Abordagem XIS ao Desenvolvimento de Sistemas de Informação

Alberto Rodrigues da Silva

IST & INESC-ID, Lisboa, Portugal

alberto.silva@acm.org

Resumo

XIS é um projecto de I&D cuja principal missão é o estudo, desenvolvimento e avaliação de mecanismos e ferramentas para produção de sistemas de informação de forma mais eficiente, alto nível, e com melhor qualidade do que actualmente acontece. O projecto XIS é influenciado pelo modelo de referência MDA e baseia-se significativamente num conjunto de boas práticas, designadamente: segue uma abordagem baseada na especificação de modelos; centrada em arquitecturas de software; e baseada em técnicas de geração automática de artefactos digitais. Neste artigo apresenta-se a visão geral do projecto XIS, introduzindo os seus principais elementos, nomeadamente a abordagem XIS, a plataforma XIS, o perfil XIS/UML e a linguagem XIS/XML. Apresenta-se sucintamente o caso de estudo "MyContacts" que é usado como demonstrador de alguns aspectos concretos. São discutidos com detalhe as principais boas práticas inspiradores do projecto XIS. Por fim, apresentam-se as principais conclusões e o trabalho a desenvolver no futuro próximo.

Palavras chave: desenvolvimento de sistemas de informação interactivos, MDA, modelos em UML e XML, geração automática, arquitecturas de software

1 Introdução

Tradicionalmente a indústria de software tem apresentado um grande ênfase e investimento ao nível das actividades ligadas à produção (como sejam programação, testes e integração de componentes e de sistema) em oposição às actividades mais ligadas ao projecto e à concepção (como sejam a engenharia de requisitos, a análise e o desenho).

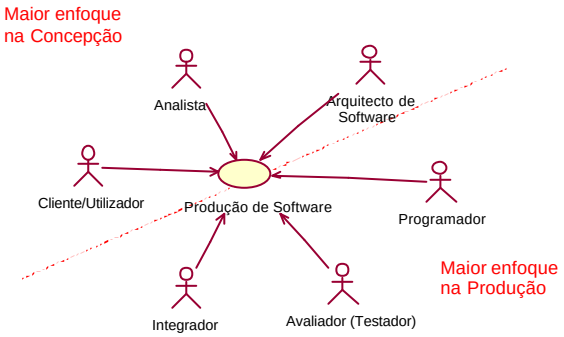

Figura 1: Principais actores na produção de software.

De acordo com a abordagem XIS [GSI --], o ênfase deve ser reforçado nas actividades de concepção e projecto, e consequentemente, o esforço nas actividades de produção deve ser minimizado e realizado tanto quanto possível de forma automática, tal como acontece noutras indústrias como por exemplo na indústria automóvel, alimentar ou farmacêutica. Os aspectos relacionados com o "como fazer" são relevantes, mas são tratados principalmente pelos

arquitectos de software que são os responsáveis por providenciar arquitecturas elegantes, flexíveis e reutilizáveis.

A ideia subjacente ao projecto XIS não é nova; desde há pelo menos duas décadas que investigadores e engenheiros constatam a crescente dimensão e complexidade dos projectos de software e que preconizam consequentemente que a sua execução deveria ser mais produtiva, os seus custos e prazos mais facilmente previsíveis e controlados, e que deveria ser aplicado menor esforço na produção e maior na concepção. Esta ideia não sendo original, não tem tido, contudo, grande sucesso: foram vários os projectos , em particular na área de ferramentas CASE, que perseguindo estes objectivos acabaram por ser abandonados por razões várias [Orlikowski 1993],[Vessey et al. 1995]. Cremos no entanto que o panorama se alterou actualmente de forma positiva. Pelo menos três factores técnicos, que não se encontravam reunidos no passado, contribuem decisivamente para suportar esta visão. Primeiro, existe uma linguagem de modelação visual de software, o UML (*Unified Modeling Language*) [OMG --], [Booch et al. 1999], [Silva e Videira 2001] que é um standard *de facto*, reconhecido e usado pela maioria da industria. Segundo, existe uma linguagem standard para representação de dados e metadados, o XML (*Extend Markup Language*) [W3C 1999], e em particular o XMI (*XML Metadata Interchange*) [OMG 2002]. Terceiro, existe o conhecimento real de padrões e arquitecturas de software de grande qualidade, como resultado de anos de experiências, reflexão e reengenharia das melhores práticas em numerosos projectos e situações [Gamma et al. 1994]],[Buschmann et al. 1996],[Hofmeister et al. 1999],[SPS --],[Juric et al. 2002]. Por conseguinte, cremos que hoje se encontram reunidas melhores condições técnicas para que a ideia subjacente ao projecto XIS se possa concretizar com sucesso. O projecto XIS é inspirado significativamente na filosofia MDA (*Model Driven Architecture*) [OMG 2001] que tem sido promovida no âmbito da OMG.

Este artigo encontra-se organizado em seis secções. A Secção 1 descreve sucintamente a motivação e enquadramento para a criação do projecto XIS. A Secção 2 apresenta a visão geral do projecto XIS, introduzindo os seus principais elementos, nomeadamente a abordagem XIS, a plataforma XIS, o perfil XIS/UML e a linguagem XIS/XML. Nas Secções 3, 4 e 5 são apresentadas com mais detalhe os principais princípios inspiradores do projecto XIS. Por fim, na Secção 6 apresentam-se as conclusões e perspectiva-se o trabalho futuro.

2 Visão Geral

O projecto XIS pretende explorar técnicas e mecanismos vários de forma a acelerar e melhorar a actividade de gestão e de produção de software. Para atingir tal objectivo é definido e proposto um conjunto integrado de elementos, designadamente:

- **Abordagem XIS**: consiste numa abordagem de desenvolvimento de software fortemente inspirada nos seguintes princípios ou boas práticas: (1) baseada em modelos, especificados em UML de forma mais conceptual, abstracta e completa possível; (2) centrada em arquitecturas de software; e (3) baseada em técnicas de geração automática de artefactos digitais.
- **Plataforma e repositório XIS**: consiste numa ferramenta CASE que tem como objectivo suportar os intervenientes técnicos no processo de desenvolvimento de software segundo a abordagem XIS. O repositório mantém informação relativamente a modelos, aplicações, arquitecturas de software, e ao próprio processo de desenvolvimento (e.g., etapas de geração, de teste e de integração dos componentes de software). A ferramenta é um sistema robusto com interface Web, multi-utilizador, multi-aplicação e com múltiplas arquitecturas de software.

