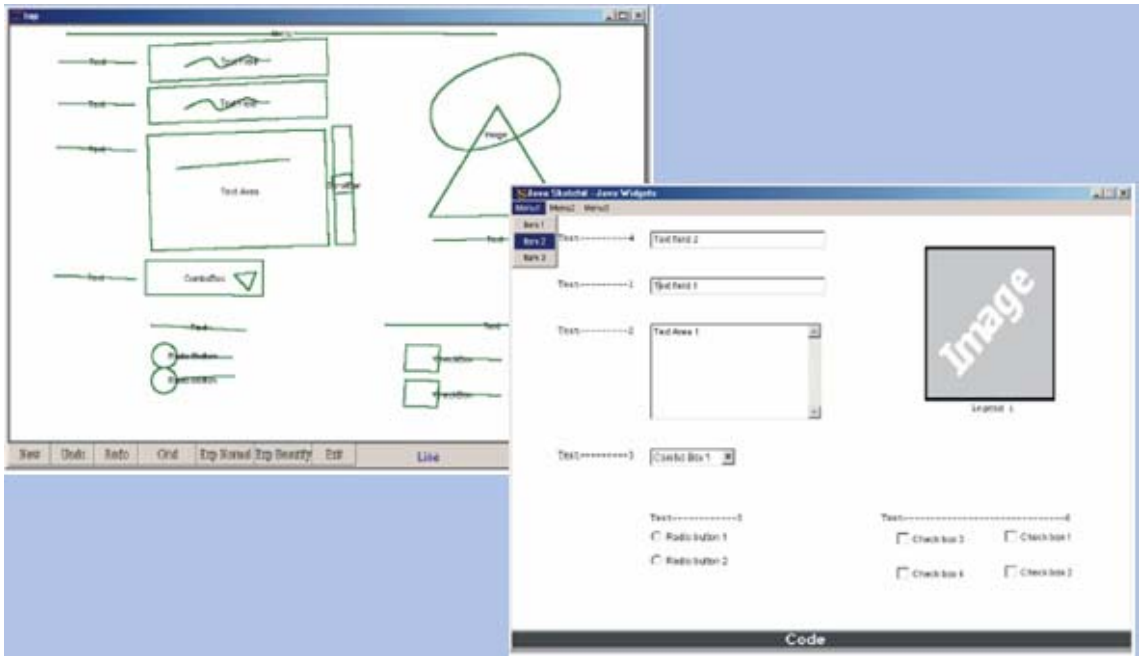


Figura 2.2: Exemplo de um esboço e respectiva interface após mecanismo de *rendering* (extraído de [Coyette 2006])

A Figura 2.2 ilustra à esquerda um exemplo de um esboço representado através de uma ferramenta de reconhecimento de esboços (SketchiXML) e à direita a respectiva interface gerado.

A maioria dos desenhadores de interfaces gráficas considera que o desenho de esboços é a forma mais eficiente para representar a primeira visão de uma futura interface gráfica. Tendo em conta que os esboços podem ser desenhados rapidamente e com facilidade, a sua utilização pode ocorrer em qualquer fase da concepção da interface

gráfica, podendo estes ser desenhados em conjunto com os utilizadores. O recurso ao desenho de esboços permite que o desenhador de interfaces se concentre em aspectos de estrutura, em vez de se preocupar com aspectos menos importantes (e.g. alinhamento, tipo de letra, cores). O desenho de interfaces gráficas através da utilização de esboços reforça a criatividade do desenhador de interfaces, mas a tarefa de reconhecimento dos componentes do esboço não é fácil, por isso o processo de geração pode ser difícil ou pode produzir resultados que não são os mais desejados.

2.2.3. Linguagens Baseadas em XML

Esta abordagem propõe a representação de interfaces através da utilização de **linguagens baseadas em XML** com o objectivo de gerar interfaces gráficas para diversas plataformas computacionais, através de mecanismos denominados de *rendering*. A interface gráfica é especificada de acordo com uma linguagem formal baseada em XML o que traz uma série de vantagens tais como: (1) validação de sintaxe automática; (2) portabilidade; (3) reutilização e (4) *rendering* de interfaces gráficas para diversas plataformas.

Existem diversas iniciativas neste âmbito [Souchon e Vanderdonckt 2003] entre as quais se destacam: (1) a UIML [Abrams, Phanouriou et al. 1999]; (2) a XIML [Puerta e Eisenstein 2002]; (3) a AUIML [Azevedo, Merrick et al. 2000] e (4) a linguagem Xforms [W3C 2007].

Esta abordagem não se foca nos aspectos relacionados com a aparência, permitindo que o desenhador da interface, se possa concentrar mais em aspectos relacionados com a estrutura e interacção dos diferentes elementos que fazem parte da interface gráfica. Também não existe a preocupação relativa à plataforma computacional para a qual se desenha a interface gráfica, visto que esta é representada de forma independente da plataforma. Também são garantidas fácil portabilidade e troca de especificações de interfaces gráficas entre diversos desenhadores, bem com a reutilização das mesmas.

É necessário construir *renderers* específicos para cada plataforma computacional para a qual se pretende gerar a interface gráfica.

A Figura 2.3 ilustra à esquerda um exemplo de uma interface gráfica descrita numa linguagem baseada em XML (UIML) e à direita a respectiva interface gerada.

```
<GROUP NAME="File" CLASS="menu">
  <ELEM NAME="New" CLASS="menuitem"/>
  <ELEM NAME="Quit" CLASS="menuitem"/>
  <ELEM NAME="Close" CLASS="menuitem"/>
</GROUP>
<GROUP NAME="Tools" CLASS="menuortoolbar">
  <ELEM NAME="Line" CLASS="itemoricon"/>
  <ELEM NAME="Box" CLASS="itemoricon"/>
  <ELEM NAME="Oval" CLASS="itemoricon"/>
  <ELEM NAME="Trap" CLASS="itemoricon"/>
  <ELEM NAME="Eraser" CLASS="itemoricon"/>
  <ELEM NAME="Pencil" CLASS="itemoricon"/>
  <ELEM NAME="Brush" CLASS="itemoricon"/>
  <ELEM NAME="Sprayer" CLASS="itemoricon"/>
  <ELEM NAME="Text" CLASS="itemoricon"/>
  <ELEM NAME="Magnifier" CLASS="itemoricon"/
</GROUP>
<GROUP NAME="Help" CLASS="menu">
  <ELEM NAME="PHelp" CLASS="menuitem"/>
  <ELEM NAME="About" CLASS="menuitem"/>
</GROUP>
```

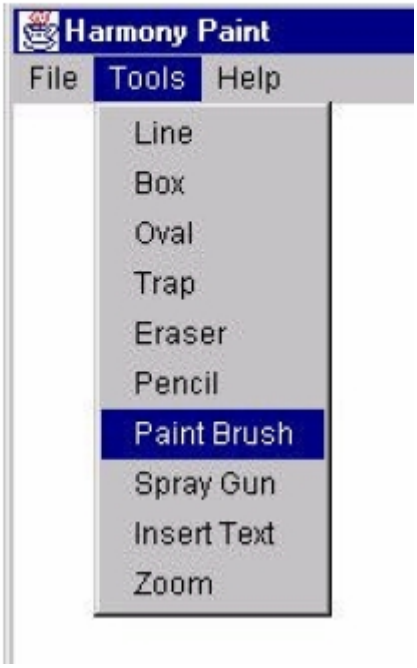


Figura 2.3: Exemplo de um extracto da definição de uma interface em UIML e respectiva interface gerada após mecanismo de rendering (adaptado de [UIML.org 1997])

2.2.4. Linguagens de Modelação

A abordagem que utiliza linguagens de modelação baseia-se no paradigma de desenvolvimento baseado em modelos (*Model Driven Development*, MDD), segundo a recomendação MDA [OMG ---a]. Esta abordagem consiste no desenho de interfaces gráficas, utilizando modelos baseados em UML (*Unified Modeling Language*) [OMG ---b] para depois e através da aplicação de técnicas de geração produzir de forma automática interfaces gráficas e outros artefactos de software.

O MDD é um paradigma de desenvolvimento de software que dá primazia à utilização de modelos. Tem por objectivo descrever as funcionalidades do sistema, utilizando um conjunto de modelos, passando o foco do desenvolvimento do software da codificação para a modelação.

Neste âmbito, o MDA é uma recomendação da OMG que preconiza a transformação de modelos independentes da plataforma (*Platform Independent Model*, PIM) em modelos específicos de plataforma (*Platform Specific Model*, PSM). Os modelos PIM descrevem a

estrutura e funcionalidade de um sistema de uma forma independente da plataforma, encontram-se num nível de abstracção mais elevado, tendem a reflectir o domínio do problema e a não reflectir a tecnologia a utilizar. Em contrapartida, os modelos PSM utilizam conceitos mais tecnológicos, mais próximos da implementação de determinada plataforma, logo encontram-se a um nível de abstracção mais baixo.

Figura 2.4: Transformação de modelos PIM em modelos PSM [Sparx_Systems 2007]

Existe um consenso alargado, no que diz respeito à utilização do UML como linguagem de modelação standard no desenho de software e de sistemas de informação em geral. No entanto e no que diz respeito às tarefas específicas da modelação de interfaces gráficas tal consenso parece ainda não existir.

No capítulo 6 são discutidas com maior detalhe quatro iniciativas para a modelação de interfaces gráficas, nomeadamente:

- **User-Experience (UX) Modeling Language** [Kozaczynski e Thario 2002; Heumann 2003] da *Rational IBM Software Corporation*, [IBM];
- **Unified Modeling Language for Interactive Applications (UML*i*)**, [Silva e Paton 2003; Silva e Paton. 2003] desenvolvida pela *Universidade de Manchester*;
- **UML-based Web Engineering Approach (UWE)**, [Koch e Kraus 2002; Koch 2006; Norrie 2007] proposta do *Institute of Computer Science* da *Universidade de Munich*;
- **Wisdom Profile** [Nunes e Cunha 2000] e os **Protótipos Canónicos Abstractos** [Constantine, Windl et al. 2003; Campos e Nunes 2004] proposta, pela *Universidade da Madeira*;

Finalmente na secção 6.5 apresentamos uma análise comparativa entre estas iniciativas e a nossa proposta para desenho e produção de interfaces gráficas resultante deste trabalho de investigação.

2.2.5. Análise Comparativa

Há vários aspectos a ter em conta quando se analisam diferentes abordagens para desenho e produção de interfaces gráficas. É importante que a abordagem de especificação de interfaces gráficas seja **formal**, de forma que seja possível suportar mecanismos de geração. Por exemplo a especificação de interfaces gráficas, através de ferramentas de desenho e reconhecimento de esboços, é informal o que dificulta o reconhecimento dos esboços no momento da geração da interface gráfica. É também desejável que as interfaces gráficas sejam especificadas de uma forma **independente da plataforma**. Deste modo será possível criar mecanismos de geração de código para diversas plataformas computacionais específicas. É também importante garantir a **rastreabilidade** dos diferentes passos do processo de especificação e geração, bem como garantir a possibilidade de criação de mecanismos de *reverse enginnering* de todo o processo.

A Tabela 2.1 sintetiza a análise comparativa das diferentes abordagens referidas anteriormente, mostrando as suas respectivas vantagens e limitações.

Abordagem Item	Interface Builder	Ferramentas de Sketching	Linguagens baseadas em XML	Linguagens de Modelação UML
Conceitos Básicos	Barras de ferramentas com controlos para UI	Linguagens de Esboços	Linguagens XML	Linguagens de Modelação
Formalismo	+ +	-	+	+
Independência da Plataforma	-	+	+ +	+ +
Rastreabilidade	NA	+	+ +	+ +
Produtividade	+ +	+	-	+
Mecanismos de Geração	Inexistente. Construção imediata	Reconhecimento de Esboços	Rendering	Geração de modelos-para-código

Tabela 2.1: Tabela comparativa das diversas abordagens.

No âmbito desta investigação optamos pela abordagem de desenho e produção de interfaces gráficas segundo o paradigma MDD. De seguida identificamos e apresentamos várias iniciativas que se enquadram nesta abordagem.

2.3. Padrões de Desenho de Interfaces Gráficas

O desenho de interfaces gráficas é uma tarefa muito importante no trabalho de desenvolvimento de software. A usabilidade do sistema é uma preocupação nas tarefas de desenho de interfaces gráficas, entendendo-se a "usabilidade", como a facilidade de utilização um determinado produto. Devido à grande variabilidade humana torna-se difícil criar interfaces gráficas que sejam adequadas para todos os utilizadores.

A usabilidade pode ser medida através de diversos indicadores [Welie, Veer et al. 1999], tais como: (1) facilidade de aprendizagem; (2) facilidade de memorização; (3) desempenho ou velocidade com que os utilizadores realizam tarefas; (4) taxa de erros, isto é, o número de erros que o utilizador cometeu; (5) satisfação do utilizador e (6) completude da tarefa, isto é, quanto da tarefa foi completada.

Os padrões de desenho de interfaces gráficas descrevem soluções de sucesso maioritariamente aceites pela sua usabilidade, isto é, porque são mais fáceis de entender, porque são mais agradáveis à vista ou porque beneficiam a interacção dos utilizadores com o sistema. Um padrão de desenho de interfaces gráficas tem que ter vantagens em pelo menos um destes indicadores para ser considerado um padrão que melhore a usabilidade da interface gráfica. Existem vários padrões de desenho de interfaces gráficas que podem ser utilizados em diversas aplicações. Há padrões de desenho de interfaces gráficas que melhoram aspectos de *design* e de estética mas que não resolvem necessariamente problemas de interacção com os utilizadores e de usabilidade.

Nesta secção descrevem-se alguns padrões de desenho de interfaces gráficas e discutem-se as suas vantagens em termos de usabilidade.

2.3.1. Padrões de Escolha

Selecção Booleana

O padrão de desenho de interfaces gráficas **SELECÇÃO BOOLEANA** (*Single Choice*) [Wesson e Cowley 1999] permite que o utilizador seleccione um determinado item. O utilizador tem apenas duas opções seleccionar ou não seleccionar o item, ou seja trata-se de uma escolha to tipo booleano.

Critérios de Usabilidade Satisfeitos: Este padrão permite reduzir a taxa de erros, visto que limita as hipóteses de escolha do utilizador para aquelas que são realmente válidas.

Exemplo: Este padrão é concretizado tipicamente por controlos do tipo *Check Boxes* como podemos ver ilustrado na Figura 2.5.

Figura 2.5: Exemplo do Padrão Selecção Booleana

Escolha

O padrão de desenho de interfaces gráficas **ESCOLHA** (*Choice*) [Wesson e Cowley 1999; Allgar e Finlay --] permite que o utilizador possa escolher entre vários itens disponíveis.

Quando Usar: Este padrão deve ser usado se o utilizador tiver que escolher um item de um conjunto restrito de itens disponíveis (entre dois a cinco itens), estando todos estes elementos visíveis ao utilizador. Desta forma o utilizador sabe quais são todas as opções que tem disponíveis.

Critérios de Usabilidade Satisfeitos: Este padrão é de fácil aprendizagem e permite reduzir a taxa de erros visto que limita as hipóteses de escolha do utilizador para aquelas que são realmente válidas, contribuindo para uma maior facilidade de aprendizagem.

Exemplo: Este padrão é tipicamente concretizado por um conjunto de *Rádio Buttons* conforme sugerido na Figura 2.6.

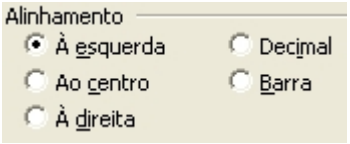


Figura 2.6: Exemplo do Padrão Escolha

Escolha a partir de um conjunto longo

O padrão de desenho de interfaces gráficas **ESCOLHA A PARTIR DE UM CONJUNTO LONGO** (*Choice from a large set*) [Wesson e Cowley 1999; Allgar e Finlay --] permite que o utilizador possa escolher entre vários itens disponíveis. Os itens exibidos devem ser exibidos sob a forma de uma lista e devem ser ordenados da melhor forma, de modo a facilitar a escolha do utilizador (e.g. por ordem alfabética ou numérica ou pelos utilizados mais recentemente).

Quando Usar: Este padrão deve ser usado se o utilizador tiver que escolher um item de uma lista longa de itens disponíveis (mais de dez itens).

Critérios de Usabilidade Satisfeitos: Este padrão é de fácil aprendizagem e permite reduzir a taxa de erros visto que limita as hipóteses de escolha do utilizador para aquelas que são realmente válidas, contribuindo para uma maior facilidade de aprendizagem.

Exemplo: Este padrão pode ser concretizado através da utilização de *Drop Down Lists* ou de *List Boxes*, conforme sugerido na Figura 2.7.

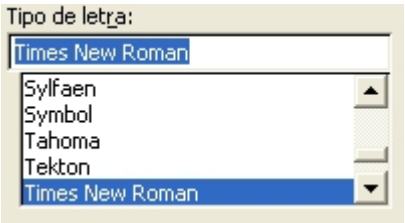


Figura 2.7: Exemplo do Padrão Escolha a partir de um conjunto longo.

2.3.2. Padrão Filtro Contínuo

O padrão de desenho de interfaces gráficas **FILTRO CONTÍNUO** (*Continuous Filter*) [Welie e Troetteberg 2000] consiste na utilização de uma área de inserção de texto através da qual o utilizador pode filtrar em tempo real apenas os itens que lhe interessam. Este padrão permite que o utilizador dinamicamente possa estreitar o leque de escolha, dependendo do *feedback* que é dado imediatamente pelo filtro contínuo. Este padrão pode ser utilizado quando é necessário realizar pesquisas num grande conjunto de informação ordenada, mas no qual o utilizador não tem a certeza de poder vir a encontrar a informação que procura. O conjunto filtrado é exibido

concorrentemente à escrita do item que está a ser pesquisado. O padrão **FILTRO CONTÍNUO** permite aumentar a rapidez da pesquisa bem como a sua eficácia pois permite que o utilizador possa encontrar outros itens relevantes, permite que possa ajustar a sua pesquisa em tempo real e também permite ao utilizador interromper a escrita do item da pesquisa, indo directamente para o item pretendido.

Quando Usar: Este padrão deve ser usado se o utilizador pretende pesquisar uma string num leque de hipóteses mais alargado.

Critérios de Usabilidade Satisfeitos: Este padrão contribui para a melhoria do desempenho e para a melhoria da satisfação do utilizador.

Exemplo: Este padrão pode ser concretizado através de mecanismos de filtragem contínua e através de mecanismos de *auto-complete.* A Figura 2.8 ilustra um exemplo do Padrão FILTRO CONTÍNUO

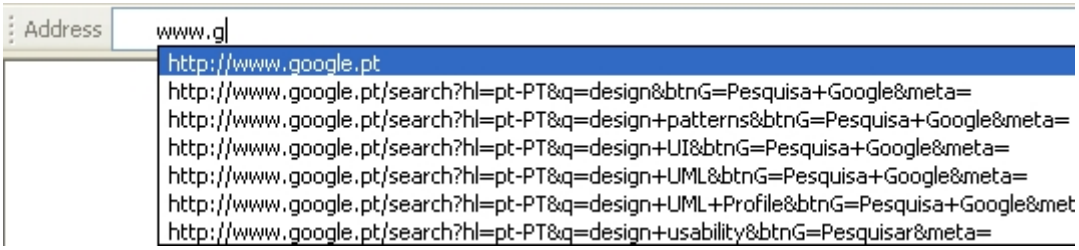


Figura 2.8: Exemplo do Padrão Filtro Contínuo

2.3.3. Padrão Navegação por Menus

O padrão de desenho de interfaces gráficas **NAVEGAÇÃO POR MENUS** [Welie e Troetteberg 2000] consiste em mostrar uma lista das funções que o utilizador tem possibilidade de seleccionar, tornando estas funções acessíveis através de uma acção (e.g. toque de um rato ou tecla de atalho). Se o grupo de funções disponíveis é grande (> ~ sete) as funções devem ser agrupadas separadamente em grupos distintos. As funções devem ser agrupadas ou ordenadas, tendo em conta a semântica, a semelhança a frequência de uso ou então, alfabeticamente. Para agrupar separadamente as funções é possível recorrer a sub menus ou a separadores (ver Figura 2.9). É habitual a colocação de *icons* junto do nome das funções, para auxiliar a memorização e para permitir que o utilizador encontre a função que pretende com maior rapidez. Também é habitual a utilização de teclas de atalho que permitem que o utilizador aceda às

funções mais rapidamente através do teclado, sem necessidade de recorrer ao rato (e.g. Ctrl+P).

Este padrão de desenho de interfaces gráficas permite que o utilizador possa ter uma percepção imediata de todas as possibilidades de funções existentes. Este tipo de padrão em lista é familiar aos humanos (e.g. habituados a listas de menu em restaurantes), logo contribui para que estes reconheçam rapidamente a sua funcionalidade.

Quando Usar: Este padrão pode ser usado quando é importante disponibilizar ao utilizador uma lista com o conjunto de funções disponíveis.

Critérios de Usabilidade Satisfeitos: Este padrão contribui para a facilidade de memorização principalmente em casos em que o número de possibilidades é grande, o que contribui para a melhoria da satisfação do utilizador. A utilização de *icons* e de teclas de atalho aliados a este padrão permitem melhorar o desempenho.

Exemplo: Este padrão é tipicamente concretizado através da utilização de menus, sub-menus, itens de menu separadores de menu, conforme sugerido na Figura 2.9.

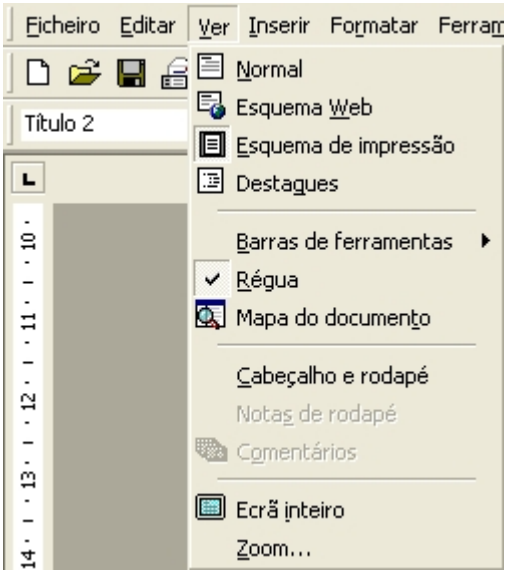


Figura 2.9: Exemplo do Padrão Navegação por Menus

2.3.4. Padrão Apresentação em Grelha

O padrão de desenho de interfaces gráficas **APRESENTAÇÃO EM GRELHA** (*Grid Layout*) [Welie e Troetteberg 2000] consiste na utilização de uma grelha com o número mínimo

de linhas e colunas e com células do maior tamanho possível, separando um espaço de interacção em diferentes áreas de forma a distinguir e ajudar a perceber os diferentes tipos de informação. Os elementos de interacção relacionados conceptualmente podem ser agrupados na mesma célula. Este padrão permite que os utilizadores interajam com vários elementos de interacção de uma forma organizada, clara, agradável e de mais fácil leitura, diminuindo o tempo necessário para visualizar os elementos de cada espaço de interacção. Ao diminuir o número de linhas e colunas diminui o tempo necessário para analisar a informação.

Quando Usar: Este padrão deve ser usado quando é necessário exibir no mesmo espaço, muitos elementos de interacção que podem ser agrupados conceptualmente.

Critérios de Usabilidade Satisfeitos: Este padrão contribui para a facilidade de aprendizagem e para facilidade de memorização. É reduzido o tempo necessário para ler a informação logo aumenta o desempenho. A apresentação é agradável à vista logo aumenta a satisfação do utilizador.

Exemplo: Este padrão é tipicamente concretizado através da utilização de *Group Boxes*, conforme sugerido na Figura 2.10.

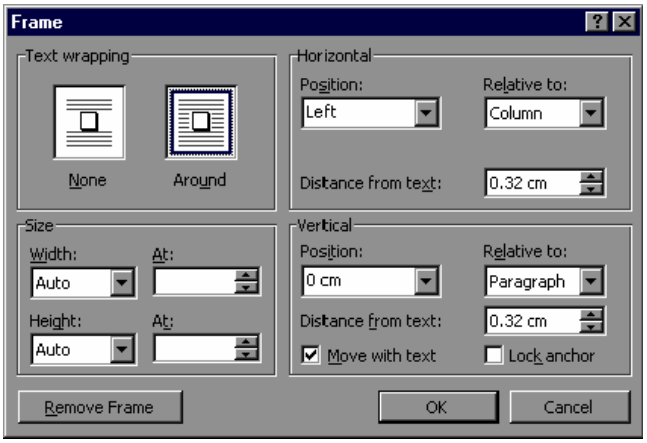


Figura 2.10: Exemplo do Padrão Apresentação em Grelha

2.3.5. Padrão Tabuladores

O padrão de desenho de interfaces gráficas **TABULADORES** (*Card Stack*) [Welie e Troetteberg 2000; Tidwell 2005] consiste em agrupar os diversos elementos de interacção em espaços separados (separado através de tabuladores), permitindo que o

utilizador possa apenas seleccionar e visualizar um espaço de cada vez. Cada espaço deve estar bem identificado. Ao agrupar os elementos de interacção em espaços separados, fica facilitada a tarefa de encontrar os diferentes elementos de interacção e reduz o espaço necessário para mostrar todos os elementos de interacção ao mesmo tempo, no mesmo espaço de interacção. Em resumo resolve o problema de exibir grandes quantidades de informação ou elementos de interacção quando o espaço disponível é limitado, permitindo subdividir os vários elementos de interacção em categorias mais próximas do modelo conceptual conhecido pelo utilizador.

Quando Usar: Este padrão pode ser usado quando é necessário exibir demasiados elementos de interacção para o espaço disponível.

Critérios de Usabilidade Satisfeitos: Este padrão contribui para a facilidade de aprendizagem, facilidade de memorização, desempenho, e satisfação do utilizador.

Exemplo: A Figura 2.11 ilustra um exemplo do Padrão TABULADORES

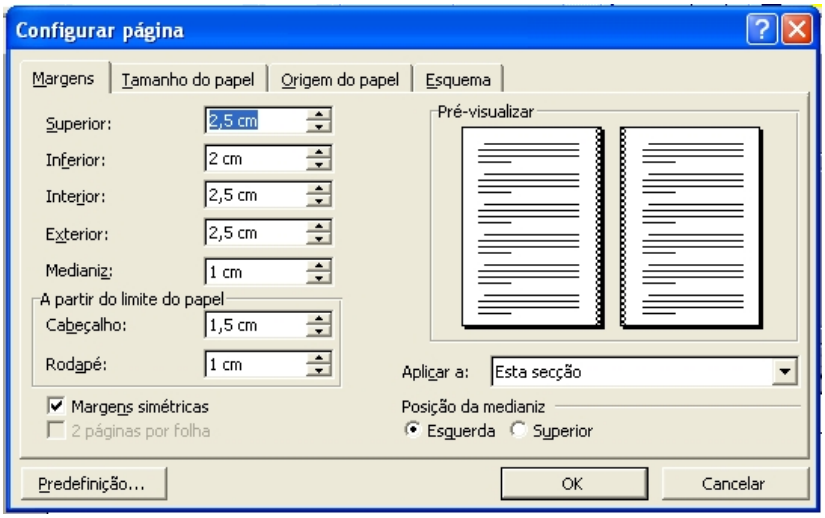


Figura 2.11: Exemplo do Padrão Tabuladores

2.3.6. Padrão Tabela

O padrão de desenho de interfaces gráficas **TABELA** (*Tabular Set*) [Tidwell 1999] consiste na visualização de um conjunto de itens de forma estruturada segundo o formato de uma tabela. As tabelas são utilizadas para mostrar vários itens sob uma vista tabular e também podem servir para facilitar um vasto leque de acções com vista a manipular o conteúdo da mesma. Cada linha da tabela corresponde a um registo ou

tuplo, enquanto que cada coluna da tabela corresponde a um campo ou atributo. Existem dois tipos de tabelas: as tabelas *display-only* que apenas permitem visualizar os dados, podendo o utilizador apenas alterar o formato em que são exibidos (e.g. alterar a ordenação, acrescentar ou retirar colunas, filtrar o conteúdo) e as *editable tables* através das quais o utilizador pode também modificar a informação directamente na tabela (e.g. inserir ou editar registos). A capacidade de alterar o formato em que é exibido uma tabela, quer através de mecanismos de ordenação, quer de selecção de colunas, facilita a exploração da informação e permite ao utilizador configurar a apresentação da interface gráfica de acordo com as suas necessidades.

Quando Usar: Este padrão pode ser usado quando é necessário exibir itens ou registos de forma estruturada, dando ao utilizador a possibilidade de manipula-los.

Critérios de Usabilidade Satisfeitos: Este padrão contribui para a satisfação do utilizador, ao permitir a visualização da informação de forma estruturada e ao permitir que o utilizador possa configurar a apresentação da interface gráfica.

Exemplo: Este padrão é tipicamente concretizado através da utilização de *Data Grids* ou *Data Tables*, conforme sugerido na Figura 2.12.

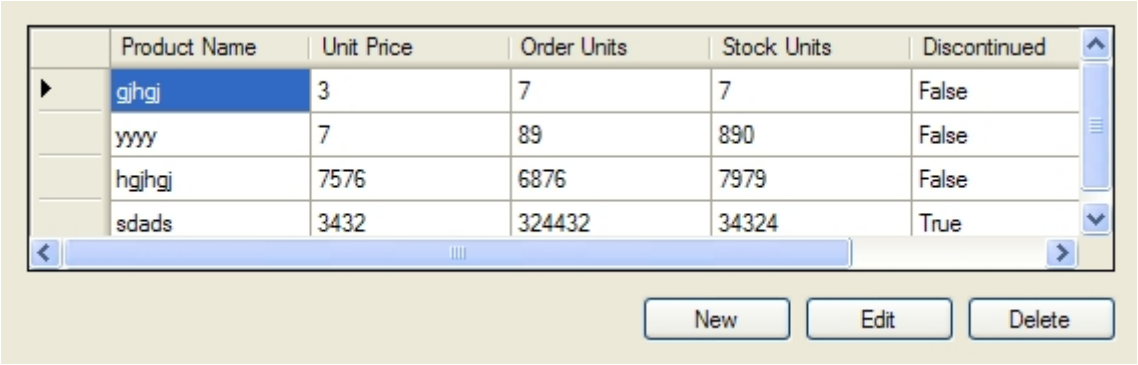


Figura 2.12: Exemplo do Padrão Tabela

2.3.7. Padrão Dupla Lista

O padrão de desenho de interfaces gráficas **DUPLA LISTA** (*Double List*) [Laakso 2003] é utilizado para fazer selecções e consiste na utilização de duas listas: uma que contém todos os itens passíveis de ser seleccionados e outra, que contém os itens seleccionados. Quando o utilizador selecciona os elementos da lista, dependendo do objectivo em

causa, estes podem ser movidos ou copiados de uma lista para outra, indicando que foram associados ou desassociados.

Quando Usar: Este padrão é utilizado quando é necessário associar vários itens a partir de uma lista longa, que não pode ser visualizada toda ao mesmo tempo.

Critérios de Usabilidade Satisfeitos: Este padrão contribui para a facilidade de aprendizagem e facilidade de memorização. O *layout* é simples de utilizar e é agradável à vista logo aumenta a satisfação do utilizador.

Exemplo: Este padrão pode ser concretizado através da utilização de *Data Grids, Data Tables* ou *List Boxes* conforme sugerido na Figura 2.13.

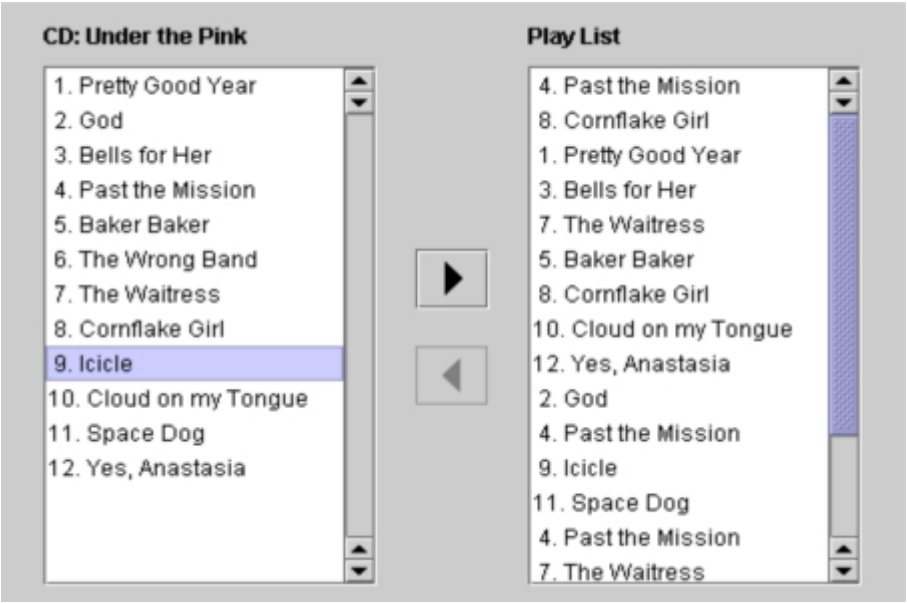


Figura 2.13: Exemplo do Padrão Dupla Lista

2.4. Conclusão

Neste capítulo começamos por definir alguns conceitos básicos utilizados na área de HCI (Interacção Pessoa-Máquina).

