# ARQUITECTURAS DE SISTEMAS DE INFORMAÇÃO E A INICIATIVA MDA

MARCO COSTA* e ALBERTO RODRIGUES DA SILVA**

**ABSTRACT**

Information Systems Architecture is a well-known field of research for more than twenty years, but many of the identified problems still exist. One of key issue is the absence of a generative programming technique that could include an architecture plan as input and generate a set of useful specifications and code directly implemented by programmers. In this paper we compare several existent conceptual approaches to information systems architectures like Index, Zachman and CIMOSA. We also describe the Object Management Group MDA approach and two concrete initiatives, Codagen and XIS.

**RESUMO**

A Arquitectura de Sistemas de Informação é uma área de investigação há mais de duas décadas. No entanto alguns dos problemas identificados ainda subsistem. Continua a procurar-se uma ferramenta que consiga produzir, a partir de um conjunto de planos arquitecturais, um conjunto de especificações e código. Este artigo apresenta uma comparação e discussão entre diversas abordagens existentes para a formalização de arquitecturas de sistemas de informação, designadamente: Index, Zachman e CIMOSA. Introduz também a abordagem MDA promovida pelo Object Management Group, e com base nela os projectos Codagen e XIS como exemplos concretos.

## 1 Introdução

O crescimento do papel das tecnologias da informação nas organizações tem levado à introdução de um número crescente de tecnologias e metodologias que são usadas para resolver problemas específicos. As organizações têm sido vistas com frequência como um conjunto de subsistemas cuja interligação e integração não têm sido realizadas com grande sucesso. Tem-se desenvolvido abordagens específicas para a implementação de projectos de ERP, CRM, CIM, Data Warehousing, etc. No entanto, cada uma destas abordagens é desenquadrada ou não alinhada com os objectivos concretos das organizações. Além disso, quando existe alguma interligação entre os subsistemas das organizações, estas não são capazes de se

* Assistente na Univ. Independente, Doutorando do Inst. Superior Técnico. marco.costa@tagus.ist.utl.pt

** Professor de Sistemas de Informação no Instituto Superior Técnico. Investigador Sénior no INESC-ID. alberto.silva@acm.org

relacionarem entre si, sejam elas parceiras, clientes, fornecedoras, etc. Este problema cuja importância assumiu evidente importância no início da década de 1990 levou a uma tomada de consciência por parte do Governo Federal dos EUA que culminou no Clinger-Cohen Act de 1996 [1]. Com esta lei o Congresso dos EUA obriga as agências federais a desenvolverem e a manterem uma arquitectura do sistema de informação integrada. Surgiu assim a necessidade de encontrar meios que permitam descrever as arquitecturas não só do sistema de informação – na medida em que este não está isolado – como da própria organização. Existem diversas infra-estruturas conceptuais já clássicas, como as matrizes de Zachman [2] e Index [3], ou o modelo CIMOSA [4], que tentam segmentar diversos níveis de abstracção nas suas constituintes.

O aparecimento deste tipo de modelos colocou a descoberto a dificuldade em conceber e utilizar uma Arquitectura do Sistema de Informação (ASI), o que representa um problema essencial não só para o sistema de informação, como para a própria organização. Existem razões para este facto que abordaremos na segunda secção. Na terceira secção far-se-á uma breve descrição de algumas infra-estruturas conceptuais para ASI, procurando realçar os seus atributos mais importantes. Este estudo será necessário para se compreender, na quarta secção, os pressupostos que servem de base à nova proposta da OMG neste campo, o MDA.

O estudo e utilização da ASI tem como possível consequência a produção de artefactos que possibilitam um desenvolvimento mais rápido e uma manutenção mais simples das aplicações da organização. Uma das formas de conseguir um desenvolvimento mais rápido é a geração automática de uma parte do código da aplicação enquadrada no ramo da Informática denominado "Programação Generativa". Através de diversas técnicas consegue-se já determinar aspectos essenciais da aplicação, mesmo antes de se saber em que linguagens de programação, sistemas operativos ou componentes de hardware esta será implementada [15 e 16]. Nas secções 5 e 6 serão descritas duas abordagens (Codagen e XIS) com fortes componentes de programação generativa baseada em ASI e de acordo com a MDA. Por fim, na secção 7 apresentam-se as principais conclusões e reflexões deste trabalho.

## 2 Arquitecturas e Organizações

Embora existam diversas definições consideraremos que "um sistema de informação é um conjunto integrado de recursos (humanos e tecnológicos) cujo objectivo é satisfazer adequadamente a totalidade das necessidades de informação de uma organização e dos respectivos processos de negócio "[5]. Esta definição sublinha o facto de um sistema de informação existir para satisfazer necessidades da organização, podendo

esta ser uma empresa, um organismo público, uma cooperativa, etc., ou num sentido mais lato uma qualquer associação destas entidades. As tecnologias da informação deixaram já o contexto departamental, onde nasceram, para passarem a enquadrar-se no contexto multiorganizacional actual.

Consideramos uma arquitectura a "estrutura dos componentes, as suas relações e os princípios e normas que governam o seu desenho e a sua evolução através do tempo"(IEEE 610.2). Sendo assim, a arquitectura de um sistema de informação será constituída por um conjunto de artefactos (esquemas, listas de definições, descrições textuais, etc.) que revelam a estrutura dos componentes desse sistema de informação, as suas relações e os princípios e normas que governam o seu desenho e a sua evolução através do tempo. Se um pequeno subsistema de informação da organização pode levantar problemas de concepção, implementação e manutenção importantes, num contexto interorganizacional esta situação pode levar ao caos a que assistimos com demasiada frequência. Para resolver este problema foram propostas aquilo a que chamaremos de **infra-estruturas conceptuais para arquitecturas de sistemas de informação (ICASI)**, referidas por vezes também como *frameworks*, e das quais daremos exemplos posteriormente. Por vezes as ICASI são confundidas com as próprias ASI que podemos considerar como instâncias das primeiras.

Existem diversas razões para realizar uma ASI, entre as quais destacamos:

- *Permite colocar ordem no caos*. Sendo um modelo que decompõe uma realidade complexa em partes mais simples e explicitando as suas relações pertinentes, consegue diminuir-se a complexidade da abordagem ao sistema.
- *Fornece uma infra-estrutura para o desenvolvimento e a operação*. Só depois de compreendido o sistema se consegue tratá-lo de forma correcta, eficiente e económica.
- *Prepara para alterações dos requisitos ou de abordagens da implementação*. Uma ASI bem constituída, associada a uma metodologia correcta, torna o processo de mudança mais simples.
- *Permite que cada elemento (individual ou colectivo) esteja enquadrado relativamente ao projecto, ao executar a sua tarefa.* Quando cada elemento da organização está motivado e conhece os objectivos que a organização quer atingir com a sua actividade, o seu desempenho é melhor e o projecto é finalizado mais depressa e com menores custos.
- *Estimula a reutilização.* A partir de uma ou mais soluções podem ser encontrados padrões arquitecturais que resolvem outros problemas que venham a existir, ou pelo menos a chegar a outras soluções mais depressa.