- **Perfil UML para XIS** (ou simplesmente perfil XIS/UML): consiste num conjunto de extensões UML específicas do projecto XIS que permite a especificação visual, alto nível e intuitiva de sistemas de informação.
- **Linguagem XIS/XML**: consiste numa linguagem definida em XML que permite a especificação textual, estruturada, legível e compacta de informação necessária à construção de sistemas de informação.

A Figura 2 apresenta uma visão geral e simplificada do fluxo de actividades preconizado pela abordagem XIS. São evidenciados, para os principais actores, as suas respectivas actividades. A abordagem XIS tem, genericamente, como *input* os requisitos do sistema (e.g., requisitos funcionais, não funcionais, e de desenvolvimento), produzindo como *output* os vários artefactos digitais, correspondentes à concretização efectiva do sistema.

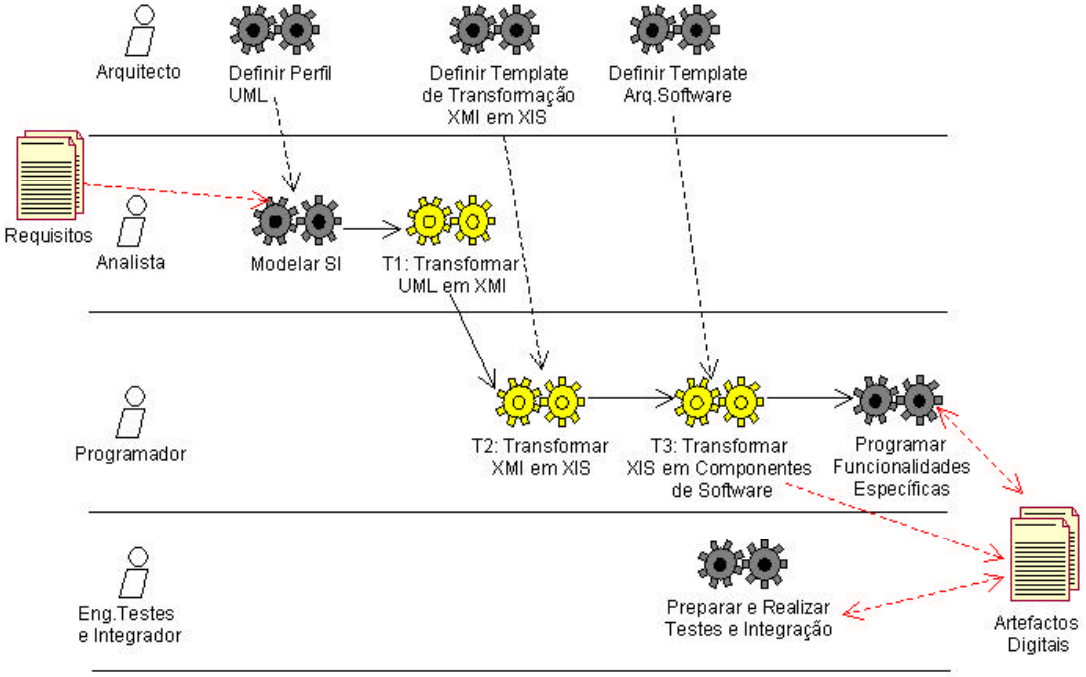

Figura 2: Visão geral da abordagem XIS.

Começa-se por evidenciar as actividades realizadas pelo arquitecto de software, que consistem genericamente na definição de todos aspectos arquitecturais de suporte ao adequado funcionamento da plataforma XIS, nomeadamente: (1) definição do perfil XIS/UML, de suporte à modelação; (2) definição do *template* de suporte à transformação T2, de modelos em formato XMI para formato XIS/XML; e (3) definição de *templates* de arquitecturas de software de suporte à transformação T3, de modelos em formato XIS/XML para artefactos digitais, tais como componentes de software. Note-se que estas actividades desenvolvidas pelo arquitecto são realizadas de forma independente do desenvolvimento específico de cada sistema de informação concreto.

Seguidamente, com base nos requisitos do sistema (tarefa que deverá ser realizada e validada normalmente entre os analistas, clientes e utilizadores) é da responsabilidade do analista modelar adequadamente o sistema de informação respectivo. Note-se que a correcção, qualidade e completude desta actividade (i.e., "Modelar SI") é crucial para as actividades subsequentes. T1, T2 e T3 correspondem à aplicação sucessiva de três transformações, que genericamente permitem transformar o modelo especificado em UML e extensões XIS/UML num conjunto significativo de artefactos digitais (ver Secção 5 para mais detalhes).

Atendendo que não se consegue modelar e captar ao nível do modelo XIS/UML todos os requisitos do sistema, é necessário a intervenção do programador para definir componentes específicas, tipicamente código fonte com perfil de fachadas, adaptadores, controladores e lógica de negócio. (Esta intervenção está sugerida ao nível da actividade "Programar Funcionalidades Específicas".)

Finalmente, é necessário a intervenção do engenheiro de testes e de integração para planear e realizar os vários testes e garantir que os requisitos são suportados de acordo com os níveis de qualidade inicialmente definidos. (Esta intervenção está sugerida ao nível da actividade "Preparar e Realizar Testes e Integração".)

3 XIS é Baseado na Especificação de Modelos

Um princípio fundamental que norteia a abordagem XIS é o facto de esta ser baseada na especificação de modelos segundo uma aproximação gráfica (usando UML) e ou declarativa (usando XML). Apresenta-se nesta secção os aspectos gerais deste princípio e a sua adopção no âmbito do projecto XIS.

3.1 UML, Mecanismos de Extensão e Perfis UML

O UML (*Unified Modeling Language*) é um standard OMG, reconhecido e utilizado pela indústria de software e mesmo noutras áreas da engenharia ou da gestão. O UML é principalmente uma linguagem de modelação visual de artefactos de sistemas de informação, que pode ser usada para representar inúmeras facetas de um sistema (e.g., visão física das plataformas computacionais, ou visão física, lógica ou de utilização de componentes de software) eventualmente representáveis a diferentes níveis de abstracção e detalhe (e.g., ao nível dos requisitos de utilizador, de análise ou de desenho) [Booch et al. 1999],[Silva e Videira 2001].

