UML 2.0 - Unified Modeling Language 2.0

António Mota, Hugo Valente, Ivo Navega,
Pedro Pacheco, Tiago Silva, José Pacheco

Faculdade de Engenharia da Universidade do Porto,
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL

Resumo Em Junho de 2001, foi iniciado um OMG Request for Proposal, tendo em vista uma reformulação da linguagem UML. Foi assim iniciado o desenvolvimento da UML 2.0, que contou com a colaboração de mais de 50 empresas, que forneceram as suas melhores tecnologias e práticas. Em Agosto de 2003, foi oficialmente adoptada a sua especificação final. Espera-se que esta linguagem tenha um grande efeito em propósitos de estandardização e precisão, tendo como consequência uma revolução na indústria de desenvolvimento de software.
Este artigo pretende dar a conhecer as principais diferenças, inovações, novas especificações, e alterações nas normas relacionadas (XMI, OCL e MOF), da UML 2.0 relativamente às versões anteriores.

1 Introdução

1.1 O que é UML?

A linguagem de modelação unificada (UML) [1] é uma linguagem gráfica para a visualização, especificação, construção e documentação de artefactos de um sistema (preferencialmente) com uma componente intensiva de software, apesar de actualmente já possibilitar a modelação de sistemas que não sejam baseados em software. Quando se pensa em projectar algo de novo, torna-se conveniente recorrer a modelos que representem aquilo que irá ser desenvolvido. Esses modelos constituem assim uma representação abstracta de uma realidade projectada para o futuro. A UML oferece uma forma standard de criar esses modelos, permitindo a simplificação do complexo processo de concepção de software através do uso de uma forte componente gráfica (tirando partido da imagem como elemento de comunicação) e a utilização de um conjunto limitado de símbolos. Teve a sua origem na Rational Software, sendo no entanto agora administrada pelo *Object Management Group* (OMG).

De seguida referem-se algumas das suas características:

- **É apenas uma sintaxe** – a UML é apenas uma linguagem. Diz quais os elementos de modelação, os diagramas disponíveis e as regras a eles associados. Não diz quais os diagramas a criar nem quando. Isso diz respeito à metodologia usada: *Rational Unified Process* (RUP), *Feature Driven Development* (FDD), etc..

- **É abrangente** – a UML pode ser usada para modelar uma grande variedade de sistemas e está concebida para poder ser actualizada de modo a satisfazer qualquer requisito de modelação.
- **É independente da linguagem usada** – a UML é independente da linguagem de alto nível a usar no código (Java, C++, etc).
- **É independente do processo de criação dos modelos** – o processo pelo qual os modelos são criados é independente da definição da linguagem. É necessário um processo desses para além do uso da UML por si só.
- **Independente da ferramenta usada** – a UML faculta uma grande margem de manobra para a criatividade usada na criação de ferramentas de modelação visual com UML, já que esta linguagem é independente da ferramenta usada (Visio, Rational Rose, etc).
- **Linguagem bem documentada** – o guia de notação da UML está disponível como referência para todas as sintaxes disponíveis na linguagem.
- **A sua aplicação não é rígida** – o guia da notação UML não é suficiente para que se saiba usá-la, já que se trata de uma linguagem de modelação genérica que por isso necessita de ser adaptada a cada situação em particular.

O uso de qualquer uma das ferramentas que suportam a UML permite analisar os requisitos futuros da aplicação e conceber uma solução que os contemple, representando os resultados através dos 13 diagramas standards da UML 2.0.

Esses diagramas podem ser divididos em três grupos:

- **Diagramas de estrutura** – diagrama de actividades, máquina de estados e casos de utilização;
- **Diagramas de interacção** – diagrama de comunicação, vista geral de interacção, sequência e temporal;
- **Diagramas de comportamento** – diagrama de classes, estrutura de composições, componentes, distribuição, objectos e pacotes.

A UML representa assim o culminar das melhores práticas usadas na modelação orientada por objectos, sendo uma unificação dos métodos mais usados e, acima de tudo, um esforço para tornar a modelação orientada por objectos num processo mais simples.

1.2 Como Nasceu?

A UML nasceu quando James Rumbaugh se juntou a Grady Booch na Rational Software, em Outubro de 1994. Ambos possuíam a sua própria sintaxe orientada por objectos e necessitavam de as combinar numa só. Semanticamente, as duas sintaxes (OMT e Booch) eram bastantes similares, residindo as diferenças essencialmente nos símbolos usados, que portanto necessitavam de ser unificados. Estes dois métodos estavam ambos a desenvolverem-se, ainda que em separado, e ambos eram reconhecidos como líderes dentro dos métodos orientados por objectos. Foi da fusão destes dois métodos que nasceu a UML 0.8, em Outubro de 1995.

