FUNDAMENTAL® de UML

2ª EDIÇÃO ACTUALIZADA E AUMENTADA

Mauro Nunes • Henrique O'Neill

Abordagem simples e prática
Tudo o que precisa para compreender
e construir os principais diagramas

FCA

Curso Completo

Os livros desta colecção e das linguagens de programação abordam, de uma maneira simples e objectiva, **praticamente** todas as capacidades dos programas tratados. As inúmeras ilustrações e exemplos com instruções, passo a passo, **levam** no a dominar com rapidez as matérias apresentadas.

— **AUTOCAD 2002** (José Garcia)
— **CRYSTAL REPORTS** (Sérgio Vasconcelos Oliveira)
— **HARDWARE - Montagem, Actualizada, Detecção e Reparação de Avarias em PC'S e Periféricos - 3ª Edição Actualizada** (José Gouveia/Alberto Magalhães)
— **HTML 4 & XHTML** (Pedro Coelho)
— **PHOTOSHOP** 7 (Fernando Tavares Ferreira)
— **PROGRAMAÇÃO EM JAVA** 2 (Pedro Coelho)
— **UTILIZAÇÃO DO LOTUS** NOTES **t** (Jorge Neves)
— **WINDOWS SERVER 2003** (Samuel Santos/António Rosa)

Nesta colecção, pretendemos oferecer uma panorâmica sobre vários produtos de Software, pertencendo a maioria a categoria de **"software livre"**, em inglês **"open source"**, e alguns à de **"software** grátis".

— **COMO INSTALAR UM SERVIDOR COMPLETO DE E-MAIL** (Mário Gamito/Ricardo Oliveira)
— **PYTHON** (Pedro Morais/José Nuno Pires l
— **PROGRAMAÇÃO** EM **PERL** (Levi Lúcio/Vasco Amaral)
— **PROGRAMAÇÃO** COM **PHP** 4 (Carlos Serrão/Joaquim Marques)

Uma colecção especialmente dirigida aos estudantes de Engenharia Informática e Informática de Gestão, assim como aos profissionais destas áreas que pretendam actualizar os seus conhecimentos. **Inclui** obras que apresentam, de uma forma clara e ***pragmática***, os conceitos fundamentais e o estado da arte

— PROGRAMAÇÃO COM CLASSES EM **C++ - 2ª** Edição **Actualizada** (Pedro Guerreiro)

Uma colecção dedicada aos profissionais de Sistemas de Informação, assim como a outros profissionais de informática que pretendem desenvolver os seus conhecimentos, e aos estudantes das licenciaturas e mestrados.

— GESTÃO DO **CONHECIMENTO** - O NOVO PARADIGMA DAS ORGANIZAÇÕES (Cândido Fialho/António Serrano)

A nova cojecção da FCA dedicada aos profissionais de projecto e desenvolvimento de *software*, e aos **estudantes** das licenciaturas e mestrados.

— **GESTÃO DE PROJECTOS DE SOFTWARE** (António Miguel)

Uma colecção que trata o impacto das tecnologias de **informação** na sociedade e **suas** influências a nível das pessoas, das empresas e das instituições.

— **INFORMATIZAÇÃO DO PODER LOCAL** (Francisco Melo Pereira)

Para Profissionais

Uma colecção dedicada aos profissionais da Informática, **abordando** matérias de uma forma acessível, mas profunda. Sã» também úteis para os que querem tirar uma certificação.

— **HARDWARE PARA PROFISSIONAIS - 2ª Edição Actualizada e Aumentada** (António Sampaio)
— **TCP/IP EM REDES MICROSOFT - 5ª Edição Actualizada** (Paulo Loureiro)

OUTRAS PUBLICAÇÕES

— **DICIONÁRIO DE INFORMÁTICA E NOVAS TECNOLOGIAS** (José A. de Matos)
— **EXERCÍCIOS RESOLVIDOS COM O EXCEL XP E 2000** (Adelaide Carvalho)
— **PALMTOPS** (Oscar Martins/Fernando Franco)
— **SOLUÇÕES INFORMÁTICAS NA GESTÃO DE RECURSOS HUMANOS - 2ª Edição Actualizada** (Sérgio Sousa/Maria José Sousa)
— **VISUAL BASIC.NET - PROGRAMAÇÃO PRÁTICA** (Nuno Nina)