O UML providencia um número elevado de conceitos e notações particularmente concebidos de forma a satisfazer os requisitos típicos de modelação de sistemas de informação. Contudo, surgem situações em que se torna desejável a introdução de conceitos e/ou de notações adicionais para além dos definidos originalmente. Para contemplar tais situações, o UML providencia mecanismos que o permitem estender de forma consistente: restrições, marcas e estereótipos. Estes mecanismos permitem [OMG 1999] (1) introduzir novos elementos de modelação para uma maior expressividade e compreensão dos modelos UML a criar; (2) definir itens standard que não são considerados suficientemente interessantes ou complexos para serem definidos directamente como elementos do metamodelo UML; (3) definir extensões específicas das linguagens de implementação ou específicas dos processos de desenvolvimento; e (4) associar arbitrariamente informação semântica e outra aos elementos do modelo.

Pelas suas características de especificidade, as extensões não são normalmente compreendidas, suportadas e aceites pela generalidade dos utilizadores UML. De forma a organizar e promover uma evolução mais pacífica destas extensões, os promotores do UML propuseram o conceito de "perfil". Um **perfil UML** é um nome dado a um grupo predefinido de estereótipos, marcas-com-valor e restrições que conjuntamente especializam e configuram o UML para um determinado domínio de aplicação ou para um determinado processo de desenvolvimento [OMG 1999].

3.2 Perfil UML para XIS

O "Perfil UML para XIS" (ou simplesmente "Perfil XIS/UML") consiste num conjunto de extensões UML que permita a modelação de sistemas de informação segundo a abordagem XIS. Este perfil é fortemente inspirado num conjunto de boas práticas reconhecidas na

concepção e construção de sistemas de informação e tecnologias emergentes, designadamente: (1) arquitecturas de software [Grasner 1988], [Buschmann et al. 1999], [Hofmeister et al. 1999], [Juric et al. 2002]; (2) entidades de negócio; (3) *workflows* de interfaces de utilizador predefinidos; (4) *workflows* de interfaces de utilizador genéricos, e.g., segundo uma linguagem declarativa (e.g., UIML [UIML 2000]); e (5) *workflows* de negócio, e.g., segundo uma linguagem declarativa (e.g., BPEL4WS [Curbera 2002]). Os três primeiros aspectos têm constituído a parte principal do esforço actualmente desenvolvido, enquanto que os restantes aspectos serão alvo de investigação e trabalho futuro. Neste artigo apresenta-se a definição do perfil XIS/UML tendo em conta o seu estado actual.

Um padrão reconhecido para a organização estrutural de sistemas de informação interactivos é o MVC (*Model View Controller*), que divide uma aplicação interactiva em três componentes fundamentais: (1) modelo, (2) controlador, (3) vista. O modelo contém os dados e o núcleo funcional. Os controladores são responsáveis pelo tratamento das interacções com os utilizadores e as vistas realizam a apresentação da informação ao utilizador, sendo que as vistas e os controladores em conjunto constituem a interface com o utilizador. De acordo com o padrão MVC o perfil XIS/UML descreve sistemas designados por *XisSystemModel* sob três vistas ou sub-modelos arquitecturiais fundamentais que são designados por *XisModelView*, *XisControlView* e *XisViewView*. A estas três vistas é adicionada a vista do domínio de negócio que é designada por *XisDomainView*. A Figura 3 apresenta o meta-modelo correspondente aos conceitos acima referidos, a que correspondem, no nível de modelo, a pacotes UML que agregam diferentes tipos de diagramas (e.g., diagramas de classes e de estados) estendidos segundo o perfil XIS/UML.

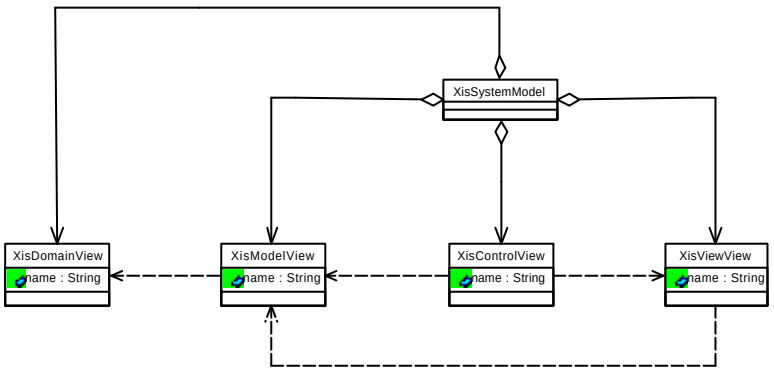

Figura 3: Relação entre os diferentes modelos/vistas segundo o perfil XIS/UML.

Modelo do Domínio (Vista de domínio, «XisDomainView»)

O modelo do domínio representa as classes e suas relações correspondentes às entidades (e respectivas relações) identificadas tipicamente no domínio do problema. A Figura 4 ilustra uma representação simplificada do respectivo modelo para o sistema MyContacts.

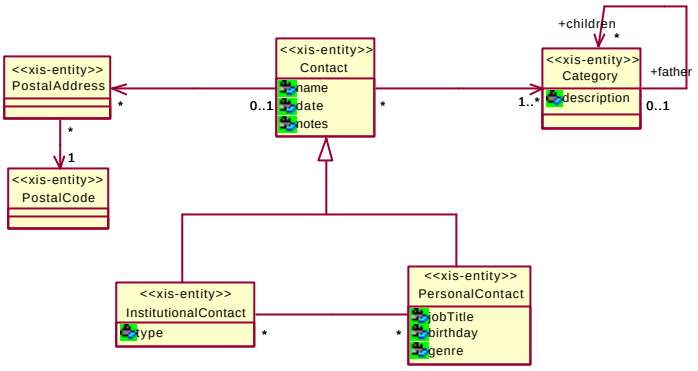

Figura 4: Modelo do domínio do sistema MyContacts.

