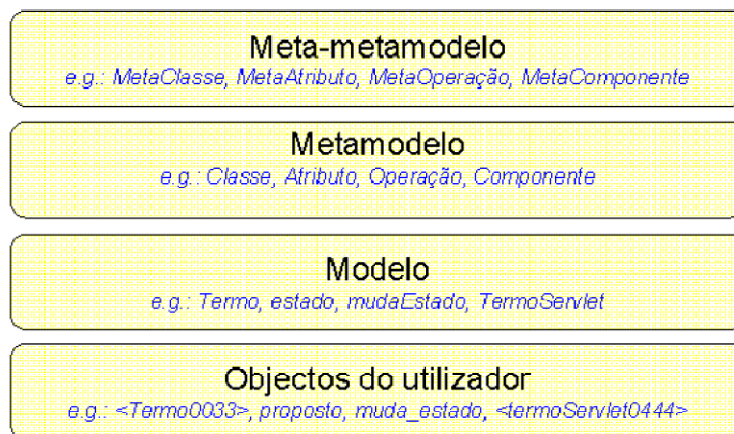


ARQUITETURA UML

A Estrutura do UML a Quatro Camadas

O UML está estruturado numa arquitetura de quatro camadas conforme sugerido na figura: meta-modelo; metamodelo; modelo; e objetos do utilizador.



Na camada mais baixa da arquitetura – “Camada Objectos do Utilizador” – encontram-se os elementos que constituem as instâncias concretas dos elementos representados num modelo definido pelo utilizador do UML (i.e., o analista ou consultor, e não o utilizador do sistema final): os objetos, os cenários de um caso de utilização, as instâncias de componentes, etc. Por exemplo, o objeto `Termo0033`, o valor `proposto`, a operação `muda_estado`, ou a instância do `servlet` `termoServlet0444` ilustram elementos desta camada.

Na segunda camada – “Camada Modelo” – encontram-se os elementos que constituem as abstrações definidas pelo utilizador do UML no seu processo de definição de modelos. Por exemplo, a classe `Termo`, o atributo `estado`, a operação `mudaEstado`, o componente `TermoServlet` ou o caso de utilização “Propor `Termo`” ilustram elementos desta camada.

Na terceira camada – “Camada Metamodelo” – encontram-se os elementos que constituem as abstrações definidas pelos criadores do UML, ou sejam, os elementos que constituem a estrutura de conceitos do UML, com base nos quais se podem definir modelos (daí a designação de “metamodelo”). Por exemplo: `Classe`, `Atributo`, `Operação`, `Componente`, `Nó`.

Estas três primeiras camadas são aquelas a que a generalidade dos utilizadores UML acabam por recorrer na maior parte das vezes, quer como utilizadores (da terceira camada) quer como criadores (de elementos das duas primeiras camadas). Por isso é natural que se coloque a questão: para quê a quarta camada? De fato, esta camada – “Camada Meta-metamodelo” – será raramente usada pelo utilizador normal do UML. No entanto, é necessária para os responsáveis pelo desenho e comparação deste tipo linguagens. Esta camada permite definir os elementos que serão responsáveis pela especificação de metamodelos. Por exemplo: `MetaClasse`, `MetaAtributo`, `MetaOperação`, `MetaComponente`, `MetaNó`.

A Camada Metamodelo

O metamodelo UML encontra-se estruturado nos seguintes pacotes lógicos: *Foundation*, *Behavioral Elements* e *Model Management*, os quais por sua vez são decompostos noutros subpacotes.

O metamodelo é descrito de forma semi-formal usando as seguintes perspectivas: (1) sintaxe abstrata, (2) regras e (3) semântica. A sintaxe abstrata é providenciada através de um modelo descrito num subconjunto do próprio UML, consistindo nos diagramas de classes UML e por uma descrição em linguagem natural. Por último, a semântica é descrita principalmente em linguagem natural. De fato, o metamodelo UML é descrito por uma combinação de notações gráficas, linguagem natural e linguagem formal, o que segundo os seus proponentes oferece um compromisso adequado entre expressividade, rigor e facilidade de leitura [OMG99].

Pacote Foundation

O pacote *Foundation* é o componente da linguagem que especifica a estrutura estática dos modelos. Contem os seguintes pacotes *Core*, *Extension Mechanisms*, e *Data Types*

- O pacote *Core* especifica os conceitos básicos do metamodelo e define a estrutura arquitetural que permite a associação de construtores adicionais, tais como metaclasses, metaassociações, e metaatributos; define os principais construtores do metamodelo (abstratos e concretos) necessários ao desenvolvimento de modelos de objetos.
- O pacote *Extension Mechanisms* especifica como os elementos do modelo podem ser configurados e estendidos com nova semântica. Define a semântica de estereótipos, restrições e marcas com valores através da definição dos seguintes construtores concretos, respectivamente *Stereotype*, *Constraint* e *TaggedValue*.
- O pacote *Data Types* define as estruturas de dados básicas da linguagem, tais como tipos de dados primitivos (e.g., *Integer*, *String*, *Time*), enumerados (e.g., *Boolean*, *ggregationKind*, *VisibilityKind*), e outros (e.g., *Expression*, *Mapping*, *Name*, *Multiplicity*).