No Outono desse mesmo ano, Ivar Jacobson juntou-se à Rational e trouxe consigo os diagramas de casos de utilização, saindo em 1996 a versão 0.91. Durante esse ano foram muitas as empresas (IBM, HP, Microsoft, Oracle, Rational Software, TI, etc.) que viram na UML uma mais valia para o seu negócio. A colaboração entre essas empresas deu origem à UML 1.0 e, mais tarde, à versão 1.1. O Object Management Group adoptou a especificação da UML 1.1 em Novembro de 1997, tendo depois sido feitas algumas alterações que deram origem às versões 1.3, 1.4 e 1.5. Actualmente está ser desenvolvida a versão 2.0. As modificações que foram feitas relativamente às versões anteriores são o principal assunto deste artigo.

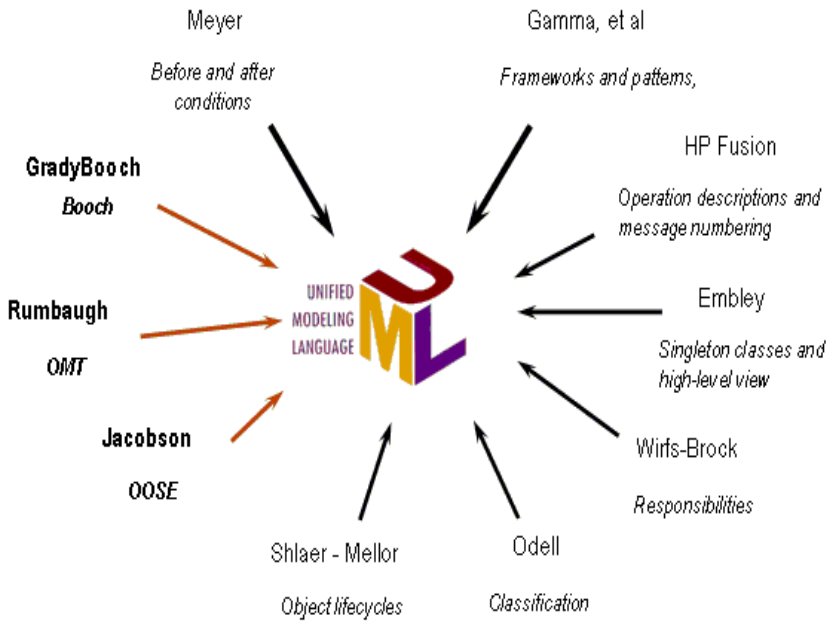

Figura 1. Colaborações entre as várias empresas na UML

1.3 Quais as principais ideias e conceitos que introduz?

Os objectivos do esforço da unificação eram manter a simplicidade, remover elementos já existentes mas que se sabia não funcionarem na prática, adicionar elementos de outros métodos que se mostrassem mais eficazes e inventar elementos novos apenas quando necessário, de modo a evitar que a UML se tornasse desnecessariamente complexa.

Existem vários conceitos que a UML introduz e junta, dos quais se destacam:

- Mecanismos de expansibilidade;
- Diagrama de actividades;

- Refinamento (vários níveis de detalhe);
- Interfaces e componentes;
- Linguagem de restrição.

Algumas destas ideias estavam já presentes em vários métodos ou teorias individuais, mas a UML juntou-as de modo coerente.

1.4 Para que serve?

O crescimento tecnológico veio permitir que toda a informação possa ser suportada em computadores. Assim, ao nível das organizações, o sistema de informação tende a ter um significativo suporte informático.

Como consequência deste facto, torna-se necessário poder recorrer a uma linguagem que facilite a comunicação entre aqueles que têm de lidar com a informática: actuais e potenciais utilizadores que definem as suas necessidades, gestores que avaliam se os sistemas informáticos satisfazem essas necessidades e informáticos que desenvolvem as funcionalidades pretendidas.

O desenvolvimento de um modelo para um sistema de software antes de proceder à sua construção (implementação) ou renovação é essencial. Bons modelos são fulcrais para a comunicação entre as diferentes equipas que participam no projecto e para assegurar a solidez da arquitectura do sistema. À medida que a complexidade dos sistemas cresce, o mesmo acontece com a importância de usar boas técnicas de modelação. Existem muitos outros factores que contribuem para o sucesso de um projecto, mas a existência de uma boa política de modelação (através de uma rigorosa linguagem de modelação) é um factor essencial.

