

7ª e 8ª aula: Modelagem de UML

Técnicas e Notações mais Utilizadas

A proliferação das metodologias que tinham como base os conceitos da orientação por objetos levou ao aparecimento de diversas notações e técnicas de modelação, que muitas vezes são partilhadas entre várias delas. Os recentes esforços de unificação permitiram que algumas dessas técnicas se tenham destacado.

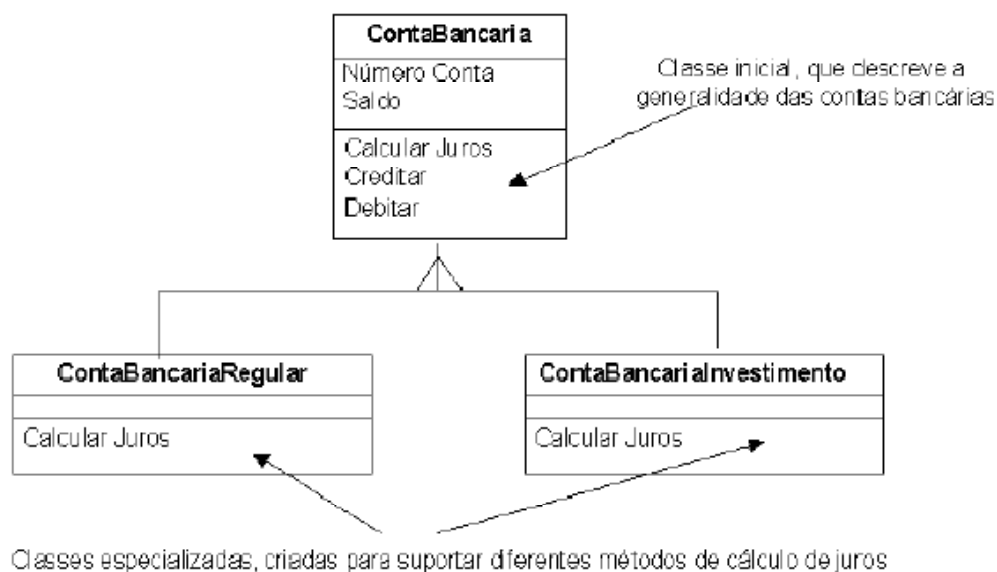
Resumidamente, indicamos de seguida as principais notações a considerar:

- Diagrama de casos de utilização
- Diagrama de classes
- CRC (Class-Relationship-Collaboration) cards
- Diagrama de pacotes
- Diagrama de estados
- Diagramas de interação
- Diagrama de actividades
- Diagrama de eventos
- Diagrama de contexto
- Diagrama de fluxo de dados.

Independentemente dos nome dos diagramas utilizados por cada metodologia, cada uma apresenta notações para modelar a visão estática do sistema (a estrutura dos objetos, relações de agregação e de especialização, e a comunicação entre objetos) e outras notações para os conceitos dinâmicos (interações, mudanças de estado, sequências de acções e mecanismos de temporização).

Representação da Realidade

Nas abordagens orientada por objetos, as entidades do mundo real são captadas directamente por objetos: um carro é um objecto do tipo “carro”, uma conta bancária é um objecto do tipo “conta bancária”, etc. Tal como as entidades do mundo real, os objetos apresentam dados e comportamento, bem como identidade própria.



Outros Aspectos

O desenvolvimento de software segundo uma abordagem orientada por objetos apresenta outras vantagens comparativamente com a abordagem estruturada, resumidamente:

- Providencia blocos de construção de alto nível que **reduzem os custos de desenvolvimento**, pela promoção da reutilização e encapsulamento de software. Este facto permite reduzir as interdependências e facilitar a manutenção posterior.
- Facilita a transferência de conhecimento através da adopção de “padrões de desenho”, **reduzindo o custo de aprendizagem**.
- Providencia um framework (aplicacional ou de middleware) para facilitar a extensão do sistema, **reduzindo o custo de novas versões**.
- **Reduz o custo de alteração** do sistema, em resposta às expectativas e requisitos dos utilizadores

EXEMPLO Imagine que é o responsável pela gestão de informação sobre alunos de uma universidade, sendo para isso relevante reproduzir no sistema de informação as acções que normalmente ocorrem ao longo do ano na universidade: matrículas de alunos num ano lectivo e em cadeiras; inscrições nas aulas práticas; inscrições em exames; pagamentos de propinas; consultas de notas atribuídas; pedidos de certificados.