CASO NÃO ENCONTRE QUALQUER UMA DESTAS OBRAS, NO SEU FORNECEDOR HABITUAL, UTILIZE A NOSSA NOTA DE ENCOMENDA (VER FIM DO LI OU ATRAVÉS DO e-mail: [illegible]@fca.pt

FUNDAMENTAL de UML®

***1*- edição actualizada e aumentada**

Mauro Nunes / Henrique O'Neill

FCA – EDITORA DE INFORMÁTICA

RUA D. ESTEFÂNIA, 183-1.º ESQ. — 1000-154 LISBOA
TEL. 21 353 27 35 (S. Editorial) FAX 21 352 26 84
TEL. 21 351 14 48 (Serviço Clientes)
E-mail: fca@fca.pt
Visite a nossa página em **http://www.fca.pt**
s/te seguro (certificado pela Thawte)

Prefácio

OBJECTIVO E AUDIÊNCIA

O objectivo do Fundamental de UML é efectuar uma abordagem simples e prática à linguagem de modelação visual UML.

Este livro é direccionado a todos os que procuram um manual prático e simples sobre as principais técnicas de modelação na UML. Ajuda a compreender e a construir os diagramas mais importantes na especificação e análise de Sistemas de Informação.

Nesta 2ª edição melhora-se a componente didáctica do livro, apresentando em cada capítulo um conjunto de perguntas de revisão e de exercícios resolvidos. O capítulo 10 foi aumentado através da apresentação de um novo caso de estudo, onde se propõe a especificação e desenvolvimento de um sistema de informação para uma Universidade. O livro foi igualmente objecto de uma revisão, tendo sido a notação gráfica presente nos diagramas compatibilizada com a versão 1.5 da UML, de Março de 2003. Com esta nova edição, pretendem os autores reforçar a capacidade do Fundamental de UML como um elemento de formação no domínio da modelação visual de sistemas de informação.

ESTRUTURA DA OBRA

O Capítulo 1 fornece uma introdução à necessidade de efectuar o desenvolvimento de Sistemas de Informação. A actividade de levantamento de requisitos é abordada no Capítulo 2-Diagrama de *use cases.* A estrutura de informação em termos de objectos, classes e suas relações é introduzida no Capítulo 3-Diagrama de Classes.

© FCA - EDITORA DE INFORMÁTICA

O Capítulo 4-Diagrama de Actividades explica os principais conceitos necessários para a modelação de actividades. Em seguida, no Capítulo 5-Diagramas de Interacção é fornecida uma visão sobre a modelação das interacções entre objectos. O Capítulo 6-Diagrama de Estados completa os aspectos dinâmicos de modelação do sistema, abordando a representação dos diversos estados dos objectos.

No Capítulo 7-Desenho do Sistema são apresentados alguns princípios gerais para a definição da arquitectura da aplicação. Os diagramas de componentes e de instalação são apresentados no Capítulo 8-Diagramas Físicos. No Capítulo 9-Processo de Modelação é abordado o método de desenvolvimento e apresentadas ferramentas informáticas de apoio à modelação.

Por fim, no Capítulo 10-Casos de Estudo são apresentados modelos de Sistemas de Informação, que demonstram de forma integrada as diversas técnicas disponibilizadas pela UML.

SIMBOLOGIA UTILIZADA

Ao longo dos diversos capítulos são apresentadas sugestões práticas para facilitar a utilização da linguagem UML. As sugestões são realçadas dentro da seguinte moldura:

Sugestão...

Quando é definido um conceito, este é realçado a **negrito** para facilitar a sua identificação.