Existem diversas razões para usar a UML, das quais se destacam:

- **A construção de software necessita de um plano** – um produto de software minimamente competitivo torna-se também bastante complexo, tornando difícil a sua alteração / actualização. A existência de um modelo visual que necessite de ser aprovado antes do início da sua implementação visa diminuir essa complexidade, tanto na fase de desenvolvimento do produto, como nas fases seguintes de modificação e/ou actualização. A UML faculta aos seus utilizadores uma linguagem visual de modelação expressiva e pronta a usar.
- **A compreensão de um problema deve constituir a primeira fase da sua resolução** – o software é muito abstracto e difícil de visualizar. Uma linguagem de modelação como a UML permite que o software possa ser visualizado em múltiplas dimensões, permitindo que o sistema computacional seja completamente compreendido antes de ser implementado. Além disso, a UML pode ser usada para produzir vários modelos com um acréscimo do nível de detalhe. O âmbito geral do software pode ser definido fácil e rapidamente no início do projecto, através de um modelo com pouco detalhe. De seguida poder-se-ão adicionar vários níveis de detalhe à medida que se queira pormenorizar.

- **A UML é adequada tanto para sistemas novos como para a reestruturação de antigos** – é um erro pensar que, para que se possa usar uma nova técnica de modelação num sistema antigo, seja necessário a "re- documentação" total desse sistema para que a mudança possa ocorrer. A verdade é que apenas as partes que são afectadas pela mudança necessitam de ser modeladas.
- **Um software bem documentado é uma vantagem, já que o torna mais visível e transparente, reduzindo a dependência da empresa face a pessoas chave (que podem abandonar essa empresa a qualquer momento).**
- **A UML é uma linguagem unificada e universal** – a UML pode ser aplicada em muitas áreas de desenvolvimento de software e foi concebida de forma a fornecer mecanismos de especialização e extensão que permitam cobrir os requisitos específicos de cada caso em particular.
- **A UML possui especificações independentes das linguagens de programação usadas e dos processos de desenvolvimento.**
- **Um projecto é concebido por equipas de especialidades diferentes**, sendo extremamente importante uma boa comunicação (objectiva, clara e sem ambiguidades) entre essas equipas. Pelo facto de utilizar um conjunto de símbolos padrão, a UML facilita essa comunicação.

2 Normas

2.1 Normas Relacionadas: XMI, OCL, MOF

XMI. O *XML Metadata Interchange* (XMI), surgiu em resposta a um pedido efectuado em 1997 pela OMG. Esse pedido queria obter sugestões para um modelo de troca de informação pela Internet, e que pudesse ser guardado no computador como um simples ficheiro de sistema ou enviado pela internet através de uma base de dados ou repositório de informação. Várias empresas responderam ao apelo (uma vez que eram as mais beneficiadas), nascendo a norma XMI.

XMI define um modelo aberto de troca de objectos e modelos de negócio pela internet, de modo standard. As ferramentas de modelação que existiam tinham o seu formato próprio para partilha de informação, o que fazia com que as equipas de desenvolvimento ficassem dependentes das ferramentas utilizadas. Deste modo, com o modelo criado, equipas de desenvolvimento que trabalham em ambiente distribuído, podem trocar informação de maneira muito simples. O XMI permite a interligação e cooperação de várias pessoas sobre um projecto, independentemente do hardware e software, incluindo sistemas operativos, usados para o desenvolvimento. Consegue-se assim construir aplicações seguras e distribuídas.

A norma XMI tem como principal objectivo disponibilizar uma forma fácil de efectuar a troca de informação pela Internet. Para isto, são conjugados nesta norma três standards [15]: UML, especificado pela OMG; *eXntensible Markup*

Language (XML), standard da *World Wide Web Consurtion* (W3C); e *Meta Object Facility* (MOF), também da OMG. A arquitectura criada, permite que ferramentas informáticas partilhem objectos de duas formas possíveis: programaticamente usando interfaces CORBA, especificado nos standards MOF e UML; ou enviando as especificações dos modelos UML e MOF usando o standard XML. Cria-se então um elevado grau de integração entre as ferramentas e repositórios de informação, ao mesmo tempo que se permite uma rápida adaptação aos standards da OMG.

A norma XMI, como já apresentada é uma ferramenta muito poderosa. Contudo, torna-se muito mais poderosa, na medida em que define regras para a criação de DTD [1]. Os DTD's, tornam possível a troca de diagramas UML mas também de um qualquer outro tipo de diagrama. Criou-se uma norma muito útil para o desenvolvimento de software.

Esta norma foi criada devido às necessidades existentes das equipas de desenvolvimento em partilhar informação pela internet de forma simples e segura.

XMI 2.0 A norma XMI define muitos aspectos importantes relativos à descrição de objectos usando XML, nomeadamente [17]:

- A representação de objectos em termos de elementos e atributos em XML é a sua base.
- Inclui mecanismos que possibilitam a referência entre objectos no mesmo ficheiro ou em ficheiros diferentes.
- A identidade dos objectos, permite que sejam referenciados por outros objectos recarrendo a ID's [2] ou UUID's [3].
- A gestão de versões de objectos e suas definições é suportada no modelo XMI.
- A validação de ficheiros XMI, é feita usando DTD's e Esquemas [4].