As classes do modelo do domínio são identificadas com o estereótipo «xis-entity», têm um nome, uma lista de atributos (não é relevante a especificação de operações para este tipo de classes) e encontram-se relacionadas com outras classes que tenham sentido. Para os atributos, para além da informação captada ao nível do UML (e.g., nome e tipo do atributo), são definidas algumas marcas de forma a captar informação adicional, designadamente para efeitos de definição (1) do modelo de dados (e.g., `size`, `pkey`, `identity` e `null`); e (2) de suporte à construção automática de UI (e.g., `label`, `tooltip`, `isEditable`, `isRequired`, `regularExpression` e `errorMessage`).

Caso de Estudo "MyContacts"

O MyContacts é um sistema avançado de gestão de contactos que providencia as seguintes funcionalidades principais: gestão de contactos, emissão de relatórios vários, emissão de etiquetas, a importação e exportação de informação em formato XML, sincronização de dados entre versões da mesma família de produtos MyContacts.

Um "contacto" é caracterizado por um nome, data de registo, endereços postais e endereços electrónicos, e por uma ou mais observações. O endereço postal é composto por: nome da rua e detalhes do andar, localidade, código postal, e país. O endereço electrónico é composto pelos seguintes atributos: uma lista de endereços de correio electrónico, URL, uma lista de telefones. Relativamente a cada telefone interessa manter o tipo de telefone (e.g., fixo, móvel, fax) e um atributo para observações (e.g., "este telefone é da casa da vizinha..."). Há dois tipos básicos de contactos: pessoas e instituições. O contacto do tipo "instituição" tem apenas um atributo adicional que é o tipo de instituição (e.g., "Empresa S.A.", "Instituto do Estado", "Escola"). O contacto do tipo "pessoa" tem adicionalmente os seguintes atributos: título profissional (e.g., "Sr", "Eng."), sexo, e data de nascimento. Para além disso importa manter para cada contacto pessoal, caso existam, as referências para as eventuais instituições que a pessoa esteja envolvida. Importa também manter um sistema de categorias, segundo uma estrutura hierárquica (do género das "páginas amarelas") de forma que cada utilizador possa classificar os seus contactos e, consequentemente, os possa procurar de forma mais eficiente. Deve ser possível associar mais que uma categoria a cada contacto.

Pretende-se desenvolver uma família de aplicações "MyContacts" integradas entre si, todas elas desenvolvidas em Java, mas com as seguintes versões:

- *MyContacts-Mobile*, para dispositivos móveis, sobre a versão do J2ME (Java 2 Micro Edition);
- *MyContacts-Standard*, para PC de secretária, sobre a versão do J2SE
- *MyContacts-Enterprise*, para ambiente empresarial , desenvolvido sobre a versão do J2EE.

Concluindo, o MyContacts é o primeiro sistema de infomação desenvolvido segundo a abordagem XIS e consiste basicamente num sistema de gestão de contactos, à semelhança por exemplo do Outlook da Microsoft, tendo como principal particularidade o suporte de um conjunto integrado de aplicações para diferentes tipos de plataformas, nomeadamente J2SE, J2EE e J2ME. Este sistema serve de exemplo nas secções seguintes à explanação do projecto.

As relações entre estas classes são identificadas com o estereótipo «xis-association» e captam a informação definida ao nível do UML (e.g., nome da relação, identificação dos papéis e multiplicidade das classes participantes).

Adicionalmente, é também possível especificar relações de generalização/especialização entre os objectos («xis-inheritance») e listas de valores que definem tipos enumerados («xis-enumeration»).

Modelo das Entidades de Negócio (Vista de Modelo, «XisModelView»)

No modelo das entidades de negócio o objectivo é a definição de entidades de negócio, i.e. de classes com estereótipo «xis-business-entity», que corresponde à agregação de várias entidades definidas no modelo do domínio (i.e., classes com estereótipo «xis-entity») de forma a constituir-se entidades com um nível de granularidade superior e que tenha maior sentido na perspectiva da interface com o utilizador.

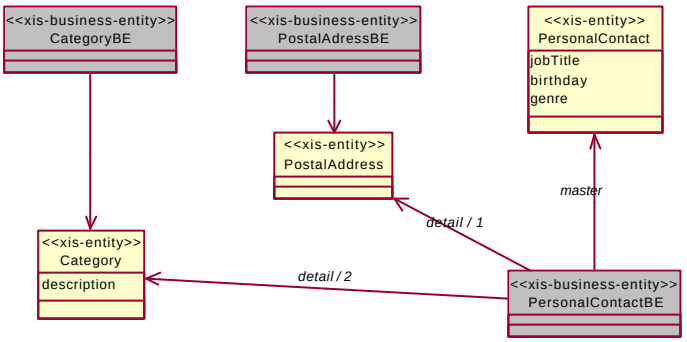

Figura 5: Uma vista do modelo das entidades de negócio do sistema MyContacts.

Algumas das entidades de negócio tem um mapeamento directo com apenas uma entidade definida no modelo de domínio (e.g., a "CategoriaBE" e "EndereçoPostalBE" da Figura 5) enquanto que outras (e.g., "ContactoPessoalBE" da Figura 5) agregam várias. Nesta última situação é conveniente adicionar-se alguma semântica às relações, a qual é suportada através de duas marcas: `role` (que pode ter o valor "*master*", "*detail*", "*lookup*", "*nlookup*"), que serve para distinguir a entidade principal (*master*) das restantes (*detail*), ou eventualmente especificar relações de consulta ou navegação ("*lookup*" ou "*nlookup*"); e `index`, que serve para identificar a ordem de apresentação das entidades do tipo *detail*. De forma opcional, apenas para efeito de melhor compreensão do modelo visual, pode-se explicitar no nome das relações o papel da entidades participantes (tal como sugerido na Figura 5 relativamente às relações entre "ContactoPessoalBE" e as demais entidades envolvidas).

Modelo dos *Worflows* (Vista de Controlo, «XisControlView»)

No modelo dos *workflows* (ou "vista de controlo") o principal objectivo é a definição de classes do estereótipo «xis-ui-workflow», i.e., de controladores que especificam *workflows* de interface com o utilizador, o que corresponde essencialmente à especificação de um diagrama de estados com: (1) estados (i.e., classes do estereótipo «xis-ui-state») correspondentes aos estados de um *workflow* de interface com o utilizador; e (2) transições (i.e., classes do estereótipo «xis-ui-transition») correspondentes às transições entre estados. Uma transição é constituída por: um evento que a faz disparar («xis-ui-event»); uma condição («xis-ui-condition») que tem de ser válida para que a transição ocorra; e uma lista de operações («xis-ui-operation») que poderão ser realizadas na transição.