Seguidamente foram apresentados resumidamente quatro abordagens para o desenho e produção de interfaces gráficas, nomeadamente: (1) utilização de ferramentas para construção de interfaces; (2) utilização de ferramentas para desenho e reconhecimento de esboços; (3) utilização de linguagens baseadas em XML e (4) utilização de linguagens de modelação. Foi realizada uma análise comparativa destas quatro abordagens através da qual analisamos diversos aspectos tais como: (1) conceitos básicos da abordagem; (2) formalismo da notação; (3) independência da plataforma; (4) mecanismos de geração e (5) rastreabilidade. É importante que a abordagem utilizada tenha uma notação **formal** para permitir que através dela possam ser criados mecanismos para geração de diversos artefactos. A abordagem utilizada deve permitir especificar as interfaces gráficas de forma **independente da plataforma**. É conveniente que a abordagem utilizada permita garantir a **rastreabilidade** dos diferentes passos do processo de especificação e geração de interfaces, bem como garantir a possibilidade de criação de mecanismos de *reverse enginnering*.

No âmbito desta investigação optamos pela abordagem de desenho e produção de interfaces gráficas que utiliza linguagens de modelação. Por conseguinte neste capítulo introduzimos e apresentamos um conjunto de iniciativas com impacto no desenho de interfaces gráficas, no âmbito desta abordagem, mas cujo detalhe e análise comparativa serão descritas no capítulo 6.

Terminamos, apresentando e descrevendo alguns padrões de desenho de interfaces gráficas e discutindo as suas vantagens em termos de usabilidade. No capítulo 4 apresentaremos a nossa sugestão de modelação para alguns destes padrões.

No próximo capítulo apresentamos o perfil XIS, que é a nossa proposta para desenho e produção de sistemas interactivos que inclui vistas para modelação de interfaces gráficas e que se enquadra no âmbito da abordagem de desenho e produção de interfaces gráficas, recorrendo à utilização de linguagens de modelação.

3. Perfil XIS

Neste capítulo apresentamos o perfil XIS (*eXtreme modeling of Interactive Systems*) que permite modelar sistemas interactivos segundo uma abordagem MDD. Apresentamos a organização do perfil XIS dividido em três vistas principais: (1) a *Entities View* (*Domain View* e *BusinessEntities View*); (2) a *Use-Cases View* (*UseCases View* e *Actors View*) e (3) a *User-Interfaces View* (*NavigationSpace View* e *InteractionSpace View*). Apresentamos também os diversos estereótipos que podem ser utilizados para cada uma destas vistas. Para facilitar a explicação, são apresentados exemplos com base no caso de estudo "DocPro – Sistema de Gestão Documental e de Projectos", introduzido na secção 1.4.

3.1. Visão Geral do Perfil XIS

O **XIS** (*eXtreme modeling of Interactive Systems*) [Silva 2003b; Silva, Saraiva et al. 2007b] é um perfil UML [OMG 1999] que permite modelar sistemas interactivos de uma forma independente da plataforma, seguindo a abordagem ProjectIT. O XIS enquadra-se em termos gerais na área de modelação de sistemas de informação e na área de desenvolvimento de software baseado em modelos (MDD, *model driven development*). O perfil XIS permite o desenho de sistemas interactivos através de modelos PIM (*Platform Independent Model*). Através de mecanismos de transformação do tipo modelo-para-código é possível transformar os modelos XIS em código para diversas linguagens e plataformas computacionais específicas (e.g. Windows Forms [Microsoft ---b], ASP.NET [Microsoft ---a], JSP/Java).

Apesar de ser um elemento chave do projecto ProjectIT, o XIS é um perfil UML e consequentemente pode ser utilizado por qualquer ferramenta CASE. O perfil XIS segue quatro princípios fundamentais [Silva, Saraiva et al. 2007b], nomeadamente: (1) modularização; (2) separação de conceitos; (3) aproximação conduzida por casos de utilização e (4) transformação de modelos.

Modularização: Quando se modelam sistemas de maior dimensão, a divisão em módulos é fundamental. No perfil XIS a modularização é conseguida através da utilização de pacotes que é um elemento UML meramente organizacional, que permite

agregar diferentes elementos de um sistema de forma que semântica ou estruturalmente faça sentido [Silva e Videira 2005]. Outro conceito que permite a divisão em módulos é o conceito de entidades de negócio (*business entities*) que são agregações lógicas de várias entidades de domínio e que será explicado em maior detalhe na secção 3.2.2.

Separação de Conceitos: Os sistemas devem ser modelados de forma isolada mas integrada através da utilização de vistas. O perfil XIS adere fortemente a este princípio, fornecendo um conjunto integrado de vistas, designadamente: a *Entities View*, a *Use-Cases View* e a *User-Interfaces View*. A Figura 3.1 mostra a organização em vistas do perfil XIS. Outras preocupações e outras vistas podem vir a ser integradas de futuro (e.g., vistas para especificar requisitos não funcionais tais como: segurança, desempenho, escalabilidade).

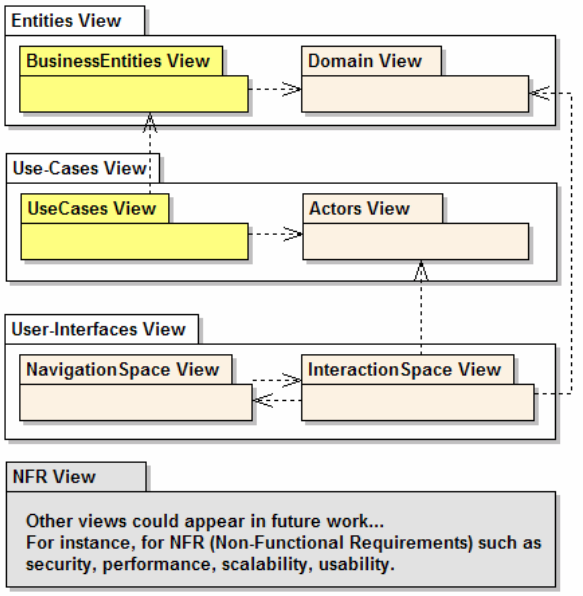


Figura 3.1: A organização em vistas do perfil XIS

Aproximação conduzida por casos de utilização: através do perfil XIS é possível identificar os actores e os casos de utilização (ver secção 3.3) de forma a identificar as funcionalidades do sistema e obter informação sobre os diversos papéis desempenhados no sistema e respectivas permissões relacionadas.

Transformação de modelos: O perfil XIS preconiza uma abordagem de *extreme modeling*, porque promove a utilização de técnicas de transformação de modelo-para-modelo, o que promove altos níveis de produtividade.

Este trabalho de investigação incide fortemente nas tarefas relacionadas com a especificação, o desenho e o desenvolvimento de sistemas interactivos, em particular

no que diz respeito ao desenho de interfaces gráficas e respectivos mecanismos de geração dos diversos artefactos que compõem o sistema. Tendo em vista este objectivo, foi definida uma vista no perfil XIS que se preocupa com os aspectos de modelação de interfaces gráficas - a *User-Interfaces View*.

De seguida descreve-se em maior detalhe os aspectos relacionados com cada uma das vistas que compõem o perfil XIS. Para facilitar a explicação utilizam-se exemplos baseados no caso de estudo "DocPro – Sistema de Gestão Documental e de Projectos" cuja especificação foi descrita na secção 1.4.

3.2. Entities View

Após a especificação dos requisitos do sistema o passo seguinte consiste na identificação e modelação do domínio do problema e das suas entidades. A modelação do domínio é realizada através da vista ***Entities View*** do perfil XIS. Esta vista, por sua vez, encontra-se subdividida em duas vistas: (1) a *Domain View* e (2) a *BusinessEntities View*

Na vista *Domain View*, são representadas as entidades do modelo de domínio e respectivas relações. Na vista *BusinessEntities View* são definidas as *business entities*, entidades a um nível de granularidade mais elevado, que agregam várias entidades do modelo de domínio e ou várias *business entities*.

3.2.1. Domain View

A vista ***Domain View*** consiste na representação do modelo de domínio através de um diagrama de classes. As entidades do domínio são representadas por classes com atributos.

Estereótipo	Elemento estendido	Descrição
«XisEntity»	Classe	Representa uma classe do modelo de domínio.
«XisEntity Attribute»	Atributo	Representa os atributos das classes do modelo de domínio.
«XisEnumeration»	Classe	Representa a definição de um novo tipo enumerado.
«XisEnumeration Value»	Atributo	Representa os elementos que fazem parte de determinado tipo enumerado.

Tabela 3.1: Estereótipos da vista *Domain View*

Os relacionamentos entre as entidades são modelados, utilizando associações simples, agregações e generalizações. Também é possível ao modelador definir novos tipos enumerados. O perfil XIS fornece um conjunto de estereótipos que podem ser aplicados nesta vista e que se encontram descritos na Tabela 3.1.

Em vez de utilizar um diagrama de classes standard, a *Domain View* utiliza estes estereótipos visto que estes fornecem um conjunto de marcas que serão utilizadas nos mecanismos de geração de modelo-para-código.

A Figura 3.2 ilustra um exemplo da vista *Domain View* para o sistema DocPro.

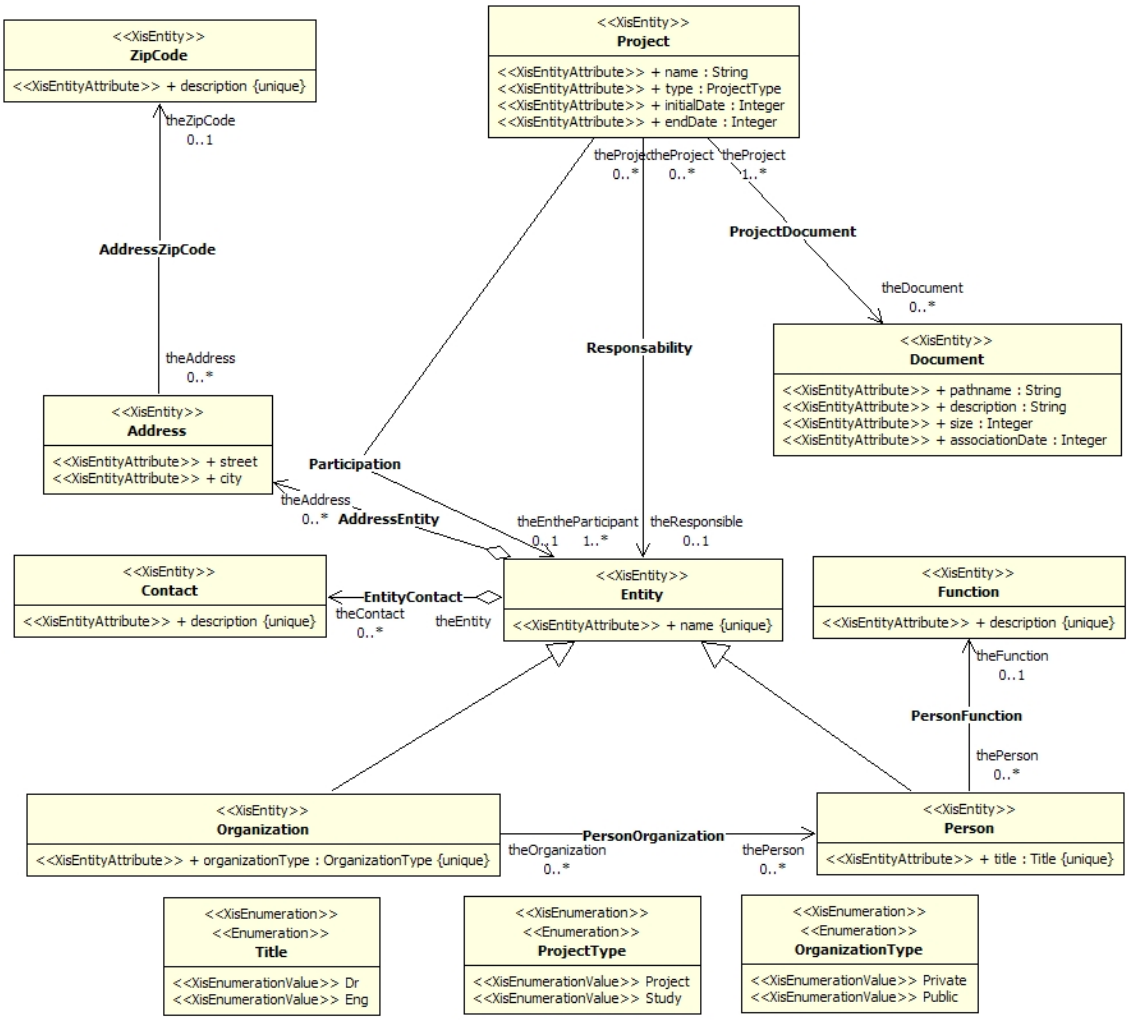


Figura 3.2: Exemplo da vista *Domain View* para o sistema DocPro

3.2.2. BusinessEntities View

O objectivo da vista ***BusinessEntities View*** é definir entidades a um nível de granularidade mais elevado, designadas por *business entities*. As *business entities*

existem para fornecer uma entidade única, que agrupa várias entidades do modelo de domínio e que pode ser manipulada mais facilmente no contexto de um ou vários casos de utilização. Uma *business entity* é especificada, designando uma entidade do modelo de domínio (*Domain View*) como sendo *master entity* e designando uma ou várias entidades do modelo de domínio como sendo as suas *detail entities*. Para que o contexto definido por uma *business entity* faça sentido é necessário que no modelo de domínio as *detail entities* estejam associadas com a *master entity*. Uma *business entity* pode também ser especificada através da agregação de outras *business entities*, nesse caso a *business entity* não estaria relacionada directamente com conceitos do modelo de domínio, mas estaria relacionada directamente com conceitos que foram definidos através de outras *business entities*. O perfil XIS fornece um conjunto de estereótipos que podem ser aplicados nesta vista e que se encontram descritos na Tabela 3.2.

Estereótipo	Elemento estendido	Descrição
«XisBusiness Entity»	Classe	Representa uma *Business Entity*.
«XisBusiness Master»	Associação	É utilizada para identificar a entidade do modelo de domínio (*Domain View*) que é *master* para esta *Business Entity*.
«XisBusiness Detail»	Associação	É utilizada para identificar as entidades do modelo de domínio (*Domain View*) que são *detail* para esta *Business Entity*.
«XisBusiness Composition»	Associação	É utilizada para identificar que uma *Business Entity* é composta por outras *Business Entities*.

Tabela 3.2: Estereótipos da vista *BusinessEntities View*

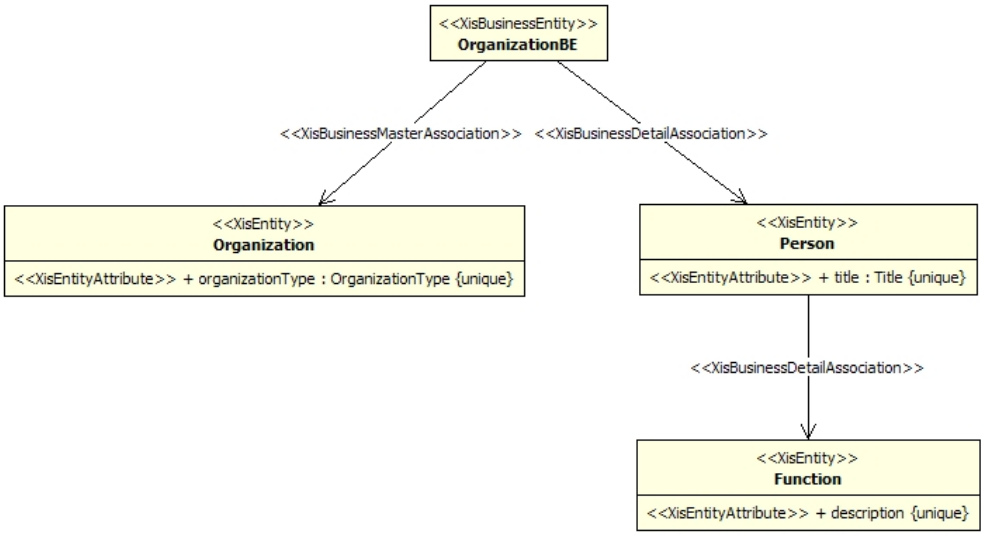


Figura 3.3: Exemplo da definição da *business entity* OrganizationBE.

A Figura 3.3 ilustra um exemplo de uma definição da *business entity* – OrganizationBE. Esta *business entity* tem como *master entity* a entidade Organization e como *detail entities* as entidades Person e Function.

Associado aos estereótipos XisBusinessComposition, XisBusinessMaster e XisBusiness-Detail foi definida a marca "Operations" que consiste numa lista de *strings* com as acções que fazem sentido no contexto de uma determinada *business entity*. Estas acções podem ser acções standard tais como: "new", "edit", "select" ou "delete" (acções standard que são reconhecidas pelo gerador de código da ferramenta ProjectIT-Studio), ou podem ser acções específicas que serão posteriormente definidas pelo programador.

3.3. Use-Cases View

A vista ***Use-Cases View*** é utilizada para definir os casos de utilização de um determinado sistema e para definir os respectivos actores para cada caso de utilização. A vista *Use-Cases View* do perfil XIS, por sua vez, está subdividida em duas vistas: (1) a *Actors View* e (2) a *UseCases View*.

3.3.1. Actors View

A vista ***Actors View*** especifica os actores que podem interagir com o sistema. Os actores do sistema podem estar relacionados através de relações de generalização. O perfil XIS fornece apenas um estereótipo que pode ser aplicado nesta vista e que se encontra descrito na Tabela 3.3.

Estereótipo	Elemento estendido	Descrição
«XisActor»	Actor	Representa um papel que um utilizador pode desempenhar no contexto de determinado sistema.

Tabela 3.3: Estereótipo da vista *Actors View*

É utilizado este estereótipo, visto que fornece um conjunto de marcas que serão utilizadas nos mecanismos de geração de modelo-para-código.

A Figura 3.4 ilustra um exemplo de uma vista *Actors View* para o sistema DocPro no qual podemos verificar através das relações de generalização que os actores UOperator e UManager podem realizar todas as acções permitidas ao actor UUser.

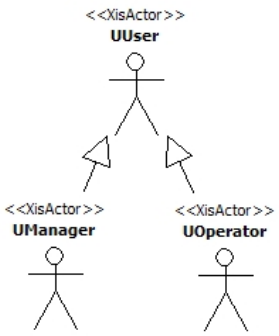


Figura 3.4: Exemplo da vista *Actors View* para o sistema DocPro

3.3.2. UseCases View

A vista ***UseCases View*** descreve as relações existentes entre os actores identificados na vista *Actors View* e as acções que estes estão autorizados a realizar sobre determinada XisBusinessEntity. O perfil XIS fornece um conjunto de estereótipos que podem ser aplicados nesta vista e que se encontram descritos na Tabela 3.4.

Estereótipo	Elemento estendido	Descrição
«XisUseCase»	Caso de Utilização	Representa um caso de utilização como um conjunto de acções que um actor tem permissão para realizar sobre determinada XisBusinessEntity.
«XisOperates OnAssociation	Associação	Representa uma associação entre um XisUseCase e uma XisBusinessEntity, indicando a lista de acções que podem ser desencadeadas sobre esta.

Tabela 3.4: Estereótipos da vista *UseCases View*

A Figura 3.4 ilustra parte de uma vista *Actors View* para o sistema DocPro na qual podemos verificar que o actor UOperator pode gerir entidades. Esta gestão é realizada sobre business entity OrganizationBE.

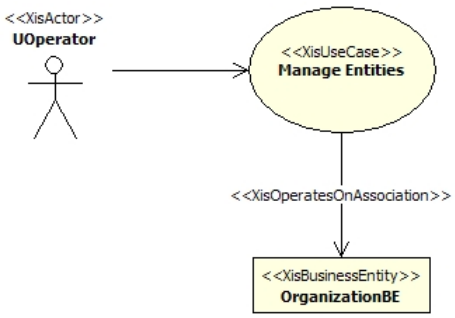


Figura 3.5: Exemplo de parte da vista *UseCases View* para o sistema DocPro

3.4. User-Interfaces View

Esta secção aborda em maior detalhe as vistas do perfil XIS que permitem modelar as interfaces gráficas. O objectivo da vista ***User-Interfaces View*** é especificar as diferentes possibilidades de navegação existentes entre os vários espaços de interacção que compõem o sistema, bem como especificar o conteúdo, estrutura e organização geral de cada um desses espaços de interacção.

Existem dois modelos importantes para especificar e representar interfaces gráficas. O primeiro é o modelo de navegação e é utilizado para especificar a navegação entre os diferentes espaços de interacção que compõem o sistema. O modelo de navegação é útil para suportar a documentação da estrutura do sistema, facilitando as tarefas futuras de alteração e de melhoria da navegabilidade. O segundo é modelo do espaço de interacção e é utilizado para representar o conteúdo e organização geral de cada espaço de interacção.

A vista *User-Interfaces View* do perfil XIS, por sua vez, está subdividida em duas vistas nomeadamente: (1) a *NavigationSpace View* e (2) a *InteractionSpace View*.

3.4.1. NavigationSpace View

A vista ***NavigationSpace View*** define o fluxo de navegação que pode existir entre os diferentes espaços de interacção que compõem o sistema. Esta vista define um mapa de navegação na forma de grafo, cujos nós são referências para espaços de interacção e cujos *links* representam as transições entre estes. As transições são normalmente desencadeadas a partir da intervenção dos utilizadores. Esta vista é um bom suporte à documentação da estrutura do sistema e facilita as eventuais alterações e melhorias da navegabilidade do sistema. O perfil XIS fornece dois estereótipos que podem ser aplicados nesta vista e que se encontram descrito na Tabela 3.5.

Estereótipo	Elemento estendido	Descrição
«XisInteractionSpace»	Classe	Representa um espaço de interacção.
«XisNavigationAssocition»	Associação	Representa um fluxo de navegação entre dois espaços de interacção.

Tabela 3.5: Estereótipos da vista *NavigationSpace View*

A Figura 3.6 ilustra um exemplo da vista *NavigationSpace View*, representando parte da navegação do sistema DocPro. O estereótipo XisNavigationAssociation é aplicado a todas as associações representadas na figura. De forma a manter o modelo de navegação mais simples e de mais fácil leitura, os elementos que fazem parte de cada espaço de interacção não são especificados nesta vista. A especificação destes elementos ocorre na *InteractionSpace View*.

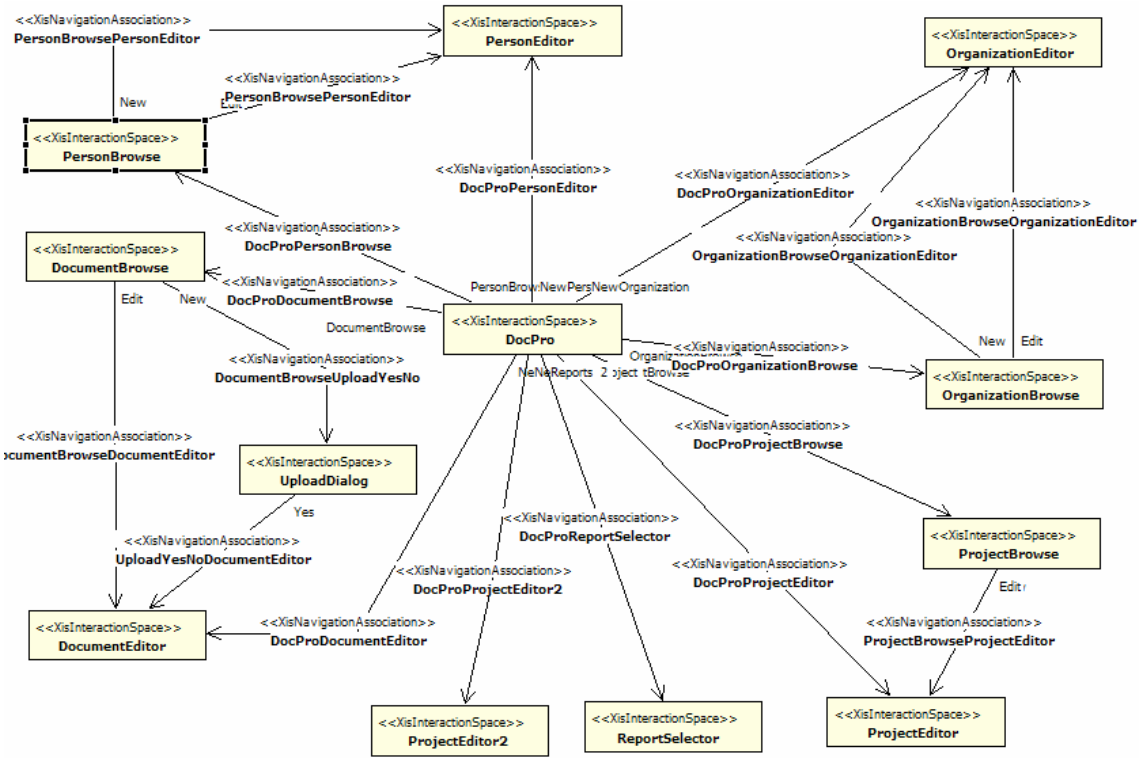


Figura 3.6: Exemplo da vista *NavigationSpace View* para o sistema DocPro

3.4.2. InteractionSpace View

A vista ***InteractionSpace View*** tem como principal objectivo descrever o conteúdo de cada espaço de interacção, ou seja, especificar os diferentes elementos que compõem cada espaço de interacção.

A *InteractionSpace View* utiliza algumas técnicas de *sketching*, baseadas no *layout* gráfico dos diagramas UML, de forma a influenciar o tamanho e a posição relativa dos diferentes elementos que compõem cada espaço de interacção. As abordagens que utilizam a informação gráfica do diagrama não são normalmente recomendadas, porque tradicionalmente cada ferramenta de modelação grava e visualiza os diagramas

de uma forma específica e particular. Contudo acreditamos que isto pode ser ultrapassado devido à especificação OMG *Diagram Interchange* [OMG 2006]. O *Diagram Interchange* define e fornece um standard para representar a informação gráfica de um diagrama UML de forma independente da ferramenta.

Estereótipo	Elemento estendido	Descrição
«XisInteraction Space»	Classe	Representa um espaço de interacção que contém os diversos elementos de interacção com o utilizador. Corresponde de forma abstracta à noção de janela ou *form* HTML.
«XisInteraction Element»	Classe	Classe abstracta que tem duas especializações que representam elementos de interacção simples e compostos.
«XisInteraction CompositeElement»	Classe	Representa um elemento de interacção composto que contém outros XisInteractionElements.
«XisDomain Association»	Associação	Representa uma associação entre um XisInteractionCompositeElement e uma XisEntity da *Domain View*, indicando a entidade principal dos elementos contidos no XisInteractionCompositeElement.
«XisInteraction SimpleElement»	Classe	Classe abstracta que tem três especializações que descrevem elementos de interacção simples que representam dados e elementos que podem desencadear acções.
«XisDataElement»	Classe	Classe abstracta que tem duas especializações que descrevem elementos contidos no espaço de interacção que representam dados.
«XisDomain Element»	Classe	Representa um elemento de interacção que está associado a uma entidade da *Domain View*, ou seja um elemento que faz parte do domínio.
«XisDomain Attribute Association»	Associação	Representa uma associação entre um XisDomainElement e uma XisEntity da *Domain View*, indicando que o elemento de interacção está relacionado com um atributo de uma entidade da *Domain View*.
«XisOtherElement»	Classe	Representa um elemento de interacção que não está associado a uma entidade da *Domain View*, ou seja um elemento que não faz parte do domínio (e.g. uma *label* com um título ou uma imagem).
«XisDataTable»	Classe	Representa um elemento de interacção que mostra uma tabela com o resultado de *query* SQL.
«XisActionElement»	Classe	Representa um elemento de interacção através do qual o utilizador invoca uma acção (e.g. um botão ou um *link*).
«XisPerforms Navigation Association»	Associação	Representa uma associação entre um XisActionElement e um XisInteractionSpace, indicando uma possibilidade de desencadear uma navegação.
«XisElement Permission»	Classe	Especifica o acesso dos actores do sistema a cada um dos XisInteractionElements do espaço de interacção.

Tabela 3.6: Estereótipos da vista *InteractionSpace View*

O perfil XIS fornece um conjunto de estereótipos que podem ser aplicados nesta vista e que se encontram descritos na Tabela 3.6. A Figura 3.7 ilustra um exemplo da vista *InteractionSpace View* para o espaço de interacção OrganizationEditor do caso de estudo DocPro. Este espaço contém vários elementos de interacção. A explicação mais detalhada deste e de outros espaços de interacção encontra-se descrita no Apêndice A.

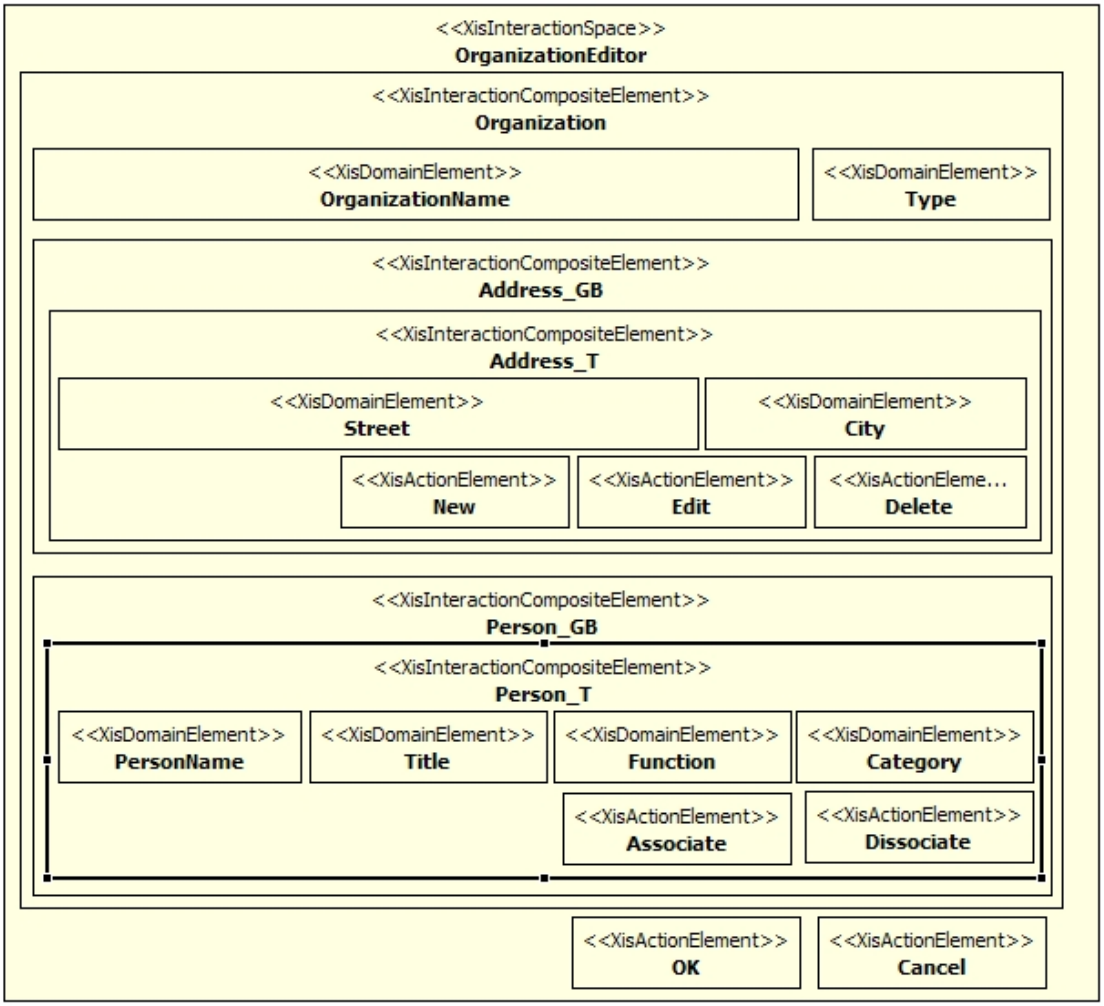


Figura 3.7: Exemplo da vista *InteractionSpace View* para o espaço de interacção OrganizationEditor

4. Desenho de Interfaces Gráficas

Neste capítulo, apresentamos a modelação de interfaces gráficas, utilizando o perfil XIS. Para além dos metamodelos, são apresentados diversos aspectos relacionados com as tarefas de desenho de interfaces gráficas, referentes às duas vistas que compõem a *User-interfaces View*: (1) a *NavigationSpace View* e (2) a *InteractionSpace View*. Apresentaremos e descreveremos a nossa sugestão para modelar alguns dos padrões de desenho de interfaces gráficas discutidos na secção 2.3.