Se estas vantagens são evidentes porquê continuarmos a encontrar grandes organizações (para não referir as mais pequenas) sem uma verdadeira ASI? Zachman apontou diversas razões [6] que comentaremos de seguida:

- *A arquitectura é contracultural*. Até agora o grande objectivo era produzir código executável e colocar o sistema a funcionar. As métricas existentes para a produção nesta área assentam em conceitos como linhas de código, pontos de função, tempo do sistema em execução, tempo de resposta, custos, orçamento, etc. Como normalmente apenas se dá valor àquilo que directamente e momentaneamente pode resolver um problema, tem sido difícil atribuir um valor à arquitectura, seja ele financeiro, de importância para a organização, etc.
- *Não é entendida como sendo uma questão de sobrevivência da organização.* Os horizontes tácticos e estratégicos são muitas vezes preteridos em detrimento das actividades operacionais. O facto de se associar erroneamente apenas ao horizonte estratégico ou táctico uma ASI leva a que seja vista como um actividade de segundo plano.
- *Falta de conhecimento metodológico.* Apesar de todas as tentativas ainda não existe um processo padrão, estável e pacificamente aceite para realizar uma ASI. Este facto contribui para uma disseminação de abordagens, modelos, processos, metodologias e notações que apenas aumentam a dificuldade para abordar estes temas, quando o deviam diminuir. Levam por isso, para nós, a um afastamento de grande parte dos recursos humanos que deveria entender estes aspectos como vitais, não como mero *assunto académico...*
- *Demora muito e exige muito trabalho.* Apesar de actualmente se procurar uma solução rápida para tudo, este não é um assunto que se resolva com uma simples ferramenta e um utilizador hábil... Necessita, pelo contrário, de um esforço colaborativo dos mais diversos sectores da organização, o que leva tempo e pode envolver um grande investimento, dependendo da dimensão e estrutura da organização.

Uma ASI, como foi referido, deve estar de acordo, ou como é comum dizer, "deve estar alinhada", com os objectivos da organização onde se insere. Numa perspectiva sistémica, o sistema de informação, enquanto subconjunto do sistema organizacional (organização) deve contribuir para que sejam atingidos os objectivos desta. Faremos em seguida uma referência a diversas ICASI, colocando em relevo aspectos que as distinguem e nos podem ajudar a escolher uma delas ou a criar uma nova.

# 3 Infra-estruturas conceptuais para arquitecturas de sistemas de informação

## *3.1 Framework de Index*

Segundo esta ICASI pode dividir-se uma ASI em quatro grandes conjuntos de artefactos: Arquitectura Aplicacional, Arquitectura de Dados, Arquitectura Tecnológica e Arquitectura Organizacional. A Figura 1 ilustra a forma como os elementos da ASI devem suportar os objectivos do negócio (entendido aqui como a organização). A Arquitectura Aplicacional define o conjunto de sistemas e aplicações da ASI. A Arquitectura de Dados define os conceitos e entidades necessárias à execução dos processos de negócio da organização. A Arquitectura Tecnológica indica os componentes de infra-estrutura e máquinas necessários para suportar as funcionalidades e requisitos das aplicações identificadas.

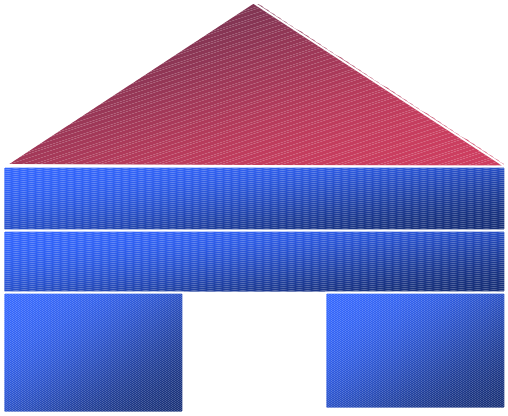


*Figura 1. Constituintes de uma ASI e objectivos de uma organização (Framework de Index) [4]*

A Arquitectura Organizacional representa a estrutura dos recursos humanos necessários para suportar adequadamente os restantes componentes dos sistemas de informação. A framework de Index cruza estes quatro conjuntos de elementos com outras quatro visões: Inventário, Princípios, Modelos e Standards, obtendo-se uma matriz com dezasseis células [18].

***Tabela 1. Framework de Index***

| | Inventário | Princípios | Modelos | Standards |
|---|---|---|---|---|
| Infra-estrutura | | | | |
| Dados | | | | |
| Aplicações | | | | |
| Organização | | | | |

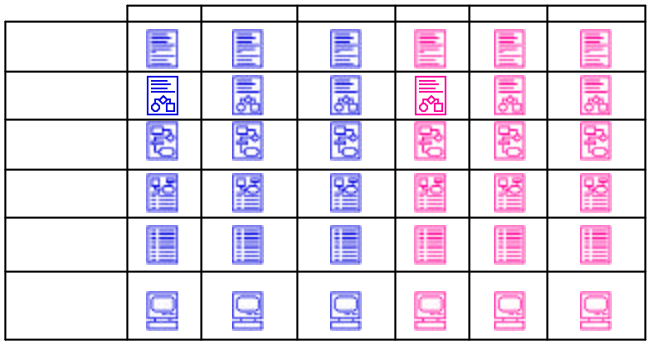
A visão Inventário define as características actuais do sistema de informação, tal como um inventário usual. Os Princípios definem as regras que são seguidas para guiar a tomada de decisão e para motivar a colaboração entre os constituintes do sistema de informação. Os Modelos são diagramas e outros artefactos, utilizados para ilustrar o conteúdo de cada linha. Os Standards compreendem uma descrição do processo aceite para a execução de cada uma das linhas.

### *3.2 Framework de Zachman*

Uma outra ICASI conhecida, a Framework de Zachman, foi proposta em 1987 e revista na sua forma actual em 1992. Embora também seja representada por uma matriz, as suas colunas podem-se considerar como uma extensão das linhas da Matriz de Index e as suas linhas representam seis perfis de intervenientes que se relacionam com o sistema. As últimas três colunas *(People, Time* e *Motivation)* foram acrescentadas mais tarde, em 1992.

Esta framework tem como inegável mérito uma arrumação das diversas partes de um problema em cada uma das células da matriz. A matriz não define as notações envolvidas em cada uma das células ou a relação entre estas. O próprio autor da framework [6] considera que algumas das células presentes nas colunas 3 e 4 não têm ainda descrições muito estáveis.

***Tabela 2. Framework de Zachman***



A framework de Zachman é criticada frequentemente por perpetuar uma separação entre dados e funções mesmo a um nível próximo da execução. Esta distinção está longe das abordagens orientadas por objectos que se utilizam hoje. Apesar de a framework de Zachman não obrigar à utilização de nenhuma notação ou metodologia de sistemas de informação, este aspecto pode condicionar a escolha de uma abordagem orientada por objectos por qualquer um dos utilizadores do sistema (linhas da matriz).

O facto desta ICASI ser instanciada num conjunto de práticas e diagramas definidos no contexto de metodologias que lhe são externas faz com que a ligação entre os diagramas gerados possa ser dificultada. A

*framework* de Zachman é suficientemente aberta para permitir a utilização de diversas metodologias, mas este facto torna difícil a sua implementação por uma ferramenta comercial devido à dificuldade que muitas vezes existe de mapear os conceitos entre metodologias.