Mauro Nunes (mauro.nunes@isctept)
Henrique O'Neill (henrique.oneill@iscte.pt)

índice

1. Introdução **1**
1.1 Introdução 1
1.2 Modelação Visual 2
1.3 Definição da *Unified Modelling Language* (UML) 3
1.4 História 4
1.4.1 A evolução das técnicas e metodologias de modelação 4
1.5 Notação 6
1.5.1 Diagramas 6
1.5.2 Abstracções de modelação 7
1.6 Desenvolvimento de Sistemas de Informação 9
1.6.1 Método iterativo e incremental 9
1.6.2 Arquitectura 11
1.7 Descrição do exemplo 12
2. Diagrama de *Use Cases* **13**
2.1 Conceito e Aplicação 13
2.1.1 Âmbito 16
2.1.2 Actores 16
2.1.3 *Use cases* de Negócio e de Sistema 17
2.1.4 Comunicação entre actores e *use cases* 18
2.1.5 Tempo 19
2.1.6 Cenário principal e Cenários Secundários 20
2.1.7 Relações de «include», «extend» e generalização 24
2.2 Exercícios 29
3. Diagrama de Classes **35**
3.1 Conceito e Aplicação 35
3.1.1 O que é um Objecto 38
3.1.2 O que é uma Classe 39
3.1.3 Tipos de dados básicos 42
3.1.4 Associações 43
3.1.5 Multiplicidade 44
3.1.6 Identificação de classes 45
3.1.7 Identificação de atributos 47
3.1.8 Identificação de associações e operações 47
3.1.9 Restrições 48
3.2 Tópicos Avançados 49
3.2.1 Classes Associativas 49
3.2.2 Generalização e Herança 50

© FCA - EDITORA DE INFORMÁTICA

3.2.3 Agregação e Composição 51
3.2.4 Diagrama de classes PhonePizza revisto 52
3.3 Exercícios 53
4. Diagrama de Actividades 57
4.1 Conceito e Aplicação 57
4.1.1 Linhas verticais de responsabilização 61
4.1.2 Actividades 61
4.1.3 Transição entre actividades 62
4.1.4 Comportamento condicional 63
4.2 Tópicos avançados 64
4.2.1 Agrupamento e decomposição de actividades 65
4.2.2 Processamento paralelo 66
4.2.3 Fluxo de objectos 67
4.2.4 Diagrama de actividades revisto 68
4.3 Exercícios 69
5. Diagramas de Interacção 75
5.1 Conceito e Aplicação 75
5.2 Diagrama de Sequência 76
5.2.1 Mensagens 77
5.2.2 Linha temporal e controlo 79
5.2.3 Processamento em paralelo 82
5.2.4 Interface com o utilizador 83
5.3 Diagrama de Colaboração 83
5.3.1 Ordenação numérica 84
5.3.2 Repetições 85
5.3.3 Estereótipos 85
5.3.4 Mensagens condicionais 86
5.3.5 Sincronização 86
5.3.6 Objectos e ligações 87
5.4 Construção de diagramas de interacção 88
5.5 Exercícios 89
6. Diagrama de Estados 95
6.1. Conceito e Aplicação 95
6.1.1 Estado 97
6.1.2 Transição entre estados 98
6.2. Tópicos avançados 99
6.2.1 Agrupamento de Estados 99
6.2.2 Concorrência entre subestados 100
6.3 Exercícios 101

© FCA - EDITORA DE INFORMÁTICA

7. Desenho do Sistema 105
7.1. Conceito e Aplicação 105
7.2 Diagrama de classes - perspectiva de Desenho 106
7.2.1 Estereótipos 106
7.2.2 Relação de Dependência 107
7.2.3 Relação de Realização 108
7.2.4 Interfaces 108
7.2.5 Diagrama de Classes com níveis 110
7.3 Pacotes 113
7.3.1 Relações entre pacotes 114
7.3.2 Representação do sistema em 3 níveis 116
7.4 Exercícios 116
8. Diagramas Físicos 119
8.1. Conceito e Aplicação 119
8.2. Diagrama de Componentes 121
8.2.1. Componentes 123
8.2.2. Interfaces 125
8.3. Diagrama de Instalação 126
8.3.1. Nós 127
8.3.2. Comunicação 129
8.3.3 Nós e componentes 129
8.4 Exercícios 131
9. Processo de Modelação 135
9.1 Conceito e Aplicação 135
9.1.1 Orientações para o desenvolvimento 136
9.2 Processo de Modelação Unificado 137
9.2.1 Actividades 137
9.2.2 Fases 138
9.2.3 Arquitectura de modelação 140
9.2.4 Resultado da Modelação 142
9.3 Aproximação prática ao desenvolvimento 143
9.4 Ferramentas de modelação com UML 145
9.4.1 Rose 2000 148
9.4.2 Visio 2000 150
9.4 MDA – Model Driven Architecture 151
10. Casos de Estudos 155
10.1 PhonePizza 155
10.1.1 Modelo Negócio 156
10.1.2 Modelo de Domínio 159
10.1.3 Modelo de *Use Cases* 162