Para testarmos a nossa abordagem de desenho e especificação de interfaces gráficas, utilizando o perfil XIS, todos os exemplos utilizados terão como base o caso de estudo DocPro, cujo enunciado se encontra descrito na secção 1.4. No Apêndice A encontra-se a representação da interface gráfica do caso de estudo DocPro, utilizando o perfil XIS, nomeadamente a representação das vistas que fazem parte da *Interaction-Space View*.

4.1. Metamodelo

Nesta secção são apresentados os metamodelos das vistas que compõem a *User-Interfaces View*. A Figura 4.1 ilustra o metamodelo da vista NavigationSpace View. Este diagrama é composto por XisInteractionSpaces e por associações do tipo XisNavigationAssociation que indicam a navegação entre dois espaços de interacção.

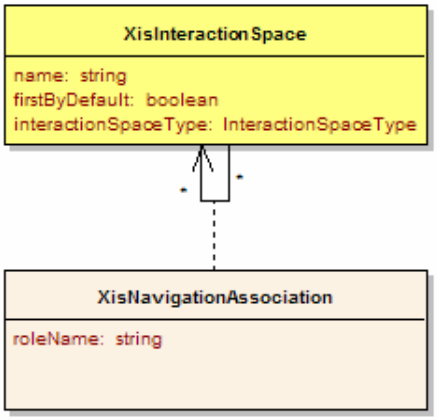


Figura 4.1: Metamodelo da *NavigationSpace View*.

A Figura 4.2 ilustra o metamodelo da vista InteractionSpace View. Cada diagrama desta vista representa um XisInteractionSpace.

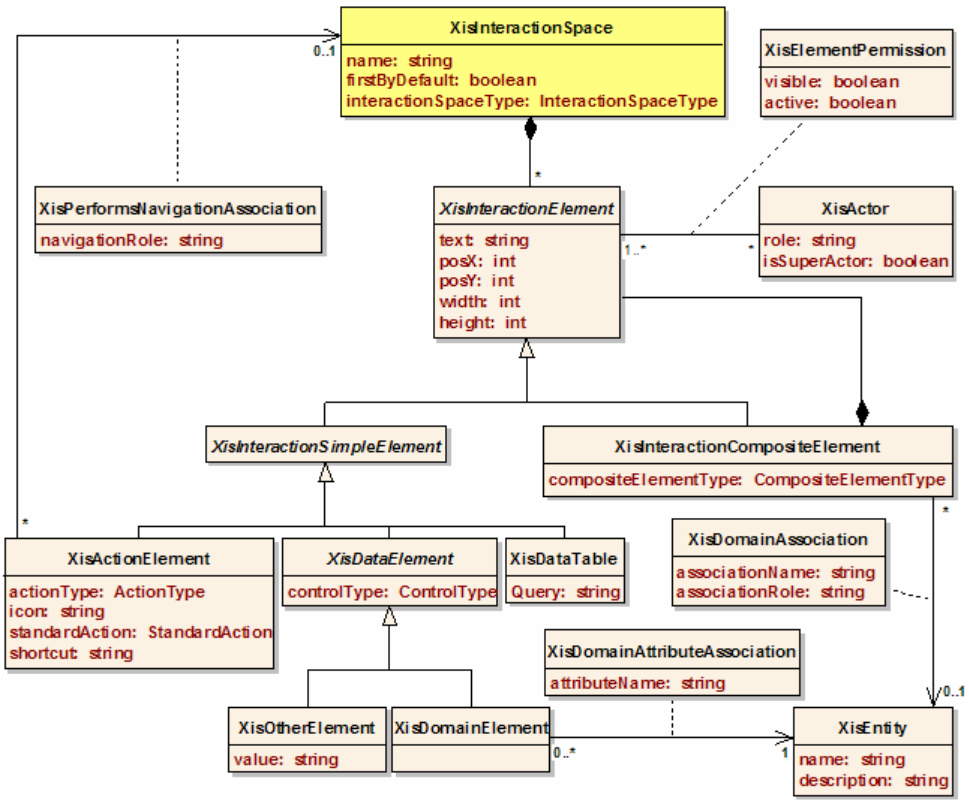


Figura 4.2: Metamodelo da *InteractionSpace View*.

Cada XisInteractionSpace contém XisInteractionElements que podem ser elementos de interacção simples (XisInteractionSimpleElements) ou compostos (XisInteraction-CompositeElements). Os XisInteractionCompositeElements, por sua vez, podem conter outros XisInteractionElements (simples ou compostos). Para definir o contexto de um XisInteractionCompositeElement, este pode estar ligado a uma XisEntity da *Domain View*. Esta ligação está representada através da associação XisDomainAssociation. Por sua vez, os XisInteractionSimpleElement podem ser XisActionElements, XisData-Elements ou XisDataTables. Um XisActionElement é uma acção através da qual pode ser desencadeada uma navegação para um XisInteractionSpace. Esta navegação está representada através da associação XisPerformsNavigationAssociation. Um XisData-Table é um elemento de interacção, que representa uma tabela que exibe o resultado de um *query* SQL. Um XisDataElement é um elemento de interacção, que representa dados e pode ser um XisDomainElement ou um XisOtherElement. Um XisOtherElement é um elemento de interacção, que não está ligado a nenhuma entidade da *Domain View*. Contrariamente, um XisDomainElement está ligado a uma XisEntity da *Domain View*. Esta relação está representada através da associação XisDomainAttributteAssociation. Através da associação XisElementPermission, podemos representar se um determinado XisActor pode ver e ou aceder a determinado XisInteractionElement.

4.2. Modelação da Navegação

A modelação da navegação geral de um sistema interactivo é representada através da *NavigationSpace View*. Na *NavigationSpace View* devemos representar todos os XisInteractionSpaces que pretendemos modelar na nossa aplicação. Através das XisNavigationAssociations podemos indicar todas as hipóteses de navegação de um espaço de interacção para outro. O nome do papel da XisNavigationAssociation do lado da origem tem que ser o mesmo, do nome do elemento de interacção que desencadeia a navegação.

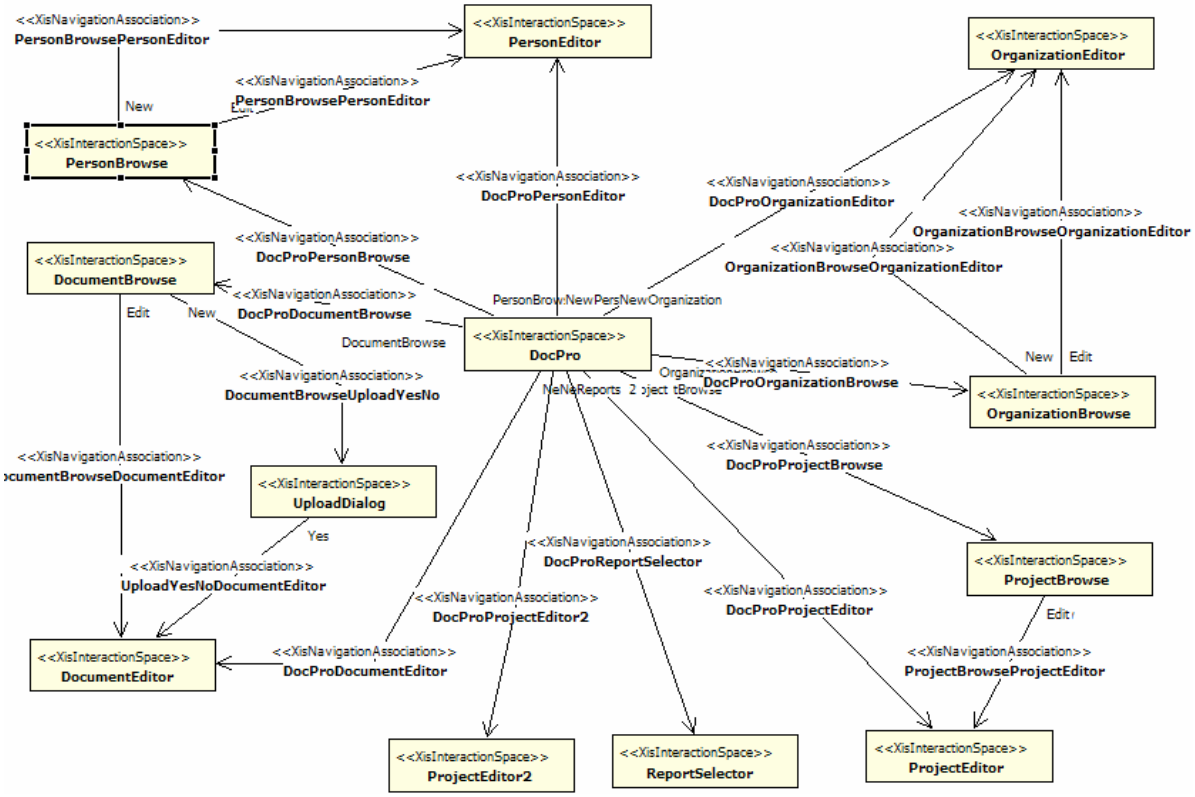


Figura 4.3: Exemplo da vista *NavigationSpace View*, parcial, para o sistema DocPro

A Figura 4.3 ilustra parte da vista *NavigationSpace View* para o caso de estudo DocPro. Podemos ver como exemplo, no canto inferior direito da figura, que a partir do espaço de interacção ProjectBrowse, temos a XisNavigationAssociation ProjectBrowseProject Editor cujo nome do papel do lado da origem é Edit, indicando que tal navegação é desencadeada por um elemento de interacção com o mesmo nome e que permitirá navegar para o espaço de interacção ProjectEditor, onde será possível editar os dados do Projecto.

4.3. Modelação do Espaço de Interacção

A modelação dos espaços de interacção é representada através da *InteractionSpace View*. De seguida iremos apresentar alguns aspectos relativos à modelação desta vista.

4.3.1. XisInteractionSpace

Um XisInteractionSpace representa um espaço de interacção que pode conter diversos elementos de interacção com o utilizador. A Figura 4.4 ilustra a parte do metamodelo da *InteractionSpace View* que define o elemento XisInteractionSpace.

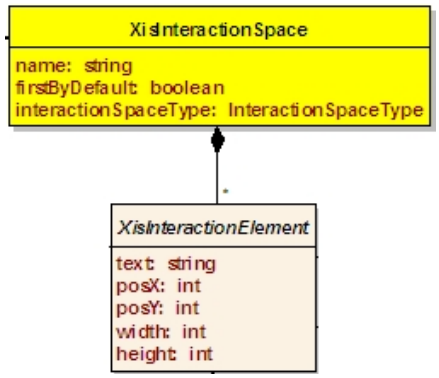


Figura 4.4: Metamodelo parcial (XisInteractionSpace) da *InteractionSpace View*

Nome	Tipo	Valor por omissão	Descrição
Name	String	- -	Indica o nome do espaço de interacção
firstBy Default	Boolean	False	Indica se é o primeiro espaço de interacção da aplicação.
interaction SpaceType	Interaction SpaceType	Form	Indica o tipo de espaço de interacção que se pretende representar, podendo apresentar um dos seguintes valores: Form ou Dialog

Tabela 4.1: Marcas para o estereótipo XisInteractionSpace

Associado ao estereótipo XisInteractionSpace foi definido um conjunto de marcas que se encontram descritas na Tabela 4.1.

Ao modelar um XisInteractioSpace cuja marca *firstByDefault* seja ***true*** estamos a especificar o primeiro espaço de interacção da aplicação.

Ao modelar um XisInteractionSpace deverá ser indicado na marca *interactionSpaceType* o tipo de janela ou formulário genérico que se pretende representar. O tipo

InteractionSpaceType é um tipo enumerado e o seu conjunto de valores pode ser alargado para ajudar a modelar outros tipos de espaços de interacção.

Ao modelar um XisInteractionSpace cuja marca *interactionSpaceType* seja **Form**, estamos a representar um espaço de interacção do tipo janela, formulário ou página *web*.

Ao modelar um XisInteractionSpace cuja marca *interactionSpaceType* seja **Dialog**, estamos a representar um espaço de interacção do tipo caixa de diálogo.

Restrições: Um XisInteractionSpace cuja marca *interactionSpaceType* seja **Dialog**:

- Apenas pode conter elementos compostos (XisInteractionCompositeElement) cuja marca *compositeElementType* seja DialogExclamation, DialogInformation, DialogWarning, DialogQuestion ou DialogError.
- Não podem estar contidos num XisInteractionSpace cuja marca *interactionSpaceType* seja **Form**.

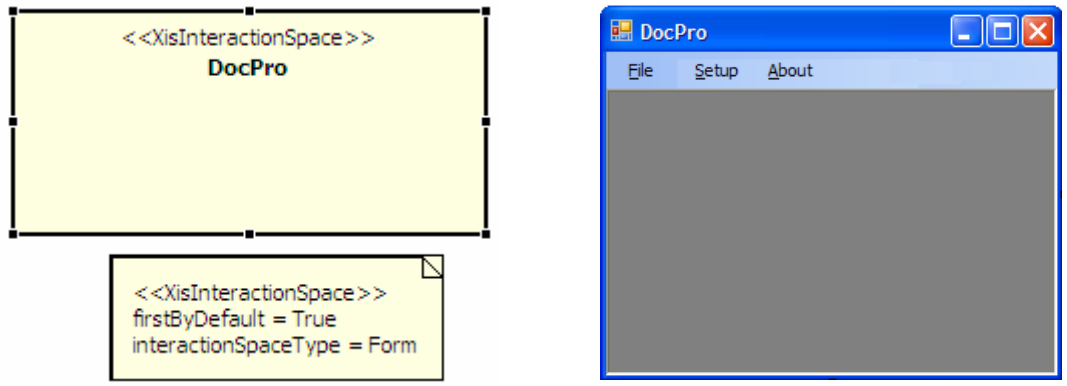


Figura 4.5: Representação de um XisInteractionSpace cuja marca *firstByDefault* é true, cuja marca *interactionSpaceType* é Form e respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.5, do lado esquerdo, representa um espaço de interacção (XisInteractionSpace) do tipo janela de formulário (*interactionSpaceType* = **Form**), com o nome DocPro e que é o primeiro espaço de interacção da aplicação (*firstByDefault* = **true**) e do lado direito a respectiva geração para a plataforma Windows Forms.NET.

4.3.2. XisDataElement

Um XisDataElement é um elemento de interacção simples (XisInteractionSimpleElement) que descreve elementos de interacção que representam dados. Um XisDataElement pode (XisDomainElement) ou não (XisOtherElement) estar associado a entidades da *Domain View*.

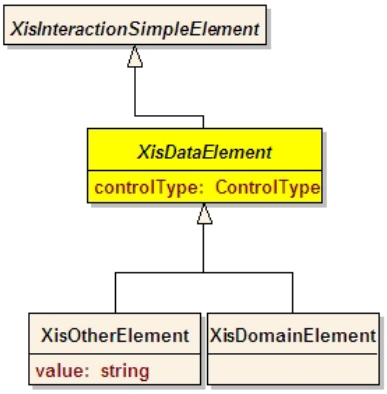


Figura 4.6: Metamodelo parcial (XisDataElement) da *InteractionSpace View*

A Figura 4.6 ilustra a parte do metamodelo da *InteractionSpace View* que define o elemento XisDataElement. Para completar a descrição do estereótipo XisDataElement foi definida uma marca como podemos ver na Tabela 4.2.

Nome	Tipo	Descrição
controlType	ControlType	Indica o tipo de elemento de interacção (um controlo genérico) que se pretende representar, podendo apresentar um dos seguintes valores: Label; Image; TextBox; CheckBox; RadioButton; ComboBox ou Date.

Tabela 4.2: Marca para o estereótipo XisDataElement

Ao modelar um XisDataElement deverá ser indicado na marca *controlType* o tipo de controlo genérico que se pretende representar. O tipo ControlType é um tipo enumerado e o seu conjunto de valores pode ser alargado para ajudar a modelar outros padrões de elementos de interacção simples.

Ao modelar um XisDataElement cuja marca *controlType* seja **Label**, estamos a representar um elemento de *output* (permite apenas exibir dados) do tipo etiqueta para exibir texto.

Ao modelar um XisDataElement cuja marca *controlType* é **Image**, estamos a representar um elemento de *output* (permite apenas exibir dados) – para exibir uma imagem.

Ao modelar um XisDataElement cuja marca *controlType* seja **TextBox** estamos a representar dois elementos: uma etiqueta e uma caixa de texto. Ou seja não é necessário colocar no modelo XIS os dois elementos (um XisDataElement cujo *controlType* seja Label e um XisDataElement cujo *controlType* seja TextBox**)** basta colocar um XisDataElement cujo *controlType* seja TextBox. Este comportamento também é comum para os seguintes *controlTypes*.

Ao modelar um XisDataElement cuja marca *controlType* seja **CheckBox**, estamos a representar um elemento de interacção simples do tipo caixa de marcação (ver o Padrão Selecção Booleana na secção 2.3.1 - Padrões de Escolha).

Ao modelar um XisDataElement cuja marca *controlType* seja **RadioButton**, estamos a representar um elemento de interacção simples do tipo botão de rádio (ver o Padrão Escolha na secção 2.3.1 - Padrões de Escolha).

Ao modelar um XisDataElement cuja marca *controlType* seja **ComboBox**, estamos a representar um elemento de interacção simples do tipo caixa combinada (ver o Padrão Escolha A Partir De Um Conjunto Longo na secção 2.3.1 - Padrões de Escolha).

Ao modelar um XisDataElement cuja marca *controlType* seja **Date**, estamos a representar um elemento de interacção que permite visualizar e ou seleccionar uma data.

Restrições: Um XisDataElement não pode estar contido num:

- XisInteractionCompositeElement cuja marca *compositeElementType* seja Menu.

Figura 4.7: Representação de um XisDomainElement cuja marca *controlType* é Date e respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.7, do lado esquerdo, representa um elemento de interacção simples do tipo Date (*controlType* = **Date**) com o nome dataFim e do lado direito, a respectiva geração para a plataforma Windows Forms.NET.

XisDomainElement

Um XisOtherElement é uma especialização de um XisDataElement e representa um elemento de interacção simples (XisInteractionSimpleElement) que está associado a uma entidade da *Domain View* (XisEntity), ou seja um elemento que faz parte do domínio. Através da associação XisDomainAttributeAssociation é indicado que

determinado XisDomainElement está associado a determinada XisEntity. A marca *attributeName* permite indicar a que atributo da XisEntity está associado.

A Figura 4.8 ilustra a parte do metamodelo da *InteractionSpace View* que contém o elemento XisDomainElement.

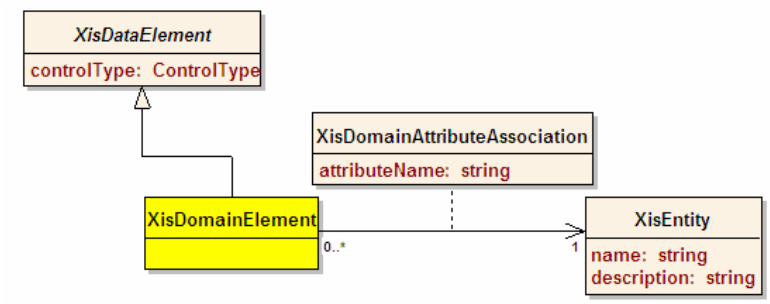


Figura 4.8: Metamodelo parcial (XisDomainElement) da *InteractionSpace View*

XisOtherElement

Um XisOtherElement é uma especialização de um XisDataElement e representa um elemento de interacção simples (XisInteractionSimpleElement) que não está associado a uma entidade da *Domain View*, ou seja um elemento que não faz parte do domínio (e.g., uma etiqueta com um título ou uma imagem).

A Figura 4.9 ilustra a parte do metamodelo da *InteractionSpace View* que define o elemento XisOtherElement.

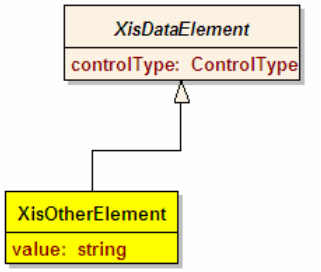


Figura 4.9: Metamodelo parcial (XisOtherElement) da *InteractionSpace View*

Como os XisOtherElements não estão associados à nenhuma fonte de dados é necessário definir uma marca que contenha o seu conteúdo. (ver na Tabela 4.3).

Nome	Tipo	Descrição
value	String	Indica uma *string* com o conteúdo do elemento de interacção.

Tabela 4.3: Marca para o estereótipo XisOtherElement

4.3.3. XisActionElement

Um XisActionElement é um elemento de interacção simples (XisInteractionSimple Element) que representa um elemento de interacção através do qual o utilizador pode invocar uma acção (e.g., um botão ou um link). Um XisActionElement permite que o utilizador invoque uma acção interna, sem navegar para outro espaço de interacção, ou uma acção, que desencadeie uma navegação para outro espaço de interacção (XisInteractionSpace).

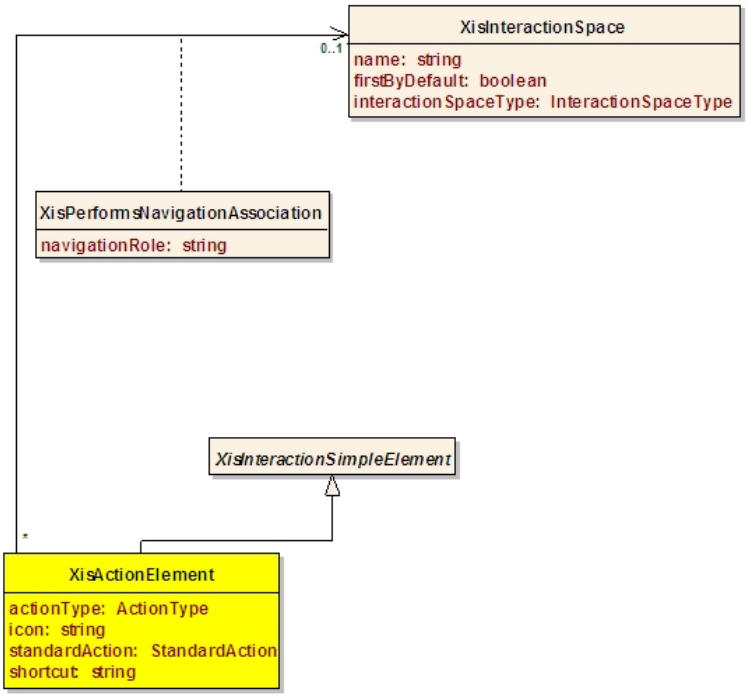


Figura 4.10: Metamodelo parcial (XisActionElement) da vista *InteractionSpace View*

A Figura 4.10 ilustra a parte do metamodelo da *InteractionSpace View* que define o elemento XisActionElement.

Nome	Tipo	Descrição
actionType	Action Type	Indica o tipo de elemento de interacção (um controlo genérico) que se pretende representar, podendo apresentar um dos seguintes valores: Button; Link; MenuItem ou MenuSeparator.
icon	String	Indica uma *string* com o caminho para o *icon* associado ao XisActionElement.
shortcut	String	Indica uma sequência de caracteres que serve de atalho para invocar a acção (e.g. Ctrl+P).
standard Action	Standard Action	Indica o tipo de acção standard que vai ser invocada, podendo apresentar um dos seguintes valores: New; Edit; Select; Delete; Navigate; Close; OK; Cancel; Yes; No; Associate; Dissociate ou Custom.

Tabela 4.4: Marcas para o estereótipo XisActionElement

Associado ao estereótipo XisActionElement foram definidas várias marcas como podemos ver na Tabela 4.4. Ao modelar um XisActionElement deverá ser indicado na marca *actionType* qual o tipo de controlo genérico que se pretende representar. O tipo ActionType é um tipo enumerado e o seu conjunto de valores pode ser alargado para ajudar a modelar outros elementos de interacção que permitam invocar acções.

ActionType Button e ActionType Link

Ao modelar um XisActionElement cuja marca *actionType* seja **Button** ou **Link**, estamos a representar um elemento de interacção simples do tipo botão ou *link* que permitirá desencadear acções.

Restrições: Um XisActionElement cuja marca *actionType* seja **Button** ou **Link**:

- Não pode ser representado dentro de um elemento composto do tipo Menu

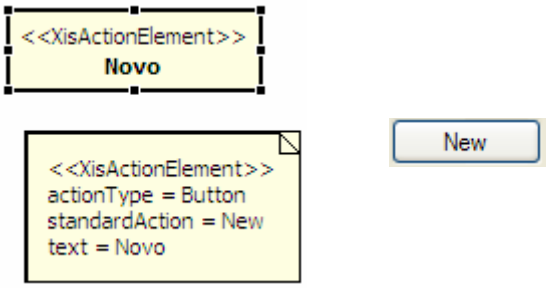


Figura 4.11: Representação de um XisActionElement cuja marca *actionType* é Button e respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.11, do lado esquerdo, representa um elemento de interacção (XisActionElement) do tipo Button (actionType = Button), com o nome Novo e do lado direito, a respectiva geração para a plataforma Windows Forms.NET.

ActionType MenuItem

Ao modelar um XisActionElement cuja marca *actionType* seja **MenuItem**, estamos a representar um elemento de interacção simples, do tipo item de menu a partir do qual será possível desencadear uma acção.

Restrições: Um XisActionElement cuja marca *actionType* seja **MenuItem**:

- Só pode ser representado dentro de um elemento composto do tipo Menu.

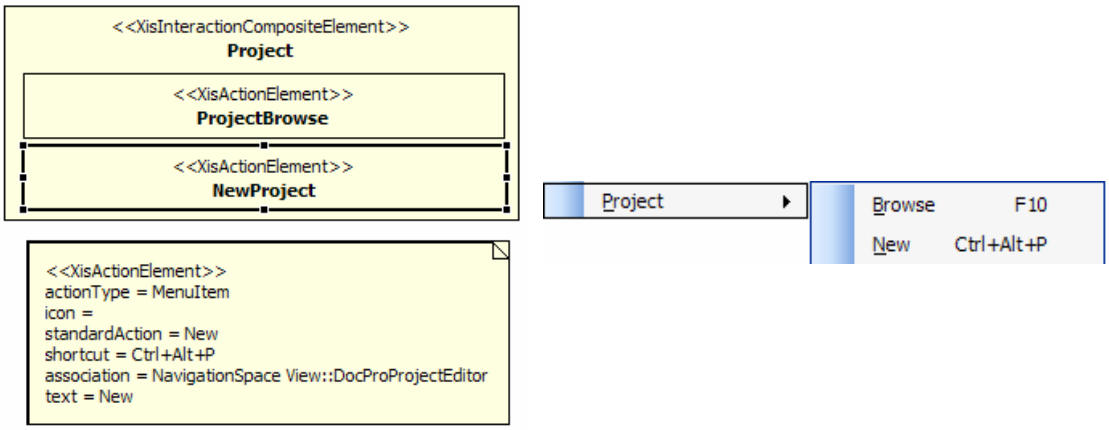


Figura 4.12: Representação de um XisInteractionCompositeElement com dos itens de menu e respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.12, do lado esquerdo, representa um elemento de interacção composto (XisInteractionCompositeElement) chamado Projectos com dois itens de menu (XisActionElement – ProjectBrowse e XisActionElement - ProjectEditor). Como podemos ver nas marcas definidas para o XisActionElement – ProjectEditor foi definido um atalho (*shortcut*) através da sequência de teclas Ctrl+Alt+P. Do lado direito, encontra-se a respectiva geração para a plataforma Windows Forms.NET.

ActionType MenuSeparator

Ao modelar um XisActionElement cuja marca *actionType* seja **MenuSeparator**, estamos a representar um elemento de interacção simples do tipo separador entre itens de menu, permitindo dividir um menu em grupos ou secções diferentes. Apesar de que um separador não permite desencadear uma acção, decidimos assumir esta solução de compromisso e utilizar um elemento simples do tipo XisActionElement. Tomamos esta decisão por uma questão de uniformização dos elementos que podem estar contidos num elemento composto do tipo Menu (XisInteractionCompositeElement cuja marca *compositeElementType* seja Menu).

Restrições: Um XisActionElement cuja marca *actionType* seja **MenuSeparator**:

- Só pode ser representado dentro de um elemento composto do tipo Menu.

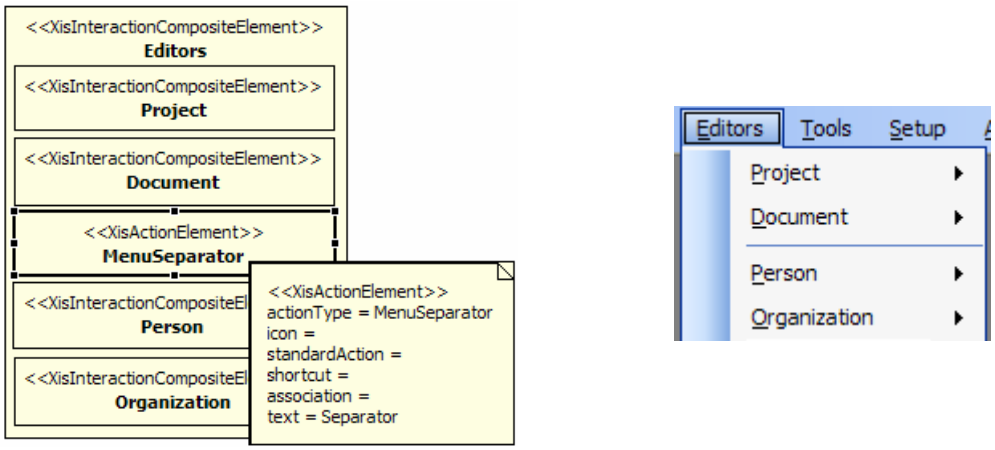


Figura 4.13: Representação de um Menu com cinco itens, sendo um deles um separador e a respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.13, do lado esquerdo, representa um elemento de interacção composto (XisInteractionCompositeElement) chamado Projectos com cinco itens de menu, sendo o terceiro um separador (XisActionElement cuja marca *actionType* é MenuSeparator) que divide o menu em duas secções e do lado direito, a respectiva geração para a plataforma Windows Forms.NET.

StandardAction

Ao modelar um XisActionElement deverá ser indicado na marca *standardAction* qual o tipo de acção que irá ser desencadeada ao activar o XisActionElement.

Um XisActionElement cuja marca *standardAction* seja **New**, **Edit**, **Select** ou **Delete**, indica que ao activar esse XisActionElement irá ser desencadeada uma acção sobre determinada entidade que permitirá respectivamente criar (novo registo), editar (alterar registo), seleccionar (consultar registo) ou eliminar. Permite definir qual o contexto para o qual a acção irá navegar, permitindo definir aspectos de interacção e de comportamento que têm de ser diferentes por exemplo para criar um novo registo (apresentação de um formulário vazio) ou para alterar um registo (apresentação de um formulário com dados, com possibilidade de alterar).

Um XisActionElement cuja marca *standardAction* seja **Navigate**, indica que ao activar esse XisActionElement irá ser desencadeada uma acção de navegação simples de um espaço de interacção para outro, ou seja sem definir nenhum contexto para o qual a acção irá navegar.

Um XisActionElement cuja marca *standardAction* seja **Close**, indica que ao activar esse XisActionElement irá ser desencadeada uma acção que fecha o actual espaço de interacção.

Um XisActionElement cuja marca *standardAction* seja **OK** ou **Cancel**, indica que ao activar esse XisActionElement irá ser desencadeada uma acção que respectivamente grava ou cancela as alterações realizadas sobre a informação contida no espaço de interacção.

Podem ser também utilizados XisActionElements cuja marca *standardAction* seja **OK**, **Cancel**, **Yes** ou **No** para representar possíveis acções resultantes de perguntas, avisos, erros, exclamações ou informações contidas em caixas de diálogo (XisInteractionSpace cuja marca *interactionSpaceType* é Dialog).

Serão utilizados XisActionElements cuja marca *standardAction* seja **Associate** ou **Dissociate**, para implementar acções de associação e desassociação do género dos implementados no Padrão Dupla Lista (ver secção 2.3.7 e ver definição de XisInteractionCompositeElements do tipo TableAssociate - secção 4.3.4).

Um XisActionElement cuja marca *standardAction* seja **Custom**, indica que ao activar esse XisActionElement irá ser desencadeada uma acção que não pode ser definida através de modelação e que poderá ser posteriormente definida pelo programador.

O tipo StandardAction é um tipo enumerado e o seu conjunto de valores pode ser alargado para ajudar a modelar outras acções que possam vir a ser desencadeadas a partir de um XisActionElement.