- Se tivesse que modelar o sistema anteriormente descrito, indique qual (ou quais) o(s) diagrama(s) que utilizaria se aplicasse uma abordagem estruturada e qual (ou quais) utilizaria se aplicasse uma abordagem orientada por objetos, justificando a sua resposta.
- Efectue a modelação dos conceitos aqui representados pelos diagramas estruturados que achar convenientes, justificando a sua resposta.
- Efectue a modelação dos conceitos aqui representados pelos diagramas relacionados com abordagens orientadas por objetos que achar convenientes, justificando a sua resposta.
- Quais as principais alterações que ocorreram no processo de desenvolvimento de software com a introdução das abordagens estruturadas?

UML

O UML apresenta, entre outras, as seguintes características principais: (1) é independente do domínio de aplicação (i.e., pode ser usado em projectos de diferentes características, tais como sistemas cliente/servidor tradicionais; sistemas baseados na Web; sistemas de informação geográficos; sistemas de tempo real); (2) é independente do processo ou metodologia de desenvolvimento; (3) é independente das ferramentas de modelação; (4) apresenta mecanismos potentes de extensão; (5) agrega um conjunto muito significativo de diferentes diagramas/técnicas dispersos por diferentes linguagens (e.g., diagramas de casos de utilização, de classes, de objetos, de colaboração, de actividades, de estados, de componentes, e de instalação).

Tipos de Elementos Básicos

A estrutura de conceitos do UML é razoavelmente abrangente consistindo num conjunto variado de notações, as quais podem ser aplicados em diferentes domínios de problemas e a diferentes níveis de abstracção. A estrutura de conceitos do UML pode ser vista através das seguintes noções: (1) “coisas” ou elementos básicos, com base nos quais se definem os modelos; (2) relações, que relacionam elementos; e (3) diagramas, que agrupam elementos.

Os elementos encontram-se organizados consoante a sua funcionalidade ou responsabilidade. Assim há elementos de estrutura, comportamento, agrupamento e de anotação. A Figura 4.2. ilustra o conjunto dos principais elementos de estrutura: classes, classes activas, interfaces, casos de utilização, actores, colaborações, componentes e nós.

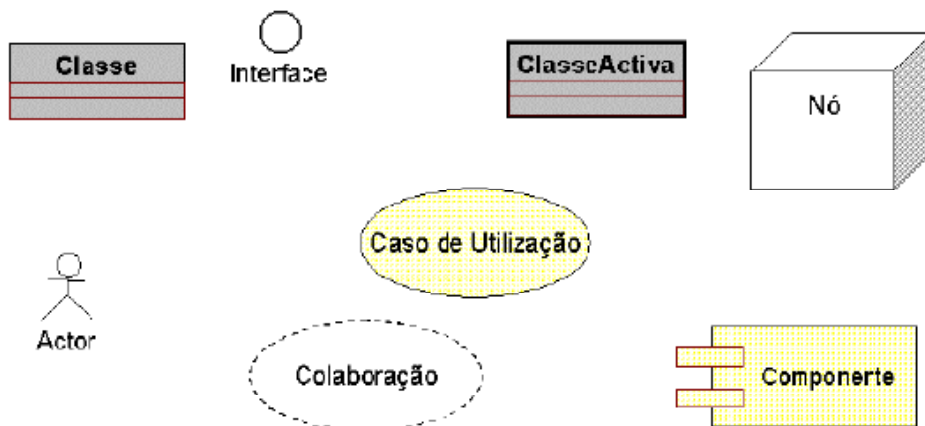


Figura 4.2: Resumo dos elementos de estrutura.

A Figura 4.3 ilustra outros elementos básicos do UML, elementos de comportamento (estados e mensagens), de agrupamento (pacotes) e de anotação (anotações ou notas).

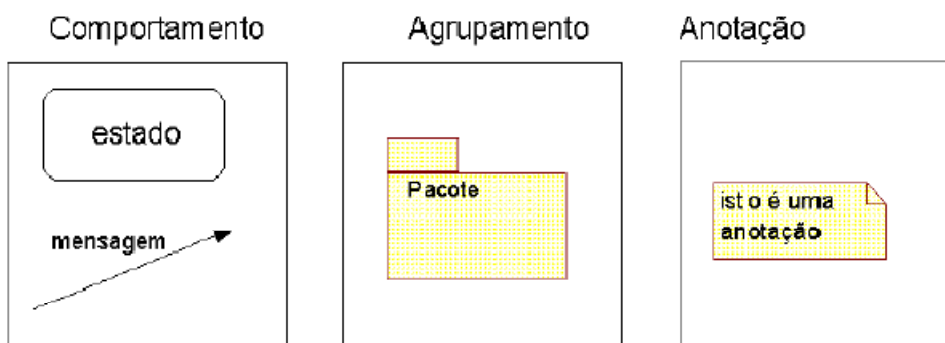


Figura 4.3: Resumo dos elementos de comportamento, agrupamento e anotação.