O modelo descreve soluções para estes tópicos através da especificação de regras usando a norma EBNF [5], para criar documentos XML, DTD's e Esquemas que partilham objectos de forma consistente. A versão 1 da norma XMI define alguns tipos de regras de produção de documentos para partilha de objectos. Essas regras são:

- Produção de XML DTD's usando modelos de objectos.
- Produção de esquemas XML utilizando os modelos de objectos.
- Produção de documentos XML com base nos objectos.

De forma a gerar esquemas XMI, foram efectuados testes para verificar como é que o XMI ficaria se se usassem novas funcionalidades nos esquemas, que não estavam disponíveis nos DTD's. Com base nestas experiências definiu-se um

[1] Document Type Definition
[2] IDentifier
[3] Universal Unique IDentifier
[4] Esquemas de XML
[5] Norma que permite a definição do contexto de ficheiros

rumo para o XMI, bem como sugestões de melhoramento, que levaram à versão 2.0. Com a ajuda do trabalho efectuado pela W3C nos esquemas XML, a nova versão 2.0 da norma XMI acrescenta algumas regras de produção à especificação; sendo elas:

- Produção de esquemas XML com base em modelos de objectos.
- Produção de documentos XML compatíveis com os esquemas XMI.
- Engenharia reversa, possibilitando obter os modelos de objectos a partir dos esquemas XML.

OCL. *Object Constraint Language*(OCL) é uma linguagem formal para especificar expressões e restrições em linguagens orientadas a objectos e outras linguagens de modelação. Com o passar dos anos OCL deixou de ser uma extensão da UML para passar a ser uma parte integrada deste.

Um diagrama de modelação de objectos não é suficiente para especificações precisas e não ambíguas. Devido à necessidade de especificar restrições adicionais aos objectos no modelo, as quais geralmente são escritas em linguagem natural, resultando em ambiguidades, foi introduzido o OCL. OCL é uma linguagem de especificação formal, a qual não implica ter um vasto conhecimento matemático para se usar (como outras linguagens formais). Foi criada para evitar essas ambiguidades e que pode ser usada por um vasto ramo de utilizadores diferentes, visto ser uma linguagem formal simples de ler e escrever.

Pelo facto de ser uma linguagem de expressões, uma expressão OCL quando é avaliada não produz alterações no modelo, no estado dos objectos e não altera o fluxo de controlo.

Aplicações do OCL

- Especificar invariantes de classes e tipos num diagrama de classes
- Especificar tipos de invariantes para Estereótipos
- Especificar pré e pós condições em operações e métodos
- Especificar guardas de transições
- Especificar restrições e operações
- Ser usada como linguagem de navegação

OCL 2.0 Definida como uma *"query language"* geral, que pode ser usada em qualquer parte dos modelos UML para expressar propriedades desejadas. Em contraste às versões anteriores, esta contém uma definição dos conceitos e semântica do OCL por meios de um metamodelo complacente ao MOF.

Também é importante referir que a estrutura de especificação de OCL foi alterada. Existe uma clara separação entre sintaxe abstracta e concreta deixando espaço para sintaxes concretas alternativas. Há uma definição clara da semântica para todas as expressões na linguagem, tanto de uma maneira formal, ou matemática, como na forma baseada em UML. Existem outras alterações que não foram referidas [5].

MOF. Um metamodelo é um conjunto de metadados interrelacionados e utilizados para definir modelos. Estes definem formalmente os elementos, a sintaxe e semântica utilizadas. A norma MOF é um standard do OMG para representar e manipular este tipo de modelos.

O MOF define uma linguagem abstracta para especificação, construção e gestão de metamodelos independentemente da tecnologia de implementação. Alguns exemplos destes tipos de metamodelos são UML, *Common Warehouse Metamodel* (CWM) e o próprio MOF.

A especificação MOF tem por base os seguintes aspectos:

- Uma definição formal para o metamodelo MOF, ou seja, uma linguagem abstracta para a definição de metamodelos.
- Regras para o mapeamento dos metamodelos MOF para uma tecnologia de implementação, por exemplo, CORBA ou Java.
- Padrão XMI para intercâmbio, em XML, dos metadados e metamodelos entre ferramentas. O XMI define um conjunto de regras que mapeiam os metamodelos MOF e os metadados em documentos XML.

MOF 2.0 O MOF pode ser usado como uma linguagem como foi dito anteriormente para definir outras linguagens, neste caso UML.

Neste momento ao mesmo tempo que está a ser criada uma versão do 2.0 da UML, também está ser elaborada uma versão 2.0 da norma MOF, o que vai permitir que no futuro o MOF e a UML se combinem para formar um universo completo de instrumentos capaz de suportar, analisar e efectuar o design de um conjunto de famílias de ferramentas de várias associações, operando via XML Metadata Interchange. Assim, a chegada da UML 2.0 vai resolver alguns problemas no que diz respeito à falta de intercâmbio e semânticas inadequadas no que respeita à interligação com o MOF (problemas estes que existiam nas versões anteriores), tentando quebrar de alguma forma inconsistências que permaneciam entre ambos.