Pacote Behavioral Elements

O pacote *Behavioral Elements* contem os subpacotes *Common Behavior*, *Collaboration*, *Use Cases*, *Activity Graphs*, e *State Machines* que permitem representar o comportamento e a dinâmica de um sistema.

Pacote Model Management

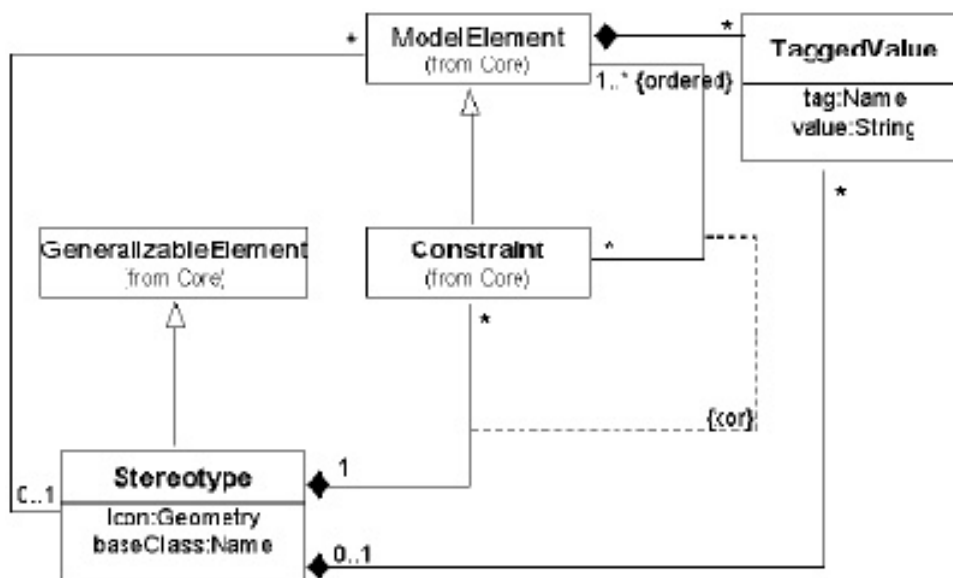
O pacote *Model Management* depende do pacote *Foundation* e consiste num diagrama de classes que ilustra como os elementos *Model*, *Package* e *Subsystem* se organizam e relacionam entre si. Estes elementos servem principalmente como unidades de agrupamento e de organização dos outros elementos do modelo.

MECANISMOS DE EXTENSÃO

O UML providencia um número elevado de conceitos e notações particularmente concebidos de forma a satisfazer os requisitos típicos de modelação de software. Contudo, podem surgir situações em que se torna desejável a introdução de conceitos e/ou de notações adicionais para além dos definidos originalmente no momento da definição do standard. Para contemplar tais situações, o UML providencia diversos mecanismos que permitem estendê-lo de forma consistente: restrições, marcas e estereótipos. Estes mecanismos permitem:

- Introduzir novos elementos de modelação para uma maior expressividade e compreensão dos modelos UML a criar.
- Definir itens standard que não são considerados suficientemente interessantes ou complexos para serem definidos diretamente como elementos do metamodelo UML.
- Definir extensões específicas das linguagens de implementação ou específicas dos processos de desenvolvimento.
- Associar arbitrariamente informação semântica e outra aos elementos do modelo.

Estes mecanismos aplicam-se aos elementos do modelo, não às suas instâncias. Representam, portanto, extensões à própria linguagem que permitem alterar a estrutura e semântica dos modelos criados. A figura abaixo ilustra a sintaxe abstrata dos mecanismos de extensão do UML. Note-se a definição e relação entre as metaclasses Stereotype, Constraint e TaggedValue.



Restrições

Consiste na especificação de semântica associada a um (ou a um conjunto ordenado de) elemento(s) do modelo. Tal especificação é escrita numa determinada linguagem de restrições, que pode ser mais ou menos formal.

Marcas com Valor

É um par <marca,valor> que permite associar arbitrariamente informação a qualquer elemento do modelo. Uma marca é um nome arbitrário, havendo contudo alguns nomes predefinidos como elementos standard

Estereótipos

Providencia um mecanismo de classificação de elementos, tal que eles se comportem, de alguma forma, como se fossem instâncias de novos construtores definidos no metamodelo. Um estereótipo pode especificar adicionalmente restrições e marcas com valor que serão aplicadas às suas instâncias. Um estereótipo pode ser usado para especificar diferenças de utilização ou de semântica entre dois elementos com uma estrutura relativamente semelhante.