### *3.3 Framework CIMOSA*

Em 1985 um consórcio de diversas empresas, denominado AMICE[4], do qual faziam parte empresas como a IBM, Aerospatiale, a British Aerospace e a Daimler Benz, entre outras, iniciou um projecto que visava a especificação de uma arquitectura de sistemas abertos para CIM (Computer Integrated Manufacturing) no âmbito da iniciativa ESPRIT da UE. Em 1990 iniciou-se a validação dos resultados entretanto obtidos, através de três projectos independentes (CIMPRES, CODE, VOICE).

Através desta framework aborda-se a organização, não só naquilo que diz respeito ao seu sistema de informação, como à própria integração entre este e o sistema de produção e a organização. Simplificando, pode-se afirmar que máquinas, computadores e pessoas ficam a pertencer a uma arquitectura comum.

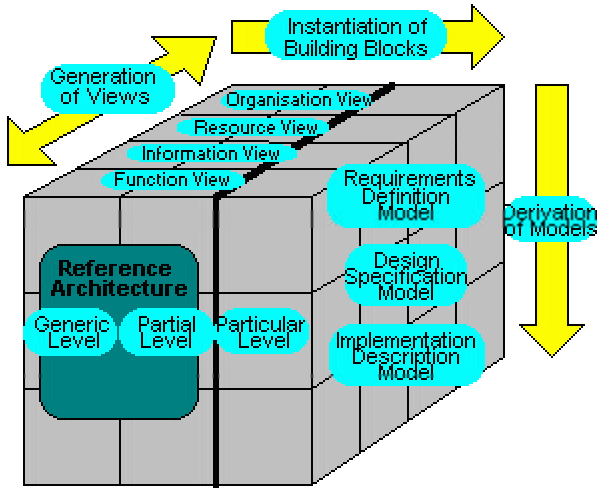



*Figura 2. A Framemework CIMOSA [4]*

Embora esta framework tenha objectivos mais alargados, outro dos problemas fundamentais que tenta resolver é o da comunicação entre organizações, ou partes desta. Esta questão assumia maior relevância em contextos como as indústrias aeronáutica e automóvel onde existia a necessidade de um esforço colaborativo entre um grande número de organizações. O CIMOSA define não só uma forma de modelar a organização, em todas as suas vertentes, como define os mecanismos para a partilha de informação e metainformação entre organizações. Não se tratava apenas de construir um modelo de uma organização, ou do seu sistema de informação, como de garantir que este modelo poderia ser entendido por outra organização relacionada (cliente, parceira, fornecedora, etc).

Para além destes aspectos, o CIMOSA deu também um contributo importante ao prever um modelo tridimensional (apresentado na Figura 2) que inclui três dimensões, em vez de duas nos modelos anteriores.

A dimensão designada por Instanciação é talvez a mais particular a este modelo porque tenta dar uma solução para o facto de em cada organização se tentar resolver problemas que já podem ter sido algures resolvidos com êxito. Este esforço constante é ultrapassado por alguns adaptando projectos já existentes. Consoante a qualidade da documentação e a generalidade da solução inicial pode ou não conseguir-se um resultado satisfatório, muitas vezes com um custo similar à construção de raiz. O CIMOSA prevê uma Arquitectura de Referência, aceite pela entidade responsável pelo standard (CIMOSA Association), com dois níveis. No primeiro existe uma arquitectura generalizada que compreende modelos genéricos aplicáveis por um grande número de organizações. Num segundo nível existem modelos de referência mais específicos, i. e. para um ramo de actividade, aplicáveis ainda por diversas organizações. Num outro nível tem-se um modelo particular à organização que deve ser a instanciação de uma arquitectura de referência.

Se excluirmos esta dimensão ao modelo obtemos uma matriz que poderia ser uma simplificação da *framework de Zachman*.

A dimensão Derivação divide os modelos em Especificação de Requisitos, Especificação do Modelo de Concepção e Descrição do Modelo de Implementação. É interessante notar que o cruzamento desta dimensão com a dimensão Instanciação resulta, no caso da Arquitectura de Referência, em padrões arquitecturais de mais alto ou baixo nível conceptual.

A dimensão Visões é análoga às linhas da matriz de Zachman e pode ser estendida caso se considere necessário.

Além das vantagens decritas que esta ICASI introduziu ou abordou, importa referir a preocupação em especificar a relação entre as diferentes células do modelo. Existem diversos exemplos de projectos realizados de acordo com a framework CIMOSA em domínios como as indústrias automóvel (Fiat, Magneti Marelli), do papel (Koehler), metalomecânica (Traub), tratamento de alumínio (ELVAL), etc.

O CIMOSA levou já à produção de um standard (ISO 14258) que define um conjunto de regras e conceitos no âmbito da automação de sistemas industriais [7]. A associação CIMOSA mantém-se activa promovendo a constituição de standards que permitam a utilização generalizada do modelo. Algumas das iniciativas estão orientadas para a ligação desta *framework* a outros standards como o XML, XMI, etc.

# 4 Iniciativas OMG – CONTEXTO DO MDA

## *4.1 Extensões UML*

A linguagem Unified Modelling Language foi definida pelo Object Management Group e é hoje um standard no campo das linguagens de modelação de sistemas de informação. O UML compreende um modelo que distingue a representação dos conceitos dos próprios conceitos. Sendo assim, um modelo UML pode ser visto como um conjunto de esquemas com uma notação pré-definida ou pode ser lido como uma descrição textual. O UML foi concebido de uma forma aberta de modo a que se pudessem acrescentar novos elementos considerados pertinentes. Os mecanismos de extensão à linguagem são [5] os estereótipos, as marcas com valor e as restrições.

Como o UML é uma linguagem genérica que cobre âmbitos variados (como sistemas de informação empresarial, sistemas de tempo real, sistemas concorrentes e distribuídos, simulação, etc.) verificou-se que a linguagem não permitia captar com a legibilidade necessária as características específicas destas e doutras áreas. Por isso, foram definidos *perfis* que são conjuntos de especificações que estendem a linguagem, utilizando os mecanismos de extensão já indicados. Existem já diversos perfis como [11]: Profile for CORBA, Profile for Enterprise Application Integration (EAI), Profile for Enterprise Distributed Object Computing (EDOC), UML Profile for Schedulability, Performance and Time. Todos os perfis referidos foram aceites pela OMG e podem ser utilizados nos âmbitos respectivos.

A criação de um perfil com notação própria pode ser justificada por diversos factores: melhor legibilidade dos diagramas resultantes; maior rapidez na construção dos diagramas devido à maior adequação entre a notação e os conceitos; inclusão de mais informação relevante nos modelos; e maior facilidade em produzir ferramentas de geração de código/mapeamentos.

As duas primeiras vantagens são evidentes se compararmos os dois elementos da Figura 3. Embora ambas digam respeito à mesma realidade, a segunda figura parece ser mais legível do que a primeira (independentemente das qualidades estéticas que lhe podem ser atribuídas). A notação do segundo elemento da Figura 3, embora seja mais perceptível, não é passível de reprodução por meios não automáticos (a menos que o desenhador tenha um talento invulgar na comunidade das TIs). Este factor já foi mais importante do que é hoje com a vulgarização das ferramentas CASE a que assistimos. Poder-se-ia, porém, encontrar uma simplificação desta notação passível de ser desenhada à mão e que continuaria a expressar melhor o modelo do que a notação standard.