Diagramas de Casos de Utilização

Um diagrama de casos de utilização descreve a relação entre actores e casos de utilização de um dado sistema (ver exemplo da Figura 4.5). Estes diagramas são utilizados preferencialmente na fase de especificação de requisitos e na modelação de processos de negócio.

Este é um diagrama que permite dar uma visão global e de alto nível do sistema, sendo fundamental a definição correta da sua fronteira.

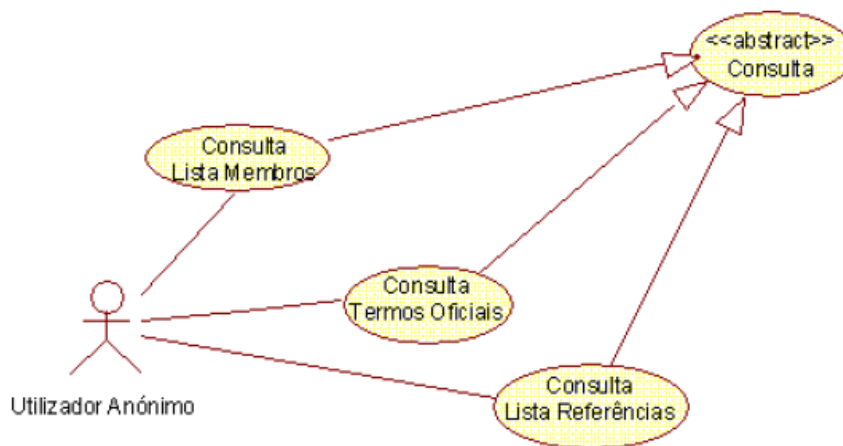


Figura 4.5: Exemplo de um diagrama de casos de utilização.

Diagramas de Modelação da Estrutura

Os diagramas de classes do UML são uma integração de diferentes diagramas de classes existentes, nomeadamente no OMT [Rumbaugh91], Booch [Booch94] e outros métodos OO. Extensões específicas de determinados processos (e.g. recorrendo a estereótipos e correspondentes ícones) podem ser definidos em vários diagramas para suportarem diferentes estilos de modelação.

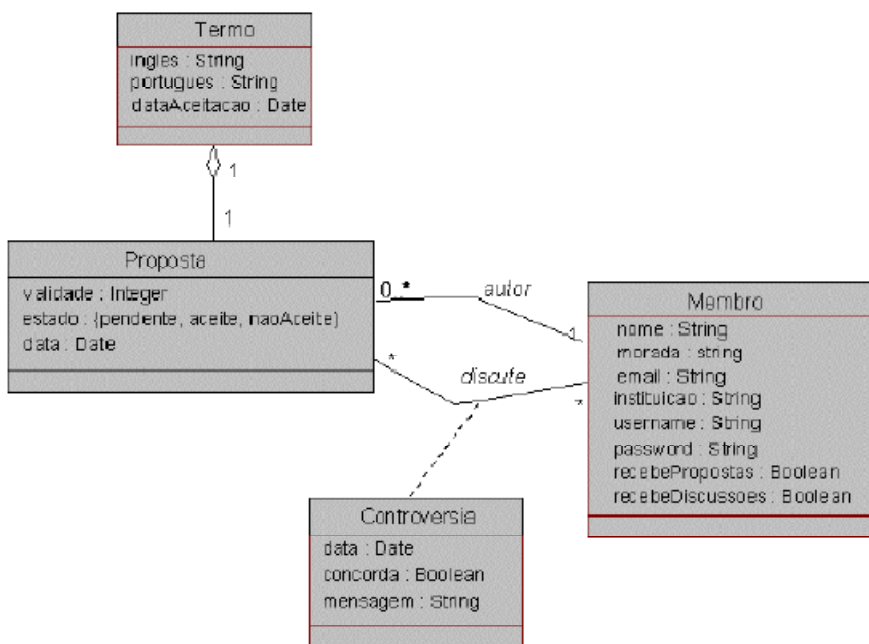


Figura 4.6: Exemplo de um diagrama de classes.