Se um elemento do modelo (de nível utilizador) é classificado por um determinado estereótipo, a classe base desse elemento tem de coincidir com a classe base especificada para esse estereótipo. As restrições e marcas com valores são implicitamente associadas a esse elemento do modelo.

PERFIS UML

A fim de 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 conjunto predefinido de estereótipos, marcas com valor, restrições e ícones que conjuntamente especializam e configuram o UML para um determinado domínio de aplicação ou um determinado processo de desenvolvimento. Note-se que um perfil não estende o UML através da introdução de novos conceitos de base.

Na especificação 1.3 do UML são descritos dois perfis, designados como “perfis standard”: (1) perfil para processos de desenvolvimento de software; e (2) perfil para modelação de processos e estruturas de negócios.

Perfil para Processos de Desenvolvimento de Software

O perfil “Processos de Desenvolvimento de Software” consiste na definição de um conjunto de estereótipos (atualmente não foram definidos quaisquer marcas com valor ou restrições) usados tipicamente em processos de desenvolvimento de software inspirados no *Unified Process* [Jacobson99], em particular pelo RUP ou o ICONIX

Estereótipos de Modelos, Pacotes e Subsistemas

Um sistema modelado com base no *Unified Process* consiste em vários modelos relacionados. Esses modelos são caracterizados pela fase do ciclo de vida que representam, designadamente: casos de utilização; análise; desenho; e implementação.

Casos de Utilização

O modelo de casos de utilização («use case model») especifica os serviços que um sistema providencia do ponto de vista dos seus utilizadores. Um sistema de casos de utilização («use case system») é um pacote de nível inicial ou de topo, e contém outros pacotes (de estereótipo «use case package») e/ou casos de utilização e relações respectivas. Por fim, um «use case package» é um pacote que contém apenas casos de utilização e relações.

Análise

Um modelo de análise («analysis model») é um modelo cujo pacote de mais alto nível é do estereótipo «analysis system». Este pacote contém pacotes de análise (de estereótipo «analysis package») e/ou pacotes de serviços (de estereótipo «analysis service package») e/ou classes de análise (de estereótipos «entity», «boundary» e «control») e respectivas relações. Um pacote de análise contém pacotes de análise, pacotes de serviços de análise, classes de análise e relações. Por fim, um pacote de serviço de análise contém apenas classes de análise e relações.

Desenho

Um modelo de desenho («design model») é um modelo cujo subsistema de mais alto nível é do estereótipo «design system». Um sistema de desenho contém subsistemas de desenho (de estereótipo «design subsystem») e/ou subsistemas de serviços de desenho (de estereótipo «design service subsystem») e/ou classes de desenho e respectivas relações. Um subsistema de desenho contém outros subsistemas de desenho, subsistemas de serviços de desenho, classes de desenho e relações. Por fim, um subsistema de serviço de desenho contém apenas classes de desenho e relações.

Implementação

Um modelo de implementação («implementation model») é um modelo cujo subsistema de mais alto nível é do estereótipo «implementation system». Um sistema de implementação contém subsistemas de implementação (de estereótipo «implementation subsystem») e/ou componentes e respectivas relações. Um subsistema de implementação contém outros subsistemas de implementação, componentes e relações.

Estereótipos de Classe

Definem-se três estereótipos para as classes de análise: «entity», «boundary» e «control» (ver figura). Para as classes de desenho não são definidos quaisquer estereótipos particulares.

Entidade («entity»)

Uma entidade («entity») é uma classe passiva, no sentido que os seus objetos não iniciam interações por si próprios. Este tipo de objetos participa em diferentes realizações de casos de utilização e geralmente persiste para além das interações em que participa (i.e., são objetos persistentes).

Controle («control»)

Um controlo («control») é uma classe cujos objetos controlam as interações entre coleções de objetos. Geralmente este tipo de classes tem um comportamento específico para determinado caso de utilização e normalmente as suas instâncias não persistem para além das interações em que participam (i.e., são objetos não persistentes).

Fronteira («boundary»)

Uma fronteira ou interface («boundary») é uma classe que se encontra na fronteira de um sistema, contudo, ainda dentro dele. A responsabilidade destes objetos é mediar a comunicação entre os atores externos do sistema e os outros objetos internos ao sistema.

Estereótipos de Associação

São apenas definidos neste perfil UML dois estereótipos particulares para associações entre classes: «communicate» e «subscribe».