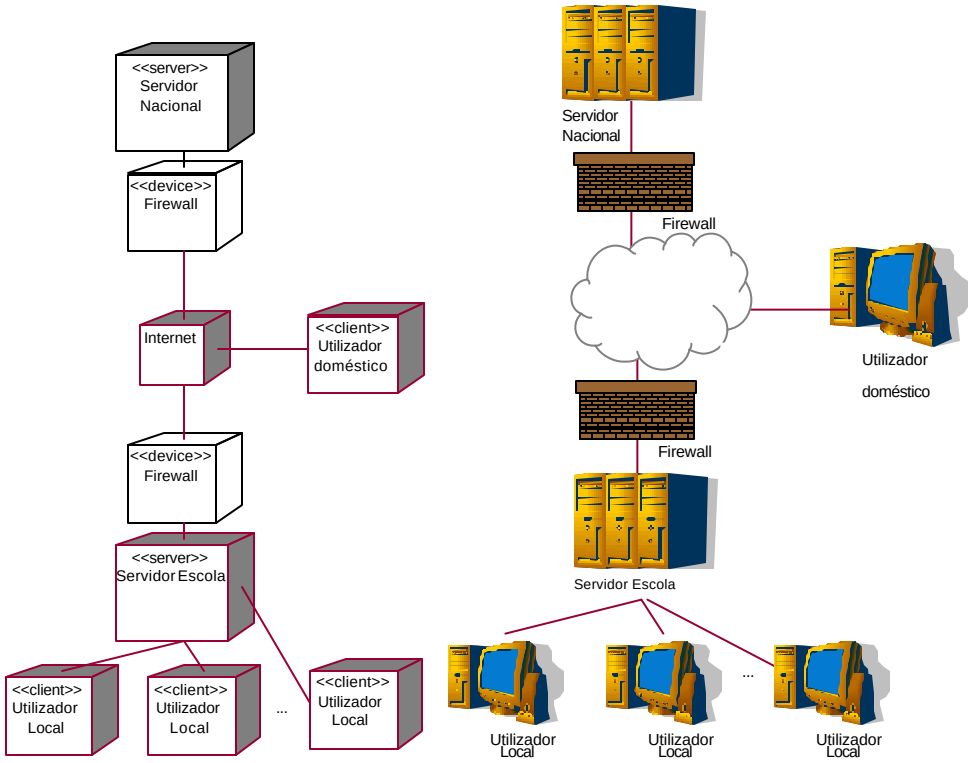



*Figura 3. Diagramas de Instalação UML utilizando a notação standard e uma notação estendida*

No entanto, existe um perigo evidente na criação de perfis; Se cada organização criar um perfil, ao fazê-lo estará a estender a linguagem criando um dialecto. Para existir a comunicação, ao nível do UML, entre dois sistemas desenvolvidos com perfis diferentes, é necessário estabelecer uma relação entre esses perfis, o que pode não ser trivial. Se esta situação fosse multiplicada por um grande número de organizações teriam de existir mapeamentos à medida para cada duas organizações. Isto só complicaria ainda mais a comunicação que se quer mais simples.

Para evitar este problema considera-se a OMG como a autoridade capaz de aceitar ou recusar um novo perfil. Sendo assim, um perfil só deve ser considerado como norma quando aceite pela OMG.

### *4.2 Model Driven Architecture*

O Object Management Group, entidade responsável por standards como o CORBA e o UML, definiu uma ICASI denominada Model Driven Architecture.

Tal como as anteriores, esta framework é independente das linguagens de programação, dos ambientes de operação e do *middleware*. A MDA define uma abordagem à especificação de sistemas de informação que separa a especificação das funcionalidades do sistema, da especificação da implementação dessa funcionalidade numa plataforma específica [10].

A Figura 4 representa a MDA como uma arquitectura cujo núcleo deve estar definido segundo standards como o UML, MOF e CWM, estando rodeado por duas outras camadas. Deverão existir vários núcleos pré-definidos [8], i.e. para sistemas de informação genéricos, computação em tempo real, etc. O número de núcleos definidos deverá, no entanto, ser restrito porque estes definem o conjunto comum de características para todas as plataformas nesta categoria. Segundo esta framework, o primeiro passo a dar será encontrar um modelo independente das plataformas, definido de acordo com um núcleo MDA existente. Cada modelo de núcleo deverá ser independente de qualquer plataforma de middleware.

Em seguida é realizado o mapeamento entre este modelo e uma plataforma como CCM, EJB ou MTS. O modelo específico à plataforma continua a ser um modelo escrito em UML, mas deve respeitar um perfil específico (que funciona como um dialecto da linguagem UML).

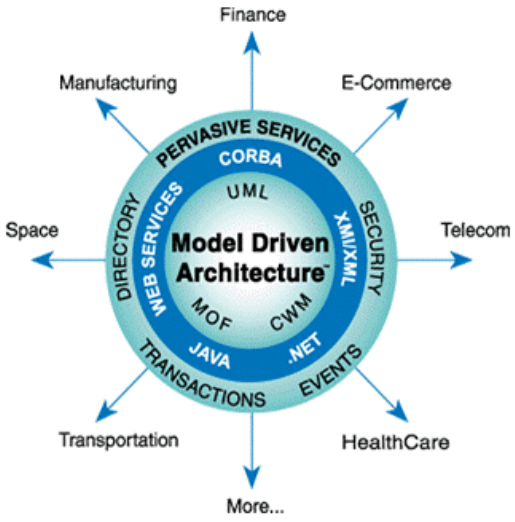



*Figura 4. Model Driven Architecture (MDA) [8]*

Existem alguns mapeamentos standard baseados em padrões que podem ajudar ferramentas a realizar esta tarefa. Na Figura 4 estas plataformas ocupam o segundo anel que envolve o núcleo. O objectivo deverá ser maximizar este mapeamento no futuro.

O terceiro passo deverá ser gerar o código da aplicação propriamente dito. Quanto mais detalhados forem os modelos anteriores e quanto mais precisos forem os mapeamentos, maior a proporção de código gerado. Esta divisão em três camadas da MDA, não sendo propriamente um conceito novo, é fundamental para que se consiga realizar sistemas enquadrados numa verdadeira ASI. O mapeamento entre modelos é um conceito

essencial à MDA. Um mapeamento é definido como um conjunto de regras e técnicas usadas para modificar um modelo de forma a obter outro [10].

A Figura 5 coloca em relevo três tipos de mapeamentos que podem ocorrer. De dentro para fora distingue-se em primeiro lugar o mapeamento entre modelos independentes das plataformas. Esta transformação é necessária quando os modelos são especializados, filtrados ou especializados, durante o processo de desenvolvimento, sem que se necessite de alguma informação sobre as plataformas.