Os diagramas de classes (ver exemplo da Figura 4.6) descrevem a estrutura estática de um sistema, em particular as entidades existentes, as suas estruturas internas, e relações entre si. Um diagrama de objetos descreve um conjunto de instâncias compatíveis com determinado diagrama de classes. Permite ilustrar os detalhes de um sistema em determinado momento ao providenciarem cenários de possíveis configurações.

Diagramas de Modelação do Comportamento

Interação entre Objetos

Os padrões de interação entre objetos são representados por diagramas de interação, os quais podem assumir duas formas complementares, cada um focando um aspecto distinto: diagramas de sequência e diagramas de colaboração.

Diagramas de Sequência

Os diagramas de sequência ilustram interações entre objetos num determinado período de tempo (ver exemplo da Figura 4.7). Em particular, os objetos são representados pelas suas “linhas de vida” e interagem por troca de mensagens ao longo de um determinado período de tempo.

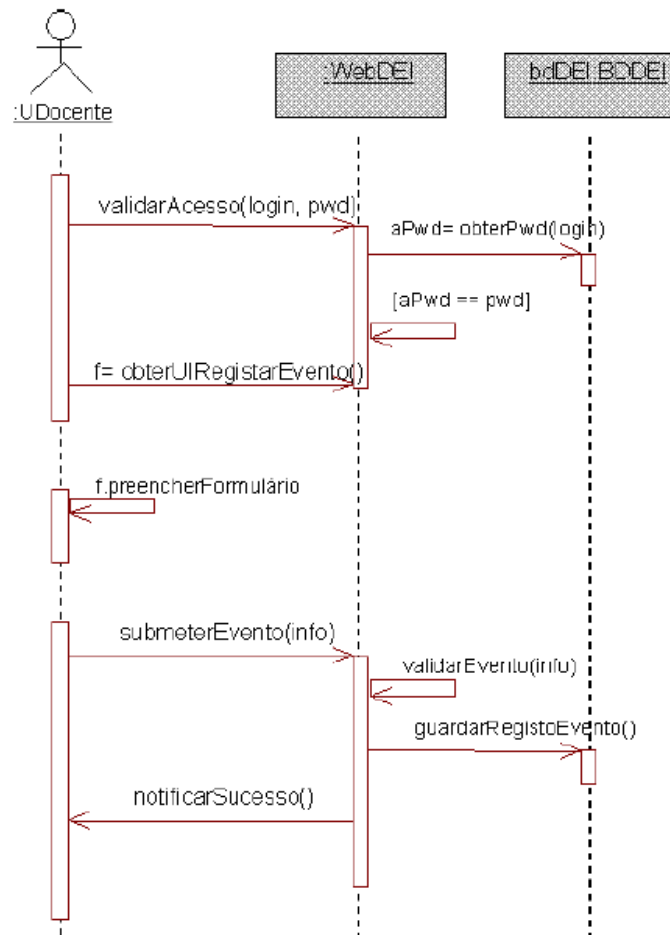


Figura 4.7: Exemplo de um diagrama de sequência.

Este tipo de diagrama baseia-se numa variedade de diagramas homólogos existentes em distintos métodos OO, segundo diferentes designações, como por exemplo interaction diagrams, message sequence charts, message trace, event trace, etc.

Diagramas de Colaboração

Os diagramas de colaboração ilustram interações entre objetos com ênfase para a representação das ligações entre objetos. Como os diagramas de colaboração não mostram o tempo como um elemento explícito (tal como acontece nos diagramas de sequência), a sequência de mensagens e de actividades concorrentes é determinada usando-se números sequenciais, com diferentes níveis hierárquicos. Colaborações são entidades de modelação principais e constituem geralmente a base para a representação visual de padrões de desenho (design patterns)

Este tipo de diagrama foi adaptado, entre outros, a partir do diagrama de objetos de Booch [Booch94], e do diagrama de interação de objetos de Fusion [Malan96].

Diagramas de Transição de Estado

Os diagramas de transição de estado descrevem as sequências de estados que um objeto ou uma interação pode passar ao longo da sua existência em resposta a estímulos recebidos, conjuntamente com as suas respostas e ações.

Diagramas de Atividades

Os diagramas de atividades (ver exemplo da Figura 4.8) são um caso particular dos diagramas de transição de estado, no qual a maioria dos estados são substituídos pelos conceitos correspondentes a ações e/ou atividades, e no qual as transições são desencadeadas devido à conclusão de ações nos estados originais.