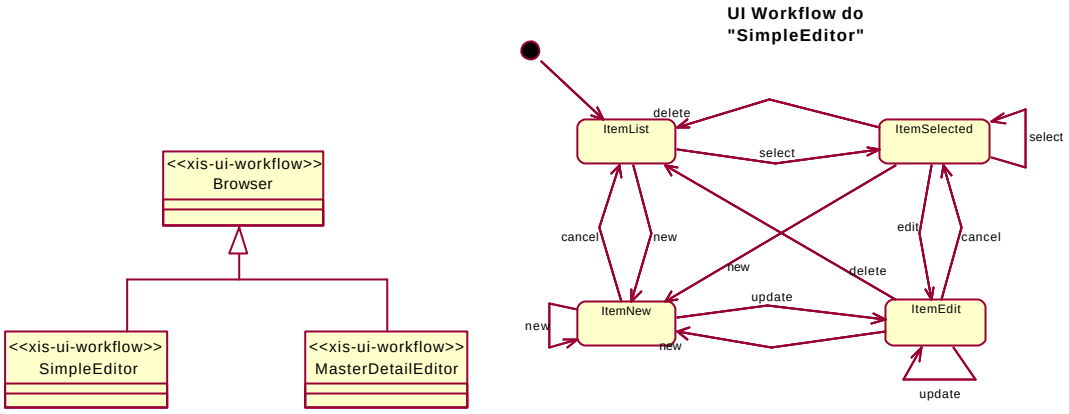

Figura 6: Detalhes do workflow "SimpleEditor", constituinte do modelo dos *workflows*.

Podem-se definir hierarquias de tipos de *workflows* de interface com o utilizador, conforme sugerido na parte esquerda da Figura 6, sendo que cada tipo é especificado graficamente através de um diagrama de estados (por exemplo, como é sugerido na parte direita da Figura 6, em que se apresenta o diagrama de estados associados ao tipo `SimpleEditor`).

Modelo das Vistas (Vista de Vista, «XisViewView»)

No modelo das vistas (ou "vista de vistas") o objectivo é a definição de classes do estereótipo «xis-ui-view». Objectos deste estereótipo são responsáveis pelo estabelecimento de associação entre uma (ou mais) entidade(s) de negócio e um controlador de *workflow* de interface com o utilizador.

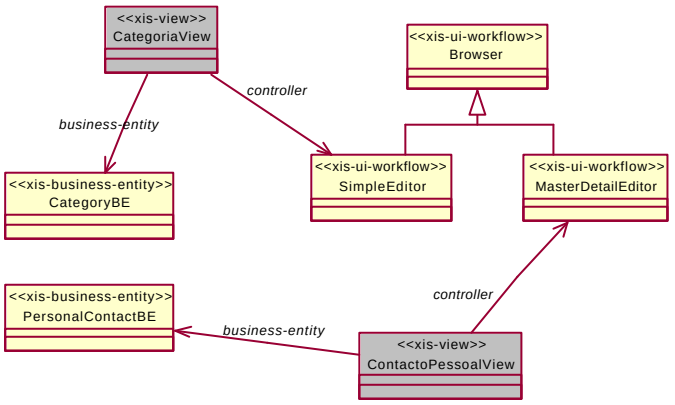

Figura 7: Uma vista do modelo das entidades de negócio do sistema MyContacts.

As vistas funcionam como objectos "cola" entre as entidades de negócio e os *workflows* de interface com o utilizador predefinidos no modelo de *workflows*. Note-se que os detalhes específicos das vistas, na perspectiva da disciplina de interacção homem-máquina (e.g., mapas de navegação entre formulários, estilos de apresentação, cores), não são suportados nem representados nos modelos XIS/UML. Tais detalhes são suportados directamente, de distintas formas, ao nível das arquitecturas de software.

3.3 Linguagem XMI

O XMI (*XML Metadata Interchange*) é o standard OMG para representação de metadata [OMG 2002]. O XMI foi definido originalmente para representar metadata e os dados

correspondentes aos modelos especificados em UML. No entanto, encontram-se em curso iniciativas para o adoptar noutros domínios de aplicação, tais como *datawarehousing* e componentes de software.

No respeitante à modelação, o XMI especifica uma estrutura de representação de modelos UML conforme o meta-modelo UML. O principal objectivo do XMI é permitir a interoperação e utilização dos modelos UML de forma independente das plataformas, linguagens, repositórios e ferramentas CASE.

3.4 Linguagem XIS/XML

O XMI tem a virtude de permitir representar modelos UML num formato XML, de modo independente de ferramentas e de repositórios. Apresenta todavia alguns inconvenientes fundamentais, designadamente: o tamanho, a fragmentação e a complexidade da representação da informação envolvida. Isto é, o XMI representa um modelo UML baseando-se integralmente nos elementos definidos no meta-modelo UML, gerando uma representação completa mas extensa e fragmentada (a definição dos elementos do modelo apresenta mecanismos de indirecção suportado por atributos do tipo `xmi.id` e `xmi.idref`) difícil de compreender por humanos e de ser processado por ferramentas computacionais.

Para além destes inconvenientes, os documentos XMI exportados por diferentes ferramentas CASE (ou mesmo diferentes versões da mesma ferramenta CASE) não representarem o mesmo modelo exactamente da mesma forma, por serem implementados diferentes dialectos de XMI. Este facto da existência de diferenças ao nível da implementação da gramática do XMI, tornam complexa a adaptação de uma nova ferramenta CASE à plataforma XIS. Neste sentido foi definido uma representação intermédia da linguagem XIS em XML, designada por XIS/XML, que adaptasse a impedância entre o XMI e a plataforma XIS. Dominando os meta-modelos do XMI e do XIS e usando XSLT e XPath é possível escrever transformações entre os documentos XML respectivos que são adaptáveis entre vários dialectos e versões de XMI.

A simplicidade da representação da linguagem XIS/XML permite, por exemplo, que um modelo de um pequeno sistema de informação seja rapidamente especificado, manualmente, sem o auxílio a ferramentas exteriores de modelação, sendo importados directamente para o repositório da aplicação.