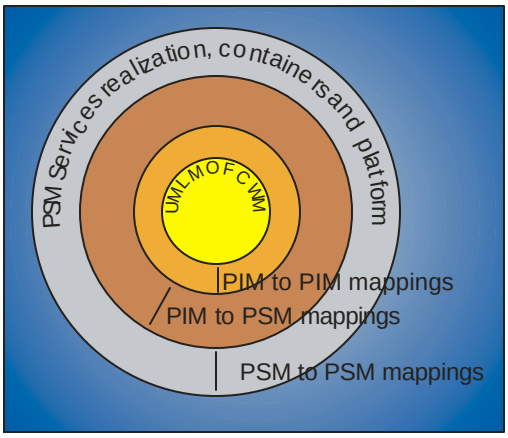



*Figura 5. Mapeamentos em MDA*

Em seguida indicam-se mapeamentos entre modelos independentes e modelos dependentes das plataformas (PIM para PSM). Esta transformação é necessária quando os PIM estão já suficientemente refinados e é necessário incluir informação sobre a plataforma, i.e., introduzir a informação relevante à implementação nas plataformas J2EE, .NET ou CORBA. No anel exterior existe um terceiro mapeamento entre modelos dependentes das plataformas (PSM). Esta transformação serve para a realização de componentes e para a geração do código propriamente dito. Esta transformação funciona como um refinamento dos PSM e recebe informação relativa a configurações, utilização e geração de bibliotecas, etc. Pode ainda ser considerado um outro mapeamento entre modelos dependentes e independentes das plataformas (PSM para PIM). Este mapeamento é necessário quando é preciso obter abstracções de implementações já existentes. Existem já no mercado diversas ferramentas que permitem este tipo de reverse engineering como o Rational Rose, o Together, o Microsoft Visio, etc. No entanto esta operação tem de ser assistida de forma a que os modelos gerados possam ser entendidos facilmente.

Tem existido actualmente um grande esforço no sentido de se produzir ferramentas que consigam realizar estes mapeamentos de uma forma metódica e organizada. A experiência actual ainda é, no entanto, pouco

animadora pois usualmente apenas é realizado 1% do mapeamento pelas ferramentas mais comuns no mercado [9].

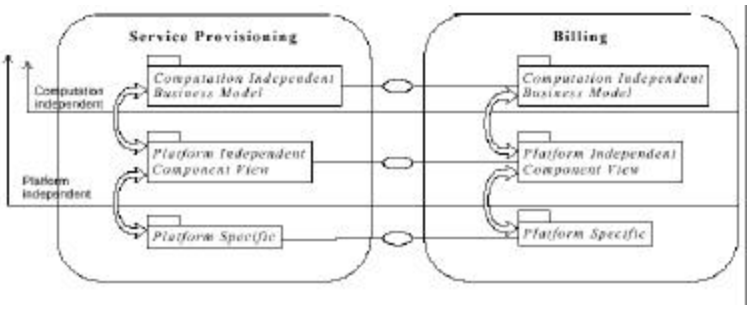



*Figura 6. Mapeamento entre sistemas (MDA)*

Existem já em fase experimental, e até mesmo já comercialmente disponíveis, ferramentas como o Codagen Architect e o XIS que podem chegar aos 70% [9 e 12]. A MDA ao definir três camadas para uma ASI, em conjunto com uma linguagem (UML), permite também realizar o mapeamento entre dois sistemas de uma organização, ou de duas organizações diferentes. Este mapeamento pode ser realizado logo ao nível dos conceitos do negócio na medida em que ambas as organizações (ou subsistemas da organização) têm uma descrição compatível dos conceitos relevantes ao negócio. Este mapeamento não é automático, nem deverá vir a ser, na medida em que envolve uma determinação semântica que tem de ser feita de forma assistida. Os mapeamentos dos níveis mais baixos podem ser tendencialmente mais automatizados à medida que se conseguir formalizar o mapeamento dos níveis superiores. Depois de feitos estes mapeamentos é possível realizar interacções entre os sistemas de informação.

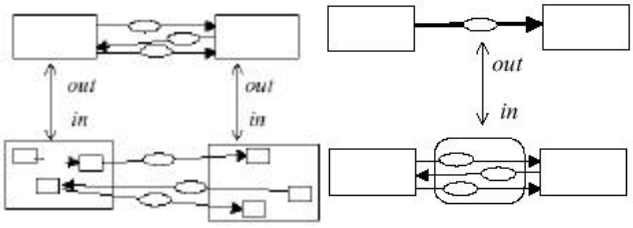



*Figura 7. Zooming de mapeamentos em MDA*

As ferramentas que suportem a MDA deverão permitir realizar o *zooming* entre diferentes níveis de abstracção como se ilustra na Figura 7. Esta operação significa que um determinado conjunto de interacções pode estar visível na sua forma simplificada, apenas com um elemento gráfico, ou com todas as suas constituintes também visíveis. Para modelos muito complexos, como são os modelos dos problemas reais das organizações, esta capacidade é de grande utilidade. É necessário agora que as ferramentas CASE que suportam UML passem a implementar esta característica.

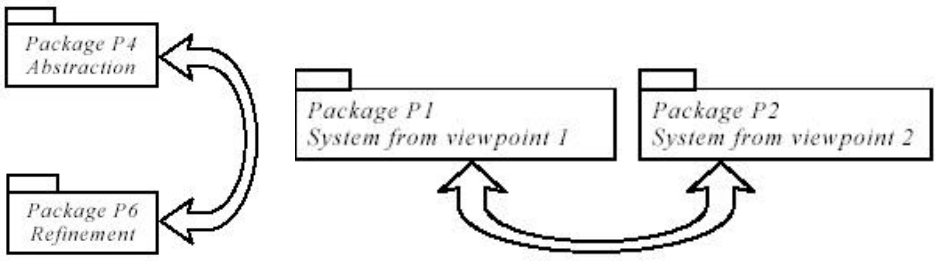



*Figura 8. Relações de refinamento e correspondência entre pacotes UML*

Um conceito UML muito importante para a MDA é o *pacote*. Este elemento é um agrupamento de outros elementos [10]. Um pacote pode importar outros elementos, tornando os elementos importados disponíveis para o seu uso. Os pacotes ajudam a compreender a MDA na medida em que os modelos que devem ser relacionados para uma integração estão normalmente em pacotes distintos.

Os modelos que explicitam duas vistas diferentes e não relacionadas por refinamento sobre o sistema podem ser definidos em dois pacotes separados. As relações existentes entre os dois pacotes são identificadas por uma correspondência entre modelos (seta grande) que define a integração entre estas duas visões do sistema a este nível de abstracção. A relação entre modelos do mesmo sistema, a níveis de abstracção diferentes definidos em dois pacotes designa-se por refinamento, como se representa na Figura 8. O refinamento é uma relação entre os elementos dos dois pacotes e não dos próprios pacotes.

Os modelos independentes da plataforma, que incluem os requisitos do sistema, podem ser descritos utilizando UML e uma linguagem formal definida também no âmbito da OMG denominada OCL (Object Constraint Language).

Os modelos dependentes da plataforma podem suscitar uma questão: se o UML não é dependente de nenhuma plataforma como explicitar um modelo que o é? Os perfis UML fornecem a chave para este problema ao definirem notações que podem ser específicas às plataformas. O perfil UML para CORBA é um exemplo, ao definir estereótipos como CORBAInterface, CORBAVAlue, CORBAStruct, etc., que são aplicadas a classes caracterizando-as em termos de uma plataforma de middleware.

O mapeamento entre modelos PIM e modelos PSM não pode ser descrito em UML Versão 1.4. na sua totalidade. Na Figura 9 representa-se um mapeamento entre uma classe do negócio *Account* e o correspondente refinamento para interfaces CORBA. Neste caso a classe de negócio *Account* pertenceria a um pacote diferente das classes dependentes da plataforma.

Os estereótipos BusinessEntiy e CORBAInterface estão já definidos e a dependência pertence a um estereótipo standard denominado <<refine>>. São usados identificadores dos espaços de nomeação

("PlatformIndepent" e "CORBASpecific") para distinguir entre duas classes com o mesmo nome mas âmbitos diferentes.