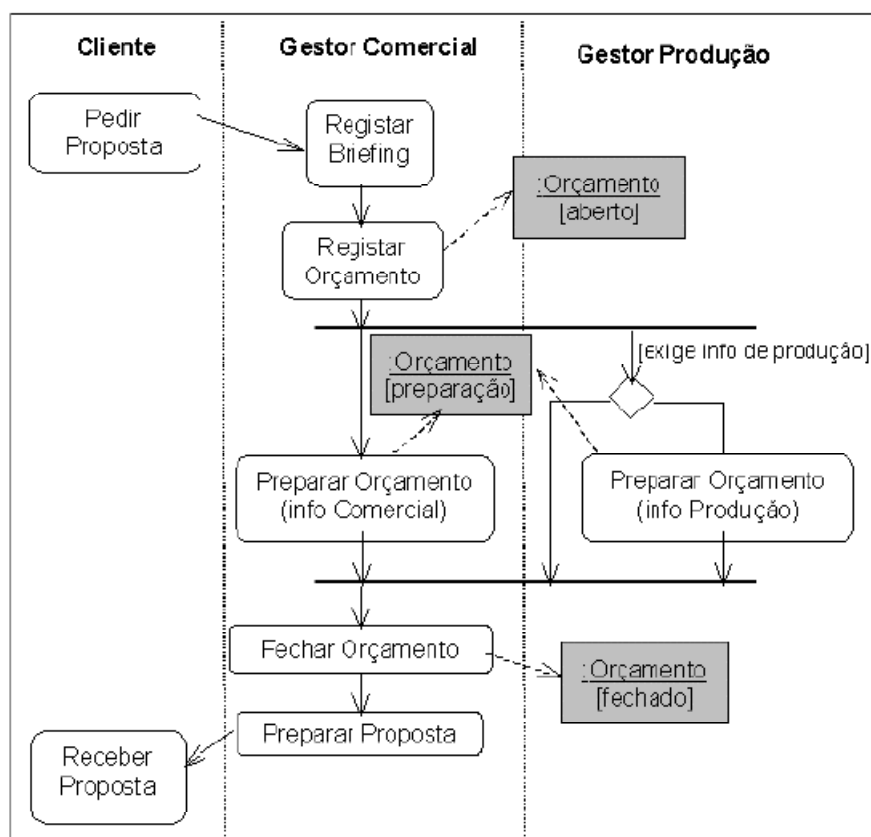


Figura 4.8: Exemplo de um diagrama de atividades.

O objetivo deste diagrama é representar os fluxos conduzidos por processamento interno, em contraste com eventos externos representados tipicamente nos diagramas de estado.

Este tipo de diagramas pode ser encarados como um caso particular de diagramas de estados e inspirados nos diagramas de workflow, fluxogramas ou naqueles baseados em SDL (Specification and Description Language).

Diagramas de Arquitetura

Os diagramas de arquitetura descrevem aspectos da fase de implementação de um sistema de software, por exemplo, relativamente à estrutura e dependências de módulos de código fonte e de módulos executáveis. Estes diagramas apresentam-se sob duas formas: diagramas de componentes e diagramas de instalação.

Diagramas de componentes

Os diagramas de componentes descrevem as dependências entre componentes de software, incluindo componentes de código fonte, código binário e executáveis. Os diagramas de componentes são representados na forma de tipos e não na forma de instâncias. Para descrever-se instâncias de componentes, usam-se os diagramas de instalação.

Diagramas de Instalação

Os diagramas de instalação (deployment diagrams) descrevem a configuração de elementos de suporte de processamento, e de componentes de software, processos e objetos existentes nesses elementos (ver exemplo da Figura 4.9).