4.3.4. XisInteractionCompositeElement

Um XisInteractionCompositeElement representa um elemento de interacção composto que contém outros elementos de interacção (XisInteractionElements), que podem por sua vez, ser elementos de interacção simples (XisInteractionSimpleElement) ou compostos (XisInteractionCompositeElement). Este elemento permite que se agrupem vários elementos de interacção, seguindo o padrão de desenho Composite [Gamma, Helm et al. 1995]. Um XisInteractionCompositeElement pode estar associado a uma XisEntity do *Domain View*. Esta associação permite definir o contexto principal dos elementos nele contidos. Se um XisInteractionCompositeElement está associado a uma determinada entity (através de uma XisDomainAssociation) então todos os elementos

de interacção contidos no XisInteractionCompositeElement pertencerão ao mesmo contexto dessa entity.

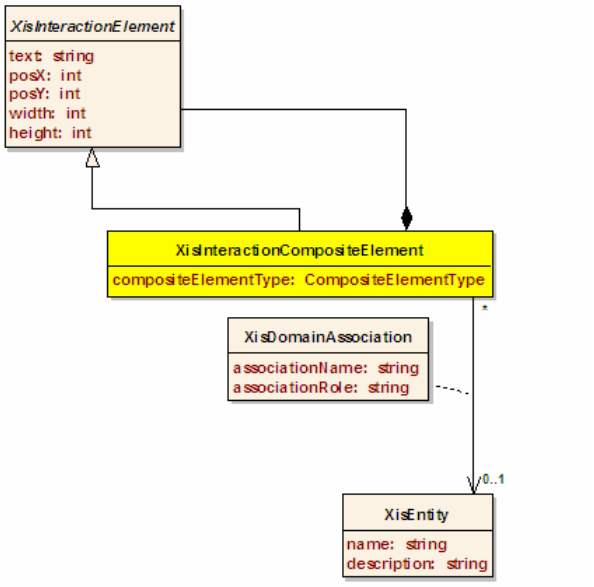


Figura 4.14: Metamodelo parcial (XisInteractionCompositeElement) da *InteractionSpace View*

A Figura 4.14 ilustra a parte do metamodelo da *InteractionSpace View* que define o elemento XisInteractionCompositeElement. Para completar a descrição do estereótipo foi definida uma marca como podemos ver na Tabela 4.5.

Nome	Tipo	Descrição
Composite ElementType	Composite ElementType	Indica o tipo de elemento composto que se pretende representar, podendo apresentar um dos seguintes valores: Menu; Tab; GroupBox; ListSelect; Table; TableAssociate; DialogExclamation; DialogInformation; DialogWarning; DialogQuestion ou DialogError;

Tabela 4.5: Marca para o estereótipo XisInteractionCompositeElement

O tipo CompositeElementType é um tipo enumerado e o seu conjunto de valores pode ser alargado, para ajudar a modelar outros padrões de elementos de interacção compostos.

CompositeElementType Menu

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **Menu**, estamos a representar um elemento de interacção composto do tipo menu (ver secção 2.3.3 - Padrão Navegação por Menus). Este XisInteractionCompositeElement,

por sua vez, pode conter outros sub-menus (outros XisInteractionCompositeElements cujo *compositeElementType* é Menu), ou que pode conter acções (XisInteractionSimpleElements - XisActionElements).

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* seja **Menu**, apenas pode conter:

- XisInteractionCompositeElements cuja marca *compositeElementType* seja Menu;
- XisActionElements cuja marca *actionType* seja MenuItem ou MenuSeparator (utilizados exclusivamente em elementos compostos do tipo Menu).

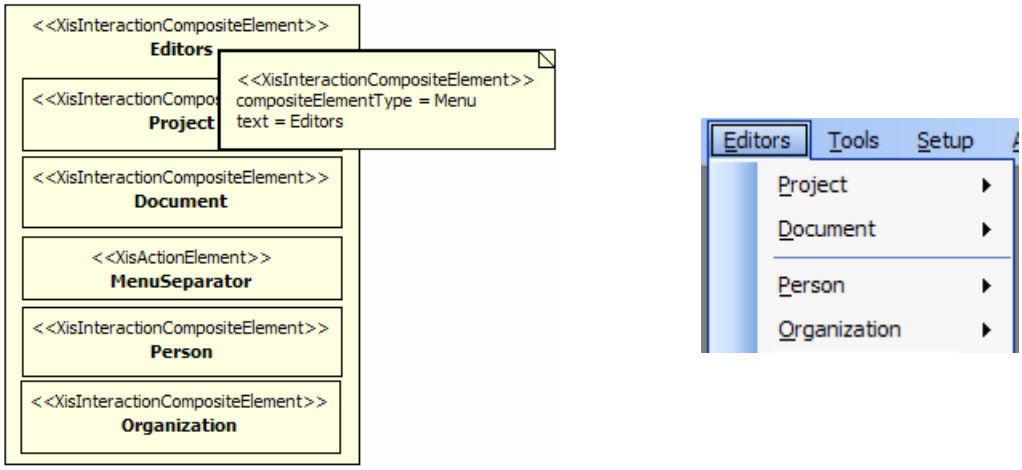


Figura 4.15: Representação de um XisInteractionCompositeElement cuja marca *compositeElementType* é Menu e respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.15, do lado esquerdo, representa um elemento de interacção composto do tipo Menu com o nome Editors e que contém cinco elementos de interacção simples, quatro acções (Project, Document, Person e Organization) e um separador de menu (MenuSeparator). Do lado direito, encontra-se ilustrada a respectiva geração para a plataforma Windows Forms.NET.

CompositeElementType GroupBox

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **GroupBox**, estamos a representar um elemento de interacção composto do tipo caixa de agrupamento. Uma caixa de agrupamento define uma área que permite agrupar um conjunto de elementos de interacção, separando-a dos restantes elementos de interacção contidos no espaço de interacção, conseguindo desta forma distinguir e separar a informação. Os elementos de interacção contidos numa caixa de agrupamento, normalmente estão relacionados e podem ser agrupados

conceptualmente numa mesma área. Por exemplo, o Padrão Apresentação em Grelha (ver secção 2.3.4) pode utilizar caixas de agrupamento para separar as diferentes áreas de um espaço de interacção. Este XisInteractionCompositeElement por sua vez pode conter outros elementos de interacção compostos, incluindo outras caixas de agrupamento (outros XisInteractionCompositeElements cujo *compositeElementType* é **GroupBox**), ou que pode conter elementos de interacção simples (XisInteractionSimpleElements).

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* seja **GroupBox**, não pode conter:

- XisInteractionCompositeElements cuja marca *compositeElementType* seja Menu;
- XisActionElements cuja marca *actionType* seja MenuItem ou MenuSeparator (utilizados exclusivamente em elementos compostos do tipo Menu).

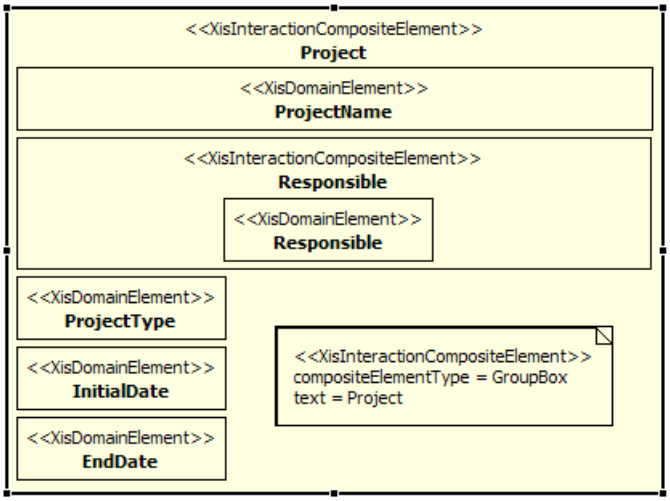


Figura 4.16: Representação de um XisInteractionCompositeElement cuja marca *compositeElementType* é GroupBox

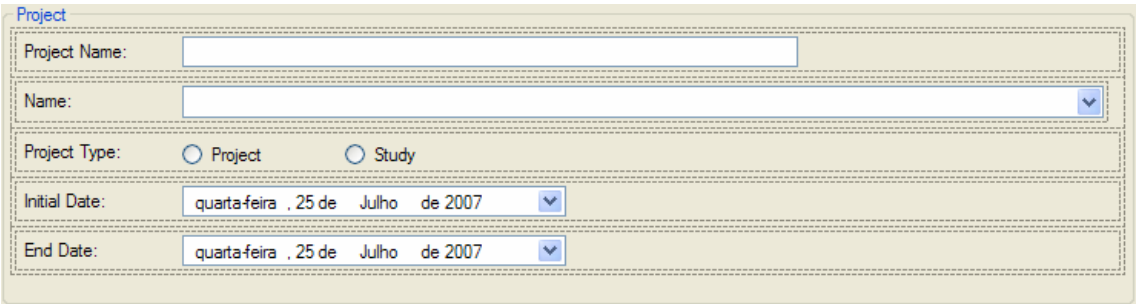


Figura 4.17: Geração do exemplo da Figura 4.16 para a plataforma Windows Forms.NET

Exemplo: A Figura 4.16 representa um elemento de interacção composto do tipo caixa de agrupamento com o nome Project e que contém quatro elementos de interacção

simples (ProjectName, ProjectType InitialDate e EndDate) e um elemento de interacção composto (Responsible). A Figura 4.17 ilustra a respectiva geração para a plataforma Windows Forms.NET.

CompositeElementType Tab

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **Tab**, estamos a representar um elemento de interacção composto do tipo tabulador (ver secção 2.3.5 - Padrão Tabuladores).

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* é **Tab**, não pode conter:

- XisInteractionCompositeElements cuja marca *compositeElementType* seja Menu;
- XisActionElements cuja marca *actionType* seja MenuItem ou MenuSeparator (utilizados exclusivamente em elementos compostos do tipo Menu).

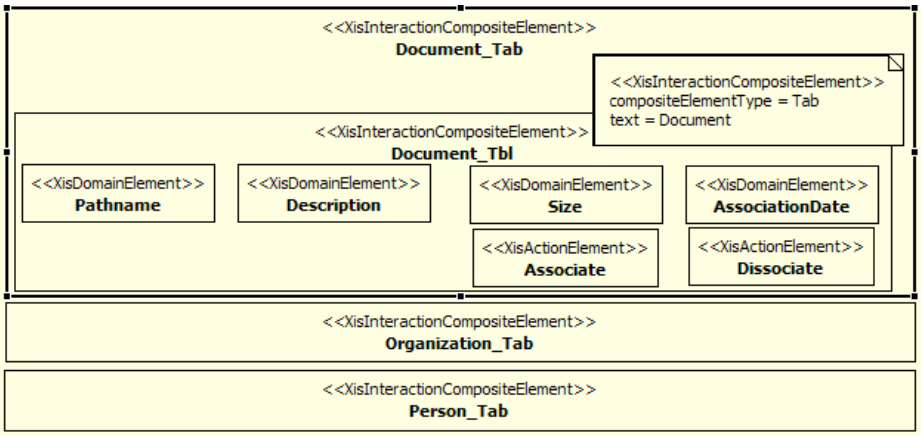


Figura 4.18: Representação de um XisInteractionCompositeElement cuja marca *compositeElementType* é Tab (Tabulador)

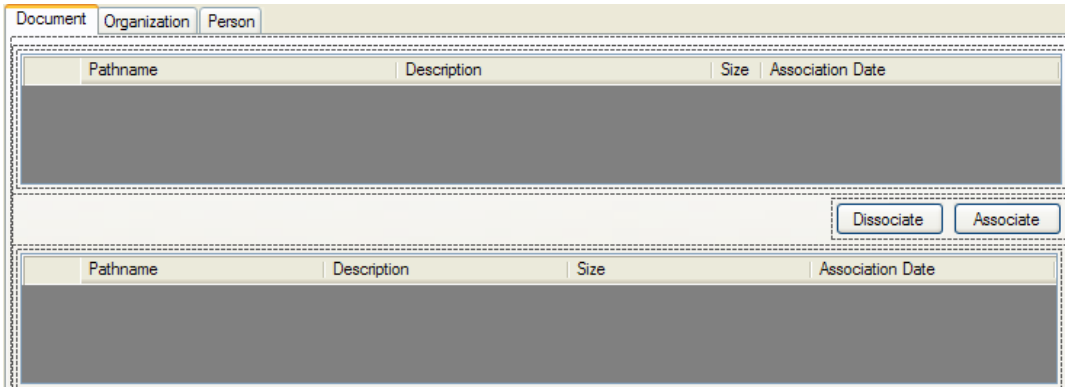


Figura 4.19: Geração do exemplo da Figura 4.18 para a plataforma Windows Forms.NET

Exemplo: A Figura 4.18 representa três elementos de interacção compostos do tipo Tabulador (Document_Tab, Organization_Tab e Person_Tab). O elemento composto Document_Tab contém vários elementos de interacção simples (quatro XisDomainElements e dois XisActionElements). A Figura 4.19 ilustra a respectiva geração para a plataforma Windows Forms.NET.

CompositeElementType ListSelect

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **ListSelect**, estamos a representar um elemento de interacção composto do tipo lista (ver o Padrão Escolha A Partir De Um Conjunto Longo na secção 2.3.1 - Padrões de Escolha). Este XisInteractionCompositeElement por sua vez pode conter outros XisDataElements para representar cada uma das colunas da lista.

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* seja **ListSelect**, apenas pode conter XisDataElements, ou seja não pode conter:

- Nenhum elemento de interacção composto (XisInteractionCompositeElement);
- Nenhuma acção (XisActionElements);
- XisDataTables.

Figura 4.20: Representação de um XisInteractionCompositeElement cuja marca *compositeElementType* é ListSelect e respectiva geração para a plataforma Windows Forms.NET.

Exemplo: A Figura 4.20, do lado esquerdo, representa um elemento de interacção composto do tipo lista com o nome Responsible e que contém um elemento de interacção simples (XisDataElement, neste caso XisDomainElement) que representa que a lista tem apenas uma coluna. Do lado direito, encontra-se ilustrada a respectiva geração para a plataforma Windows Forms.NET.

CompositeElementType Table

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **Table**, estamos a especificar um elemento de interacção composto do tipo tabela (ver secção 2.3.6 - Padrão Tabela). Este XisInteractionCompositeElement pode conter outros elementos de interacção simples (XisInteractionSimpleElements), tais como: (1) XisDataElements para representar cada uma das colunas da tabela e (2) XisActionElements para representar as acções que podem ser realizadas sobre os registos da tabela.

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* seja **Table**, não pode conter:

- Nenhum elemento de interacção composto (XisInteractionCompositeElement), apenas pode conter elementos de interacção simples (XisInteractionSimple Element);
- XisActionElements cuja marca *actionType* seja MenuItem ou MenuSeparator (utilizados exclusivamente em elementos compostos do tipo Menu);
- XisDataTables.

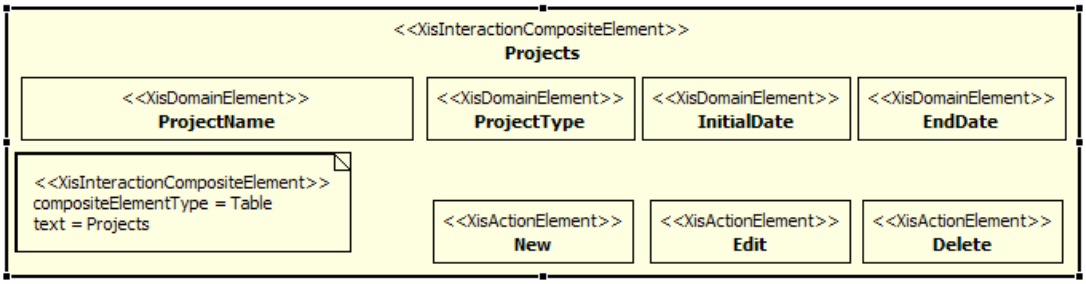
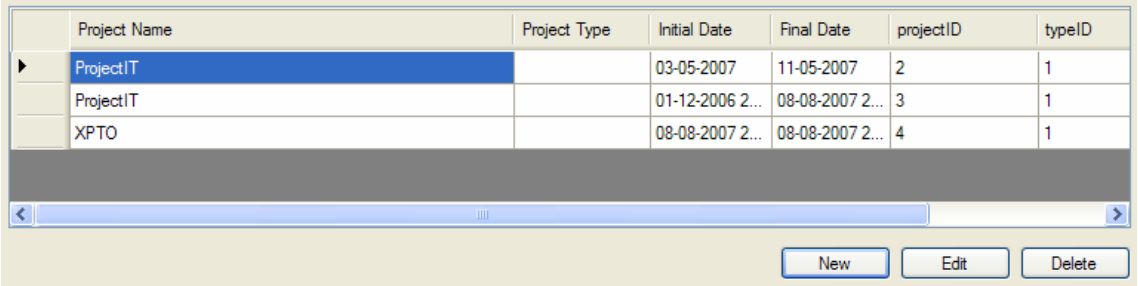


Figura 4.21: Representação de um XisInteractionCompositeElement cuja marca *compositeElementType* é Table



Project Name	Project Type	Initial Date	Final Date	projectID	typeID
ProjectIT		03-05-2007	11-05-2007	2	1
ProjectIT		01-12-2006 2...	08-08-2007 2...	3	1
XPTO		08-08-2007 2...	08-08-2007 2...	4	1

Figura 4.22: Geração do exemplo da Figura 4.18 para a plataforma Windows Forms.NET

Exemplo: A Figura 4.21 representa um elemento de interacção composto do tipo Tabela com o nome Projects e que contém seis elementos de interacção simples, nomeadamente: (1) quatro elementos de interacção (XisDataElement, neste caso XisDomainElement) que representam as colunas da tabela e (2) três acções (XisActionElement) que podem ser realizadas sobre os registos da tabela (New Edit e Delete). A Figura 4.22 ilustra a respectiva geração para a plataforma Windows Forms.NET.

CompositeElementType TableAssociate

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **TableAssociate**, estamos a representar um elemento de interacção composto do tipo Padrão Dupla Lista (ver secção 2.3.7), utilizado para fazer associações. Ou seja estamos a representar duas tabelas ou duas listas, a primeira que contém todos os itens passíveis de ser seleccionados e a segunda que contém os itens já seleccionados. Este XisInteractionCompositeElement por sua vez pode conter outros elementos de interacção simples (XisInteractionSimpleElements), nomeadamente XisDataElements para representar cada uma das colunas da tabela, e tem que conter obrigatoriamente dois XisActionElements para representar as acções de associar / desassociar um registo.

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* seja **TableAssociate**:

- Não pode conter nenhum elemento de interacção composto (XisInteraction CompositeElement), apenas pode conter elementos de interacção simples (XisInteractionSimpleElement);
- Tem que conter pelo menos dois XisActionElement cuja marca *standardAction* seja Associate e Disassociate (utilizados exclusivamente em elementos compostos do tipo TableAssociate);
- Não pode conter XisActionElements cuja marca *actionType* seja MenuItem ou MenuSeparator (utilizados exclusivamente em elementos compostos do tipo Menu);
- Não pode conter XisDataTables.

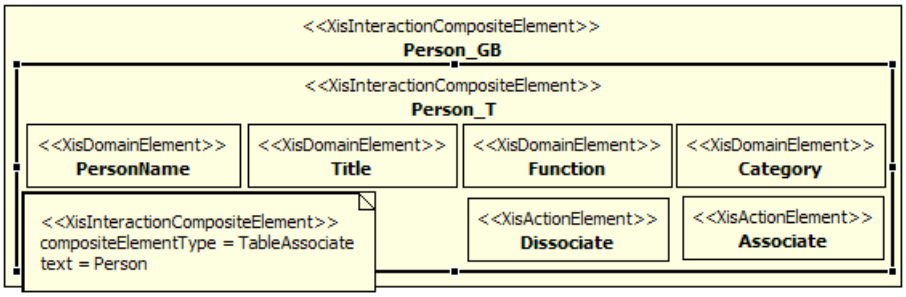


Figura 4.23: Representação de um XisInteractionCompositeElement (Pessoas_T) cuja marca compositeElementType é TableAssociate (padrão Dupla Lista, utilizando tabelas)

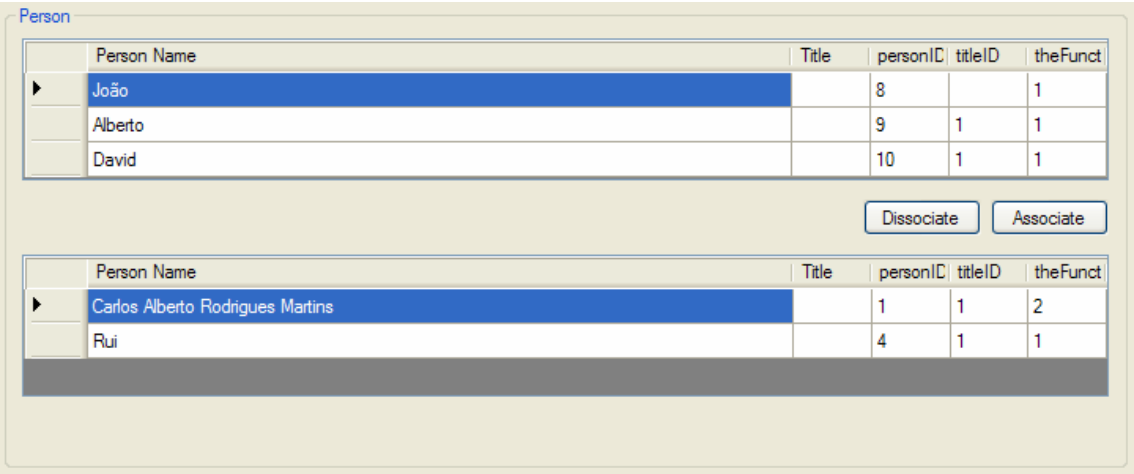


Figura 4.24: Geração do exemplo da Figura 4.23 para a plataforma Windows Forms.NET

Exemplo: A Figura 4.23 representa um elemento de interacção composto do tipo caixa de agrupamento chamado Pessoas_GB que, por sua vez, contém outro elemento de interacção composto do tipo Padrão Dupla Lista com o nome Person_T. Este, por sua vez, contém seis elementos de interacção simples, nomeadamente: (1) quatro elementos de interacção simples (XisDomainElement) que representam as colunas das tabelas e (2) duas acções (XisActionElement) que podem ser realizadas sobre os registos da tabela (Dissociate e Associate). Para efeitos de modelação e tendo em conta que as duas tabelas do padrão podem ter a mesma estrutura, optou-se por simplificar a modelação deste padrão, representando apenas uma tabela e não as duas tabelas como na realidade este padrão é tipicamente concretizado. A Figura 4.24 ilustra a respectiva geração para a plataforma Windows Forms.NET.

CompositeElementType DialogExclamation, DialogInformation, DialogWarning, DialogQuestion ou DialogError

Ao modelar um XisInteractionCompositeElement cuja marca *compositeElementType* é **DialogExclamation, DialogInformation, DialogWarning, DialogQuestion** ou **DialogError**, estamos a representar um elemento de interacção composto, respectivamente uma mensagem de exclamação, mensagem informativa, aviso, pergunta ou mensagem de erro contida numa caixa de diálogo (XisInteractionSpace cuja marca *interactionSpaceType* é Dialog).

Restrições: Um XisInteractionCompositeElement cuja marca *compositeElementType* seja **DialogExclamation, DialogInformation, DialogWarning, DialogQuestion** ou **DialogError**, tem que estar contido numa caixa de diálogo (XisInteractionSpace cuja marca *interactionSpaceType* é Dialog) e não pode conter:

- Elementos de interacção compostos (XisInteractionCompositeElement). Apenas pode conter elementos de interacção simples (XisInteractionSimpleElement);
- XisActionElements cuja marca *actionType* seja MenuItem ou MenuSeparator (utilizados exclusivamente em elementos compostos do tipo Menu);
- Não pode conter XisDataTables.

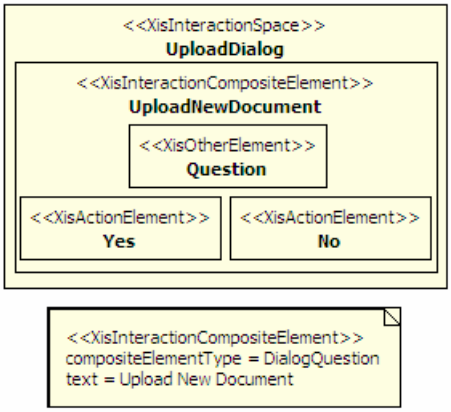


Figura 4.25: Representação de um XisInteractionCompositeElement cuja marca *compositeElementType* é DialogQuestion

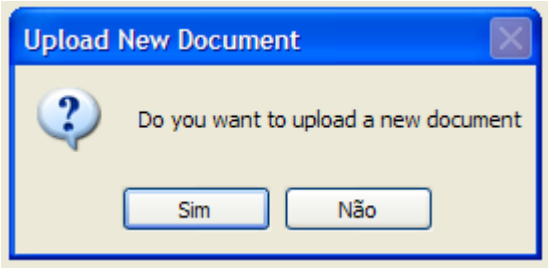


Figura 4.26: Geração do exemplo da Figura 4.25 para a plataforma Windows Forms.NET

Exemplo: A Figura 4.25 representa um espaço de interacção do tipo caixa de diálogo chamado UploadDialog que contém um elemento de interacção composto do tipo pergunta (XisInteractionCompositeElement cuja marca *compositeElementType* é **DialogQuestion)** com o nome UploadNewDocument. Este, por sua vez, contém três elementos de interacção simples, nomeadamente: (1) um elemento de interacção (XisOtherElement) que representa a pergunta propriamente dita e (2) duas acções (XisActionElement) que indicam as possíveis respostas (Yes e No) à pergunta. A Figura 4.26 ilustra a respectiva geração para a plataforma Windows Forms.NET.

4.4. Conclusão

Neste capítulo foram apresentados em maior detalhe diversos aspectos relacionados com o desenho de interfaces gráficas, utilizando o perfil XIS. Foram apresentados os metamodelos das duas vistas que compõem a *User-interfaces View*: (1) a *NavigationSpace View* e (2) a *InteractionSpace View*. O perfil XIS permite representar aspectos de navegação dos sistemas, como pudemos constatar na secção 4.2. Permite também especificar a estrutura de cada espaço de interacção, como analisado na secção 4.3. Nesta secção apresentamos diversos aspectos de modelação da *InteractionSpace View*, baseados no caso de Estudo DocPro. Apresentamos como podemos representar diferentes tipos de espaços de interacção (XisInteractionSpaces). Apresentamos também como podemos representar os elementos que fazem parte de cada espaço de interacção, nomeadamente os elementos de interacção simples (XisInteractionSimple Elements) e os elementos de interacção compostos (XisInteractionCompositeElements). Dentro dos elementos de interacção simples apresentamos os elementos de interacção que nos permitem representar dados (XisDataElements) e os elementos de interacção que nos permitem representar acções (XisActionElements).

Através do perfil XIS é possível especificar um grande leque de elementos de interacção simples e compostos. A utilização de XisInteractionCompositeElements permite representar com facilidade padrões de interacção mais ou menos complexos. A utilização no perfil XIS de marcas (interactionSpaceType, actionType, controlType, compositeElementType) cujo tipo enumerado contém uma lista de valores que pode ser estendida a qualquer momento permite representar qualquer tipo de elemento de interacção genérico existente ou que venha a existir no futuro. A utilização da marca *standardAction* permite definir qual o contexto para o qual uma acção irá navegar, permitindo definir aspectos de interacção e de comportamento.

Concluída a apresentação dos diversos aspectos relacionados com o desenho de interface gráficas, apresentaremos no próximo capítulo os aspectos relativos à geração de interfaces gráficas a partir do perfil XIS.

5. Geração de Interfaces Gráficas

Neste capítulo são explicados alguns aspectos relativos à geração de interfaces gráficas, utilizando a abordagem Project-IT. São definidos alguns aspectos relativos à ferramenta Project-IT Studio e são apresentados os diversos passos que antecedem o processo de geração, nomeadamente: (1) definição da arquitectura de software; (2) definição dos *templates* de geração de código e (3) definição do processo de geração. São também apresentados alguns aspectos de geração de modelo-para-código para plataformas computacionais específicas.

5.1. Ferramenta Project-IT-Studio

O ProjectIT-Studio/MDDCodeGenerator é o componente do ProjectIT-Studio [Silva, Videira et al. 2006] responsável pela configuração e execução dos processos de geração das aplicações.

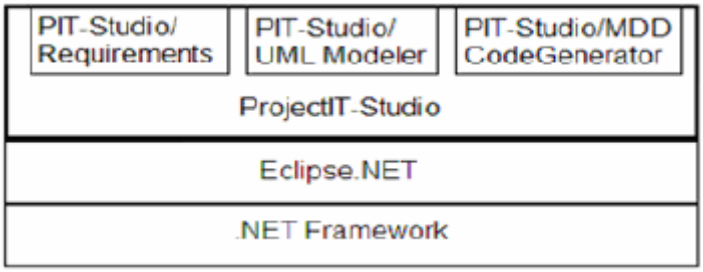


Figura 5.1: Componentes do ProjectIT-Studio (extraído de [Silva, Videira et al. 2006])

O ProjectIT-Studio integra actualmente mais dois componentes, desenvolvidos no âmbito de outros trabalhos de investigação: (1) o ProjectIT-Studio/Requirements [Videira e Silva 2004] – componente responsável pela definição e gestão de requisitos; e (2) o ProjectIT-Studio/UMLModeler [Saraiva 2005] – componente responsável pela definição e gestão dos modelos. A Figura 5.1 ilustra genericamente a arquitectura do ProjectIT-Studio.

Neste trabalho de investigação foram definidos mecanismos de geração modelo-para-código para duas plataformas computacionais: (1) a plataforma Microsoft Windows Forms.NET [Microsoft ---b] e (2) a plataforma Microsoft ASP.NET [Microsoft ---a]. Antes de iniciar o processo de geração modelo-para-código segundo a abordagem ProjectIT deverão ser realizadas um conjunto de tarefas: (1) definir a arquitectura de software; (2) definir os *templates* de geração de código e (3) definir o processo de geração. Seguindo esta metodologia, definimos duas arquitecturas de *software* através do respectivo editor de arquitecturas, uma para cada plataforma a gerar. A arquitectura de *software* é uma representação da plataforma para a qual se pretende gerar a aplicação, que por sua vez é definida pelos *templates* que a constituem (ver Figura 5.2).

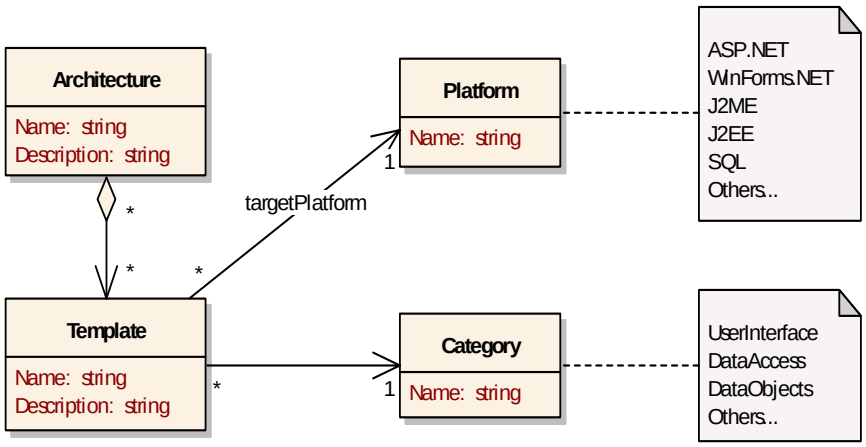


Figura 5.2: Entidades da Arquitectura de Software (extraído de [Silva 2006])

A partir do editor de arquitecturas podemos adicionar e retirar os *templates* de geração que podem vir a ser utilizados no processo de geração que está a ser definido, conforme ilustrado na Figura 5.3. O utilizador pode ver os *templates* que fazem parte da arquitectura através da lista localizada do lado esquerdo, e na do lado direito pode ver os *templates* que estão disponíveis. Para adicionar *templates* à arquitectura, o utilizador arrasta os *templates* da lista dos disponíveis para a lista dos *templates* da arquitectura, e para retirar faz a operação inversa, i.e. arrasta os *templates* da lista dos *templates* pertencentes à arquitectura para a lista dos disponíveis.