2.2 Mudanças da versão 1.* para a 2.0

A versão final da nova especificação UML, conhecida como UML 2.0 [22], tem data marcada para a sua aparição no final de 2004, início de 2005, depois de vários atrasos na sua elaboração. Mas desde 2003, são já conhecidas novas características fundamentais para o desenvolvimento desta metodologia.

MDA [20] A grande inovação do UML 2.0 em relação à versão anterior está na implementação da *Model Driven Architecture* (MDA). A MDA é uma nova maneira de escrever especificações e de desenvolver aplicações, baseado em modelos independentes da plataforma (PIM). Uma especificação MDA completa consiste num modelo UML independente da plataforma, e um ou mais modelos de uma plataforma específica (PSM).

O desenvolvimento em MDA foca em primeiro lugar a funcionalidade e o comportamento de uma aplicação ou de um sistema distribuído, independentemente das tecnologias em que vão ser implementadas. O MDA separa os detalhes

da implementação das funções do negócio. Ao contrário de outras arquitecturas, que estão ligadas a uma tecnologia em particular, com o MDA deixa de ser necessário repetir o processo de modelação sempre que uma nova tecnologia (p.e., XML/SOAP) aparece.

Onde entra a UML na implementação desta arquitectura? A UML é a chave para a aplicação desta tecnologia. Todas as aplicações MDA serão baseadas num modelo UML, tornando-se a UML numa ferramenta de criação, e não apenas numa ferramenta de documentação. Com a utilização dos modelos UML já existentes, e universalmente aceites, as aplicações que usam DMA irão assentar em 3 fundamentos básicos: Portabilidade, Interoperabilidade e Reutilização.

Com o surgimento do MDA, surge um novo futuro na criação de aplicações. Esta nova arquitectura irá permitir pegar num modelo UML e, directamente, convertê-lo em aplicações funcionais, com pouco ou nenhum trabalho de programação. O potencial desta arquitectura é enorme, contando já com software de desenvolvimento baseado em MDA [19] . Apesar disso, é um facto afirmar que ainda falta um longo caminho para a total implementação e utilização desta nova arquitectura.

A versão do 2.0 da UML vai corrigir alguns aspectos em que a UML 1.* falhou, entre os quais se salientam:

- O potential dos modelos Model-Driven Development(MDD) não são explorados ao máximo
- Capacidades inadequadas de modelação
- Demasiado complexo
- Definição de semânticas inadequadas
- Não existe capacidade de interligação entre diagramas
- Não existe total compatibilidade com o MOF

A especificação UML 1.* não é suficientemente formal, levando a incongruências em relação ao significado de cada modelo, assim como a variações na implementação por parte dos fabricantes de software de desenvolvimento em UML.

Para resolver estes problemas, a nova especificação 2.0 foi dividida em 4 importantes partes [23]:

- A infra-estrutura, o núcleo da linguagem, que é compatível com MOF, CWM e outros meta-modelos suportados pela OMG;
- A super-estrutura, sobre o núcleo, que disponibiliza funcionalidades para a construção dos modelos;
- O OCL, que permite adicionar novas restrições aos modelos, permitindo ao utilizador refinar todos os aspectos fundamentais da especificação;
- A interligação entre modelos, permitindo ligar e desenvolver vários modelos relacionados.

Esta divisão permite identificar as partes fundamentais da UML conseguindo toda as funcionar sem prejudicar o funcionamento das outras partes.

Desenvolvimento de sistemas em tempo real [21] Uma área não abrangida pela UML 1.* era a criação de sistemas em tempo real. A UML 1.* tem dois grandes problemas relacionados com esta área. Em primeiro lugar os sistemas em tempo real funcionam normalmente em modo embebido no hardware, tornando-se necessário que os modelos abranjam hardware e software. O outro problema prende-se com a afixação de datas como *'deadlines'* e períodos, pois estes estão lado a lado com os sistemas em tempo real.

Em relação à afixação de datas, a UML 2.0 fornece condições/limites de tempo e duração, assim como os novos diagramas de Temporização e de Sequência. Com a introdução do novo Diagrama de Distribuição, torna-se já possível a interligação entre o sofware e do hardware.

Profiles A UML 2.0 permite a criação e utilização de profiles. Profiles são pacotes estereotipados que contêm elementos de modelos que foram criados para uma determinada área ou objectivo. Existem já criados profiles para modelação de telecomunicações, calendarização, modelação de negócios, etc. A introdução dos profiles em UML 2.0, é feita através de metamodelos e estereótipos dos modelos já criados e adicionar-lhes novos métodos relevantes. A criação de profiles é importante, pois numa indústria em constante desenvolvimento não seria eficaz estar sempre a construir novos núcleos da especificação UML. Assim, torna-se fácil e prático evoluir a linguagem consoante as necessidades de cada utilizador.