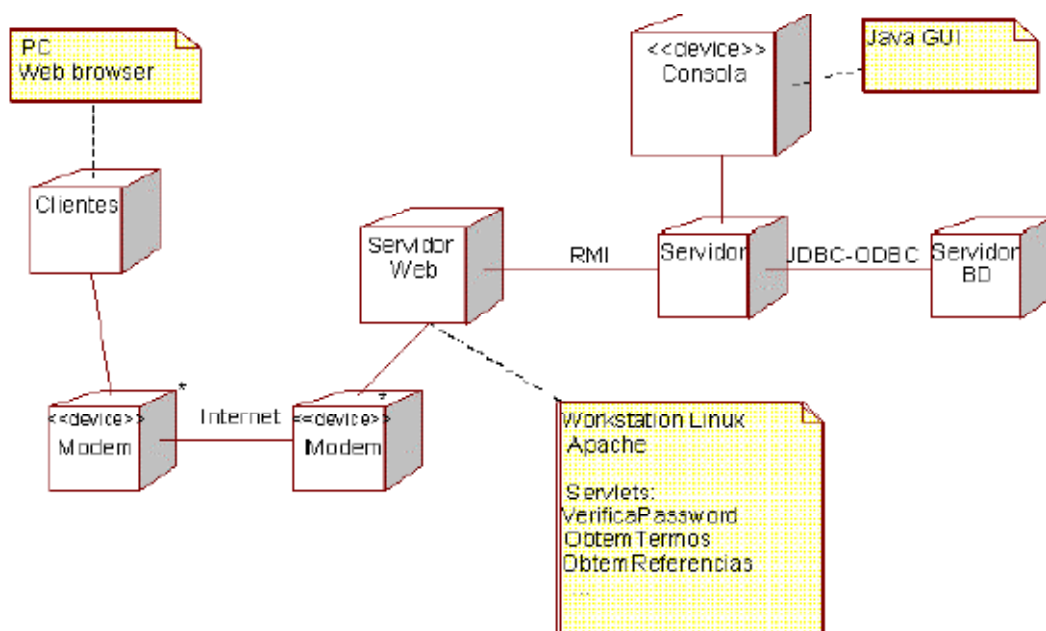


Figura 4.9: Exemplo de um diagrama de instalação.

Mecanismos Comuns

O UML contém um conjunto de mecanismos comuns que são aplicados de modo consistente ao longo dos seus diferentes diagramas. Descreve-se de seguida dois dos mecanismos comuns mais usados: anotações e mecanismos de extensão.

Notas (Anotações)

Notas ou anotações são os adornos mais importantes que existem autonomamente (i.e., sem estarem diretamente associados a outros elementos). Ver-se-á mais à frente a utilização de outros adornos (por exemplo, adornos para qualificar elementos participantes em relações).

Uma **nota** ilustra um comentário e não tem qualquer impacto semântico, já que o seu conteúdo não altera o significado do modelo no qual ela se encontra (ver Figuras 4.9 e 4.10). Por isso é normal a utilização de notas para se descrever informalmente: requisitos, restrições, observações, revisões ou explicações.

Em geral devem ser tomadas em consideração as seguintes observações na utilização de notas:

§ **Localização:** Devem ser colocadas graficamente perto dos elementos que descrevem.

§ **Visibilidade:** Podem-se esconder ou tornar visíveis (isto dependendo das ferramentas de suporte).

§ **Extensão:** Devem-se usar referências (e.g., nomes de documentos ou URL) caso se pretenda um comentário mais extenso.

§ **Evolução:** Há medida que o modelo evolui as notas mais antigas (cuja relevância e utilidade deixem de ter sentido) devem ser eliminadas dos modelos.

Mecanismos de Extensão

O UML providencia os seguintes mecanismos que permitem estender a sua linguagem de forma consistente: estereótipos, marcas e restrições.

Interessa desde já referir alguns aspectos na aplicação destes mecanismos, designadamente deve-se:

§ Introduzir um número reduzido destes elementos.

§ Escolher nomes curtos e com significado para estereótipos e marcas.

§ Sempre que a precisão puder ser relaxada, usar texto livre para especificação das restrições. Caso contrário, usar a linguagem OCL. O OCL (Object Constraint Language) [Warmer99] é uma linguagem para especificação formal de restrições e é parte integrante do UML.

§ A definição destes mecanismos de extensão do UML, em particular a introdução de estereótipos, deve ser realizada por um número restrito de especialistas em UML e deve ser devidamente documentado.

Estereótipos

Um **estereótipo** é um metatipo, isto é, um tipo que descreve tipos. Permite definir novos tipos de elementos sem se alterar o metamodelo do UML, e por conseguinte permite estender o UML. O nome de um estereótipo é representado entre os caracteres '«' e '»'. Exemplos:

§ Na modelação de processos de negócio: «trabalhador», «documento», «política».

§ Na modelação de aplicações específicas: classes de «interface», «controlo», e «entidade».

§ Na modelação de aplicações Web, usando o Web Application Extension [Conallen00]: classes de «Server Page», «Client Page», «Form», «Select Element».