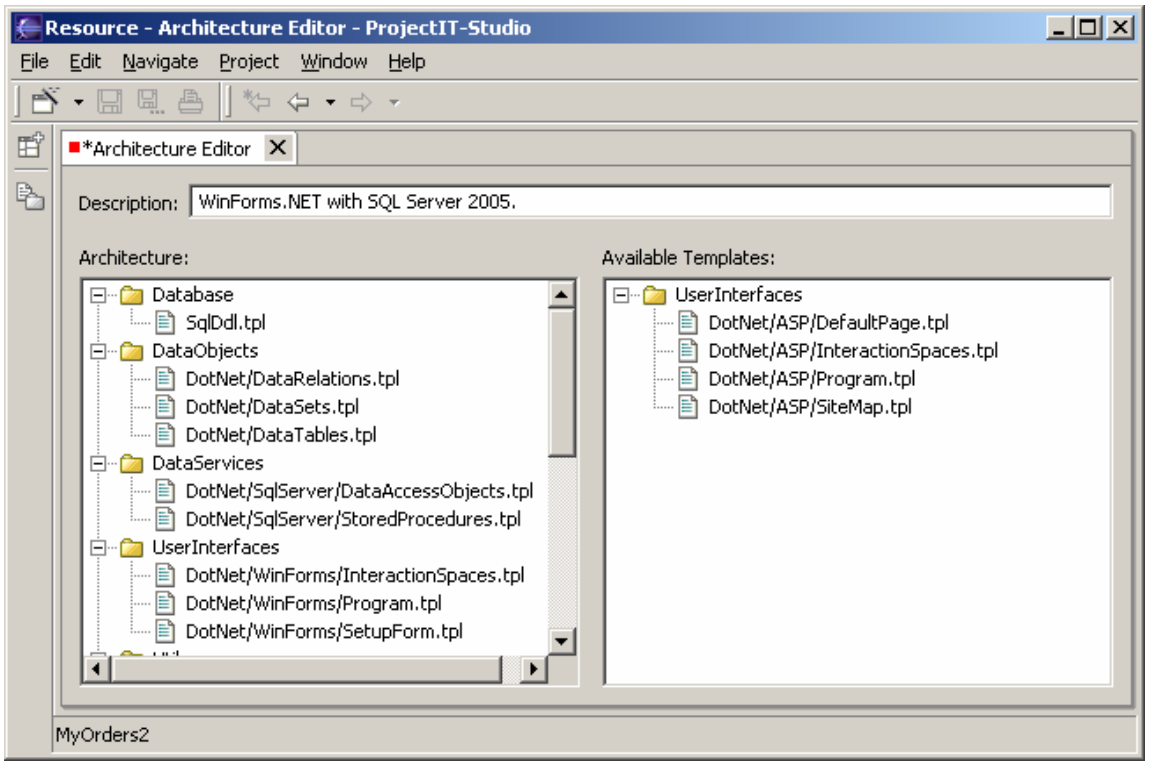


Figura 5.3: Editor de Arquitecturas de Software

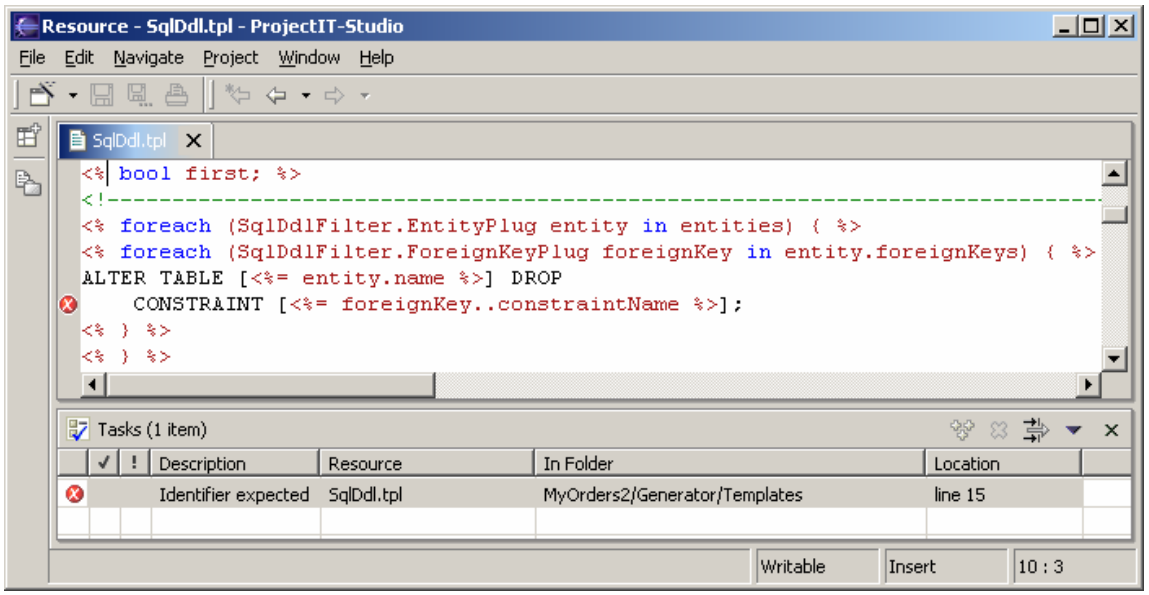


Figura 5.4: Editor de *Templates*

De seguida definimos os diversos *templates* de geração de código para estas duas plataformas que farão parte de cada uma das arquitecturas. A Figura 5.4 ilustra o Editor de *templates*. Um *template* é a definição de uma transformação modelo-para-

código, escrita numa linguagem própria, semelhante à linguagem ASP (*Active Server Pages*) [Microsoft ---a]. A linguagem tem um conjunto de directivas para definir os *templates*, que são utilizadas juntamente com instruções de código na linguagem C# [Microsoft ---d].

Finalmente definimos dois processos de geração, um para cada plataforma a gerar. O processo de geração é uma configuração que define os *inputs* do gerador de código (arquitectura de *software* e modelo XIS a gerar). O processo de geração associa o modelo, à arquitectura de *software* da aplicação (ver Figura 5.5).

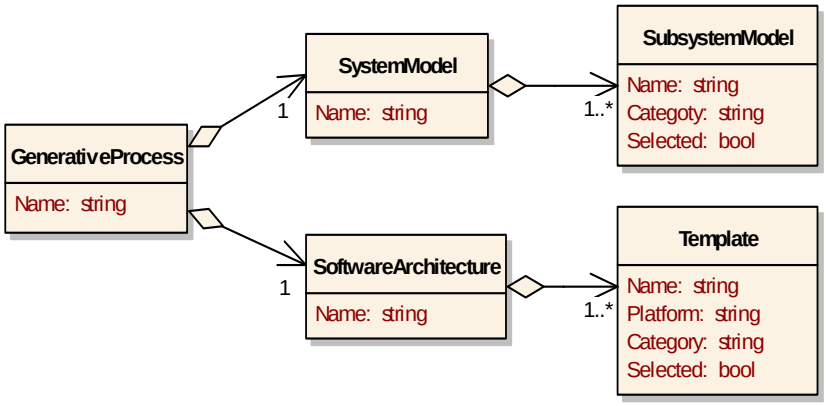


Figura 5.5: Entidades do Processo de Geração (extraído de [Silva, Videira et al. 2006])

No processo de geração podemos definir se pretendemos gerar todos ou apenas parte dos *templates* de geração e também podemos definir se pretendemos gerar todo ou apenas parte do modelo XIS seleccionado, conforme se encontra ilustrado na Figura 5.6. O modelo é composto por subsistemas (tais como as entidades do modelo de domínio, ou as interface de utilizador da aplicação), e a arquitectura de *software* é composta por *templates* (tais como de geração de *scripts* SQL ou ficheiros de código C#). A Figura 5.6 ilustra o editor de processos de geração. A interface permite definir: (1) o nome do processo de geração, isto é,. o nome da aplicação que é gerada; (2) do lado esquerdo, permite escolher a arquitectura de *software* e seleccionar os *templates* que devem ser executados e (3) do lado direito, permite escolher o modelo e seleccionar os subsistemas que são incluídos no modelo instanciado a ser utilizado na geração.

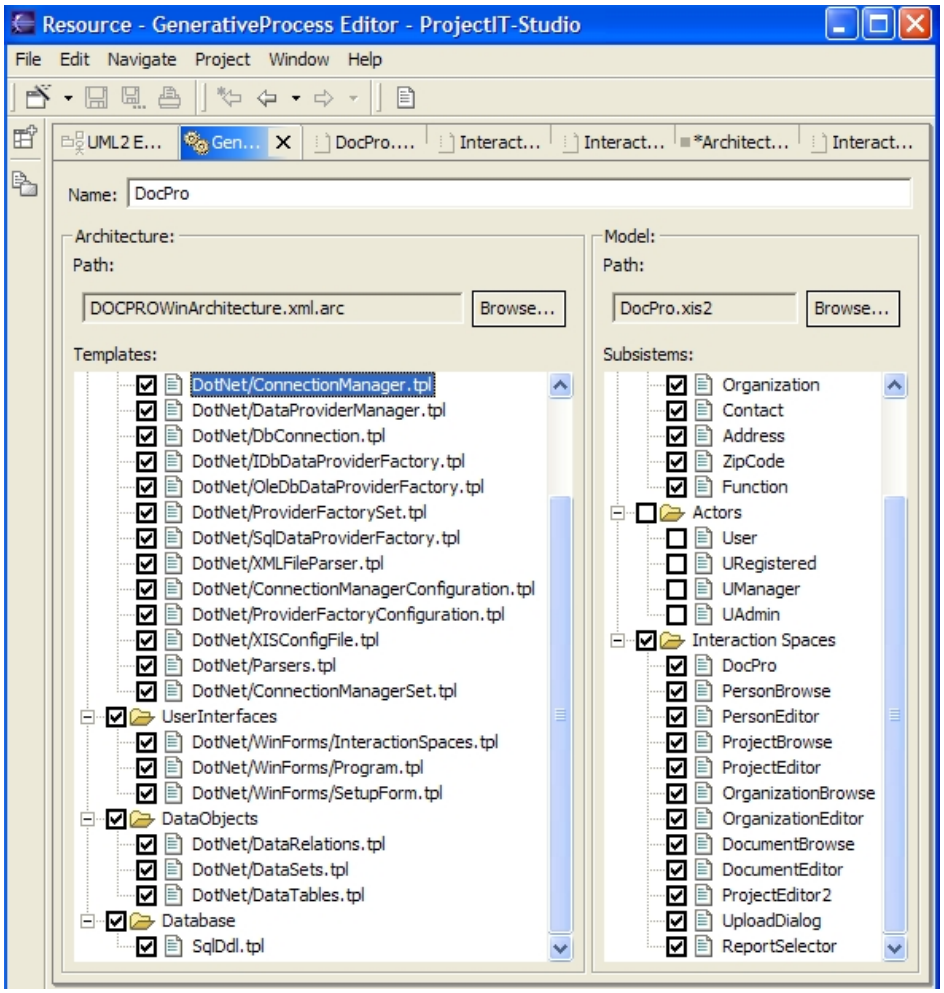


Figura 5.6: Editor de Processos de Geração

5.2. Geração para Windows Forms.NET

5.2.1. A Plataforma Windows Forms.NET

A plataforma Windows Forms.NET [Microsoft ---b] é uma plataforma da Microsoft que fornece um conjunto de classes disponibilizadas através da *framework* .NET, que permitem o desenvolvimento rápido de aplicações para o sistema operativo Windows. A plataforma Windows Forms.NET fornece um conjunto de bibliotecas para o desenho de interfaces gráficas com o utilizador através da ferramenta Windows Forms Designer.

O Windows Forms Designer dispõe de uma ***toolbox*** que exibe uma lista a partir da qual é possível arrastar os vários elementos que podem fazer parte da interface gráfica.

Também dispõe de ferramentas que ajudam no posicionamento e alinhamento desses elementos num determinado espaço de interacção.

Para a plataforma Windows Forms.NET, os espaços de interacção são, **janelas** ou **caixas de diálogo**. Um formulário ou janela é uma área em branco à qual é possível adicionar controlos de forma a definir determinada interface gráfica.

Para a plataforma Windows Forms.NET, os elementos que fazem parte da interface gráfica são designados por **controlos**. Um controlo é um objecto que pode ser colocado num espaço de interacção da interface gráfica (e.g. *TextBox*, *Label*) e que é utilizado para interagir com o utilizador. Cada tipo de controlo tem definido um conjunto de propriedades, métodos e eventos que permitem especificar o controlo tendo em conta a sua finalidade.

Um **evento** é uma acção que pode ser desencadeada a partir da intervenção do utilizador (e.g. *click*, *keyPress*) ou a partir do sistema. Quando um evento é activado é executado o código correspondente ao evento. Cada controlo dispõe de um conjunto de eventos que pode ser programado. As **propriedades** e **métodos** são respectivamente atributos e funções associados aos controlos.

5.2.2. Apresentação para Windows Forms.NET

Nesta secção abordaremos algumas questões genéricas de apresentação das janelas geradas para a plataforma Windows Forms.NET, nomeadamente alguns aspectos relativos ao *layout* de cada espaço de interacção. A versão do Visual Studio 2005 [Microsoft ---e] introduz dois novos controlos: (1) o FlowLayoutPanel e (2) o TableLayoutPanel.

O **FlowLayoutPanel** [Microsoft ---f] é um controlo do tipo *container* dentro do qual podem ser adicionados outros controlos e cujo fluxo de posicionamento segue uma determinada direcção. O controlo FlowLayoutPanel posiciona de forma automática todos os controlos nele contidos segundo a direcção indicada pelo valor da propriedade *FlowDirection*. Esta propriedade pode conter um dos seguintes valores enumerados possíveis: (1) LeftToRight; (2) TopDown; (3) RightToLeft e (4) BottonUp.

Por exemplo, cada vez que se posiciona um controlo dentro de um controlo FlowLayoutPanel cuja propriedade *FlowDirection* seja *LeftToRight*, este posiciona-se à direita do controlo previamente posicionado. Se não houver espaço à direita do

controlo, o novo controlo fica posicionado em baixo, no próximo espaço disponível. Se colocarmos vários controlos dentro deste FlowLayoutPanel e se este for suficientemente grande, todos os botões ficam posicionados lado a lado na mesma linha, mas se diminuirmos a largura do controlo FlowLayoutPanel, os controlos nele contidos são reposicionados no FlowLayoutPanel de forma a ocuparem duas ou mais linhas.

O **TableLayoutPanel** [Microsoft ---g] é um controlo do tipo *container* que como o nome sugere tem um formato de forma tabular (com linhas e colunas). Em cada célula do TableLayoutPanel apenas pode ser colocado um controlo. É possível adicionar várias linhas / colunas ao TableLayoutPanel.

À partida pode não parecer que o FlowLayoutPanel e o TableLayoutPanel tenham grande utilidade, mas na realidade a sua utilização permite construir interfaces gráficas flexíveis onde os controlos são posicionados conforme as necessidades. Estes controlos também permitem que seja definido o *layout* de determinado espaço de interacção sem que seja necessário definir a posição específica de cada controlo nele contido. Permite resolver o problema de inserção de novos controlos no meio de outros já existentes sem ter necessidade de realinhar e reposicionar manualmente os restantes controlos afectados com esta inserção. Outra vantagem é que o FlowLayoutPanel e ao TableLayoutPanel permitem agrupar os controlos nele contidos como se se tratasse apenas de uma unidade, o que facilita a sua movimentação.

No nosso processo de geração optamos por recorrer a estes dois controlos do tipo *container*. Utilizamos o controlo TableLayoutPanel para criar as várias linhas onde irão ser colocados os vários elementos de interacção. Utilizamos o controlo FlowLayaoutPanel para posicionar vários controlos seguidos, como por exemplo no posicionamento dos botões ou *links*.

À medida que os controlos vão sendo gerados e posicionados na janela ou *form* é atribuído a cada controlo o valor da propriedade *TabIndex*. Este valor vai sendo incrementado para cada controlo gerado e permite definir a sequência de passagem pelos diferentes controlos da janela ou *form* através da utilização por parte do utilizador da tecla *Tab*.

De seguida apresentamos todos os controlos e janelas que podem ser gerados para a plataforma Microsoft Windows Forms.NET.

5.2.3. Janelas e Tipos de Janelas

Nesta secção apresentaremos diversos aspectos de geração de vários tipos de janela para a plataforma Microsoft Windows Forms.NET. Para cada um dos XisInteractionSpace definidos na vista *InteractionSpace View* que foram seleccionados no processo de geração e cuja marca *interactionSpaceType* seja **Form**, é gerada uma janela (controlo *window*). O título da janela (propriedade *text* do controlo *window*) é preenchido com o nome do XisInteractionSpace (marca *name* do XisInteractionSpace). Ao gerar um XisInteractioSpace cuja marca *firstByDefault* seja *true* será gerada uma janela que é a primeira da aplicação. Neste caso e sem que o modelador tenha necessidade de o explicitar esta será gerada com vários menus predefinidos (Menu File, Menu Setup e Menu About). Esta janela será do tipo *mdiContainer* (*Multiple Document Interface Container*).

5.2.4. Controlos e Tipos de Controlos

Nesta secção apresentaremos diversos aspectos de geração de vários tipos de controlos para a plataforma Microsoft Windows Forms.NET.

Caixas de Diálogo

Para cada um dos XisInteractionSpace definidos na vista *InteractionSpace View* que foram seleccionados no processo de geração e cuja marca *interactionSpaceType* seja **Dialog**, é chamada uma caixa de diálogo através do comando *MessageBox*. Um XisInteractionSpace cuja marca *interactionSpaceType* seja **Dialog** tem que conter um XisInteractionCompositeElement que representa o conteúdo da caixa de diálogo. O título da caixa de diálogo (parâmetro *caption* da *MessageBox*) é preenchido com o valor da marca *text* do XisInteractionCompositeElement. Por sua vez este XisInteraction CompositeElement tem que conter um XisOtherElement que representa o texto da mensagem da caixa de diálogo e um ou vários XisActionElements que representam as respostas possíveis da caixa de diálogo. O texto da mensagem da caixa de diálogo (parâmetro *text* da *MessageBox*) é preenchido com o valor da marca *text* do XisOtherElement. São gerados tantos botões ou *links* (*parâmetro MessageBoxButtons* da *MessageBox*) como os XisActionElements contidos no XisInteractionCompositeElement. O texto dos botões ou *links* é preenchido com o valor da marca *text* do XisActionElement. O tipo da mensagem da caixa de diálogo gerada é indicada através

do *icon* da mensagem (parâmetro *MessageBoxIcon* da *MessageBox*) e é preenchido a partir da marca *compositeElementType* do XisInteractionCompositeElement, conforme indicado na Tabela 5.1.

compositeElementType	Mensagem da caixa de diálogo	Parâmetro *MessageBoxIcon*
DialogExclamation	Mensagem de exclamação	MessageBoxIcon.Exclamation
DialogInformation	Mensagem informativa	MessageBoxIcon.Information
DialogWarning	Aviso	MessageBoxIcon.Warning
DialogQuestion	Pergunta	MessageBoxIcon.Question
DialogError	Mensagem de Erro	MessageBoxIcon.Error

Tabela 5.1: Tipos de mensagens das caixas de diálogo geradas

Menus

Para cada XisInteractionCompositeElement cuja marca *compositeElementType* é **Menu** é gerada na janela um menu (controlo *ToolStripMenuItem*). Este por sua vez pode conter: (1) outros XisInteractionCompositeElements cuja marca *compositeElementType* seja **Menu**, que são gerados sob a forma de sub-menus (controlo *ToolStripMenuItem*); (2) XisActionElements cuja marca *actionType* seja **MenuItem**, que são gerados sob a forma de itens de menu (controlo *ToolStripMenuItem*) ou (3) XisActionElements cuja marca *actionType* seja **MenuSeparator** que são gerados sob a forma de separadores de menu (controlo *ToolStripSeparator*). O texto dos sub-menus e itens de menu (propriedade *text* do controlo *ToolStripMenuItem*) é preenchido com o valor da marca *text* do XisInteractionCompositeElement ou do XisActionElement.

Caixas de Agrupamento e Tabuladores

Para cada XisInteractionCompositeElement cuja marca *compositeElementType* é **GroupBox** ou **Tab** é gerada respectivamente uma caixa de agrupamento (controlo *GroupBox*) ou um tabulador (controlo *TabPage*). O texto da caixa de agrupamento ou do tabulador (propriedade *text* do controlo *GroupBox* ou *TabPage*) é preenchido com o valor da marca *text* do XisInteractionCompositeElement.

Tabelas

Para cada XisInteractionCompositeElement cuja marca *compositeElementType* é **Table** é gerada uma Tabela (controlo *DataGridView*). Cada XisInteractionCompositeElement, por sua vez, pode conter: (1) XisDataElements, que são gerados como colunas da tabela (normalmente controlos *DataGridViewTextBoxColumn*); ou (2) XisActionElements, que

são gerados como botões ou *links* (controlo *Button* ou *Link*), que permitirão realizar acções sobre os registos da tabela. O texto do título das colunas da tabela (propriedade *HeaderText* do controlo *DataGridViewTextBoxColumn*) é preenchido com o valor da marca *text* do XisDataElement. As linhas da tabela são geradas, utilizando cores diferentes nas linhas pares e ímpares de modo a facilitar a leitura dos registos da tabela. Os botões e *links* são gerados conforme indicado no seguinte parágrafo.

Botões e Links

Para cada XisActionElement cuja marca *controlType* é **Button** ou **Link** é gerado respectivamente um botão (controlo *Button*) ou um *link* (controlo *Link*). Utilizamos o controlo FlowLayoutPanel com a propriedade *FlowDirection* definida como *LeftToRight* com vista a agrupar os botões ou *links* e para coloca-los posicionados da esquerda para a direita. O texto dos botões ou *links* (propriedade *text* do controlo *Button* ou *Link*) é preenchido com o valor da marca *text* do XisActionElement.

Etiquetas

Para cada XisDataElement cuja marca *controlType* é **Label** é gerada uma etiqueta (controlo *Label*). O texto da etiqueta (propriedade *text* do controlo *Label*) é preenchido com o valor da marca *text* do XisDataElement.

Caixas de Texto

Para cada XisDataElement cuja marca *controlType* é **TextBox** serão gerados dois controlos: uma etiqueta (controlo *Label*) e uma caixa de texto (controlo *TextBox*). O texto da etiqueta (propriedade *text* do controlo *Label*) é preenchido com o valor da marca *text* do XisDataElement. Caso se tratem de XisDomainElements e tendo em conta que estes estão associados a um atributo da *Domain View*, é possível aproveitar alguma informação desta vista para o processo de geração. Nestes casos utilizamos a informação da marca *type* da XisEntityAttribute da *Domain View*, se a marca *type* for de um tipo numérico o texto das caixas de texto será alinhado à direita (propriedade *TextAlign = Right*). Utilizamos também a informação da marca *size* da XisEntityAttribute da *Domain View* para inferir o tamanho (propriedade *size* do controlo *TextBox*) das caixas texto a gerar. Esta informação da marca *size* também é aproveitada para a geração das caixas combinadas.

Caixas de Marcação

Para cada XisDataElement cuja marca *controlType* é **CheckBox** é gerada uma caixa de marcação (controlo *CheckBox*). O texto da caixa de marcação (propriedade *text* do controlo *CheckBox*) é preenchido com o valor da marca *text* do XisDataElement.

Botões de Rádio

Para cada XisDataElement cuja marca *controlType* é **RadioButton** é gerado um botão de rádio (controlo *RadioButton*). O texto do botão de rádio (propriedade *text* do controlo *RadioButton*) é preenchido com o valor da marca *text* do XisDataElement. Caso se tratem de XisDomainElements ou seja elementos que estão associados a um atributo da *Domain View*, se este atributo for de um tipo enumerado, serão gerados tantos botões de rádio (controlo *RadioButton*) como o número de valores possíveis do tipo enumerado. Neste caso o texto de cada um dos botões de rádio (propriedade *text* do controlo *RadioButton*) é preenchido com o valor da marca *value* do XisEnumerationValue correspondente.

Caixas Combinadas

Para cada XisDataElement cuja marca *controlType* é **ComboBox** é gerada uma caixa combinada (controlo *ComboBox*). O texto da caixa combinada (propriedade *text* do controlo *ComboBox*) é preenchido com o valor da marca *text* do XisDataElement. Caso se tratem de XisDomainElements ou seja elementos que estão associados a um atributo da *Domain View*, se este atributo for de um tipo enumerado, a caixa combinada será preenchida com a lista de valores possíveis do tipo enumerado. O texto das caixas combinadas (propriedade *text* do controlo *ComboBox*) é preenchido com o valor da marca *text* do XisDataElement.

Selectores de Datas

Para cada XisDataElement cuja marca *controlType* é **Date** serão gerados dois controlos: uma etiqueta (controlo *Label*) e um controlo que permite escolher / seleccionar uma data, um selector de datas (controlo *DateTimePicker*). O texto da etiqueta (propriedade *text* do controlo *Label*) é preenchido com o valor da marca *text* do XisDataElement.

5.3. Geração para ASP.NET

5.3.1. A Plataforma ASP.NET

A plataforma ASP.NET [Microsoft ---a] é uma plataforma da Microsoft, que fornece um conjunto de classes disponibilizadas através da *framework* .NET e que permitem o desenvolvimento rápido de *web sites*, *web services* e aplicações *web*. A plataforma ASP.NET utiliza o protocolo http (*HyperText Transfer Protocol*). O **protocolo http** é um protocolo cliente/servidor. Neste protocolo existem dois intervenientes, o primeiro interveniente desempenha o papel de cliente e o outro, o de servidor.

Uma sessão *web* é iniciada num *browser*, através do qual o cliente efectua um pedido (*request*). Este pedido é enviado ao *site web*, que desempenha o papel de servidor e por sua vez, responde ao pedido (response). Todas as ligações entre o cliente e o servidor são mantidas, apenas durante o tempo necessário à satisfação do pedido do cliente. Existem 5 objectos que assumem grande importância no desenvolvimento de aplicações *web*, nomeadamente: (1) **Server** – objecto que representa o servidor *web*; (2) **Request** – objecto que representa um pedido http; (3) **Response** – objecto que representa a resposta do protocolo http; (4) **Session** – objecto que contém informação acerca da sessão estabelecida pelo utilizador e (5) **Application** – objecto que pode ser utilizado, para guardar informação, a partilhar por todos os utilizadores que utilizam a aplicação *web*.

As **páginas ASP.NET**, também conhecidas por *web forms*, devem possuir uma extensão *.aspx*. Os ficheiros *.aspx* podem conter *tags* html (*HyperText Markup Language*), bem como *tags* definindo controlos ASP.NET. Os **controlos** permitem representar os elementos que fazem parte da interface gráfica. Um controlo é um objecto, que pode ser colocado num espaço de interacção da interface gráfica (e.g. *TextBox*, *Label*) e que é utilizado para interagir com o utilizador. Cada tipo de controlo tem definido um conjunto de propriedades, métodos e eventos que permitem especificar o controlo tendo em conta a sua finalidade. A plataforma ASP.NET apresenta um conjunto de controlos, que se subdividem em duas categorias: (1) os *Html Server Controls* e (2) os *Web Controls*. Para a plataforma ASP.NET, os espaços de interacção são páginas *web*. Uma **página *web*** é um espaço de interacção, ao qual é possível adicionar controlos, de forma a definir determinada interface gráfica. Um **evento** é uma acção, que pode ser

desencadeada a partir da intervenção do utilizador (e.g. *click*, *keyPress*) ou a partir do sistema. Quando um evento é activado, é executado o código correspondente ao evento. Cada controlo dispõe de um conjunto de eventos, que pode ser programado. As **propriedades** e **métodos** são respectivamente, atributos e funções associados aos controlos.

5.3.2. Apresentação para ASP.NET

Nesta secção abordaremos algumas questões genéricas de apresentação das páginas *web* geradas para a plataforma ASP.NET, nomeadamente alguns aspectos relativos ao *layout* de cada espaço de interacção. No processo de geração para ASP.NET optamos por colocar e posicionar os controlos, recorrendo a ajuda de tabelas (*tag* <*table*>). Em cada linha (*tag* <*tr*>) da tabela irão ser colocados os vários elementos de interacção.

De seguida apresentamos todos os controlos que podem ser gerados para a plataforma Microsoft ASP.NET

5.3.3. Páginas Web

Para cada um dos XisInteractionSpace definidos na vista *InteractionSpace View* que foram seleccionados no processo de geração e cuja marca interactionSpaceType seja **Form**, é gerado uma página (*web page*). O título da página (*tag* <*title*>) é preenchido com o nome do projecto + hífen + nome do XisInteractionSpace. Ao gerar um XisInteractioSpace cuja marca *firstByDefault* seja true será gerada uma página que é a primeira da aplicação e que contém o nome da aplicação ao centro.

5.3.4. Controlos e Tipos de Controlos

Nesta secção apresentaremos diversos aspectos de geração de vários tipos de controlos para a plataforma Microsoft Windows ASP.NET.

Menus

Se o primeiro espaço de interacção (XisInteractionSpace cuja marca *firstByDefault* é true) contém um XisInteractionCompositeElement cuja marca *compositeElementType* é **Menu,** é gerada em todas as páginas da aplicação um menu sob a forma de árvore

(controlo *TreeView*). Este menu é construído a partir do mapa do sítio (controlo *SiteMap*). O *SiteMap* define a hierarquia da navegação numa aplicação *web*. Cada menu, por sua vez, pode conter: (1) outros XisInteractionCompositeElements cuja marca *compositeElementType* seja **Menu** ou (2) XisActionElements cuja marca *actionType* seja **MenuItem**. Ambos são gerados sob a forma de nós (controlo *SiteMapNode*). O texto de cada nó (atributo *title* do controlo *SiteMapNode*) é preenchido respectivamente com o valor da marca *text* do XisInteractionCompositeElement ou do XisActionElement. Para cada acção, ou seja, cada XisActionElement cuja marca *actionType* seja **MenuItem** é preenchido o *url* de destino (atributo *url* do controlo *SiteMapNode)* com o valor da marca *end* da XisNavigationAssociation (associação de navegação correspondente). Ou seja o atributo *url* é preenchido com a indicação do espaço de interacção destino da navegação.

Caixas de Agrupamento

Para cada XisInteractionCompositeElement cuja marca *compositeElementType* é **GroupBox** é gerada uma nova linha na tabela (*tag* <*tr*>). Dentro desta linha será gerada uma nova tabela (*tag* <*table*>). A primeira linha desta tabela conterá uma etiqueta (controlo *Label*) cujo texto (atributo *text* do controlo *Label*) é preenchido com o valor da marca *text* do XisInteractionCompositeElement. Serão geradas outras linhas nesta tabela onde serão colocados os diversos elementos contidos no XisInteractionComposisteElement.

Tabelas

Para cada XisInteractionCompositeElement cuja marca *compositeElementType* é **Table** é gerada uma nova linha na tabela (*tag* <*tr*>). Dentro desta linha será gerada uma nova tabela (*tag* <*table*>). A primeira linha desta tabela conterá uma etiqueta (controlo *Label*) cujo texto (atributo *text* do controlo *Label*) é preenchido com o valor da marca *text* do XisInteractionCompositeElement. A segunda linha conterá uma grelha (controlo *GridView*). As linhas da tabela são geradas, utilizando cores diferentes nas linhas pares e ímpares de modo a facilitar a leitura dos registos da tabela. Os botões e *links* associados à *GridView* são gerados conforme indicado no seguinte parágrafo.

Botões e Links

Para cada XisActionElement cuja marca *controlType* é **Button** ou **Link** é gerado respectivamente um botão (controlo *Button*) ou um *link* (controlo *HyperLink*). Estes são colocados numa nova célula (*tag* <*td*>) com o atributo *style* definido com o alinhamento do texto à direita (*style="text-align: right"*) com vista a posicionar os botões ou *links* do lado direito. O texto dos botões ou *links* (atributo *text* do controlo *Button* ou *Link*) é preenchido com o valor da marca *text* do XisActionElement.

Etiquetas

Para cada XisDataElement cuja marca *controlType* é **Label** é gerada uma etiqueta (controlo *Label*). O texto da etiqueta (atributo *text* do controlo *Label*) é preenchido com o valor da marca text do XisDataElement.

Caixas de Texto

Para cada XisDataElement cuja marca *controlType* é **TextBox** serão gerados dois controlos: uma etiqueta (controlo *Label*) e uma caixa de texto (controlo *TextBox*). O texto da etiqueta (atributo *text* do controlo *Label*) é preenchido com o valor da marca text do XisDataElement. Caso se tratem de XisDomainElements e tendo em conta que estes estão associados a um atributo da *Domain View,* é possível aproveitar alguma informação desta vista para o processo de geração. Nestes casos utilizamos a informação da marca *size* da XisEntityAttribute da *Domain View* para inferir o tamanho das caixas texto a gerar. Esta informação da marca *size* também é aproveitada na geração das caixas combinadas.

Caixas de Marcação

Para cada XisDataElement cuja marca *controlType* é **CheckBox** é gerada uma caixa de marcação (controlo *CheckBox*). O texto da caixa de marcação (atributo *text* do controlo *CheckBox*) é preenchido com o valor da marca *text* do XisDataElement.