4 XIS é Centrado em Arquitecturas de Software

4.1 Arquitecturas de Software

O desenvolvimento de software centrado numa arquitectura é reconhecido como sendo uma boa prática tendo em conta o crescente nível de complexidade e dimensão dos sistemas [Gamma et al. 1994],[Hofmeister et al. 1999],[SPS --],[Juric et al. 2002]. Essencialmente, a função da arquitectura de software é de reflectir decisões sobre a organização do sistema de software, nomeadamente decisões relativas: (1) aos elementos estruturais que compõem o sistema; (2) ao seu comportamento, especificado através de colaborações entre eles; (3) à composição dos elementos estruturais e de comportamento em subsistemas progressivamente maiores; e (4) ao estilo arquitectural, que guia esta organização, os seus elementos, as suas colaborações e a sua composição. A arquitectura de software decide ainda sobre restrições e compromissos relativos a diferentes aspectos do sistema, tais como: utilização, funcionalidades, desempenho, tolerância a alterações, reutilização, compreensão, economia e estética.

A arquitectura de software é representada fisicamente através de um conjunto de vistas sobre distintos tipos de modelos, elaborados quer em função dos intervenientes e destinatários, quer da

fase/tarefa do processo de desenvolvimento. Assim, podemos ter a visão do modelo de casos de utilização; a visão do modelo de análise; a visão do modelo de desenho; a visão do modelo de implementação; e a visão do modelo de instalação. Estas diferentes vistas podem evidenciar o "estilo arquitectural" que o sistema deverá apresentar.

Quando nos referimos à noção de arquitectura de software, no estado actual do projecto XIS, estamos a circunscrevê-la basicamente às vistas de análise e desenho (constituirá trabalho futuro estudar e alargar o alcance desta noção). Neste âmbito, o foco principal incide na análise e aplicação de *frameworks* (infra-estruturas) de software. Uma infra-estrutura de software consiste num conjunto de classes cooperantes que providenciam funcionalidades comuns e reutilizáveis para um determinada família de (sistemas de) aplicações. Uma infra-estrutura providencia mecanismos para criação de aplicações particulares a partir da especialização de aplicações reutilizáveis e semi-completas. Exemplos de infra-estruturas são, entre outras, (1) o J2SE, ou o J2EE da Sun, ou o .NET da Microsoft para desenvolvimento de aplicações com interface Windows ou Web, com requisitos *desktop* ou empresariais; ou (2) o J2ME da Sun ou o .NET Compact Framework da Microsoft, para desenvolvimento de aplicações para dispositivos móveis de reduzidas dimensões.

Uma infra-estrutura de software define a arquitectura de um aplicação, ou seja, define a estrutura e organização global da aplicação, define como as classes e objectos podem colaborar, e define os próprios fluxos de execução possíveis. Uma infra-estrutura predefine os principais parâmetros do desenho de uma aplicação, deixando apenas para o analista/programador de uma aplicação as tarefas exclusivamente específicas da aplicação. Desta forma a ênfase de uma infra-estrutura consiste na reutilização do desenho de famílias de aplicações.

4.2 Arquitecturas de Software Disponibilizadas na Plataforma XIS

Conforme sugerido inicialmente na Figura 2 é da responsabilidade do arquitecto a definição de arquitecturas de software e o seu respectivo registo na plataforma XIS. Por outro lado, é da responsabilidade do programador escolher uma determinada arquitectura para proceder à transformação T3. A ideia geral é que se possam definir as arquitecturas que se quiser, registando-as num correspondente catálogo de arquitecturas e respectivas transformações, sendo que com mais e melhores arquitecturas na plataforma, a abordagem XIS será naturalmente mais poderosa e produtiva.

O catálogo de transformações actualmente permite a geração de modelo de dados, e implementações em arquitecturas multi-camada de sistemas interactivos com interface Web, usando as infra-estruturas de software J2SE e .NET (Sai fora dos objectivos do artigo uma análise mais detalhada sobre este tópico.)

5 XIS é Baseado em Técnicas de Geração Automática

A terceira boa prática adoptada no contexto do projecto XIS preconiza a adopção de técnicas de geração automática [Cleaveland 2001] e [GPG --]. Esta boa prática está intimamente relacionada com as referidas nas secções anteriores, funcionando de alguma forma como a "cola" ou o seu elemento integrador. Ou seja, a geração automática transforma a especificação do sistema (representada através de modelos descritos gráfica e ou declarativamente, ver Secção 3) num conjunto de artefactos constituintes do sistema final, tendo em conta uma determinada arquitectura de software (representada de forma imperativa numa determinada linguagem de programação, ver Secção 4). Um aspecto relevante desta abordagem é a possibilidade dos geradores produzirem código optimizado (e.g., em termos de desempenho),

para diferentes arquitecturas de software, para diferentes linguagens de programação, e com maior ou menor grau de flexibilidade.

5.1 Transformações XIS: Dos modelos XIS/UML ao Código Fonte

A Figura 2 apresentada anteriormente sugere o processo de geração automática providenciado pela abordagem XIS, o qual consiste efectivamente na aplicação sucessiva de três transformações, conforme formalizado pela seguinte expressão:

```
IS = T3(T2(T1(XIS/UML-Model, XIS/UML-Profile), XIS/XSL), SoftArch)
```

Ou separadamente, por:

```
XIS/UML-Model      (definido através de uma ferramenta CASE e baseado no perfil XIS/UML)
XMI-Model          = T1(XIS/UML-Model, XIS/UML-Profile)
XIS/XML-Model      = T2(XMI-Model, XIS/XSL)
IS (FinalSystem)   = T3(XIS/XML-Model, SoftArch)
```

Onde:

- **`XIS/UML-Model`**: É o modelo original do sistema especificado de acordo o perfil XIS/UML (i.e., `XIS/UML-Profile`) e desenhado através de uma ferramenta CASE (e.g., Rational Rose [Rational --].
- **`T1`**: É a primeira transformação; consiste na transformação do modelo `XIS/UML-Model` num equivalente no formato XMI. Quaisquer *parsers* XMI, como por exemplo o XMIToolkit [IBM --], disponibilizados pela generalidade das actuais ferramentas CASE, suportam naturalmente a transformação T1.
- **`T2`**: É a segunda transformação; consiste na transformação de modelos especificados segundo o XMI para modelos equivalentes de acordo com o formato XIS/XML. Esta segunda transformação é conveniente devido ao facto dos modelos especificados em XMI serem muito extensos, complexos e difíceis de entender e de manipular (ver discussão da Secção 3.4). T2 é realizado por um *parser* XSLT com base em scripts `XIS/XSL`.
- **`T3`**: É a terceira transformação; consiste na geração de múltiplos artefactos a partir dos modelos especificados em XIS/XML e tendo em conta uma determinada arquitectura de software (`SoftArch`). O gerador de código, XISGenerator, apresenta as propriedades de ser genérico, modular e versátil, e é o elemento chave da transformação T3.
- **`IS`**: É o sistema de informação final, que consiste na adequada integração e utilização dos múltiplos artefactos gerados(e.g., ficheiros de código Java, JSP, HTML, ant-build, XML, scripts SQL, .ASPX, .CSPX, ou de documentação).