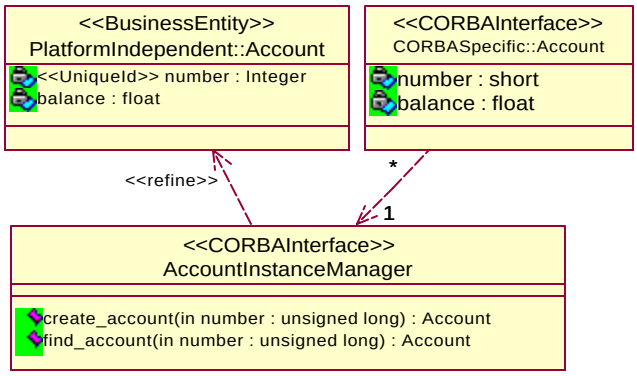



*Figura 9. Mapeamento entre um modelo PIM e um modelo PSM em UML*

Este exemplo coloca em relevo a diferenciação entre as actividades relacionadas com o desenvolvimento de um SI. O *arquitecto/modelador* foca a sua atenção em criar o modelo de negócio arquitectural independente da plataforma. O *desenhador de middleware/*serviços web usa o perfil UML CORBA para modelar aspectos do sistema específicos e dependentes da plataforma permitindo que a geração de interfaces CORBA possa ser automática. O programador usa o modelo UML, tal como as interfaces IDL, acrescentando o código necessário para implementar o serviço.

Na Figura 10 representa-se uma arquitectura com os respectivos constituintes MDA [10]. O modelo de Domínio do Negócio representa o conhecimento sobre o negócio da organização, independentemente de ferramentas de software que possam ser usadas. Os Modelos de Processo de Negócio descrevem-nos e incluem modelos da informação.

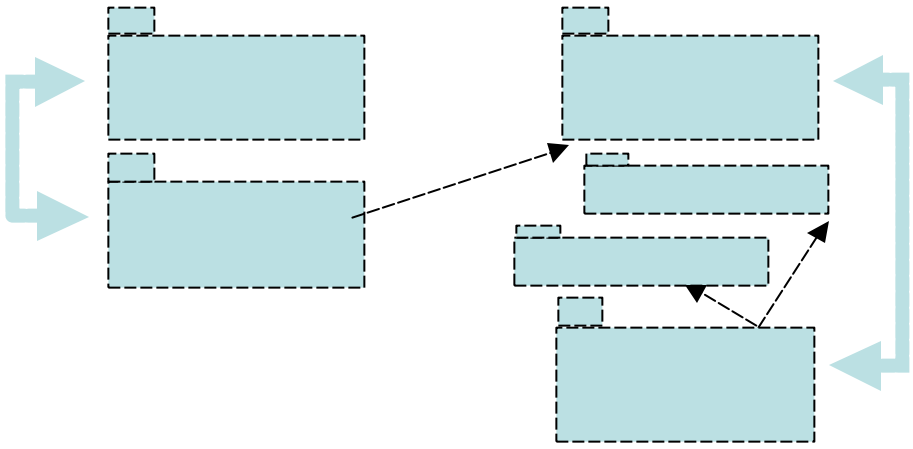


*Figura 10. Exemplo de arquitectura conforme a MDA [11]*

Versões detalhadas podem incluir também a forma como uma aplicação, S1 é usada como parte de alguns processos de negócio. Não pode violar o conhecimento sobre o negócio descrito em B1 e B2. A Especificação do Sistema Informático descreve as aplicações independentemente dos vários processos para as quais podem ser configuradas e é construída utilizando um modelo simplificado do domínio. A Concepção D1 do Sistema Informático S1 é realizada juntando outros componentes de software C1 e C2 já especificados.

Existem no entanto algumas dificuldades com que as equipas de sistemas de informação das organizações têm de se defrontar [13]:

- Quando uma arquitectura de software é modelada em UML, a sua evolução e manutenção requer uma intervenção manual morosa, repetitiva, e fastidiosa. O arquitecto tem de garantir que o modelo UML reflecte as alterações em cada instância onde a arquitectura é usada.
- Não existe ainda nenhuma abordagem sistemática que garanta que o código das aplicações está conforme a arquitectura do sistema. A única maneira de o conseguir é realizar verificações exaustivas ao código.

## 5 Abordagem Codagen

Existe um grande número de empresas ou outras organizações a trabalhar no sentido de produzir ferramentas que implementem a MDA. O objectivo final seria gerar 100% do código a partir de uma especificação que por sua vez seria gerada automaticamente a partir de uma arquitectura. Mesmo não considerando os passos intermédios necessários este objectivo ainda está, pelo menos, longe de ser alcançado. Faremos em seguida uma breve descrição da ferramenta Codagen Architect produzida pela Codagen Technologies Corp. que trata este problema com resultados prometedores [14].

Pretende-se encontrar uma forma de realizar o mapeamento PIM para PSM e o inverso PSM para PIM.

Pode-se sempre distinguir a representação UML do PSM de uma representação codificada do mesmo.

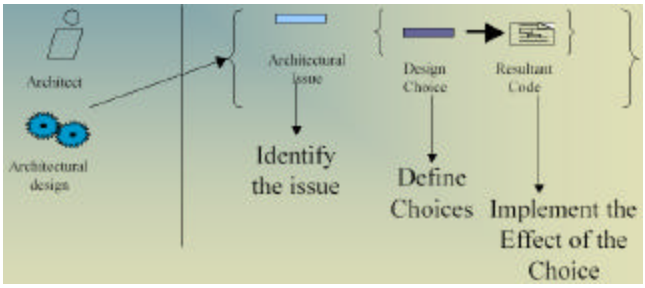



*Figura 11. Processo da abordagem Codagen[13]*

Esta abordagem considera que um PSM é um conjunto de conceitos UML e uma descrição codificada de acordo com as plataformas necessárias (CORBA, J2EE, etc.). O arquitecto de SI define os requisitos e os padrões arquitecturais necessários para que a PSM possa ser derivada para diversas plataformas. O arquitecto pode definir também os serviços comuns de infra-estrutura que um programador (mesmo que não seja um grande entendido numa tecnologia) pode utilizar. Segundo a Codagen reduz-se assim a necessidade do nível técnico necessário aos programadores.

O arquitecto pode permitir aos programadores estenderem a implementação da arquitectura com *marcadores de código*. Estes marcadores actuam como blocos que devem ser preenchidos posteriormente pelos programadores e são preservados mesmo depois do código ser gerado. A abordagem Codagen é baseada numa solução XML com características de "caixa negra" e "caixa branca". Os arquitectos usam a visão de "caixa branca" para capturar, restringir e gerir as especificações arquitecturais, a implementação e os standards. Pode incluir-se um *marcador de código* sempre que um arquitecto deseje fornecer um ponto de extensão aos programadores. Os programadores utilizam a abordagem de "caixa negra" para gerar o código associado à arquitectura. Esta abordagem permite que o arquitecto especifique o que precisa ser feito sem ter de manipular uma linguagem de *scripts* que pode ser alterada posteriormente pelos programadores. Da mesma forma, preserva o trabalho dos programadores porque os *marcadores de código* preservam o código de cada plataforma cada vez que o código é regerado.