Figura 4.11: Exemplo de estereótipos.

A Figura 4.11 ilustra três formas complementares de apresentação de estereótipos: com ícone (e.g., SensorTemperatura); com nome (e.g., ModelElement); e com nome e ícone (e.g., Overflow). A definição de um estereótipo tem de ter em conta os seguintes aspectos:

§ Propriedades (pode providenciar o seu próprio conjunto de marcas)

§ Semântica (pode providenciar as suas próprias restrições)

§ Notação (pode providenciar o seu próprio ícone)

Restrições

As **restrições** (constraints) permitem adicionar ou alterar a semântica assumida por omissão de qualquer elemento UML. Uma restrição consiste na especificação de uma condição delimitada pelos caracteres '{' e '}'. A condição pode ser especificada numa linguagem formal (e.g., OCL) ou informal (e.g., texto em português).

Uma restrição permite especificar condições que têm de ser validadas para que o modelo esteja “bem definido”.

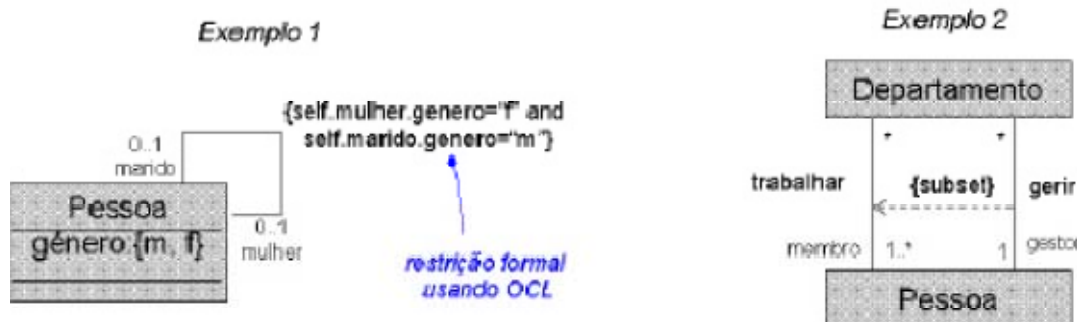


Figura 4.13: Exemplos de utilização de restrições.

Organização dos Artefactos – Pacotes

Um **pacote** (package) é em UML um elemento meramente organizacional. Permite agregar diferentes elementos de um sistema em grupos de forma que semântica ou estruturalmente faça sentido. Um pacote pode conter outros elementos, incluindo: classes, interfaces, componentes, nós, casos de utilização, e mesmo outros pacotes. Por outro lado, um elemento encontra-se definido em apenas um único pacote.

Representação Gráfica

Em UML os pacotes são representados graficamente por uma pasta (tabbed folder) com duas zonas complementares: um pequeno retângulo (designado por tabulador ou tab), normalmente com o nome do pacote; e um rectângulo maior onde normalmente se especificam os elementos constituintes desse pacote ou, pelo menos, os seus elementos públicos.

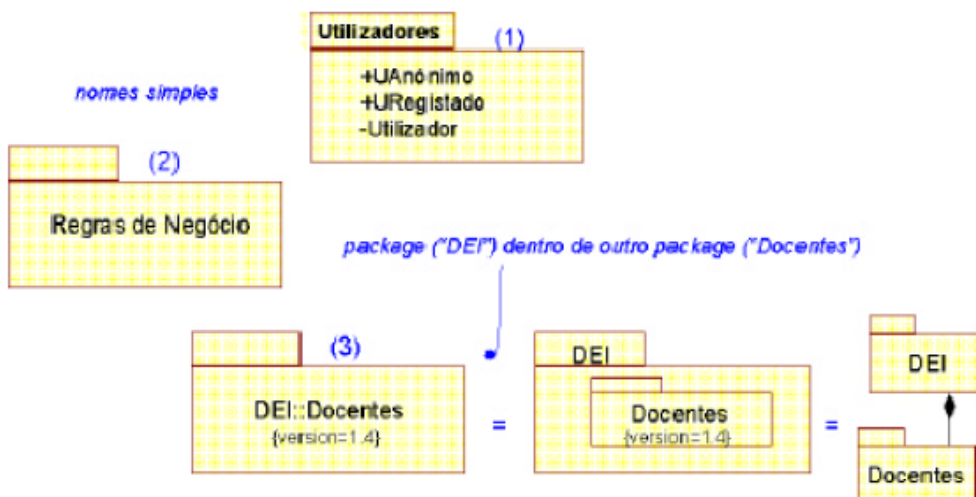


Figura 4.14: Exemplos de pacotes.