5.2 Aspectos da Transformação T3

É evidente que existem vários aspectos relevantes nas transformações referidas, em particular na transformação T3, que merecem uma explanação mais detalhada. No entanto, por motivos de enquadramento, tal deverá ser descrito em artigo futuro. Referimos aqui apenas dois aspectos

relevantes relativamente à transformação T3: repositório de suporte; e geração iterativa e incremental.

Repositório de Suporte

Embora sugerido visualmente na Figura 2 (por motivos de simplicidade), na realidade a transformação T3 não é executada directamente a partir da especificação XIS/XML. De facto, o processo T3 é mediado com informação mantida no repositório XIS. Ou seja, T3 deve ser visto mais correctamente pela aplicação de dois sub-processos sucessivos. Primeiro, um processo relativamente simples (T3.1), responsável por carregar o repositório XIS a partir da especificação XIS/XML. Segundo, um processo bastante mais complexo (T3.2), responsável pela geração dos artefactos de software constituintes de um determinado sistema de informação. O processo T3 é por conseguinte baseado na informação mantida no repositório XIS e na arquitectura de software previamente seleccionada.

O repositório XIS é suportado por um SGBD relacional, apresentando um modelo de dados relativamente complexo para suportar de forma flexível o armazenamento e gestão de uma variedade de elementos, designadamente:

- Os elementos constituintes do metamodelo XIS (i.e., definidos quer ao nível do perfil XIS/UML quer ao nível do XIS/XML), como por exemplo: entidades, atributos, enumerados, controladores, transições, eventos, ou entidades de negócio.
- Os elementos de suporte ao próprio processo de geração, tais como: arquitecturas, aplicações, processos e etapas de geração, opções de geração, artefactos, tipos de artefactos.

Geração Iterativa e Incremental

Um dos problemas das abordagens clássicas de geração de artefactos reside no facto de nem sempre se conseguir delimitar a abrangência da sua acção, adoptando-se geralmente estratégias do tipo "tudo-ou-nada". Ou seja, quando se conclui que a especificação original não estava correcta (ou quando deixou de estar porque os requisitos foram alterados...) a abordagem clássica consiste na re-geração de "todos" os artefactos envolvidos a partir das novas versões das especificações. Esta estratégica pode funcionar adequadamente em sistemas relativamente pequenos e com poucas dependências. Mas em sistemas maiores, constituídos por múltiplas componentes partilhadas, a situação poderá tornar-se facilmente incontrolada. De forma a contornar este tipo de problemas, o processo de geração do XIS preconiza uma abordagem mais fina, do tipo iterativa e incremental, consistindo na prática na possibilidade do arquitecto ou do programador poder definir o processo de geração à custa de várias etapas, em que para cada uma são determinados quais os artefactos/componentes correspondentes que deverão ser alvo de geração.

6 Conclusões e Trabalho Futuro

A ideia subjacente no projecto XIS não é nova, é de facto uma revisitação de anteriores expectativas que no passado não tiveram muito sucesso: **a ideia de que a construção de sistemas de informação fosse realizada quase automaticamente a partir de especificações relativamente abstractas e de alto nível**. Ou seja, no limite, que a tarefa clássica de programação fosse realizada automatica e não manualmente como actualmente e na generalidade sucede, com todos os custos e problemas conhecidos.

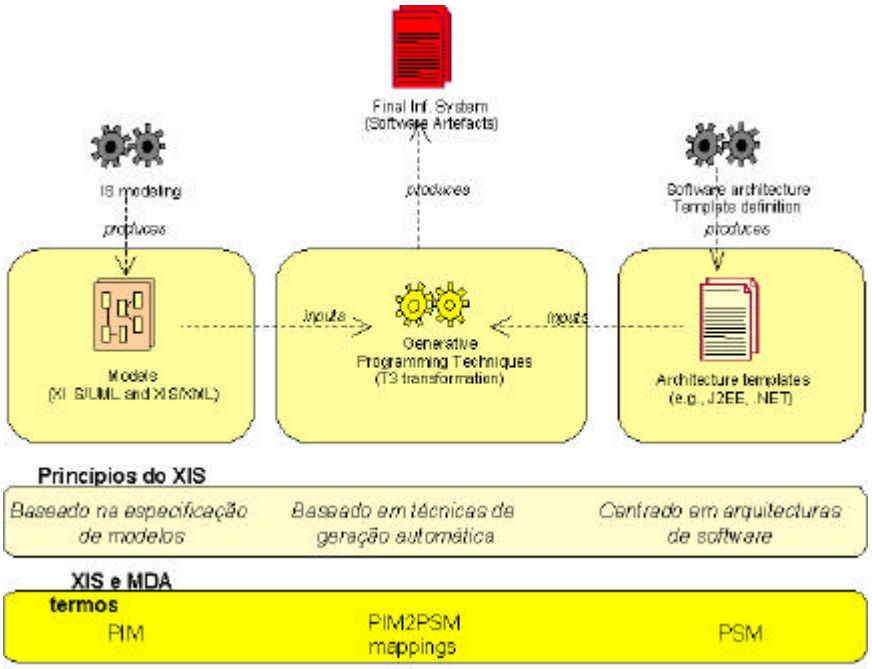

Figure 8: Relações entre o projecto XIS e o paradigma MDA.