Botões de Rádio

Para cada XisDataElement cuja marca *controlType* é **RadioButton** é gerado um botão de rádio (controlo *RadioButton*). O texto do botão de rádio (atributo *text* do controlo *RadioButton*) é preenchido com o valor da marca *text* do XisDataElement. Caso se tratem de XisDomainElements ou seja elementos que estão associados a um atributo da

Domain View, se este atributo for de um tipo enumerado, serão gerados tantos botões de rádio (controlo *RadioButton*) como o número de valores possíveis do tipo enumerado. Neste caso o texto de cada um dos botões de rádio (atributo *text* do controlo *RadioButton*) é preenchido com o valor da marca *value* do XisEnumerationValue correspondente.

Caixas Combinadas

Para cada XisDataElement cuja marca *controlType* é **ComboBox** serão gerados dois controlos: uma etiqueta (controlo *Label*) e uma caixa combinada (controlo *DropDownList*). O texto da etiqueta (atributo *text* do controlo *Label*) é preenchido com o valor da marca text do XisDataElement. Caso se tratem de XisDomainElements ou seja elementos que estão associados a um atributo da *Domain View*, se este atributo for de um tipo enumerado, a caixa combinada será preenchida com a lista de valores possíveis do tipo enumerado (controlo *ListItem*). O texto das caixas combinadas (atributo *text* do controlo *DropDownList*) é preenchido com o valor da marca *text* do XisDataElement.

Selectores de Datas

Para cada XisDataElement cuja marca *controlType* é **Date** serão gerados dois controlos: uma etiqueta (controlo *Label*) e um calendário (controlo *Calendar*) para escolher / seleccionar uma data. O texto da etiqueta (atributo *text* do controlo *Label*) é preenchido com o valor da marca text do XisDataElement.

5.4. Conclusão

Neste capítulo foram apresentados os aspectos relacionados com a geração de interfaces gráficas a partir do perfil XIS. Foram apresentados alguns aspectos da abordagem ProjectIT, nomeadamente os passos que antecedem o processo de geração, utilizando a ferramenta ProjectIT-Studio.

Neste trabalho de investigação criamos mecanismos de geração para as plataformas: (1) Microsoft Windows Forms.NET (ver secção 5.2) e (2) Microsoft ASP.NET (ver secção 5.3), indicando que controlos específicos para cada plataforma, são gerados a partir dos elementos de interacção genéricos representados através do perfil XIS.

Sendo o perfil XIS independente da plataforma, é possível a criar mecanismos de geração do perfil XIS para diversas plataformas computacionais específicas. Seria interessante a geração para outras plataformas que não da Microsoft (e.g, Java / Struts), bem como a geração para plataformas móveis.

6. Trabalho Relacionado

Parece existir um consenso generalizado, no que diz respeito à falta de suporte do UML para o desenho de interfaces gráficas. Algumas das dificuldades apontadas [Paton e Silva 2002] são:

- Não existe nenhum diagrama no UML que tenha a noção de *container* de vários objectos de interacção;
- Não existe nenhum diagrama no UML que forneça uma identificação gráfica dos papéis em abstracto que os diversos objectos de interacção desempenham na interface (e.g.: mostrar informação aos utilizadores, receber informação dos utilizadores, desencadear acções, ...);
- A modelação de alguns tipos de comportamento muito observados em sistemas interactivos (e.g., executar um conjunto de actividades em qualquer ordem, garantindo que cada actividade é apenas executada uma vez) é complexa, utilizando os diversos diagramas de comportamento do UML;
- Os diagramas UML tornam-se demasiado complexos, mesmo para modelar interfaces gráficas simples.

Tendo em conta estas dificuldades, o UML deve ser melhorado, no que diz respeito aos aspectos relacionados com o desenho de interfaces gráficas, recorrendo à utilização de perfis UML [OMG 1999]. A utilização de modelos UML é uma característica essencial, visto que estes fornecem uma notação precisa que pode ser utilizada na geração automática de interfaces gráficas. Neste capítulo analisamos e discutimos as seguintes quatro iniciativas baseadas na abordagem que utiliza linguagens de modelação, recorrendo à utilização de perfis UML:

- **User-Experience (UX) Modeling Language** [Kozaczynski e Thario 2002; Heumann 2003] da *Rational IBM Software Corporation*;
- **Unified Modeling Language for Interactive Applications (UML*i*)** [Silva e Paton 2003; Silva e Paton. 2003], desenvolvida pela *Universidade de Manchester*;
- **UML-based Web Engineering Approach (UWE)** [Koch e Kraus 2002; Koch 2006; Norrie 2007], do *Institute of Computer Science* da *Universidade de Munich*;
- **Wisdom Profile** [Nunes e Cunha 2000] e os **Protótipos Canónicos Abstractos** [Constantine, Windl et al. 2003; Campos e Nunes 2004], proposta pela *Universidade da Madeira*.

6.1. User-Experience Modeling Language

A abordagem ***User-Experience Modeling Language*** **(UX)**, definida pela *IBM Rational Software Corporation*, apresenta-se como sendo uma linguagem simples para descrever aspectos técnicos da interacção entre os utilizadores e as aplicações [Kozaczynski e Thario 2002; Heumann 2003]. Tendo em vista este objectivo, propõe um perfil UML que pode ser facilmente implementado, através de qualquer ferramenta de modelação que permita representar os modelos UML. Os modelos UX capturam diversos aspectos das interfaces gráficas de aplicações *web*. A linguagem UX considera que não são relevantes os aspectos relacionados com o *design*, tais como a cor, o tipo de letra utilizado ou o *layout*. Um modelo UX é um diagrama composto por diferentes elementos definidos no perfil UX UML. Os principais elementos do perfil UX UML são um conjunto de estereótipos apresentados na Tabela 6.1.

Estereótipo	Elemento estendido	Descrição
«Screen»	Classe	Representa uma abstracção de uma página *web*.
«Compartment»	Classe	Representa uma região bem definida que pode ser reutilizada por vários espaços de interacção.
«Input form»	Classe	Representa um conjunto de elementos de interacção que podem ser visualizados e manipulados pelo utilizador
«Form»	Atributo	Indica um elemento de interacção que está associado a um input form, ou seja é um campo do input form.

Tabela 6.1: Estereótipos utilizados no perfil UX UML

Os atributos do modelo podem ser representados através do estereótipo Form, que indica que um elemento de interacção está associado com um Input Form, ou seja é um campo do Input Form. Um elemento de interacção sem estereótipo representa um campo de *output*. Para os diversos elementos da linguagem UX foram definidas uma série de marcas. Associado ao estereótipo Form foram definidas algumas marcas que se encontram descritas na Tabela 6.2.

Nome	Tipo	Valor por omissão	Descrição
Editable	Boolean	True	Indica se o elemento de interacção pode (*true*) ou não (*false*) ser editado.
Source Form	String		Contém o nome do formulário de onde provém o campo.
Field Visibility	String 1-Visible / 2-Hidden	Visible	Indica se o espaço de interacção se encontra visível ou escondido.

Tabela 6.2: Marcas definidas para o estereótipo Form

Para o elemento base – "associação", já existente no UML, foi adicionada uma marca com a indicação da acção do sistema que origina a transição, ou seja, uma *string* com um ou mais nomes provenientes da área de negócio que ajudam a perceber a transição entre os espaços de interacção.

O principal modelo da linguagem de modelação UX é o *Participants Diagram*. O *Participants Diagram* permite especificar a navegação entre os diversos espaços de interacção, bem como, o conteúdo de cada um dos espaços de interacção, mas sem se preocupar com nenhum aspecto de apresentação.

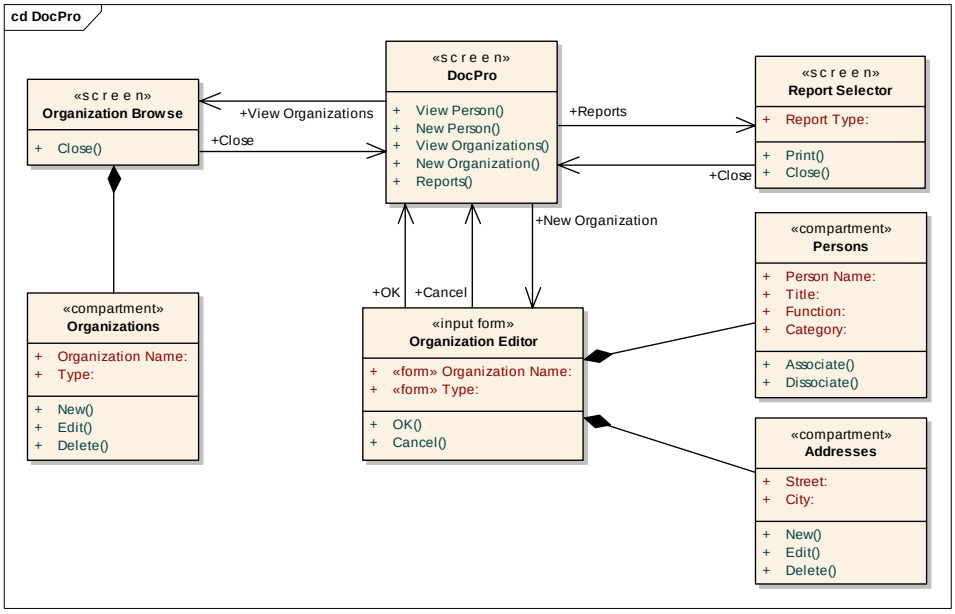


Figura 6.1: Exemplo de um *Participants Diagram* para o caso de estudo DocPro

A Figura 6.1 ilustra um exemplo de um *Participants Diagram,* representando parte do caso de estudo DocPro, através do qual podemos verificar o fluxo de navegação entre os diversos espaços de interacção. Foi utilizado um diagrama de classes com os diferentes estereótipos, Screen, Compartment e Input Form. Na segunda secção das classes encontram-se os atributos, que representam os diversos elementos que compõem o espaço de interacção. Na terceira secção das classes estão representadas as acções que podem ser desencadeadas a partir do espaço de interacção.

Ao nível dos atributos foi utilizado o estereótipo form, nomeadamente nos elementos contidos no espaço de interacção OrganizationEditor, de forma a representar os elementos que estão associados a um Input Form. As associações entre as classes representam fluxos de navegação entre os espaços de interacção. O nome a colocar na origem da associação (o nome do papel) deve ser o nome de uma das operações representadas na terceira secção das classes, de forma que seja mais fácil verificar as

transições de saída resultantes das acções do utilizador. Cada fluxo de navegação deve ter um nome único, embora possam existir dois fluxos com o mesmo nome do papel, ou seja resultantes da mesma acção, mas que podem seguir por caminhos diferentes, dando origem a dois possíveis fluxos de navegação. Por exemplo ao submeter a informação de uma nova organização poderíamos ter um fluxo que seria desencadeado quando a informação a submeter é válida e outro que era desencadeado quando a informação a submeter não é válida. Finalmente o estereótipo compartment permite representar um espaço de interacção que está contido noutro espaço de interacção.

Para cada *Participants Diagram*, podemos complementar e refinar a descrição do fluxo de navegação através da utilização de um diagrama de estados. Para representar este diagrama de estados, devemos ter em consideração que [Kozaczynski e Thario 2002]:

- Cada estado está associado a pelo menos um screen ou input form e representa o estado do fluxo no momento em que o utilizador inicia uma acção;
- Todas as transições, excepto aquelas para um ponto de escolha, devem ter o mesmo nome da associação correspondente no *Participants Diagram*;
- A transição de um estado para um ponto de escolha não tem nome;
- Ao nível da transição, a acção (e.g.: / Return to DocPro Menu) fornece o nome para a acção do sistema que suporta a transição;

Na Figura 6.2 apresentamos um exemplo de um diagrama de estados correspondente ao refinamento do *Participants Diagram* da Figura 6.1. Embora os autores indiquem que se trata de um diagrama de estados, este refinamento é representado através de um diagrama de actividades.

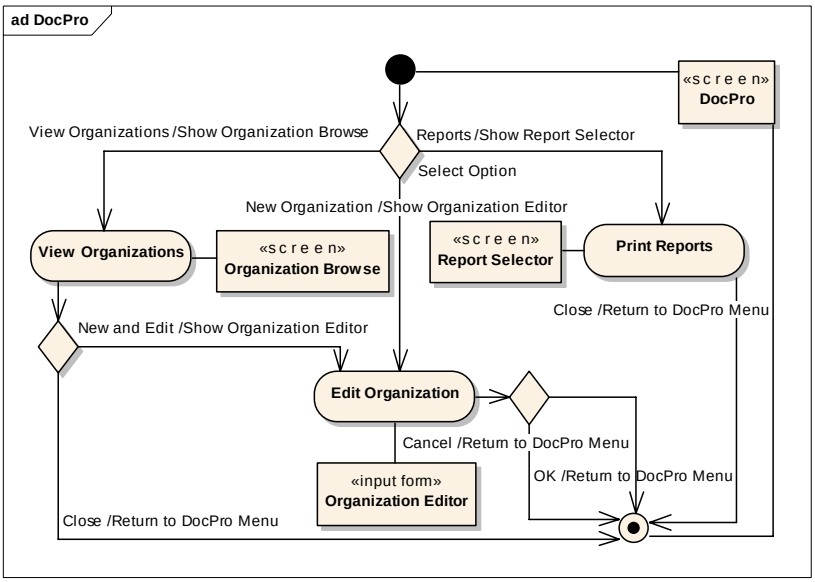


Figura 6.2: Exemplo de um diagrama de estados UX [Kozaczynski e Thario 2002]

6.2. Unified Modeling Language for Interactive Applications

A abordagem **Unified Modeling Language for Interactive Applications (UML*i*)** [Silva e Paton 2003; Silva e Paton. 2003] é baseada num projecto de investigação da *Universidade de Manchester* e consiste numa proposta para melhorar o UML nos aspectos que dizem respeito ao desenho de interfaces gráficas. Para os defensores do UML*i*, existem três tipos de modelos que permitem representar aspectos de estrutura e de comportamento dos sistemas: (1) o modelo de domínio; (2) o modelo de apresentação e (3) o modelo de comportamento.

Os **modelos de domínio** especificam as classes e objectos que representam as entidades do sistema, os elementos do domínio. Os **modelos de apresentação** representam as classes e objectos responsáveis pelo conteúdo de cada espaço de interacção. Os **modelos de comportamento** descrevem as propriedades dos elementos de comportamento (e.g.: tarefas, acções, eventos), ou seja, os elementos utilizados para alterar o estado dos elementos estruturais. Em relação aos modelos de domínio é pacífica a utilização de diagramas de classes. Em relação aos modelos de apresentação, podem também, ser construídos utilizando diagramas de classes, sendo proposto um novo diagrama, o Diagrama de Interface com o Utilizador (*User Interface Diagram*). Foram definidos um conjunto de estereótipos para utilizar no Diagrama de Interface com o Utilizador e que se encontram descritos na Tabela 6.3.

Estereótipo	Icon	Elemento estendido	Descrição
«Free Container»		Classe	Elemento de interacção de topo que não pode estar contido em nenhum outro elemento de interacção.
«Container»		Classe	Elemento de interacção que pode conter outros elementos de interacção, excepto FreeContainers.
«Inputer»		Classe	Elemento de interacção responsável por receber informação dos utilizadores.
«Displayer»		Classe	Elemento de interacção responsável por enviar informação visual aos utilizadores.
«Editor»		Classe	Elemento que é simultaneamente Inputer e Displayer.
«Action Invoker»		Classe	Elemento de interacção responsável por receber informação dos utilizadores sob a forma de eventos.

Tabela 6.3: Estereótipos utilizados no diagrama de interface com o utilizador UML*i*

Na Figura 6.3 é apresentado um exemplo de um *Diagrama de Interface com o Utilizador* representando o Free Container OrganizationEditor do caso de estudo DocPro.

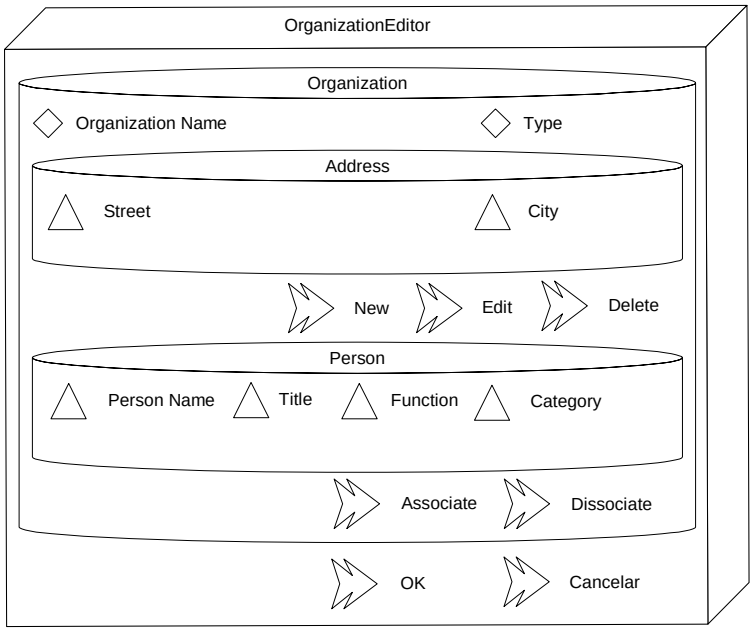


Figura 6.3: Modelo de apresentação, utilizando o diagrama de interface com o utilizador

Para modelar o comportamento é defendida a utilização de diagramas de actividades. A modelação do comportamento é simplificada na linguagem UML*i*, através de alguns mecanismos de extensão. Uma combinação de transições, difusões (*forks*) e junções (*joins*) é adequada para relacionar actividades que podem ser executadas em paralelo. Uma combinação de transições com diferentes ramos é adequada para modelar uma situação em que apenas uma de muitas actividades é executada, em que existe uma decisão. Contudo, para modelar aplicações interactivas, existem situações nas quais estas combinações de tão baixo nível originam a construção de modelos complexos. Segundo os autores, existem três tipos de comportamento comuns em aplicações interactivas:

(1) Comportamento de selecção por ordem independente, ilustrado na Figura 6.4:

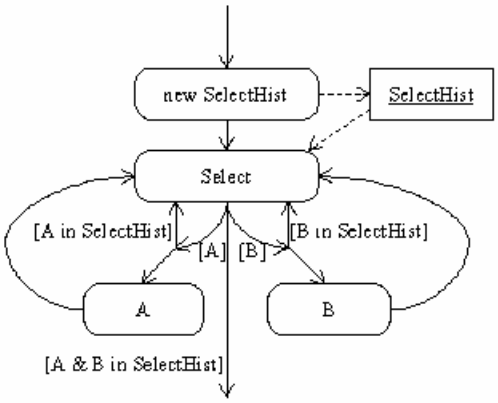


Figura 6.4: Modelação UML, representando o tipo de comportamento de selecção por ordem independente (extraído de [Silva e Paton 2000])

Na Figura 6.4, as actividades A e B podem ser seleccionadas e activadas por qualquer ordem e a pedido dos utilizadores que interagem com o sistema, mas apenas podem ser executadas uma vez. Este tipo de comportamento pode ser composto, por uma ou mais actividades que podem ser seleccionadas.

(2) O comportamento opcional que podemos ver na Figura 6.5:

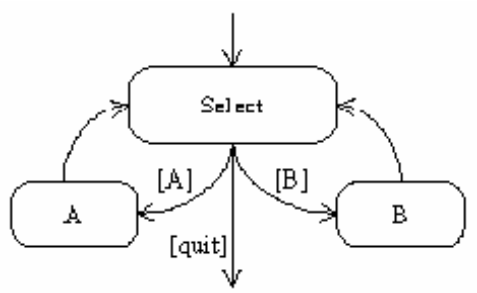


Figura 6.5: Modelação UML, representando o tipo de comportamento opcional (extraído de [Silva e Paton 2000])

Neste tipo de comportamento, os utilizadores que interagem com o sistema podem seleccionar qualquer uma das actividades, quantas vezes quiserem, inclusive nenhuma vez. Fica a cargo do utilizador indicar quando termina a selecção e quando pretende continuar. Este tipo de comportamento pode ser composto por uma ou mais actividades que podem ser seleccionadas.

(3) O comportamento de repetição que podemos ver na Figura 6.6:

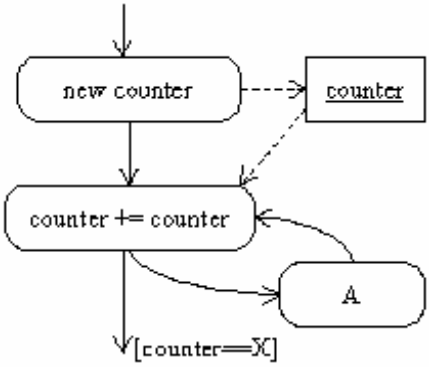


Figura 6.6: Modelação UML, representando o tipo de comportamento de repetição (extraído de [Silva e Paton 2000])

Este tipo de comportamento é composto, apenas por uma actividade. A actividade A será repetida várias vezes conforme especificado. No caso do exemplo da Figura 6.6, a actividade A será repetida X vezes.

Facilmente verificamos que a repetição destes diversos comportamentos, pode conduzir a modelos extremamente complexos e de difícil leitura. Como se tratam de comportamentos que são frequentes em sistemas interactivos, a linguagem UML*i* propõe uma simplificação da notação para representar estes comportamentos e que se encontram ilustrados na Figura 6.7.

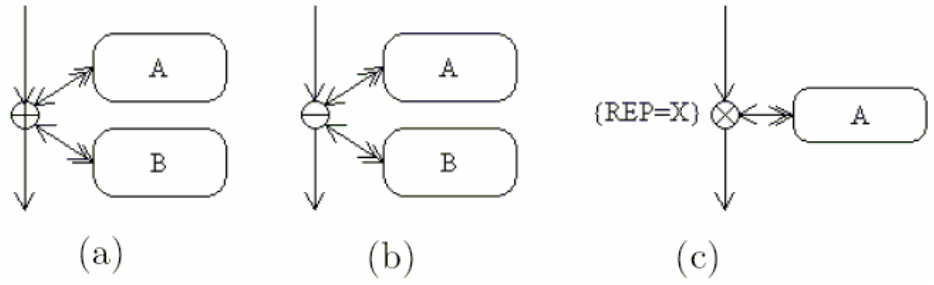


Figura 6.7: Modelação UML*i*, representando os tipos de comportamento: (a) de selecção por ordem independente; (b) opcional e (c) de repetição (extraído de [Silva e Paton 2000])

A notação para representar um comportamento de selecção por ordem independente encontra-se ilustrada na Figura 6.7 em (a). É utilizada a representação gráfica de um círculo contendo um sinal de mais (+), conectado com as actividades A e B através de transições, representadas através de uma linha com uma seta simples do lado da selecção ⊕ e uma seta dupla do lado do lado da actividade a seleccionar. A notação para representar um comportamento opcional encontra-se ilustrada na Figura 6.7 em (b). É utilizada a representação gráfica de um círculo contendo um sinal de menos (-), conectado com as actividades A e B através de transições, representadas através de uma linha com uma seta simples do lado da selecção ⊖ e uma seta dupla do lado da actividade a seleccionar. A notação para representar um comportamento de repetição encontra-se ilustrada na Figura 6.7 em (c). É utilizada a representação gráfica de um círculo contendo um sinal de multiplicar (X), conectado com as actividades A e B através de transições, representadas através de uma linha com uma seta simples do lado da selecção ⊗ e uma seta dupla do lado do lado da actividade a seleccionar. O selector de repetição ⊗ necessita de um elemento adicional, uma restrição REP que indica quantas vezes deve ser repetida a actividade. Estas notações do UML*i* podem ser consideradas como macro-notações ou notações de mais alto nível com vista a simplificar os modelos. É possível ver exemplos de diagramas de actividades UML*i* que utilizam esta notação simplificada em [Silva e Paton 2000].

6.3. UML-based Web Engineering Approach

O **UML-based Web Engineering Approach (UWE)** [Koch e Kraus 2002; Koch 2006; Norrie 2007] é um projecto do *Institute of Computer Science* da *Universidade de Munich* e tem como objectivo o desenho de interfaces gráficas para aplicações Web, utilizando uma extensão UML com mecanismos de *sketching*. O UWE define três modelos essenciais: (1) o modelo conceptual; (2) o modelo de navegação e (3) o modelo de apresentação. O **modelo conceptual** é representado através de um diagrama de classes e consiste na representação do domínio, identificação de classes e suas associações. O **modelo de navegação** representa os espaços de navegação e os elementos de acesso que podem ser utilizados para a navegação. **O modelo de apresentação** tem como objectivo o desenho em abstracto de cada espaço de interacção.

Para representar o modelo de navegação, foi definido um conjunto de estereótipos que se encontram descritos na Tabela 6.4.

Estereótipo	Elemento estendido	Descrição
«Navigation Class»	Classe	Modela uma classe cujas instâncias são visitadas pelo utilizador durante a navegação.
«Index»	Classe	É um elemento de acesso. Representa um objecto composto, que contém um número indeterminado de itens. Cada item, por sua vez é um objecto com um nome e um *link* para uma instância de uma Navigation Class.
«Guided Tour»	Classe	É um elemento de acesso. Representa um objecto que fornece acesso sequencial, às instâncias de uma Navigation Class.
«Query»	Classe	É um elemento de acesso. Representa um objecto para pesquisa, que tem uma *string* de *query* como atributo.
«Menu»	Classe	Representa um objecto composto, que contém vários de itens de menu. Cada item tem um nome e um link, para uma instância de uma Navigation Class ou para um elemento de acesso.
«Navega-bilidade Directa»	Associação	As associações, no modelo de navegação, indicam uma navegabilidade directa da Navigation Class de origem para a Navigation Class de destino.

Tabela 6.4: Estereótipos utilizados no modelo de navegação

Os estereótipos anteriormente descritos resumem-se na Figura 6.8, que ilustra o metamodelo dos elementos do modelo de navegação.

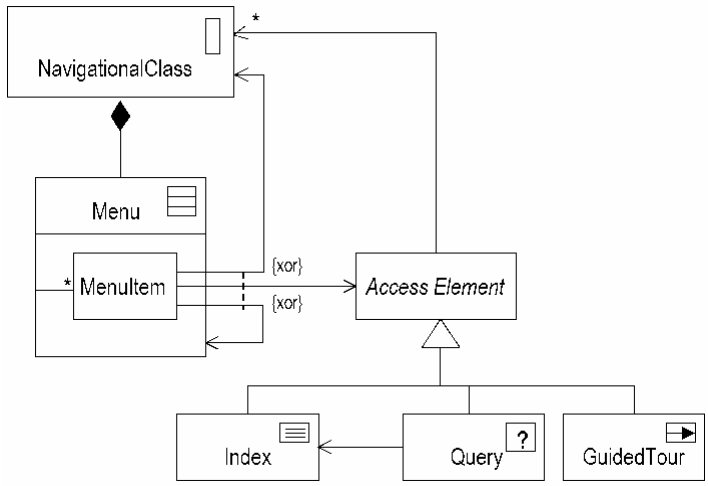


Figura 6.8: Metamodelo do modelo de navegação (extraído de [Hennicker e Koch 2001])

Na Figura 6.9 apresentamos um exemplo de um modelo de navegação, para o caso de estudo DocPro, onde se indica, que a Navigation Class DocPro contém um menu com cinco itens de menu: (View Person, New Person, View Organization, New Organization e Reports). Através destes itens de menu podemos navegar para as restantes Navigation Classes do sistema (Person Browse, Person Editor, Organization Browse, Organization Editor e Report Selector).

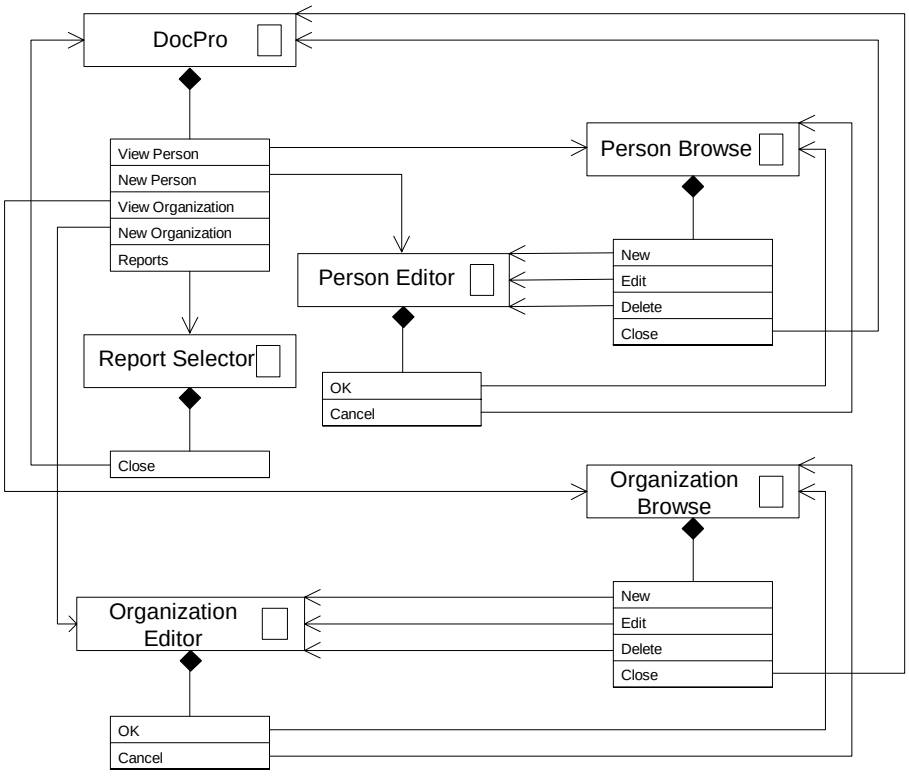


Figura 6.9: Exemplo do modelo de navegação do caso de estudo DocPro

Segundo [Koch e Kraus 2002], a etapa seguinte é construir o modelo de apresentação. Este modelo tem como objectivo, o desenho em abstracto de cada espaço de interacção. O principal diagrama do modelo de apresentação é o *Abstract User Interface Model*. Para a construção deste modelo são propostos os estereótipos apresentados na Tabela 6.5.

Estereótipo	Elemento estendido	Descrição
«UI View»	Classe	User Interface View – representa um contentor de todos os elementos abstractos de interacção, que são apresentados ao utilizador ao mesmo tempo, em determinado momento numa mesma janela.
«Presentation Class»	Classe	É uma unidade estrutural que permite partir uma UI view em grupos de elementos de interacção.
«UI Element»	Classe	User Interface Element - Classe abstracta que tem várias especializações, que descrevem elementos de interacção particulares.
«Text»	Classe	Subclasse da classe UI element. Sequência de caracteres conjuntamente com informação de formatação.
«Form»	Classe	Subclasse da classe UI element. Utilizado para pedir informação ao utilizador. Podem representar campos de input, checkboxes,...
«Anchor»	Classe	Subclasse da classe UI element. Uma Anchor tem uma apresentação sob a forma de texto, imagem, vídeo ou outro elemento interactivo, estando estes associados a um link para navegação.
«Image» **«Video»** **«Áudio»**	Classe	Subclasses da classe UI element, que representam objectos multimédia.
«Collection»	Classe	Subclasse da classe UI element. Representa um conjunto consistente de objectos de interacção.
«Anchored Collection»	Classe	Subclasse da classe UI element. Representa um conjunto de Anchors.

Tabela 6.5: Estereótipos utilizados no modelo abstracto do espaço de interacção.

O *Abstract User Interface Model* mostra o conteúdo e a estrutura de cada nó, de cada espaço de interacção. Para descrever modelos de apresentação há quem utilize técnicas de *sketching*, mas sem utilizar nenhuma notação precisa. Em [Koch e Kraus 2002] é proposta uma extensão ao UML com vista ao desenho abstracto de cada espaço de interacção. O *Abstract User Interface Model* permite representar apenas a estrutura de cada nó e nunca características relacionadas com a criatividade, como o tipo de letra, cor e outros. Contudo, o desenho abstracto fornece algumas pistas no que diz respeito ao tamanho e posicionamento relativo dos elementos de interacção.

A Figura 6.10 ilustra o metamodelo do *Abstract User Interface Model*.