Comunicação («communicate»)

A comunicação («communicate») é uma associação entre atores e casos de utilização, que traduz o envio de mensagens do ator para o caso de utilização e/ou vice-versa. Pode também ser usada entre as classes de interface, controlo e entidade, e entre atores e classes de interface. Se for relevante, pode-se explicitar a direção da comunicação através de um adorno de navegabilidade (por omissão, assume-se comunicação bidirecional).

Subscrição («subscribe»)

A subscrição («subscribe») é uma associação entre duas classes estado, cujos objetos da classe subscritora são notificados quando um determinado evento ocorre nos objetos da classe destino (designada por classe “publicadora”). A associação inclui a especificação do conjunto de eventos, cuja ocorrência implica que o subscritor seja notificado.



«control»
TermoSRegistrar



«boundary»
TermoSWindow



«entity»
Termo

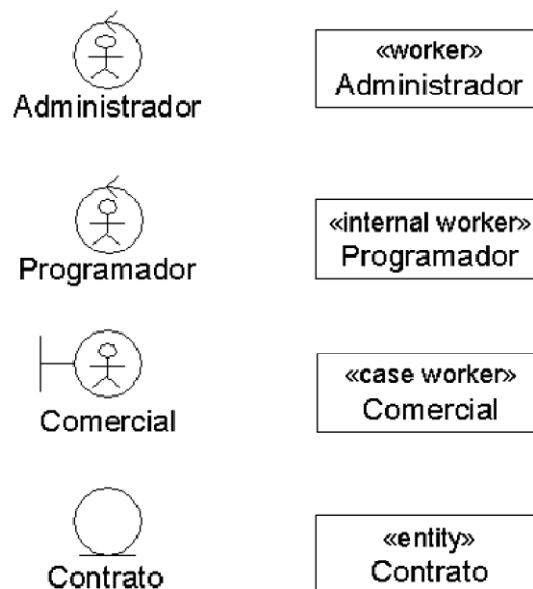
Perfil para Modelação de Negócios

O perfil “Modelação de Negócios” consiste na definição de um conjunto de estereótipos (não foram definidos quaisquer marcas com valor ou restrições) e ilustra as capacidades do UML para modelar não apenas sistemas de software, mas outras realidades, neste caso “o mundo das organizações”.

No contexto deste perfil, um trabalhador («worker») atua dentro de um determinado negócio, participa em casos de utilização e interatua com outros trabalhadores e entidades («entity»). Os trabalhadores internos («internal worker») interagem com outros trabalhadores dentro do negócio, enquanto que os trabalhadores de interface («case worker») interagem com os atores externos ao sistema.

Os objetos interagem segundo dois tipos de associações: de comunicação («communicate») e de subscrição («subscribe»), segundo o modelo de subscrição/publicação.

Um conjunto de entidades agregadas de forma combinada constitui uma unidade de trabalho («work unit»). Por fim, um conjunto de unidades de trabalho, de classes e de associações agregadas de forma combinada constitui uma unidade organizacional («organization unit») de suporte ao negócio.



Perfil para Modelação de Aplicações Web

Jim Conallen, no seu livro “*Building Web Applications with UML*” [Conallen00], propõe um processo de modelação de aplicações Web baseado em UML.

Aplicações Web são aplicações baseadas em diversos modelos computacionais (e.g., computação no cliente, computação no servidor, computação distribuída entre cliente e servidor) e tecnologias (e.g., HTTP, HTML, *frames*, âncoras, linguagens de *scripting*, DOM, XML, Java/Beans, ActiveX/COM, objetos distribuídos) que estão na base da generalidade dos atuais sistemas de informação das organizações em todo o mundo.

Conollen propõe um número significativo de extensões ao UML, designado no seu conjunto como “WAE” (*Web Application Extension*), designadamente os seguintes estereótipos, derivados a partir de:

- Metaclasses Class: Server Page, Client Page, Form, Frameset, Target, JavaScript Object, ClientScript Object.
- Metaclasses Association: Link, Targeted Link, Frame Content, Submit. Builds, Redirect, IIOP, RMI.
- Metaclasses Attribute: Input Element, Select Element, Text Area Element.
- Metaclasses Component: Web Page, ASP Page, JSP Page, Servlet, Script Library.

O processo de desenvolvimento deste tipo de aplicações é clássico nas fases de especificação de requisitos e de análise, mas é inovador na fase de desenho. É nesta fase que é aplicado o WAE de forma adequada. Apresenta-se de seguida, a título de exemplo, a definição do estereótipo `Client Page` e do estereótipo `Link`

EXERCÍCIOS

Ex.01. Um estereótipo pode estender um elemento do tipo associação? Dê um exemplo e justifique a sua resposta.

Ex.02. O que são e para que servem os perfis UML?

Ex.03. O que é a classe base de um estereótipo? Qual é a classe base do estereótipo «entity», definido no perfil UML para processos de desenvolvimento de software?