Diagramas da UML 2.0 [18] Os metamodelos foram modificados para integrar novas noções. Quatro novos diagramas foram acrescentados, algumas mudanças técnicas foram feitas nos outros diagramas. Numa forma geral, a semântica dos diagramas e as acções associadas foram melhoradas.

Diagramas de Actividades:
Descreve processos de negócio de alto nível, incluindo fluxos de dados ou modelação lógica da complexidade dentro do sistema.

Diagrama de Casos de Utilização:
Demostra casos de uso, actores e relações entre eles.

Diagrama de Classes:
Representa uma vasta gama de elementos estáticos de um modelo, assim como classes e os seus tipos, o seu conteúdo e as suas relações.

Diagrama de Componentes:
Descreve os componentes que compõem uma aplicação, sistema ou empresa. São descritos os componentes, as suas inter relações, interacções e as suas interfaces públicas.

Diagrama de Distribuição:
Representa a arquitectura de execução de sistemas. Incluindo nós, ambientes de execução de hardware ou software, assim como o middleware que os liga.

Diagrama de Máquina de Estados:
Descreve os estados em que um objecto ou interacção se pode encontrar, assim como a transição entre estados. Antes conhecido como Diagrama de Estados.[1]

Diagrama de Objectos:
Descreve objectos, as suas relações num ponto no tempo, tipicamente no caso especial de um diagrama de classes ou de um diagrama de comunicações.

Diagrama de Pacotes:
Representa a forma como os elementos dos modelos estão organizados em pacotes, assim como as dependências entre pacotes.

Diagrama de Sequência:
Modela a lógica sequencial, em concreto a ordenação temporal de mensagens entre objectos num determinado contexto.

Novos Diagramas introduzidos

Diagrama de Comunicações:
Representa instâncias de classes as suas inter relações e o fluxo de mensagens entre elas. São semelhantes aos Diagramas de Sequências, focam-se principalmente na organização estrutural de objectos que enviam e recebem mensagens. Frequentemente será necessário efectuar uma escolha entre o Digrama de Sequencias e o Digrama de Comunicações. Se representar o tempo ou sequência de eventos for o mais importante, devem ser usados os Diagramas de Sequências. Se o objectivo for mostrar o contexto deverá ser usado o Diagrama de Comunicações. Anteriormente conhecidos como Diagramas de Colaboração.

Diagramas de Estrutura de Composições:
Descreve a estrutura interna de uma classe, componente ou caso de uso, incluindo os pontos de interacção destes com outras partes do sistema. É semelhante ao Diagrama de Classe, mas representa partes e conectores. As partes não são necessariamente classes no modelo e não representam instâncias particulares, mas podem representar papeís que os objectos ou instâncias desempenham. As partes são descritas de maneira semelhante aos objectos.

Diagrama Temporal:
Descreve as mudanças de um estado ou condição de uma instância de um objecto ou papel ao longo do tempo. Tipicamente usado para mostrar a mudança de estados de um objecto com o decorrer do tempo em resposta a eventos externos. Podem ser usados para definir componentes de software e *hardware-driven*. Os Diagramas Temporais podem ser desenhados com base num valor ou baseados temporalmente num ponto de vista

Diagrama de Vista Geral de Interacção:
Uma variante do diagrama de actividades que fornece uma visão geral do fluxo de controlo dentro de um processo de sistema ou de negócio.

Os diagramas de actividades foram os que sofreram as maiores alterações, não só permitem descrever os fluxos de controlo, mas também têm ferramentas que suportam a automação desses fluxos.

Nos diagramas de casos de utilização muitas vezes quer-se mostrar como um caso de uso se extende a outro através dos pontos de extensão, que foram introduzidos na UML 1.3 e agora aprimorados, permitem mostrar a lógica necessária

para um caso de uso estender o outro. Como nas associações entre classes, nos diagramas de classes, também se pode estabelecer uma multiplicidade entre os actores e os casos de uso. Os diagramas de classes não sofreram grandes alterações, não alterando as classes mas fazendo alguns acrescentos aos atributos. A nova versão da UML cria uma relação de equivalência entre atributos como *"strings"* compartimentadas e atributos como associações. As generalizações são semelhantes às da UML 1.*, mas as associações podem ser navegadas, isto é, pode-se permitir uma navegação no sentido da seta indicada na associação, ou restringir colocando um "X".

Os diagramas de colaboração, são agora conhecidos por diagramas de comunicação, os nós de um diagrama e comunicação são chamados de linhas de vida, nomenclatura semelhante à dos diagramas de sequência, estes nós estão ligados por mensagens indicando o sentido em que a interacção se realiza. As mensagens agora podem ser mandadas simultaneamente colocando uma letra depois do número de sequência.