Importação e Exportação

Um pacote faz a **exportação**, por definição, de todos os seus elementos públicos. Mas tal fato não implica que um elemento definido noutro pacote possa aceder/referenciar um elemento exportado num outro pacote. Para que tal pudesse ocorrer deveria existir explicitamente uma relação de importação entre esses dois pacotes.

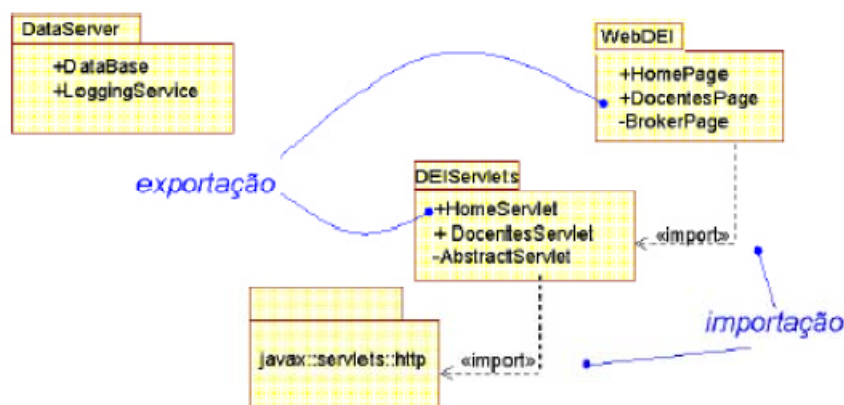


Figura 4.15: Relações de importação/exportação entre pacotes.

Generalização

A relação de generalização entre pacotes é semelhante à homóloga existente entre classes, e é usada para a especificação de famílias de pacotes, típicas em sistemas complexos ou flexíveis (e.g., significativamente parametrizáveis, multi-plataforma, multi-linguagem).

Tipos de Pacotes

Na generalidade dos exercícios académicos a dimensão e/ou a simplicidade do problema faz com que um pacote seja “simplesmente” um pacote. Contudo, em situações reais de modelação de sistemas de software de média/grande dimensão, realizada por equipas de vários indivíduos, torna-se necessário tipificar os próprios pacotes.

A especificação atual do UML propõe cinco estereótipos standard aplicados a pacotes:

§ «system»: pacote que representa o sistema inteiro; tipicamente este pacote agrega pacotes dos restantes tipos (subsistema, fachada, etc.).

§ «subsystem»: pacote que representa uma parte constituinte do sistema inteiro.

§ «facade»: pacote com elementos (tipicamente classes e interfaces) que constituem a fachada (ou a interface de programação) providenciada conjunta e coerentemente por outros pacotes. A fachada permite esconder os detalhes de arquitetura e de implementação de vários elementos eventualmente organizados em distintos pacotes. Os elementos definidos numa fachada apenas referem outros elementos definidos noutros pacotes.

§ «framework»: um framework é uma arquitetura de classes e interfaces com inúmeros pontos de variabilidade ou de extensão e com estruturas de objetos padronizadas, conhecidas por padrões de desenho. O desenvolvimento de sistemas baseados em frameworks e em componentes de software é um tópico emergente extremamente atual.

§ «stub»: um adaptador (stub) é usado quando se pretende partir um sistema em diferentes pacotes por motivos de divisão de trabalho, quer em termos físicos/espaciais, quer em termos temporais. Os adaptadores permitem que duas equipas consigam trabalhar paralelamente em diferentes locais, mas mantendo algum nível de interdependência.

Exercícios

1. Das seguintes afirmações assinale as que são verdadeiras:
 - a. - O UML é uma metodologia orientado por objetos.
 - b. - O UML é independente das ferramentas de modelação.
 - c. - O UML é um standard OMG.
 - d. - O UML é uma linguagem de programação robusta.
2. Quais são os dois aspectos importantes que se ganham com a adoção do UML.
3. Quais são os principais tipos de relações identificados na estrutura de conceitos do UML?
4. Com base em que princípio de modelação o UML propõe vários tipos de diagramas (com base nos quais se podem produzir visões complementares de um sistema)?
5. O que é uma marca com valor? Para que serve? Dê um exemplo de aplicação.
6. O que é um pacote UML? Enumere as três principais motivações/benefícios para a utilização de pacotes.