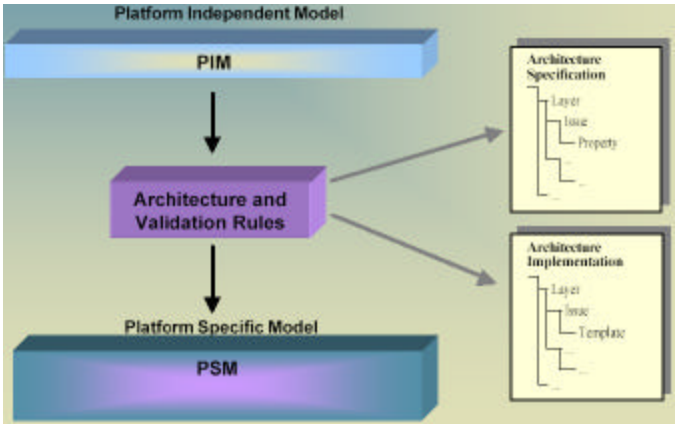



*Figura 12. Desacoplamento entre Especificação da Arquitectura e a sua Implementação (Codagen)* ***[13]***

Como é representado na Figura 12, a especificação e a implementação da arquitectura compreendem um conjunto de assuntos ou restrições (elementos do modelo de negócio) organizados por camadas. Cada camada serve para organizar os assuntos ou restrições para cada nível de abstracção. Esta categorização

serve para identificar os requisitos que devem ser tidos em conta para uma correcta implementação da tecnologia associada a cada camada. Para cada restrição de arquitectura são realizadas decisões de concepção (propriedades) que deverão ajudar o desenhador ou programador a resolver essa questão. São indicados também os mapeamentos entre cada decisão de concepção e um ou mais elementos UML do PIM (pacote, classe, atributo, operação, etc). Existem também valores dados por omissão para cada par (decisão de concepção, elemento UML). A especificação da arquitectura é guardada num documento XML. A solução compreende um conjunto de *templates* XML que geram o código dependente da plataforma. Desde que o arquitecto organize os *templates* por camada e por assunto, pode forçar a especificação da arquitectura durante a implementação. De uma forma mais detalhada, a abordagem segue as seguintes fases conforme se referem.

**Fase de Arquitectura de Concepção (*Design*).** Durante a fase de concepção, o arquitecto realiza um conjunto de passos que definem uma especificação arquitectural e define as diferentes camadas que podem existir no PIM, i.e., as camadas de Base de Dados e Negócio. Para cada nível são definidos os requisitos de arquitectura para cada camada, i.e., para a camada Base de Dados poderiam ser Persistência e Concorrência.

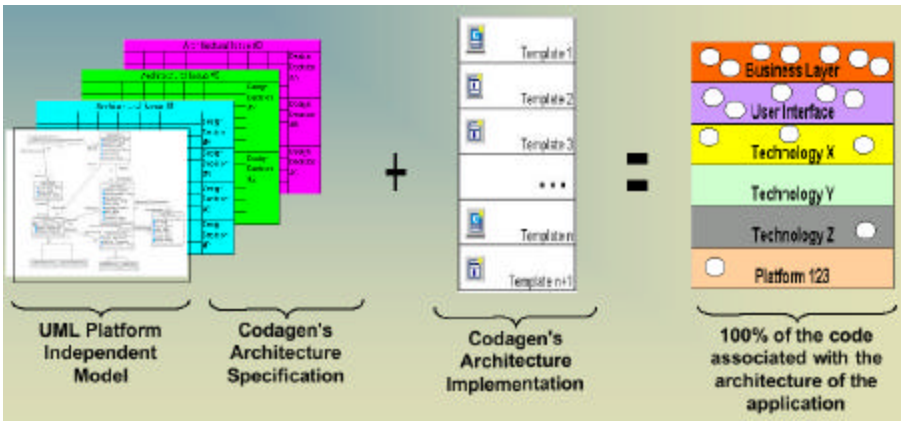



*Figura 13. Artefactos de input e produtos resultantes (Codagen)*

Para cada requisito arquitectural é definido um conjunto de decisões de concepção, i.e., para o requisito Persistência poderíamos encontrar um atributo isPersistent que indica se um atributo ou classe é persistente. Para o mesmo requisito poderíamos ter também um atributo *isKey* que indicava se um atributo é chave ou não. Mapeia-se em seguida cada decisão de concepção com um ou mais elementos UML no PIM (pacote, classe, atributo, operação, associação, etc.) de forma a obter o par valor (decisão de concepção, elemento UML). Usando o mesmo exemplo, a decisão *isPersistent* poderia ser associado tanto a uma classe como a

um atributo, e a decisão *isKey* poderia ser associada a um atributo. Define-se em seguida os valores por omissão para cada par (decisão de concepção, elemento UML). O par (*isKey*, atributo) poderia ter o valor Boolean *false* por omissão.

**Fase de Arquitectura da Implementação**. O arquitecto utiliza uma especificação arquitectural baseada em XML para construir os *templates* XML que preparam a geração do modelo de código específico a cada plataforma. Este modelo de código (*code* PSM) é a parte do modelo específico que não pode ser descrito em diagramas UML. Os *templates* incluem as regras para a tradução entre o PIM e o *code* PSM e incluem regras de validação das decisões de concepção. De facto, estes *templates* capturam a arquitectura da implementação, validam a especificação e automatizam simultaneamente a passagem do PSM a código final. Como a arquitectura está organizada em camadas, é simples alterar uma camada de implementação por outra e gerar o *code* PSM para cada camada.

**Fase de Geração de Código**. De forma a adaptar o modelo arquitectural a cada plataforma específica de uma aplicação, o programador precisa apenas de validar o valor dado por omissão para cada par (decisão de concepção, elemento UML) ou dar-lhe um novo valor. Finalmente, o programador gera o código associado ao PSM. A abordagem Codagen pode gerar código para cada PSM definida por camadas como *web services*, CORBA, J2EE ou .NET.

**Separação da Especificação da Arquitectura da sua Implementação**. Uma vez que o arquitecto tenha terminado de conceber as regras para validar e para traduzir o PIM no PSM, a especificação da arquitectura é separada da sua implementação.

Segundo a Codagen esta separação apresenta diversas vantagens:

- O PIM não inclui detalhes de implementação, o que protege o investimento na análise das camadas de domínio e permite que o modelo de camadas de domínio possa evoluir a passo com os requisitos do negócio.
- O modelo de concepção passa a ser a norma (blueprint). Não há necessidade, nem oportunidade, para o programador alterar a especificação ou implementação.
- Como a norma é baseada em XML e não é necessário utilizar nenhum *script* para a sua implementação, o arquitecto retém um controlo total sobre a norma. Isto permite forçar a utilização da arquitectura.
- Sob controlo do arquitecto, a arquitectura da aplicação pode evoluir livremente em sincronia com os requisitos tecnológicos da plataforma em constante mutação.

- A modificação da arquitectura não requer uma modificação do PIM.
- A Codagen possui *add-in's* para algumas ferramentas CASE UML como as da Rational Software, TogetherSoft e Microsoft. Assim, o arquitecto não precisa de gerir um novo paradigma para modelar o PIM.

Actualmente o Codagen Architect não permite realizar o *reverse-engineeering* dos PSM para PIM. A Codagen considera colocar esta capacidade nas novas versões do produto.