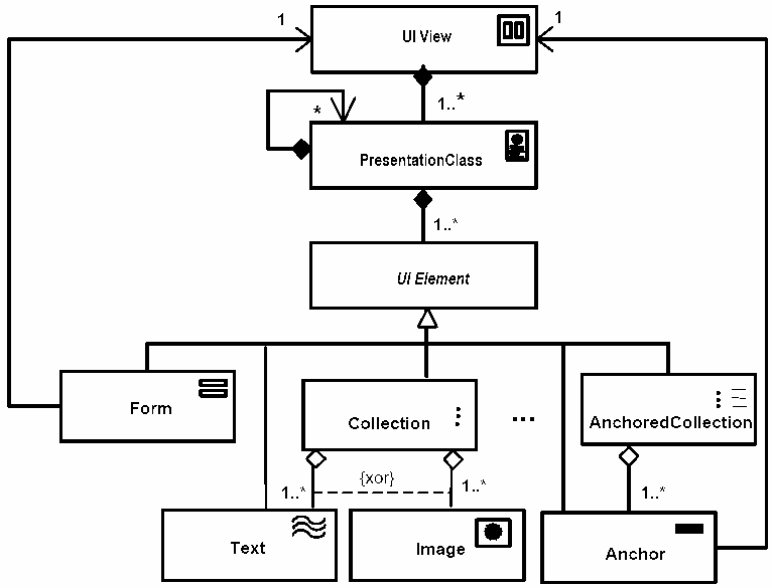


Figura 6.10: Metamodelo do *Abstract User Interface Model* (extraído de [Hennicker e Koch 2001])

A Figura 6.11 ilustra uma vista de um espaço de interacção, que é uma parte do *Abstract User Interface Model*. Trata-se da Presentation Class Organization Editor

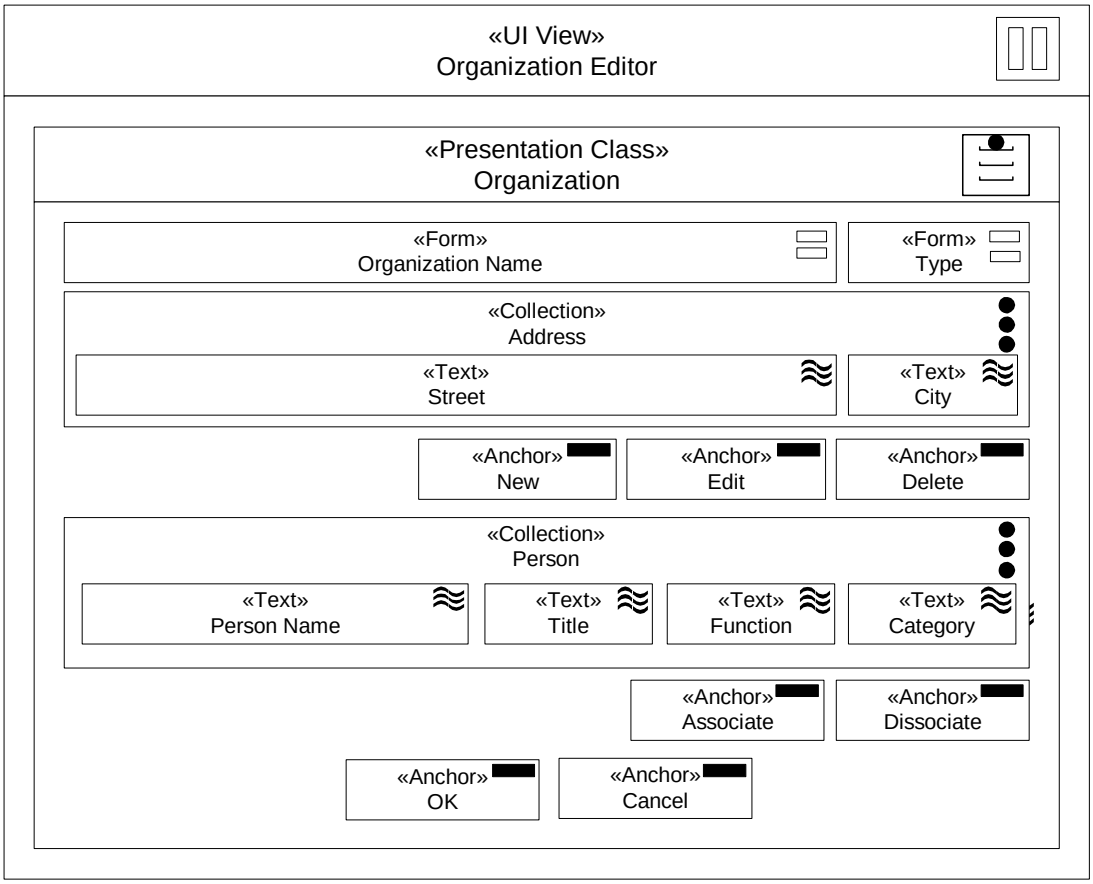


Figura 6.11: Exemplo de uma UI View - *Abstract User Interface Model* para o caso de estudo DocPro

Após o desenho das diferentes vistas, podem ser desenvolvidos cenários que mostram a sequência de navegação entre as vistas de uma forma mais intuitiva do que a que é obtida através do modelo de navegação. Na Figura 6.12 temos um exemplo de um cenário que descreve a navegação entre diferentes vistas da interface gráfica para o caso de estudo DocPro. Este modelo chama-se *Storyboard Model*.

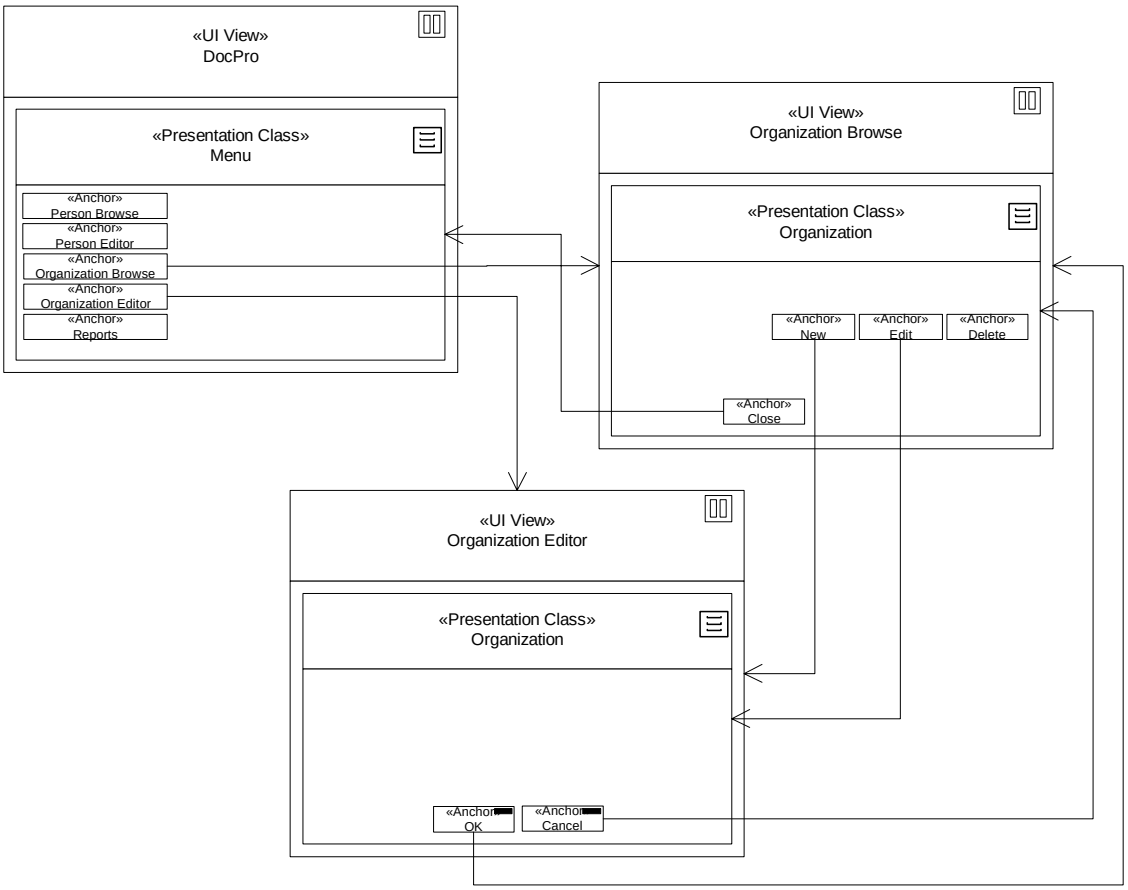


Figura 6.12: Cenário representado através de um *Storyboard Model*

6.4. Wisdom UML Profile e Protótipos Canónicos Abstractos

O **Wisdom UML Profile** [Nunes e Cunha 2000] é uma proposta de perfil UML para a documentação, especificação e desenho de sistemas interactivos. Este perfil, proposto pela *Universidade da Madeira*, utiliza e estende a notação e semântica do UML, beneficiando do conhecimento existente na área de interacção pessoa-máquina (HCI). Os criadores do perfil Wisdom (Whitewater Interactive System Development with Object Models) também entendem, que embora o UML se esteja a expandir cada vez mais em diferentes domínios de aplicação, tornando-se numa linguagem standard de facto para a análise e desenho de sistemas orientados por objectos, este continua a ser insuficiente, para suportar a análise e desenvolvimento de sistemas interactivos.

A abordagem Wisdom propõe vários modelos, entre os quais, o modelo de análise que define a arquitectura interna do sistema e o modelo de interacção que define a arquitectura do interface com o utilizador. Na fase de desenho, o modelo de interacção é refinado em dois modelos: (1) o modelo de diálogo e (2) o modelo de apresentação. O modelo de diálogo especifica a estrutura de diálogo de uma aplicação, utilizando uma adaptação baseada em UML, da notação ConcurTaskTrees (CTT) [Paternò 1999].

Estereótipo	Elemento estendido	Descrição
«Interaction Space»	Classe	Responsável por receber e apresentar informação aos utilizadores, servindo de suporte para as suas tarefas.
«Navigates»	Associação	Associa dois «interaction space», indicando que o utilizador se movimenta de um «interaction space», para o outro. Esta associação pode ser unidireccional ou bidireccional.
«Contains»	Associação	Associa dois «interaction spaces», indicando que o «interaction space» de origem contém o «interaction space» de destino. Está associação apenas pode ser unidireccional.
«Input Element»	Atributo	Indica informação recebida a partir do utilizador, informação que pode ser manipulada.
«Input Collection»	Atributo	Indica um conjunto relacionado de elementos de input - «input element»
«Output Element»	Atributo	Indica informação apresentada ao utilizador, informação que não pode ser manipulada.
«Output Collection»	Atributo	Indica um conjunto relacionado de elementos de *output* - «output element»
«Action»	Operação	Representa uma operação que o utilizador pode realizar na interface e que provoca uma alteração significativa no sistema.

Tabela 6.6: Estereótipos utilizados no modelo de apresentação Wisdom

O **modelo de apresentação** define a forma como as diferentes entidades de apresentação são estruturadas para realizar a interacção física com o utilizador. A Tabela 6.6 apresenta a lista de estereótipos a utilizar no modelo de apresentação.

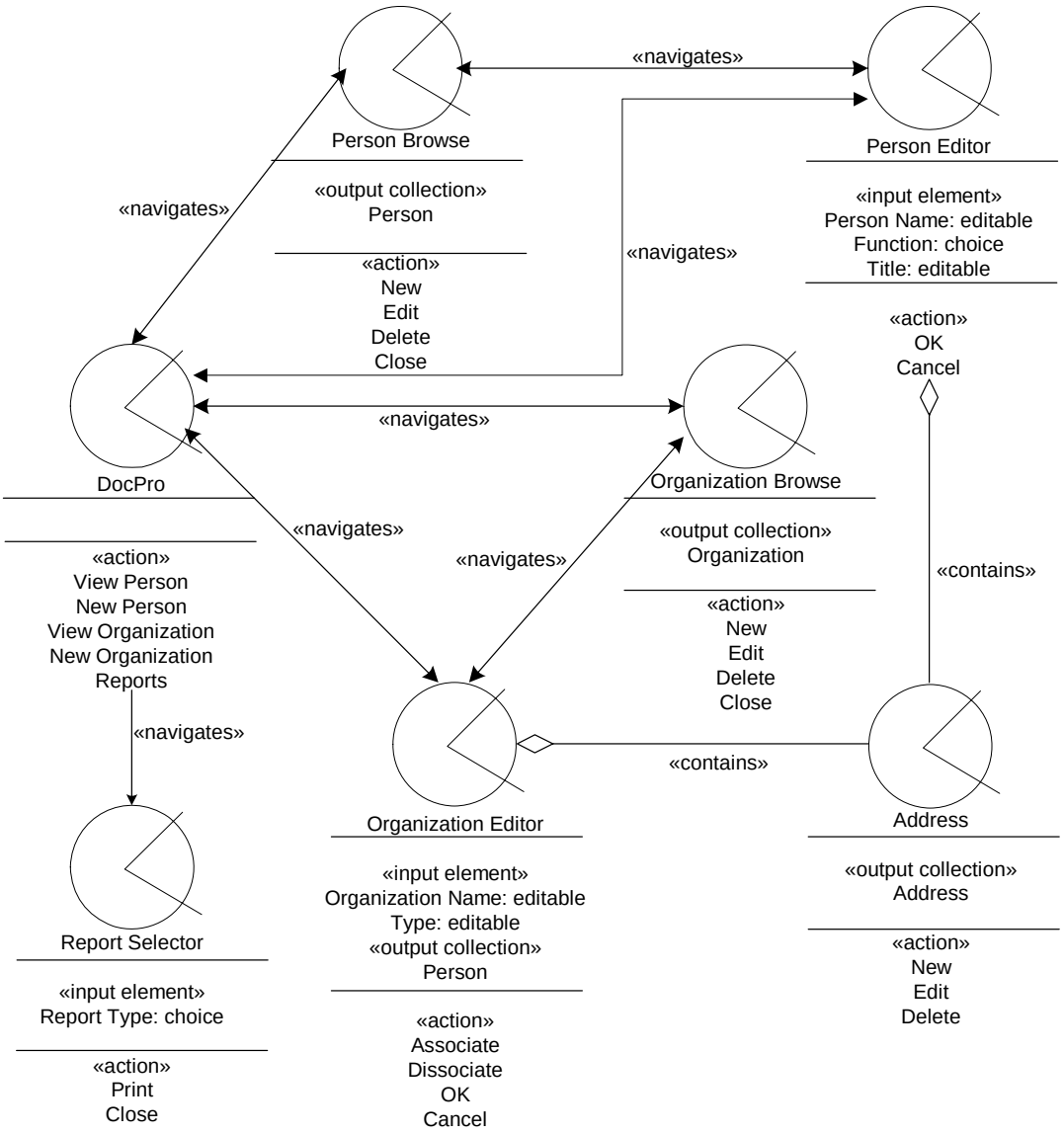


Figura 6.13: Exemplo de um modelo de apresentação Wisdom, representando parte do caso de estudo DocPro.

A Figura 6.13 ilustra um exemplo de um modelo de apresentação Wisdom, representando parte do nosso caso de estudo DocPro. Através da utilização do estereótipo «navigates», podemos verificar as diferentes possibilidades de navegação entre os diferentes espaços de interacção do sistema. Através da utilização do estereótipo «contains», podemos verificar que o espaço de interacção Address está contido nos espaços de interacção Person Editor e Organization Editor.

Após a representação do modelo de apresentação Wisdom, este poderá ser mapeado para um ***Protótipo Canónico Abstracto***. Os *Protótipos Canónicos Abstractos* [Constantine, Windl et al. 2003] são o resultado de um método para o desenvolvimento rápido de protótipos para interfaces gráficas, desenvolvido pela empresa *Constantine & Lockwood Ltd*, conjuntamente com alguns clientes. Um *Protótipo Canónico Abstracto* [Constantine, Windl et al. 2003] é um protótipo abstracto, construído a partir de um conjunto de componentes, utilizando uma notação abstracta canónica. Um protótipo abstracto permite descrever os conteúdos e organização geral de um espaço de interacção, incluindo tamanho e posicionamento relativo, sem detalhar a sua aparência ou o seu comportamento. Da experiência obtida pela empresa *Constantine & Lockwood Ltd*, chegou-se à conclusão [Constantine, Windl et al. 2003] de que para os designers inexperientes, a utilização de protótipos abstractos, parecia conduzir a melhores resultados do que aqueles obtidos sem a sua utilização, enquanto que para os designers mais sofisticados e experientes, a utilização de protótipos abstractos fazia aumentar a criatividade, conduzindo a soluções mais inovadoras.

Da mesma forma que os diversos ambientes de desenvolvimento e desenho de interfaces gráficas fornecem um conjunto de ferramentas standard, que podem ser seleccionadas, existe um conjunto de componentes abstractos que podem ser utilizados para construir os protótipos abstractos. A notação simbólica destes componentes abstractos é definida a partir de dois símbolos genéricos e extensíveis: (1) um material ou contentor genérico que representa conteúdo, informação ou elementos de interacção que podem ser manipulados ou apresentados ao utilizador e que é representado graficamente por um quadrado □ e (2) uma ferramenta ou acção genérica que representa um operador ou mecanismo que pode ser utilizado para manipular ou transformar os materiais e que é representado graficamente por uma seta ↗. Um terceiro componente, genérico designado por híbrido ou material activo é obtido através da combinação dos dois componentes definidos anteriormente e representam qualquer componente com características de material e de ferramenta, (e.g., um material que consiste num elemento de interacção que apresenta conteúdo e que pode ser editado. O material activo é representado graficamente pela combinação de um quadrado e uma seta ↗□.

A Figura 6.14 ilustra a lista completa da notação abstracta canónica [Constantine, Windl et al. 2003], utilizada para construir protótipos canónicos abstractos, que como já vimos encontra-se dividida em três grupos: (1) materiais canónicos abstractos; (2)

ferramentas canónicas abstractas e (3) materiais activos canónicos abstractos. O primeiro elemento de cada grupo, que está marcado com '*' é um elemento genérico, sendo os restantes elementos do grupo especializações. Todas as operações são especializações da operação genérica action/operation*, todos os materiais são especializações do material genérico container* e todos os materiais activos são especializações do material activo genérico active material*.

SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
	action/operation*	Print symbol table, Color selected shape
	start/go/to	Begin consistency check, Confirm purchase
	stop/end/complete	Finish inspection session, Interrupt test
	select	Group member picker, Object selector
	create	New customer, Blank slide
	delete, erase	Break connection line, Clear form
	modify	Change shipping address, Edit client details
	move	Put into address list, Move up/down
	duplicate	Copy address, Duplicate slide
	perform (& return)	Object formatting, Set print layout
	toggle	Bold on/off, Encrypted mode
	view	Show file details, Switch to summary

SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
	container*	Configuration holder, Employee history
	element	Customer ID, Product thumbnail image
	collection	Personal addresses, Electrical Components
	notification	Email delivery failure, Controller status

SYMBOL	INTERACTIVE FUNCTION	EXAMPLES
	active material*	Expandable thumbnail, Resizable chart
	input/accepter	Accept search terms, User name entry
	editable element	Patient name, Next appointment date
	editable collection	Patient details, Text object properties
	selectable collection	Performance choices, Font selection
	selectable action set	Go to page, Zoom scale selection
	selectable view set	Choose patient document, Set display mode

Figura 6.14: Notação Canónica Abstracta completa

A Figura 6.15 ilustra um exemplo de um *Protótipo Canónico Abstracto*, correspondente ao espaço de interacção Organization Editor, para o caso de estudo DocPro. Através dos *Protótipos Canónicos Abstractos* podemos modelar aspectos relativos ao posicionamento, tamanho, *layout* e composição de cada espaço de interacção.

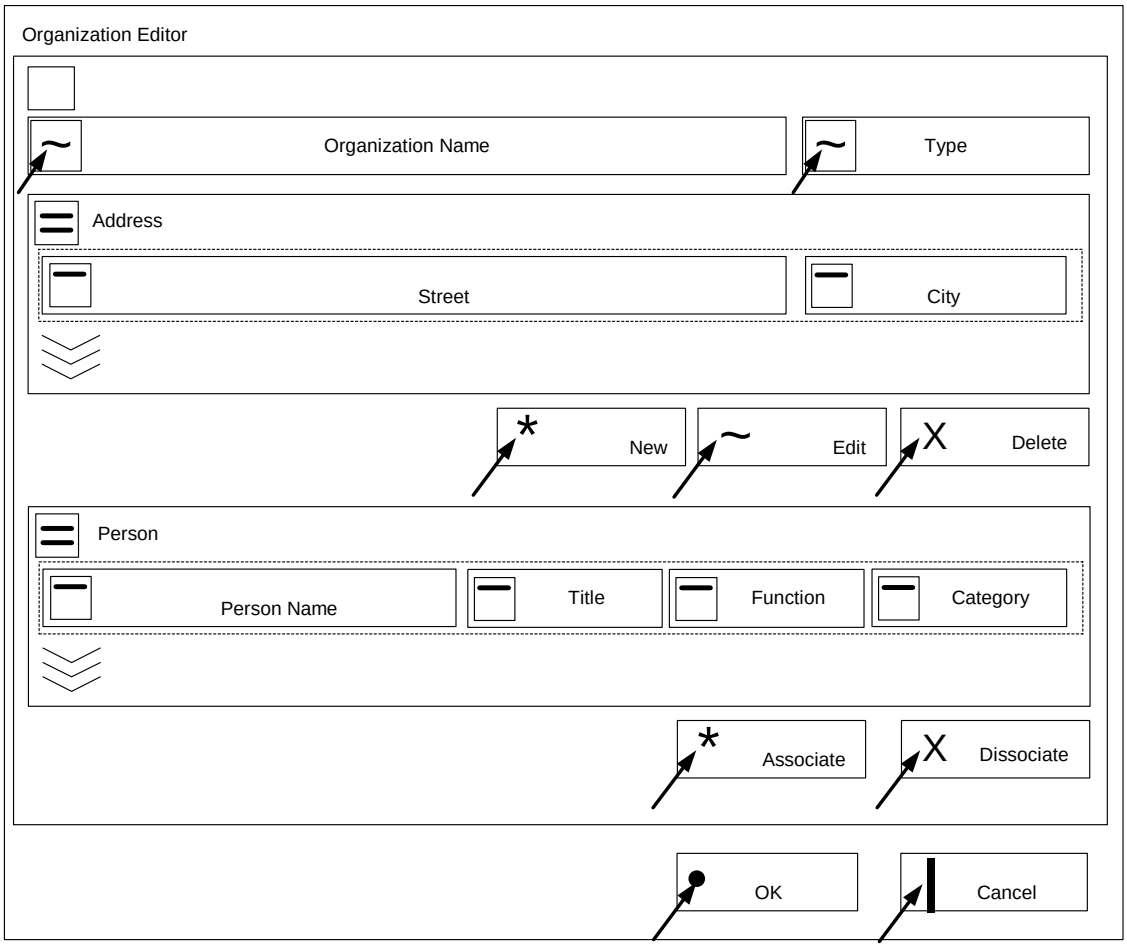


Figura 6.15: Exemplo de um *Protótipo Canónico Abstracto* para o espaço de interacção Organization Editor do caso de estudo DocPro

O objectivo da parceria *Wisdom / Protótipos Canónicos Abstractos* é passar a detalhar em pormenor, a estrutura da apresentação de cada um dos espaços de interacção, obtendo um nível de desenho mais refinado. Pretende-se criar uma semântica comum e sincronizada, de modo a suportar as diversas fases no processo de desenho de interfaces gráficas, complementando as fraquezas de um modelo com as vantagens do outro modelo. Para atingir este objectivo, é necessário especificar um mapeamento [Campos e Nunes 2004] entre o modelo de apresentação *Wisdom* e os *Protótipos Canónicos Abstractos*. Este mapeamento permitirá representar alguns aspectos relacionados com a apresentação, decorrentes da utilização do modelo de apresentação Wisdom, tais como a representação da posição, tamanho, *layout* e composição e permitirá adicionar algum formalismo necessário na notação canónica abstracta. A Figura 6.16 ilustra a proposta de especificação para um possível mapeamento entre os estereótipos do modelo de apresentação *Wisdom* e os componentes canónicos abstractos. Através deste mapeamento, é possível especificar a estrutura geral da

interface gráfica, refinando o modelo de apresentação Wisdom em um ou vários contextos de interacção canónicos, expandindo e detalhando o modelo no que diz respeito a aspectos relacionados com a apresentação.

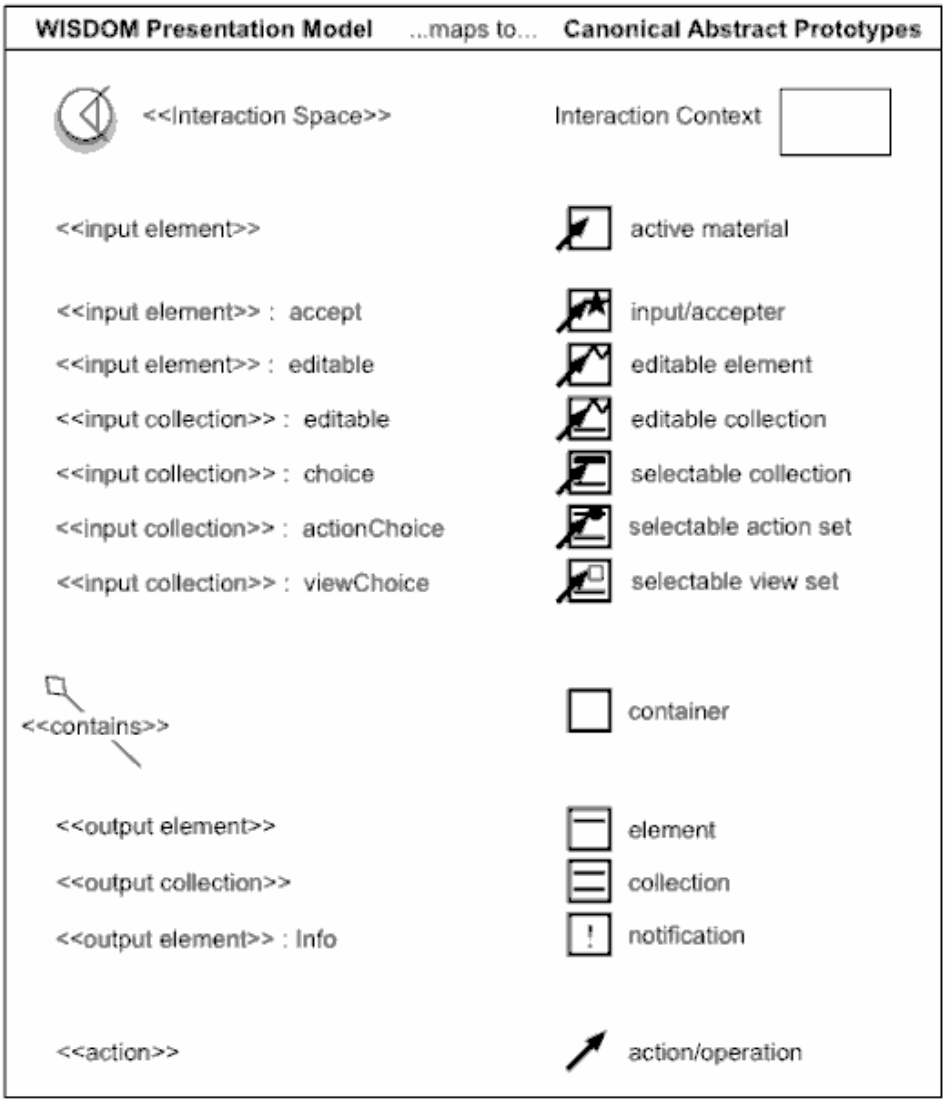


Figura 6.16: Correspondência entre estereótipos Wisdom e os componentes Canónicos Abstractos

Para apoiar e ajudar a desenvolver de forma prática este projecto, foi desenvolvida a ferramenta *CanonSketch* com vista a suportar o desenho de *Protótipos Canónicos Abstractos* a partir do modelo de apresentação Wisdom. Para obter mais pormenores sobre esta ferramenta deverá ser consultado [Campos e Nunes 2004].

Em [Costa 2007] foram definidos mecanismos de geração de código, para um contexto de desenvolvimento Web, para uma plataforma suportada pela linguagem PHP e pela *framework* Hydra

6.5. Análise Comparativa

Existem vários aspectos que são considerados importantes, quando se trata de modelar interfaces gráficas [Anderson 2000]:

- Os modelos devem ser facilmente descritos à mão, isto é, sem o recurso a nenhuma ferramenta electrónica, com vista a facilitar a troca de ideias;
- Se os elementos de interacção forem representados por ícones, estes devem ser suficientemente claros, devem comunicar o seu significado, devem facilitar a identificação em abstracto do papel desempenhado pelo elemento que estão a representar e devem ser suficientemente distintos de forma a evitar confusões e mal entendidos;
- Deve ser possível, destacar visualmente a noção de conteúdo entre os vários elementos que compõem a interface gráfica;
- Devem existir ferramentas de modelação que suportem com facilidade a notação proposta;
- Devem existir modelos que permitam representar a navegação entre os diferentes espaços de interacção, bem como, a representação de acções que podem alterar o estado ou o comportamento de um elemento de interacção, mas que não resultam necessariamente, na navegação para outro espaço de interacção;
- Também é importante a existência de modelos para representar o conteúdo dos espaços de interacção. Constantine & Lockwood designa-o por modelo de conteúdo e define-o como sendo um modelo que representa em abstracto com alguma fidelidade e precisão o entendimento de como os materiais (elementos da interface gráfica) e as ferramentas (elementos que permitem invocar acções) são distribuídos pelo espaço de interacção [Constantine e Cockwood 1999]. Estes modelos do conteúdo do espaço de interacção, permitem representar a estrutura e a organização geral de cada espaço de interacção, sem que exista qualquer compromisso na escolha de um controlo GUI em particular.

A Tabela 6.7 resume os principais conceitos do perfil XIS, em comparação com as propostas analisadas neste capítulo. A Tabela 6.8 apresenta resumidamente uma análise comparativa das características do perfil XIS, em comparação com as propostas analisadas neste capítulo.

Iniciativa Item	UX	UML*i*	UWE	Wisdom	XIS
Modelação do Domínio	Não	Domain Model	Conceptual Model	Domain Model	Entities View
Modelação da Navegação	Participants Diagram, State Machine	Não	Navigation Model, Storyboard Model	Presentation Model	NavigationSpace View
Relações de Navegação	Associação Transição	Não	Navegabilidade Directa	«Navigates»	«XisNavigation Association»
Modelação do Espaço de Interacção	Participants Diagram (apenas conteúdo)	User Interface Diagram	Abstract User Interface Model	Canonical Abstract Prototypes	InteractionSpace View
Noção de Contentor	«Screen» «Compartment»	«Free Container» «Container»	«UI View» «Presentation Class» «… collection»	«Interaction Space» «Contains» «… collection»	«XisInteraction Space» «XisInteraction CompositeElement»
Conteúdo do Espaço de Interacção	Sim	Sim	Sim	Sim	Sim
Apresentação do Espaço de Interacção	Não	Sim	Sim	Sim	Sim
Modelação do Comportamento	Modelação da Navegação. Elementos Editable (*True* ou *False*)	Diagrama de Actividades. «Inputter» «Displayer» «Editor»	Modelação da Navegação. «Text» «Form»	Modelação da Navegação. Utilização de CTTs	Modelação da Navegação. Elementos visíveis e activos por actor

Tabela 6.7: Comparação dos conceitos de modelação, para as diversas iniciativas baseadas na abordagem de produção de interfaces gráficas, utilizando linguagens de modelação

Iniciativa Item	UX	UML*i*	UWE	Wisdom	XIS
Facilmente descritos à mão	+	+	-	-	+
Visualmente claros / Facilidade de leitura	+	-	+	-	+
Especificação aspectos de apresentação (tamanho e posicionamento)	- -	+	+	+	+
Utilização de modelos PIM	+ (*web-based*)	+ +	+ (*web-based*)	+ +	+ +
Transformação de modelos-para-modelos	- - Não	- - Não	+ Gerar o Story board Model a partir do Navi-gation Model e do Abstract User Interface Model	+ Geração dos Protótipos Canónicos Abstractos a partir do modelo de apresentação	+ + Geração da User-Interface View a partir da BusinessEntities View e da UseCases View
Transformação de modelos-para-código	+ Geração para plataformas *Web*. Java Struts Framework	+ Possibilidade de geração para diversas plataformas	- Possibilidade de geração para diversas plataformas *Web*	+ + Geração para diversas plataformas. Framework Hydra – PHP	+ + Geração para diversas plataformas WinForms.NET e ASP.NET

Tabela 6.8: Comparação das características das diversas iniciativas, baseadas na abordagem de produção de interfaces gráficas, utilizando linguagens de modelação

A iniciativa **UX** define um perfil que permite o desenho e modelação de aplicações *web*, nomeadamente no que diz respeito a aspectos de navegação e discute as transformações de modelos UML UX para código, especialmente para a *framework* Java Struts. A iniciativa UX utiliza diagramas simples, facilmente descritos à mão. É possível destacar a noção de contentor, através da utilização dos estereótipos *Screen* e *Compartment*.

Os digramas UX representam bem os aspectos de navegação de forma semelhante ao proposto na *NavigationSpace View* do XIS, mas não definem nenhum modelo para representar de forma abstracta cada espaço de interacção, como é proposto no XIS através da *InteractionSpace View*. A partir do *Participants Diagram* é possível representar os elementos de interacção contidos em cada espaço de interacção, mas não conseguimos representar a sua apresentação. A não existência de um modelo abstracto que represente cada espaço de interacção, impossibilita a inferência a partir dos modelos UX de aspectos relacionados com o tamanho e posicionamento relativo dos elementos de interacção.