A UML deixou de ser usada apenas para descrever sistemas de software, está a progredir rapidamente no sentido de proporcionar um ambiente de modelação visual que acompanhe as exigências da tecnologia de software actual e necessidades de comunicação. A UML tem que progredir de maneira a descrever e automatizar processos de negócio assim como tornar-se uma linguagem para desenvolver sistemas de plataformas diferentes. Uma nota especial foi definida: a especificação deve ser perfeitamente compatível e portátil com a versão anterior. Assim, quem não achar necessário usar as novas metodologias poderá usar a UML como antigamente.

3 Conclusões

A UML focava bem a definição das estruturas dos sistemas e o seu comportamento, mas era bastante limitada no que diz respeito à descrição de objectivos de teste e procedimentos de teste. Contudo, com o desenvolvimento dos sistemas de engenharia de geração de código automático, a necessidade de testes sólidos aumentou. Um outro problema do formato da UML era o facto de "não ter semântica", ou seja, era difícil ter uma percepção global das interligações que existiam entre os vários fragmentos de um modelo.

Em Junho de 2001, foi iniciado um OMG *Request for Proposal* (RFP), tendo em vista, entre outras coisas, uma reformulação da UML que incluísse a especificação de testes para aspectos estruturais e comportamentais de modelos UML computacionais e a capacidade de operar com tecnologias de teste de sistemas do tipo caixa-negra, existentes.

Em Abril de 2003 o consórcio U2 Partners (consórcio de vendedores e utilizadores de UML, dedicados a tornar a UML fácil de aplicar, implementar e utilizar) completou as alterações finais à terceira revisão da proposta da superestrutura (define construtores ao nível do utilizador para especificar a estrutura e o comportamento de sistemas) da UML 2.0 e submeteu-a ao OMG para consideração. O OMG *Analysis and Design Task Force* (ADTF) recomendou

unanimemente que o OMG adoptasse a super-estrutura, em Junho de 2003. A OMG classificou-a como especificação final adoptada, em Agosto de 2003.

A adopção da superstrutura final foi um ponto culminante em três anos e meio em que decorreu o principal processo de revisão.

Mais de 50 empresas contribuíram, com as suas melhores tecnologias e as suas melhores práticas, para o desenvolvimento da UML 2.0.

Esta especificação inclui um complemento às definições das versões anteriores da UML e portanto implicou um desenvolvimento da indústria do principal software de modelação notacional. A UML 2.0, representa o próximo passo revolucionário que irá permitir uma grande capacidade de expressão e a especificação da comunicação entre sistemas. Espera-se que esta linguagem tenha um grande efeito em propósitos de estandardização, nos mais diversos tipos de utilização e que tenha uma grande capacidade de precisão, bem como as ferramentas que servem de suporte à sua implementação.