## 6 A abordagem XIS

O XIS é um projecto de investigação e desenvolvimento do INESC-ID cuja missão é analisar, desenvolver e avaliar mecanismos e ferramentas para a produção de sistemas de informação de uma forma mais abstracta, eficiente, produtiva e de mais alto nível, das que são normalmente utilizadas na actualidade [12]. O projecto XIS é influenciado pelo modelo de referência MDA e baseia-se significativamente num conjunto de boas práticas emergentes, designadamente segue uma abordagem baseada na especificação de modelos; centrada em arquitecturas de software; e baseada em técnicas de geração automática de artefactos digitais.

A Figura 14 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.

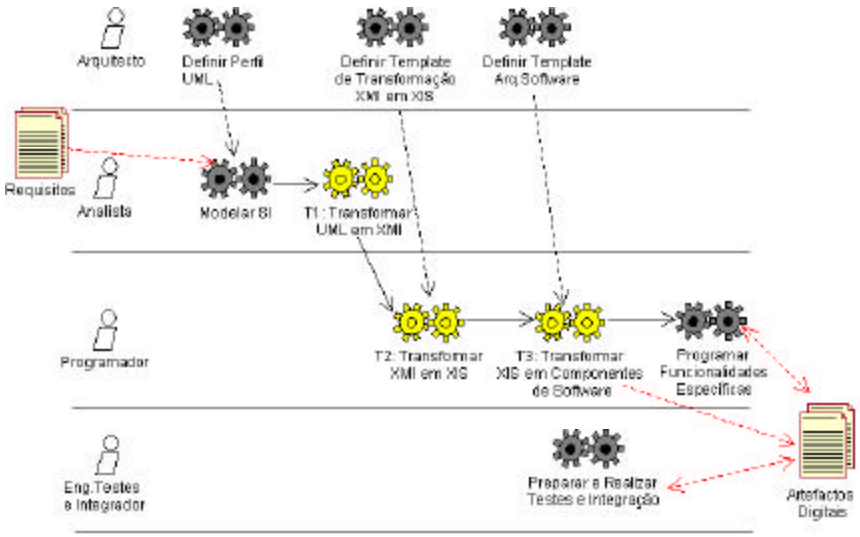



Figura 14. Visão geral da abordagem XIS.

Além de uma componente metodológica, o projecto define e propõe um conjunto integrado de elementos, designadamente: (1) **Plataforma e repositório XIS**, que consiste numa ferramenta CASE que tem como objectivo suportar os intervenientes técnicos no processo de desenvolvimento de software segundo a abordagem XIS. Esta ferramenta é um sistema robusto com interface Web, multi-utilizador, multi-aplicação e com múltiplas arquitecturas de software. (2) **Perfil UML para XIS**, que consiste num conjunto de extensões UML para o projecto XIS que permite a especificação visual, alto nível e intuitiva de informação necessária à construção de sistemas de informação. (3) **Linguagem XIS/XML**, que consiste numa linguagem definida em XML para o projecto XIS que permite a especificação textual, estruturada, legível e compacta de informação necessária à construção de sistemas de informação.

O projecto XIS inclui elementos inovadores como a possibilidade do processo de geração poder ser realizado remotamente através de uma aplicação que não reside na própria organização alvo. O XIS permite também gerar código para diversas linguagens de programação e arquitecturas de software.

## 7 Conclusões

Existe actualmente uma consciencialização por parte da comunidade das tecnologias da informação para as vantagens na utilização de modelos arquitecturais nos desenvolvimentos de sistemas de informação das organizações. Além das próprias organizações, algumas iniciativas referidas demonstram que estas preocupações já são reconhecidas a um nível governamental (i.e., *Clinger-Cohen Act*, ESPRIT, etc.). Depois de um período de criação de metodologias, ICASI e notações de sistemas de informação nos anos 80 e 90 a indústria parece ter encontrado uma norma sobre a qual produzir um trabalho mais estável. As iniciativas da OMG como o UML e o MDA, devido não só às suas potencialidades como à sua aceitação pelos maiores intervenientes no mercado, parecem constituir a infra-estrutura conceptual que faltava até aqui para que as organizações possam produzir ou adquirir aquilo que for necessário ao seu sistema de informação. Segundo a OMG a MDA oferece alguns benefícios como: o aumento da produtividade para arquitectos e programadores; a diminuição do custo do desenvolvimento das aplicações e da sua gestão e a melhoria na interoperação e na portabilidade. É muito difícil afirmar actualmente se estes benefícios são reais quando se compara a utilização da MDA com a de outra ICASI na medida em que existe uma enorme quantidade de factores que pode enviesar os resultados. No entanto se compararmos a utilização de uma *framework* (seja a MDA ou outra das indicadas) com a situação actual da maior parte das organizações – caos em termos de ASI –, os benefícios não podem deixar de ser evidentes. Depois do sonho inicial das

ferramentas CASE poderem tomar o controlo da organização e dos seus sistemas, surgiu a decepção durante mais de uma década, na medida em que mecanismos como o *round-trip generation* só agora começam a ser uma realidade. Iniciativas como o Codagen Architect ou o XIS levarão a que os sistemas de informação das organizações passem a estar alinhados com os objectivos da organização, sejam de mais fácil manutenção e reajam a um mundo em mudança, com custos inferiores.

## Bibliografia


[1] West, D., Bittner, K., Glenn, E. Ingredients for Building Effective Enterprise Architectures http://www.therationaledge.com/content/nov_02/f_enterpriseArchitecture_dw.jsp

[2] Framework Zachman http://www.istis.unomaha.edu/isqa/vanvliet/arch/isa/isa.htm

[3] Boar, B. "Constructing Blueprints for Enterprise IT Architectures", Wiley 1999

[4] CIMOSA, http://cimosa.cnt.pl

[5] Da Silva, A., Videira, C., "UML Metodologias e Ferramentas", Centro Atlântico, 2001

[6] Zachman, Enterprise Architecture: The Past and the Future, 2000 http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=1702

[7] Winchester, G. (Ed.), Industrial automation systems-Concepts and rules for enterprise models http://www.mel.nist.gov/sc5wg1/std-dft.htm

[8] Soley, R., et.al., Model Driven Architecture, 11/2000 ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf

[9] Codagen Technologies and model drive-architecture, http://www.omg.org/mda/mda_files/CodagenMDA-Final.pdf

[10] Architecture Board ORMSC1, Model Driven Architecture (MDA), Doc. nr ormsc/2001-07-01 http://cgi.omg.org/docs/ormsc/01-07-01.pdf

[11] OMG, www.omg.org

[12] Da Silva, A., Visão Geral do Projecto XIS, Grupo de Sistemas de Informação, LAVC/INESC-ID

[13] Codagen Tec. Inc., Codagen Technology and Model-Driven Architecture (MDA) http://www.omg.org/mda/mda_files/CodagenMDA-Final.pdf

[14] Codagen Tec. Inc., White Paper on MDA http://www.codagen.com/scripts/login.asp?docID=115113568&return=/mda

[15] K. Czarnecki et al., Beyond Objects: Generative Programming, ECOOP'97 Workshop on Aspect-Oriented Programming. http://citeseer.nj.nec.com/czarnecki97beyond.html

[16] Markus Voelter, A Collection of Patterns for Program Generation.

http://www.voelter.de/publications/index.html

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

[18] Richard Wurman. *Information Architects*. Palace Press International, 1997.