Esta abordagem de construção de software é designada por "desenvolvimento de sistemas baseado em modelos" e acreditamos que **desta vez** seja possível concretizar o sonho tendo em conta o estado de maior maturidade da engenharia e das respectivas tecnologias, em particular do UML, do XML e suas linguagens, e das boas práticas de padrões e de arquitecturas de software que têm vindo a emergir. Esta abordagem tem vindo a ser discutida e promovida pela OMG no contexto do grupo de trabalho do MDA (*Model Driven Architecture*) tendo vindo a ser analisada e suportada progressivamente pelas principais empresas da indústria. Todavia, existem diferenças entre MDA e a aproximação XIS que merecem ser discutidas convenientemente. Primeiro, o MDA é um modelo de referência de alto nível ou paradigma enquanto que XIS é um projecto concreto com uma proposta de aproximação, linguagens específicas e ferramentas de suporte. Segundo, o MDA recomenda a definição de dois tipos de modelos – PIM (*platform independent models*) e PSM (*platform specific models*) –, bem como as respectivas transformações PIM2PIM, PIM2PSM e PSM2PSM. Na aproximação XIS existem apenas modelos PIM, os quais embora sejam de desenho, são especificações de sistemas de informação independentes da plataforma de desenvolvimento, seguem o padrão MVC e podem ser representados através do perfil XIS/UML ou directamente através da linguagem XIS/XML. A transformação dos modelos XIS para uma determinada plataforma e linguagem de programação é definida, conforme ilustrado na Figura 2, pelos arquitectos quando definem os *templates* correspondente a uma determinada arquitectura de software, e finalmente pelos programadores quando selecionam os modelos e os *templates* necessários para desencadear a transformação T3 correspondente.

A Figura 8 sintetiza a aproximação XIS, sendo evidenciado os seus princípios e relações relativamente à terminologia MDA.

O projecto XIS tem como objectivo principal servir de contexto para a realização de vários trabalhos de investigação a desenvolver estrategicamente num período de 2 a 4 anos. Neste âmbito foi recentemente concluído um trabalho final de curso e uma tese de mestrado, encontrando-se em estudo e arranque trabalhos de mestrado e doutoramento que irão continuar e aprofundar os objectivos e a visão geral aqui introduzidos. Complementarmente, pretende-se a

médio prazo vir a aplicar os resultados do projecto XIS na realização de outros projectos concretos, de índole de desenvolvimento e de consultoria.

Actualmente o projecto têm vindo a desenvolver a sua investigação com um significativo enfoque em sistemas de informação empresariais e distribuídos em larga escala. Há, no entanto, inúmeros aspectos que serão alvo de análise e investigação em trabalho futuro, designadamente: (1) alargar o suporte a outras arquitecturas de software, quer derivadas do Java (e.g., derivadas do J2SE, J2EE e J2ME) quer do .NET da Microsoft; (2) mecanismos para especificar e produzir vistas correspondentes a interacções homem-máquina, de forma mais flexível do que actualmente acontece; (3) mecanismos para especificar e produzir vistas correspondentes a interfaces de comunicação entre aplicações, tipicamente recorrendo-se às tecnologias dos *Web Services* e de *workflow* de aplicações; e (4) formalizar e melhorar as capacidades providenciadas pela plataforma XIS, permitindo integrá-la com capacidades de gestão de projecto, gestão de requisitos e de suporte iterativo e incremental aos mecanismos de geração, teste e integração de componentes de software.

Concluindo, deve ser sublinhado que **XIS não é um plano de investigação conceptual; é um projecto em desenvolvimento já com resultados concretos e sistemas produzidos**. O leitor interessado é convidado a contactar o autor ou a visitar o *web site* do projecto [GSI--] para mais informações, por exemplo, para acesso à definição integral e formal do perfil XIS/UML.

Referências

AgileAlliance, *Manifesto for Agile Software Development*,2000. http://www.agilemanifesto.org

Booch, G., Rumbaugh, J., Jacobson, , I., The Unified Modeling Language User Guide, Addison-Wesley, 1999.

Buschmann, F., Meunier, R., Rohner, H., Sommerlad, P., Stal, M. *Pattern-Oriented Software Architecture – A system of patterns.* Volume 1. Wiley. 1996.

Cleaveland, J. C., *Program Generators with XML and JAVA*, Prentice-Hall, 2001.

Curbera, F., et al., *Business Process Execution Language for Web Services*, Version 1.0, 2002. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Gamma, E., Helm, R., Johnson, R., Vlissides, J., *Design Patterns – Elements of Reusable Object-Oriented Software*, Addison Wesley, 1994.

GPG, Generative Programming Group. http://www.generative-programming.org

Grasner, G., Pope, S., "A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80". *Journal of Object-Oriented Programming*, 1 (3), 1988.

GSI, Grupo de Sistemas de Informação, INESC-ID, *The XIS Project*, http://berlin.inesc-id.pt/projects/xis/

Hofmeister, C., Nord, R., Soni, D., *Applied Software Architecture*, Addison Wesley, 1999.

IBM, IBM AlphaWorks, *XMIToolkit*. http://www.alphaworks.ibm.com/tech/xmitoolkit/

Juric, M., et al., *J2EE Design Patterns Applied*, Wrox Press, 2002.

OMG, Object Management Group. http://www.omg.org

OMG, *White Paper on the Profile mechanism, Version 1.0, OMG Document ad/99-04-07*, 1999.

OMG, *Model Driven Architecture - A Technical Perspective*, 2001. http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01

OMG, *XML Metadata Interchange (XMI®)Version1.2, 2002.* http://www.omg.org/cgi-bin/doc?formal/2002-01-01

Orlikowski, W. J., *CASE tools as Organizational Change: Investigating incremental and radical changes in systems development*, MIS Quarterly, vol. 17, no. 3, 1993.

Rational, Rational Rose, http://www.rational.com

Silva, A.R. da, Videira, C., *UML, Metodologias e Ferramentas CASE*. Centro Atlântico (Portugal), 2001.

SPS, *The Software Patterns Series*, Addison Wesley, 1996-2002.

UIML.org. *UIML v2.0 Draft Specification* , 2000. http://www.uiml.org/

Vessey, I., Jarvenpaa, S. L., Tractinsky, N., *Evaluation of Vendor Products: CASE Tools as Methodology Companions*, Communications of the ACM, vol. 38, no. 1, 1995.

W3C, *Extensible Markup Language (XML) 1.0*, 1999. http://www.w3.org/TR/REC-xml

Nota: As referências electrónicas acima referidas foram acedidas com sucesso entre Maio e Julho de 2003.