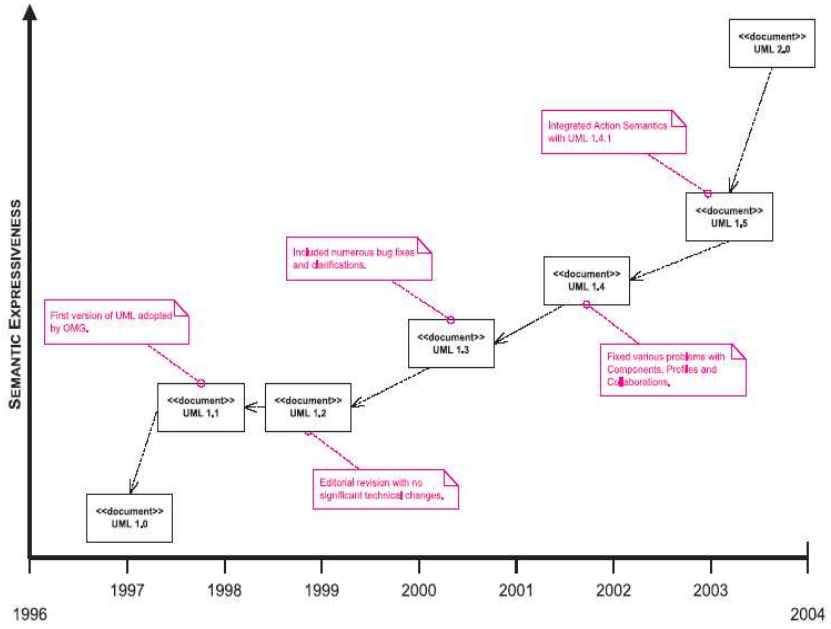

Figura 2. Evolução da UML 1.0 para a UML 2.0

Referências

1. Object Management Group, "Unified Modeling Language", Última actualização 29 de Março de 2004,
http://www.uml.org

2. U2 Partners, "Object Constrain Language Specification", version 1.1, 1 Setembro 1997,
http://umlcenter.visual-paradigm.com/umlresources/obje_11.pdf
3. Object Management Group, "UML 2.0 OCL Specification", Outubro 2003,
http://www.omg.org/docs/ptc/03-10-14.pdf
4. Sten Loecher e Stefan Ocke, "A Metamodel – Based OCL – Compiler for UML and MOF", Department of Computer Science Dresden University of Technology Dresden, Germany,
http://i12www.ilkd.uni-karlsruhe.de/ baar/oclworkshopUml03/papers
/03_meta_model_based_ocl_compiler.pdf
5. Klasse Objecten, Soest, the Netherlands, "OCL 2.0 Submission", Última actualização 27 de Fevereiro de 2004,
http://www.klasse.nl/ocl/ocl-subm.html
6. Luciana de Araujo Spagnoli e Karin Becker, "Um estudo sobre o Desenvolvimento Baseado em Componentes", Faculdade de Informática, Programa de Pós-Graduação em Ciência da Computação, PUCRS Brasil, Maio de 2003,
www.inf.pucrs.br/tr/tr026.pdf
7. Randy Miller, "What's New in UML 2? The Use Case Diagram", Borland Software Corporation, Última actualização 30 de Junho de 2003,
http://bdn.borland.com/article/0,1410,30166,00.html
8. I. Schieferdecker, Z. R. Dai, J. Grabowski, A. Rennoch: The UML 2.0 Testing Profile and its Relation to TTCN-3. Fraunhofer FOKUS - Competence Center for Testing, Interoperability and Performance, University of Lübeck - Institute for Telematics, 2003.
http://www.swe.informatik.uni-goettingen.de/publications/IS_ZD_JG_AR/
TestCom2003_UTP_Final.pdf
9. Bran V. Selic: On the Semantic Foundations of Standard UML 2.0. IBM Rational Software Canada, Abril, 2004.
http://www-128.ibm.com/developerworks/rational/library/content/04April/
semanticfoundations.pdf
10. Cris Kobryn: UML3.0 and the future of modeling. CEO, PivotPoint Technology Corp., Maio, 2003.
http://www.kobryn.com/docs/papers/SoSym_Mar04_p4_Kobryn.pdf
11. UML 2.0 Standard Officially Adopted at OMG Technical Meeting in Paris. Object Management Group, Agosto, 2004.
http://www.omg.org/news/releases/pr2003/6-12-032.htm
12. Making Better Standars. European Telecommunications Standards Institute, 2004.
http://portal.etsi.org/mbs/Languages/UML/uml2.asp
13. Cris Kobryn: UML 2.0 Preview. Object Management Group, Fevereiro, 2003.
http://syseng.omg.org/INCOSE%20IW%202003/UML2-Preview%20Kobryn.ppt
14. OMG e Unisys. XML Metadata Interchange (OMG XMI) Distributed Metadata Interchange for the WEB Generation, 1994
http://www.omg.org/docs/omg/99-04-04.pdf
15. Cover Pages. XML Metadata Interchange (XMI). Última modificação: 03-05-2002
http://xml.coverpages.org/xmi.html
16. XML Journal. XMI: The OMG's XML Metadata Interchange Standard. 2004
http://www.sys-con.com/xml/article.cfm?id=45
17. Object Managment Group. XML Metadata Interchange (XMI) Specification. 02-05-2003
http://www.omg.org/docs/formal/03-05-02.pdf

18. Novos Diagramas da versão UML 2.0, 2004
http://www.xpdian.biz/UML2.0.html
19. Compuware Corporation, OptimalJ - Desenvolvimento de Aplicações J2EE usando a arquitectura MDA, 2004
http://www.compuware.com/products/optimalj/
20. Object Management Group, Model Driven Architecture, 2003 http://www.omg.org/mda/
21. Kirsten Berkenkötter: Using UML 2.0 in Real-Time Development, Outubro de 2003
http://www-verimag.imag.fr/EVENTS/2003/SVERTS/PAPERS-WEB/04-Berkenkoetter-UMLRT-critic.pdf
22. ComputerWorld, Entrevista com Grady Booch e Bran Selic, a respeito do novo UML 2.0, 22 de Março de 2004
http://www.computerworld.com/developmenttopics/development/story/0,10801,91325,00.html
23. Object Management Group, Especificações Finais UML 2.0, 2003
- Infra-estruturas, http://www.omg.org/cgi-bin/doc?ptc/2003-09-15
- Super-estruturas, http://www.omg.org/cgi-bin/doc?ptc/2003-08-02
- OCL, http://www.omg.org/cgi-bin/doc?ptc/2003-10-14
- Interligação entre modelos, http://www.omg.org/cgi-bin/doc?ptc/2003-09-01