Relativamente aos aspectos de comportamento, é possível representar a navegação entre os diferentes espaços de interacção. Para cada espaço de interacção é possível indicar, através da marca *Editable,* se um elemento de interacção é de *input* ou de *output*, mas não é possível alterar o seu estado / comportamento para diferentes momentos ou contextos da interacção.

No que diz respeito aos aspectos de geração, embora não se tenha encontrado nenhuma referência a este tipo de transformação, parece ser possível desenvolver mecanismos de geração de modelo-para-modelo, com vista a gerar o diagrama de estados a partir do *Participants Diagram*. É possível a geração dos modelos UX para código, nomeadamente para diversas plataformas *web*, tendo sido desenvolvida a geração para a *framework* Java Struts.

A iniciativa **UML*i*** propõe um perfil que permite representar aspectos conceptuais, de apresentação e de comportamento dos sistemas, nomeadamente através da utilização de três tipos de modelos principais: (1) o modelo de apresentação; (2) o modelo de domínio e (3) o modelo de comportamento. Os modelos UML*i* são facilmente descritos à mão, são visualmente claros e de fácil leitura, com excepção do digrama de actividades, utilizado para modelar o comportamento.

A iniciativa UML*i* representa a apresentação para cada espaço de interacção através da utilização do *User Interface Diagram*. Este diagrama apresenta algumas semelhanças com o *InteractionSpace View* do XIS. O *User Interface Diagram* utiliza *icons* para representar o papel em abstracto de cada elemento de interacção (elemento de input, elemento de output, elemento que invoca acções), o que pode facilitar a leitura dos modelos. É possível destacar a noção de contentor através da utilização dos estereótipos *Free Container* e *Container*. A partir do *User Interface Diagram* é possível inferir alguma informação respeitante ao posicionamento relativo dos elementos de interacção, mas não relativamente ao tamanho dos mesmos.

Através do *User Interface Diagram* é possível, para cada espaço de interacção, indicar se um elemento de interacção é *Inputter*, *Displayer* ou *Editor*, mas não é possível alterar o seu estado / comportamento para diferentes momentos ou contextos da interacção. Relativamente a outros aspectos de comportamento e de navegação, não existe nenhum diagrama que permita representar a navegação entre os diferentes espaços de interacção, mas é possível representar o comportamento de um determinado espaço de interacção e dos elementos de interacção nele contidos, através de um diagrama de actividades UML*i*.

No que diz respeito a aspectos de geração, não encontramos nenhuma referência à utilização de mecanismos de geração de modelo-para-modelo. É possível a geração dos modelos UML*i* para código, mas não encontramos nenhuma referência a experiências nesse sentido.

A iniciativa **UWE** define um perfil que permite representar aspectos conceptuais, de navegação e de apresentação, nomeadamente através da utilização de três tipos de modelos: (1) o modelo conceptual; (2) o modelo de navegação e (3) o modelo de apresentação. O modelo de navegação denota que esta iniciativa está fortemente vocacionada para o desenho e modelação de aplicações *web*, nomeadamente dando grande foco à representação de menus, itens de menu e outros elementos que permitem navegar para os diferentes espaços de interacção. Os modelos UWE apresentam uma notação mais extensa, o que contribui para que não sejam tão facilmente descritos à mão. A utilização de icons torna os modelos visualmente claros e de fácil leitura.

A iniciativa UWE representa a apresentação de cada espaço de interacção, através da utilização do *Abstrat User Interface Model*, apresentando algumas semelhanças com o *InteractionSpace View* do XIS. A iniciativa UWE utiliza icons para representar o papel

em abstracto de cada elemento de interacção (elemento de *input*, elemento de *output*, elemento que invoca acções), o que pode facilitar a leitura dos modelos, principalmente o *Abstract User Interface Model* e o *Storyboard Model*. É possível destacar a noção de contentor, através da utilização dos estereótipos *UI View* e *Presentation Class*. A partir do *Abstract User Interface Model,* é possível inferir alguma informação, respeitante ao tamanho e posicionamento relativo dos elementos de interacção.

Relativamente a aspectos de comportamento, é possível representar a navegação entre os diferentes espaços de interacção. Para cada espaço de interacção é possível indicar se um elemento de interacção é *Text* (elemento de *output*), ou *Form* (elemento de *input*), mas não é possível alterar o seu estado / comportamento para diferentes momentos ou contextos da interacção.

No que diz respeito a aspectos de geração, embora não se tenha encontrado nenhuma referência para este tipo de transformação, parece ser possível desenvolver mecanismos de geração de modelo-para-modelo, com vista a gerar o *Storyboard Model,* a partir do modelo de navegação e do *Abstract User Interface Model*. É possível a geração dos modelos UWE para código, nomeadamente para diversas plataformas *web*, mas não encontramos nenhuma referência a experiências nesse sentido.

Numa primeira fase, a iniciativa **Wisdom** permitia representar diversos aspectos conceptuais, de navegação e de apresentação dos sistemas, mas não permitia o desenho abstracto de cada espaço de interacção. A iniciativa Wisdom focava-se principalmente no modelo de apresentação, que é muito semelhante ao *Participants Diagram* da iniciativa UX.

O modelo de apresentação Wisdom permite especificar os diferentes espaços de interacção do sistema e o fluxo de navegação entre eles. Os modelos Wisdom utilizam icons, o que torna os modelos visualmente claros e de fácil leitura. É possível destacar a noção de contentor, através da utilização do estereótipo *Interaction Space*, dos diversos estereótipos do tipo *container* e da associação *Contains*. É possível representar os diferentes elementos que estão contidos em cada espaço de interacção (quer sejam elementos de interacção simples, quer sejam outros espaços de interacção), mas sem representar a estrutura de cada espaço de interacção. A iniciativa Wisdom ultrapassou esta lacuna, utilizando os Protótipos Canónicos Abstractos, que permitem representar de forma abstracta, cada nó da interface gráfica.

Os Protótipos Canónicos Abstractos apresentam uma notação extensa o que contribui para que não sejam tão facilmente descritos à mão. Os Protótipos Canónicos Abstractos utilizam icons para representar o papel em abstracto de cada elemento de interacção (*input element*, *output element*, *action*). A partir dos Protótipos Canónicos Abstractos, é possível inferir alguma informação respeitante ao tamanho e posicionamento relativo dos elementos de interacção.

Relativamente a aspectos de comportamento, é possível representar a navegação entre os diferentes espaços de interacção. Para cada espaço de interacção, é possível indicar se um elemento de interacção é um input ou output element. A iniciativa Wisdom recorre à utilização de ConcurTaskTrees (CTTs) para descrever o comportamento da interacção das interfaces com o utilizador.

No que diz respeito a aspectos de geração, é possível desenvolver mecanismos de geração de modelo-para-modelo, com vista a gerar os Protótipos Canónicos Abstractos a partir do modelo de apresentação. É possível a geração dos modelos Wisdom para código, nomeadamente para a linguagem PHP na *framework* Hydra.

O XIS distingue-se destas iniciativas, através dos seguintes aspectos principais:

- **Representação utilizando modelos PIM**. O XIS utiliza modelos independentes da plataforma (PIM) para representar as suas diferentes vistas. Algumas das iniciativas analisadas não apresentam esta vantagem, nomeadamente as iniciativas UX e UWE cujos modelos estão definidos para representar aplicações *web*.
- **Modelação de várias vistas interrelacionadas**. O XIS permite modelar sistemas interactivos através da utilização de várias vistas: (1) a *Entities View*; (2) a *Use-Cases View* e (3) a *User-Interfaces View*. Se por um lado, a utilização de vistas, permite que os diferentes conceitos do sistema possam ser modelados de forma independente, também é possível manter uma visão integrada das diferentes vistas. Através desta visão integrada, é possível relacionar as diferentes vistas, podendo umas vistas aproveitar informação de outras vistas. Isto permite, entre outros exemplos, ligar os elementos da vista *User-Interface View*, a elementos da Entities View, de modo a obter informação relativamente ao tamanho e tipo de elementos do modelo de domínio.
- **Suporta técnicas de geração de modelos para diversas plataformas computacionais específicas**. Apesar de ser possível criar mecanismos de geração de código a partir de todas estas linguagens e de algumas delas já o

estarem a fazer (UX e Wisdom), o XIS apresenta uma forte experiência nesta área. Já na primeira versão do XIS, foram criados mecanismos de geração de modelos para código tendo sido substancialmente melhorados e complementados nesta segunda versão, através da criação das vistas para representação das interfaces gráficas. Actualmente o XIS apresenta mecanismos de geração para (1) a plataforma Microsoft Windows Forms.NET [Microsoft ---b] e (2) a plataforma Microsoft ASP.NET [Microsoft ---a].

7. Conclusão e Trabalho Futuro

Neste capítulo são apresentadas as conclusões gerais, as diversas etapas e os objectivos atingidos neste trabalho de investigação. Também são identificados os aspectos em aberto que podem ser aprofundados em trabalhos futuros.

7.1. Conclusão

Neste trabalho de investigação foram analisadas diversas abordagens para o desenho e produção de interfaces gráficas, nomeadamente: (1) utilização de ferramentas para construção de interfaces gráficas (*UI builders tools approach*); (2) utilização de ferramentas para desenho e reconhecimento de esboços (*UI sketching tools approach*); (3) utilização de linguagens baseadas em XML (*XML-based languages approach*) e (4) utilização de linguagens de modelação (*MDA tools approach*). Foram analisadas em maior detalhe diversas iniciativas baseadas na abordagem que utiliza linguagens de modelação para a produção de interfaces gráficas. Esta análise serviu de ponto de partida para estender a versão 1 do perfil XIS [Silva 2003b; Silva 2003a; Silva, Lemos et al. 2003]. Esta primeira versão apresentava algumas limitações, designadamente, não permitia modelar as interfaces gráficas com o utilizador de forma explícita. Sendo assim, neste trabalho, o perfil XIS foi estendido de forma a criar vistas específicas para modelar interfaces gráficas.

A nova versão do perfil XIS encontra-se dividida em três vistas principais: (1) a *Entities View* (*Domain View* e *BusinessEntities View*), (2) a *Use-Cases View* (*UseCases View* e *Actors View*) e (3) a *User-Interfaces View* (*NavigationSpace View* e *InteractionSpace View*), sendo esta última dedicada a modelar os diversos aspectos relacionados com as interfaces gráficas. Tentamos descrever com algum detalhe como poderíamos especificar diversos padrões de interacção através do perfil XIS, representado controlos genéricos independentes de qualquer plataforma computacional.

A abordagem seguida acompanha os requisitos da recomendação Model Driven Architecture (MDA) [OMG ---a], por isso o perfil XIS é independente da plataforma, possibilitando criar a partir deste, *templates* para geração para diversas plataformas computacionais específicas. Neste contexto, foram desenvolvidos *templates* para

geração de código para duas plataformas específicas: (1) a plataforma Microsoft Windows Forms.NET [Microsoft ---b] e (2) a plataforma Microsoft ASP.NET [Microsoft ---a]. Estes *templates* de geração recebem como *input* os modelos criados através do perfil XIS.

Finalmente e para avaliar o perfil XIS, foi realizada uma análise comparativa com outras quatro iniciativas analisadas, baseadas na abordagem que utiliza linguagens de modelação para a produção de interfaces gráficas, nomeadamente: (1) User-Experience (UX) [Kozaczynski e Thario 2002; Heumann 2003]; (2) UMLi [Silva e Paton 2003; Silva e Paton. 2003]; (3) UWE [Koch e Kraus 2002; Koch 2006; Norrie 2007] e (4) Wisdom [Nunes e Cunha 2000].

Todas as fases deste trabalho foram exemplificadas e testadas, tendo por base o caso de estudo "DocPro - Sistema de Gestão Documental e de Projectos" descrito na secção 1.4 e cuja modelação se encontra descrita no Apêndice A.

7.2. Trabalho Futuro

Como em todos os trabalhos de investigação, existe um conjunto de aspectos que por um motivo ou outro não podem ser desenvolvidos quer por falta de tempo, quer por questões de estratégia da investigação, mas que poderão ser enquadrados em futuros trabalhos. De seguida sintetizam-se alguns destes aspectos em aberto.

Representação de novos padrões de interacção e geração de novos tipos de controlos: No perfil XIS foram definidas um conjunto de marcas associadas aos elementos de interacção (XisInteractionSpace, XisInteractionCompositeElement, XisActionElement e XisDataElement), nomeadamente as marcas *interactionSpaceType*, *controlType*, *actionType* e *compositeElementType*. Estas marcas são do tipo enumerado, podendo conter um conjunto de valores predefinido. Este mecanismo permite, que a lista possa ser estendida, possibilitando a representação de novos controlos genéricos ou de novos padrões de elementos de interacção, deixando em aberto as possibilidades de extensibilidade da notação. Para cada novo elemento a representar através do perfil XIS, teriam que ser criados mecanismos de geração e alterados os respectivos *templates* de forma a gerar estes novos controlos específicos. Em resumo, fica em aberto um enorme leque de possibilidades de crescimento da notação para representação de

novos padrões de interacção e respectiva geração de novos controlos, tais como, *toolbars*, *panels*, etc.

Representação e geração de relatórios: Não foi pensado nem definido como seria possível representar e gerar relatórios. À primeira vista, pensamos que um relatório poderia ser representado como sendo um tipo especial de espaço de interacção que conteria apenas elementos de output e cujo resultado se destinaria a ser impresso ou eventualmente pré-visualizado. No entanto, por falta de tempo, não se explorou e testou esta hipótese de representação e respectiva geração.

Representação do comportamento, recorrendo à linguagem OCL: O perfil XIS permite modelar alguns aspectos de comportamento, como sejam: (1) aspectos relacionados com a navegação; (2) possibilidade de indicar se um elemento de interacção é elemento de *output* ou de *input* e (3) definição de comportamento consoante o contexto definido através da marca *standardAction*. No entanto existe um conjunto de aspectos de comportamento que não foram pensados e aos quais o perfil XIS não consegue responder (e.g. alteração do estado / comportamento de um determinado elemento para diferentes momentos ou contextos da interacção, ou seja tornar ou não disponível determinado elemento de interacção, tendo em conta validações feitas a partir dos dados). Seria pertinente, alargar esta investigação à possibilidade de representar diversos aspectos de comportamento, recorrendo à linguagem OCL (*Object Constraint Language*). O OCL é uma linguagem para especificação formal de restrições e é parte integrante do UML.

Controlo de acesso aos elementos de interacção diferenciado por actor: Através do perfil XIS está prevista a possibilidade de representar um conjunto de regras de controlo de acesso aos elementos de interacção, ou seja se um determinado elemento de interacção está visível ou activo para determinado actor do sistema. Apesar desta possibilidade estar prevista em termos de modelação, através da utilização da associação XisElementPermission, este aspecto foi pouco desenvolvido, não tendo sido concretizado em termos de geração de código.

Transformações de modelo-para-modelo: Por questões de produtividade seria interessante criar alguns mecanismos de geração de modelo-para-modelo. Seria interessante e altamente produtivo poder gerar a *User-Interfaces View* a partir da

BusinessEntities View e da *Use-Cases View*. A partir da *BusinessEntities View* poderíamos obter os elementos que fariam parte de determinado espaço de interacção e a partir da *Use-Cases View* poderíamos especificar o controlo de acesso para determinados elementos de interacção por parte de determinados actores.

Transformação dos modelos na linguagem XIS em linguagens baseadas em XML: Para além da abordagem de desenho e produção de interfaces gráficas, recorrendo à utilização de linguagens de modelação, também foi analisada entre outras, a abordagem que utiliza linguagens baseadas em XML (ver secção 2.2.3). Para uma futura investigação interessaria estudar a possibilidade de combinar estas duas abordagens. Poderíamos, por exemplo, transformar os modelos representados na linguagem XIS em representações baseadas em linguagens XML já existentes (e.g UIML, XIML, AUIML ou XForms). Esta transformação permitiria aproveitar os mecanismos de *rendering* já existentes e os que viessem a ser produzidos de futuro, garantindo a geração para diversas plataformas computacionais específicas.

Geração para diversas plataformas específicas: Neste trabalho de investigação foram definidos *templates* para geração de código para duas plataformas específicas: (1) a plataforma Microsoft Windows Forms.NET e (2) a plataforma Microsoft ASP.NET. Como o perfil XIS é independente da plataforma, é possível criar *templates* para geração de código para outras plataformas computacionais específicas, ficando esta possibilidade em aberto. Seria pertinente a geração para outras plataformas que não da Microsoft (e.g, Java / Struts), bem como a geração para plataformas móveis. Também podem ser estudados e melhorados alguns aspectos de apresentação para as plataformas já geradas.

References

[Abrams, Phanouriou et al. 1999] Abrams, Marc, Constantinos Phanouriou, et al. (1999). UIML: An Appliance-Independent XML User Interface Language. Proceedings of the 8th International World Wide Web Conference, http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html.

[Allgar e Finlay --] Allgar, Elizabeth e Janet Finlay. (--). "Patterns project: pattern languages ", consultado em: http://www.leedsmet.ac.uk/inn/comp/research/isle/patterns/learning.htm.

[Anderson 2000] Anderson, David J. (2000). Extending UML for UI. Tupis 2000 Workshop - Towards a UML Profile for Interactive Systems Development, http://www.uidesign.net/2000/papers/TUPISproposal.html.

[Azevedo, Merrick et al. 2000] Azevedo, Pedro, Roland Merrick, et al. (2000). OVID to AUIML - User-Oriented Interface Modelling. Tupis 2000 Workshop - Towards a UML Profile for Interactive Systems Development, http://math.uma.pt/tupis00/submissions/azevedoroberts/azevedoroberts.html.

[Borland --] Borland. (--). "Delphi." consultado em: http://www.borland.com/uk/products/delphi/index.html.

[Caetano, Goulart et al. 2002] Caetano, Anabela, Neri Goulart, et al. (2002). JavaSketchIt: Issues in Sketching the Look of User Interfaces. AAAI Spring Symposium - Sketch Understanding, http://immi.inesc.pt/~mjf/publications/java-aaai02.pdf.

[Campos e Nunes 2004] Campos, Pedro F. e Nuno J. Nunes (2004). CanonSketch: a User-Centered Tool for Canonical Abstract Prototyping. Engineering for Human-Computer Interaction (EHCI) / Design, Specification, and Verification of Interactive Systems (DSV-IS), http://dme.uma.pt/people/faculty/nuno.nunes/publications/DSVIS2004.pdf.

[Constantine e Cockwood 1999] Constantine, Larry L. e Lucy A. D. Cockwood (1999). Software for Use, ACM Press, Addison-Wesley Publishing Co.

[Constantine, Windl et al. 2003] Constantine, Larry, Helmut Windl, et al. (2003) "From Abstraction to Realization: Canonical Abstract Prototypes for User Interface Design." consultado em: http://www.foruse.com/articles/canonical.pdf.

[Costa 2007] Costa, Duarte Nuno Fernandes Homem (2007). Geração Automática de Interfaces com o Utilizador: Uma Abordagem Baseada em MDA para a Plataforma PHP, Universidade da Madeira. Mestrado.

[Coyette 2006] Coyette, Adrien. (2006). "Un petit dessin vaut mieux qu'un long discours." consultado em: http://www.usixml.org/index.php5?mod=download&file=Coyette-Forum-Dec2006.ppt.

[Coyette e Vanderdonckt 2005] Coyette, Adrien e Jean Vanderdonckt (2005). A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces. Proceedings of 10th IFIP TC13 International Conference on Human-Computer Interaction: INTERACT'05,

http://www.isys.ucl.ac.be/bchi/publications/2005/Coyette-Interact2005.pdf.

[Favre 2004] Favre, Jean-Marie (2004). Towards a basic theory to model Model Driven Engineering. WISME 2004,

http://www-adele.imag.fr/users/Jean-Marie.Favre/papers/VisualizationInTheContextOfModelDrivenEngineering.pdf

[Fonseca e Jorge 2002]Fonseca, Manuel J. e Joaquim A. Jorge (2002). Sketching User Interfaces with Visual Patterns. 1st Ibero-American Symposium in Computer Graphics (SIACG'02), Conference, Revista VIRtual,

http://immi.inesc.pt/~mjf/publications/siacg02-javaSketchIt.pdf.

[Gamma, Helm et al. 1995] Gamma, Erich, Richard Helm, et al. (1995). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

[Hennicker e Koch 2001] Hennicker, Rolf e Nora Koch (2001). Modeling the User Interface of Web Applications with UML. Workshop of the pUML-Group at the UML 2001, Conference, A. Evans, R. France and A. Moreira,

http://www.pst.informatik.uni-muenchen.de/personen/kochn/pUML2001-Hen-Koch.pdf.

[Heumann 2003] Heumann, Jim (2003). User experience storyboards: Building better UIs with RUP, UML, and use cases. The Rational Edje,

http://www.ibm.com/developerworks/rational/library/content/RationalEdge/nov03/f_usability_jh.pdf.

[IBM] IBM. "IBM Rational Software." consultado em:

http://www-306.ibm.com/software/rational/.

[INESC-ID --] INESC-ID (--). "INESC-ID Instituto de Engenharia de Sistemas e Computadores Investigação e Desenvolvimento em Lisboa ", Editor,

http://www.inesc-id.pt/intranet/laboratoriogrupo/view/group_generalinfo.php?CC=II01.

[Koch 2006] Koch, Nora (2006). Transformation techniques in the model-driven development process of UWE. Second international workshop on model driven web engineering (MDWE'06),

http://www.lcc.uma.es/~av/mdwe2006/camera_ready_papers/koch-mdwe-2006-final.pdf.

[Koch e Kraus 2002] Koch, Nora e Andreas Kraus (2002). The Expressive Power of UML-based Web Engineering (UWE). Second Int. Worskhop on Web-oriented Software Technology (IWWOST´02),

http://www.pst.informatik.uni-muenchen.de/~kochn/IWWOST02-koch-kraus.PDF.

[Kozaczynski e Thario 2002] Kozaczynski, Wojtek e Jim Thario (2002). Transforming User Experience Model To Presentation Layer Implementations. OOPSLA 2002 - Second Workshop on Domain-Specific Visual Languages, Conference, ACM,

http://se2c.uni.lu/tiki/se2c-bib_download.php?id=262.

[Laakso 2003] Laakso, Sari A. (2003). "User Interface Design Patterns." consultado em: http://www.cs.helsinki.fi/u/salaakso/patterns/index.html.

[Landay e Mayers 2001] Landay, James A e Brad A Mayers (2001). Sketching Interfaces: Toward More Human Interface Design. IEEE Computer Magazine, http://www.cs.berkeley.edu/~landay/research/publications/silk-ieee-published.pdf.

[Martins e Silva 2007] Martins, Carlos e Alberto Rodrigues da Silva (2007). Modeling User Interfaces with the XIS UML Profile. 9th International Conference on Enterprise Information Systems - ICEIS 2007, Conference, Springer, http://berlin.inesc-id.pt/alb/static/papers/2007/cm-iceis2007.pdf.

[Microsoft ---a] Microsoft. (---a). "Microsoft ASP.NET." consultado em: http://www.asp.net/.

[Microsoft ---b] Microsoft. (---b). "Microsoft Windows Forms." consultado em: http://msdn2.microsoft.com/pt-br/netframework/aa497342.aspx.

[Microsoft ---c] Microsoft. (---c). "Visual Basic." consultado em: http://msdn2.microsoft.com/pt-br/vbasic/default.aspx.

[Microsoft ---d] Microsoft. (---d). "Visual C#." consultado em: http://msdn2.microsoft.com/pt-br/vcsharp/default.aspx.

[Microsoft ---e] Microsoft. (---e). "Visual Studio 2005 Developer Center." consultado em: http://msdn2.microsoft.com/en-us/vstudio/default.aspx.

[Microsoft ---f]Microsoft. (---f). "Windows Forms Programming - FlowLayoutPanel Control Overview ", consultado em: http://msdn2.microsoft.com/en-us/library/f9e8s203.aspx.

[Microsoft ---g] Microsoft. (---g). "Windows Forms Programming - TableLayoutPanel Control Overview ", consultado em: http://msdn2.microsoft.com/en-us/library/h21wykkx.aspx.

[Newman, Lin et al. 2003] Newman, Mark W., James Lin, et al. (2003). "DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice." Human-Computer Interaction, Editor, http://dub.washington.edu/projects/denim/pubs/denim-hci_journal.pdf.

[Norrie 2007] Norrie, Moira. (2007). "UWE: UML-based Web Engineering." consultado em: http://www.globis.ethz.ch/education/webeng/lec8-uwe-lge.pdf.

[Nunes e Cunha 2000]Nunes, Nuno Jardim e João Falcão e Cunha (2000). Towards a UML profile for interaction design: the Wisdom approach. Tupis 2000 Workshop - Towards a UML Profile for Interactive Systems Development, Conference, Springer, http://dme2.uma.pt/njn/tiki-download_file.php?fileId=37.

[OMG 1999] OMG. (1999). "OMG UML Working Group. White Paper on the Profile mechanism, Version 1.0." consultado em.

[OMG 2006] OMG. (2006). "Unified Modeling Language: Diagram Interchange – Specification Version 1.0." consultado em:

http://www.omg.org/cgi-bin/apps/doc?formal/06-04-04.pdf.

[OMG ---a] OMG. (---a). "Model Driven Architecture." consultado em:

http://www.omg.org/mda/

[OMG ---b] OMG. (---b). "Unified Modeling Language." consultado em:

http://www.uml.org/.

[Paternò 1999] Paternò, Fabio (1999) "ConcurTaskTrees: An Engineered Approach to Model-based Design of Interactive Systems." consultado em:

http://giove.cnuce.cnr.it/book-task-paterno.pdf.

[Paton e Silva 2002] Paton, Norman W. e Paulo Pinheiro da Silva. (2002). "UML*i* The Unified Modeling Language for Interactive Applications." consultado em:

http://www.cs.man.ac.uk/img/umli/index.html.

[Plimmer e Grundy 2005] Plimmer, Beryl e John Grundy (2005). Beautifying Sketching-based Design Tool Content: Issues and Experiences. The Sixth Australasian User Interface Conference (AUIC), Conference, ACM,

http://crpit.com/confpapers/CRPITV40Plimmer.pdf.

[Puerta e Eisenstein 2002] Puerta, Angel e Jacob Eisenstein (2002). XIML: A Universal Language for User Interfaces. International Conference on Intelligent User Interfaces (IUI'02), Conference, ACM,

http://www.ximl.org/documents/XimlWhitePaper.pdf.

[Saraiva 2005] Saraiva, João (2005). Relatório Final de Trabalho Final de Curso – Desenvolvimento Automático de Sistemas, Instituto Superior Técnico - Universidade Técnica de Lisboa.

[Schmidt 2006] Schmidt, Douglas C. (2006). Model Driven Engineering. IEEE Computer Society,

http://www.cs.wustl.edu/~schmidt/GEI.pdf.

[Silva 2003a] Silva, Alberto Rodrigues da (2003a). Abordagem XIS ao Desenvolvimento de Sistemas de Informação. CAPSI'2003 - IV Conferência da Associação de Sistemas de Informação, Conference, APSI - Associação Portuguesa de Sistemas de Informação

http://berlin.inesc.pt/alb/static/papers/2003/capsi2003-amrs.pdf.

[Silva 2003b] Silva, Alberto Rodrigues da (2003b). The XIS Approach and Principles. Proceedings of the 29th Euromicro Conference, Conference, IEEE Computer Society.

[Silva 2004] Silva, Alberto Rodrigues da. (2004). "O Programa de Investigação "ProjectIT", Technical report, V 1.0." consultado em:

http://berlin.inesc.pt/alb/uploads/1/193/pit-white-paper-v1.0.pdf.

[Silva, Lemos et al. 2003] Silva, Alberto Rodrigues da, Gonçalo Lemos, et al. (2003). The XIS Generative Programming Techniques. Proceedings of the 27th COMPSAC Conference, Conference, IEEE Computer Society.

[Silva, Saraiva et al. 2007a] Silva, Alberto Rodrigues da, João Saraiva, et al. (2007a). "Integration of RE and MDE Paradigms: The ProjectIT Approach and Tools."

IET Software Journal (Special issue "On the Interplay of .NET and Contemporary Development Techniques), Editor.

[Silva, Saraiva et al. 2007b] Silva, Alberto Rodrigues da, João Saraiva, et al. (2007b). XIS – UML Profile for eXtreme Modeling Interactive Systems. MOMPES 2007 - International Workshop Series on Model-based Methodologies for Pervasive and Embedded Software Conference, IEEE Computer Society,

http://berlin.inesc.pt/alb/static/papers/2007/as-XISUMLProfile-MOMPES2007.pdf.

[Silva e Videira 2005] Silva, Alberto Rodrigues da e Carlos Videira (2005). UML, Metodologias e Ferramentas CASE, Centro Atlântico.

[Silva, Videira et al. 2006] Silva, Alberto, Carlos Videira, et al. (2006). The ProjectIT-Studio, an integrated environment for the development of information systems. Proceedings of the Second International Conference of Innovative Views of .NET Technologies (IVNET'06), Conference, SBC & Microsoft,

http://berlin.inesc-id.pt/alb/static/papers/2006/ivnet2006-pit-v1.0c.pdf.

[Silva e Paton 2000] Silva, Paulo Pinheiro da e Norman W. Paton (2000). UML*i*: The Unified Modeling Language for Interactive Applications. UML 2000 - Third International Conference on the Unified Modeling

http://www.ksl.stanford.edu/people/pp/papers/PinheirodaSilva_UML_2000.pdf.

[Silva e Paton 2003] Silva, Paulo Pinheiro da e Norman W. Paton (2003). Improving UML Support for User Interface Design: A Metric Assessment of UMLi. ICSE 03 International Conference on Software Engineering, Conference, IEEE Computer Society, ACM,

http://www.ksl.stanford.edu/people/pp/papers/PinheirodaSilva_ksl_02_04.pdf.

[Silva e Paton. 2003] Silva, Paulo Pinheiro da e Norman W. Paton. (2003). User Interface Modeling in UML*i*. IEEE Software. Vol. 20**:** Pages 62-69,

http://www.cs.utep.edu/paulo/papers/PinheirodaSilva_SOFTWARE_2003.pdf.

[Silva 2006] Silva, Rui (2006). Relatório Final de Trabalho Final de Curso - Produção Automática de Software, Instituto Superior Técnico - Universidade Técnica de Lisboa.

[Souchon e Vanderdonckt 2003] Souchon, Nathalie e Jean Vanderdonckt (2003). A Review of XML-Compliant User Interface Description Languages. DSV-IS 2003: 10th International Workshop on Design, Specification and Verification of Interactive Systems, Conference, Springer,

http://www.isys.ucl.ac.be/bchi/publications/2003/Souchon-DSVIS2003.pdf.

[Sparx_Systems 2007] Sparx_Systems (2007) "UML 2 Case Tool by Sparx Systems." consultado em:

http://sparxsystems.com.au/bin/MDA%20Tool.pdf.

[The_Eclipse_Foundation --] The_Eclipse_Foundation (--). "Eclipse." Editor,

http://www.eclipse.org/.

[Tidwell 1999] Tidwell, Jenifer. (1999). "Common Ground: A Pattern Language for Human-Computer Interface Design Jenifer Tidwell ", consultado em:

http://www.mit.edu/~jtidwell/interaction_patterns.html.

[Tidwell 2005] Tidwell, Jenifer (2005). Designing Interfaces - Patterns for Effective Interaction Design, O'Reilly,
http://designinginterfaces.com/.

[UIML.org 1997] UIML.org. (1997). "Tutorial Booklet." consultado em:
http://www.uiml.org/tutorials/uiml1/uiml_v10_tutorial.pdf.

[Videira e Silva 2004] Videira, Carlos e Aberto Rodrigues da Silva (2004). ProjectIT-Requirements, a Formal and User-oriented Approach to Requirements Specification. Actas de las IV Jornadas Iberoamericanas en Ingeniería del Software e Ingeniería del Conocimiento (JIISIC).
http://berlin.inesc.pt/alb/static/papers/2004/cv-jiisic2004.pdf.

[W3C 2007] W3C (2007). "W3C - The Forms Working Group." Editor,
http://www.w3.org/MarkUp/Forms/.

[Welie e Troetteberg 2000] Welie, Martijn van e Hallvard Troetteberg (2000). Interaction Patterns in User Interfaces. PLoP 2000,
www.idi.ntnu.no/~hal/publications/design-patterns/PLoP2k-Welie.pdf

[Welie, Veer et al. 1999] Welie, Martijn van, Gerrit C. van der Veer, et al. (1999). Breaking down Usability. Proceedings of Interact 99
www.cs.vu.nl/~martijn/gta/docs/Interact99.pdf

[Wesson e Cowley 1999] Wesson, Janet e Lester Cowley (1999). Selecting Interaction Objects: A Patterns Approach. Interact 99,
www.it.bton.ac.uk/staff/rng/CHI2K_PLworkshop/PositionPapers/Wesson.doc.