10.1.4 Modelo de Desenho 173
10.1.5 Modelo de Implementação 175
10.1.6 Modelo de Instalação 176
10.2 SIUniversitas® 177
10.2.1 Modelo de *Use Cases* 181
10.2.2 Modelo de Desenho 190
10.2.3 Modelo de Implementação 194
10.2.4 Modelo de Instalação 195
Anexo Regras de transposição 197
I.1 Conceitos e Aplicação 197
I.1.1 Conceitos básicos 197
I.2 Regras 198
I.3 Optimização do Modelo Relacional 203
Anexo Descrições do caso PhonePizza 207
II.1 Descrição de *use cases* 207
II.2 Descrição das classes 210
Glossário 213
Bibliografia 221
índice Remissivo 223

Introdução 1

1.1 Introdução

A introdução de tecnologias de informação continua a alterar profundamente o modo como as organizações evoluem e os negócios se processam. Um elemento intrínseco a qualquer organização é o seu sistema de informação, constituído por pessoas, dados, procedimentos e equipamentos. O desenvolvimento 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.

A informatização exige que sejamos capazes de descrever com rigor o modo como as nossas organizações funcionam, para que os sistemas de informação possam satisfazer plenamente as nossas necessidades. Este requisito é igualmente importante quer se venha a optar pela aquisição de uma aplicação informática existente no mercado ou por um desenvolvimento específico.

As aplicações informáticas modernas tendem a ser cada vez mais flexíveis, mas não estão preparadas para satisfazer todas as necessidades de informação dos seus potenciais utilizadores. Frequentemente temos de ser capazes de definir o que pretendemos ter de uma aplicação informática, de forma a avaliar se esta é capaz de responder a essas necessidades ou se requer adaptações.

Assim, torna-se necessário podermos recorrer a uma linguagem que faciite 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

© FCA - EDITORA DE INFORMÁTICA

satisfazem essas necessidades e informáticos que desenvolvam as funcionalidades pretendidas.

A utilização da UML - *Unified Modelling Language* abre perspectivas para responder ao desafio de desenvolvimento de novos sistemas de informação, cada vez mais complexos, robustos, fiáveis e ajustados às necessidades dos utilizadores.

1,2 MODELAÇÃO VISUAL

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. Tomemos por exemplo a construção de um novo edifício ou de uma nova máquina. Para tal, os arquitectos e engenheiros criam representações das suas ideias através de desenhos técnicos ou de maquetas, para serem compreendidas e validadas. Estes modelos possuem uma forte componente gráfica e utilizam um conjunto limitado de símbolos com um significado específico. Esta aproximação procura eliminar a ambiguidade e redundância geralmente associada a uma descrição escrita, tirando partido da imagem como elemento de comunicação.

Estes modelos tendem a ser tão mais elaborados quanto mais complexo for aquilo que se pretende desenvolver: um simples esboço será suficiente para orientar a construção de um pequeno anexo para arrumação de material de jardinagem; o projecto de um grande edifício exigirá um conjunto complexo de diagramas que permitam coordenar o trabalho de todos aqueles envolvidos na sua construção. Independentemente da complexidade do problema, a linguagem utilizada nestes diagramas deverá ser isenta de ambiguidade, permitir descrever as partes essenciais do problema a modelar e ser simples para ser entendida por todos.

O desenvolvimento de sistemas informáticos depara com desafios semelhantes aos que se encontram noutras áreas de criação técnica, como a arquitectura ou a engenharia. Estes desafios talvez sejam maiores, já que os sistemas informáticos se vão tornando mais complexos e permanecem mais tempo entre nós, exigindo uma permanente actualização de forma a responder às contínuas solicitações de melhoria colocadas pelos seus utilizadores e a tirar partido de novos desenvolvimentos tecnológicos.

1.3 DEFINIÇÃO DA *UNIFIED MODELLING LANGUAGE* (UML)

UML é a sigla de *Unified Modelling Language,* que pode ser traduzido por Linguagem de Modelação Unificada.

A UML é uma linguagem que utiliza uma notação padrão para especificar, construir, visualizar e documentar sistemas de informação orientados por objectos.

Pela abrangência e simplicidade dos conceitos utilizados, a UML facilita o desenvolvimento de um sistema de informação. Permite integrar os aspectos de natureza organizacional que constituem o negócio e os elementos de natureza tecnológica, que irão constituir o sistema informático, ajudando a dominar a complexidade das regras de negócio e definir os processos e fluxos informativos.

Pelo facto de utilizar um conjunto de símbolos padrão, a UML funciona como um meio de comunicação entre os diversos elementos envolvidos no processo, utilizadores, gestores e equipa de desenvolvimento. A linguagem pode ser utilizada para documentar o sistema ao longo de todo o ciclo de desenvolvimento, começando com a tarefa inicial de análise dos processos de negócio da organização e prolongando-se até à tarefa de manutenção evolutiva do sistema informático.

A UML permite ainda responder a requisitos técnicos relevantes para uma evolução dos sistemas informáticos, como a arquitectura

© FCA - EDITORA DE INFORMÁTICA

da aplicação informática (*software*), a capacidade de reutilização dos componentes desenvolvidos e a independência em relação ao equipamento.

A abrangência da UML justifica assim a utilização do termo *unificada.*

1.4 História

1.4.1 A evolução das técnicas e metodologias de modelação

Na curta história do desenvolvimento de sistemas de informação, e do *software* em geral, poderemos identificar duas grandes fases: uma fase inicial em que se adoptou uma aproximação estruturada ou funcional e uma outra fase, mais recente, em que a aproximação se baseia no paradigma dos objectos (Rumbaugh et al, 1991). A UML é o resultado de um longo processo de maturação no domínio da modelação de sistemas de informação segundo o paradigma dos objectos.

São muitos os autores que contribuíram para o conhecimento que existe actualmente sobre o modo como os sistemas de informação devem ser modelados e construídos (fig. 1.1). Todos estes trabalhos têm muito em comum, mas cada um deles propõe nomenclaturas e abordagens específicas para a modelação de sistemas de informação.

A diversidade de propostas tornava difícil a comunicação e reduzia a utilidade prática desta área de conhecimento, em grande parte devido à diferença de nomenclaturas utilizadas nos modelos semânticos, notação sintáctica e diagramas. Atentos a este problema, três dos mais relevantes autores neste domínio - Booch, Jacobson e Rumbaugh - começaram a trabalhar em conjunto para apresentar uma proposta unificadora dos seus trabalhos individuais.

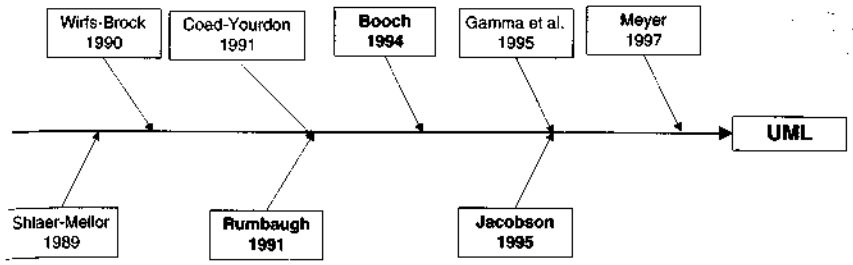


FIGURA 1.1 - CONTRIBUIÇÕES PARA Â UML

Este trabalho veio a dar origem à UML (Linguagem de Modelação Unificada), apresentada publicamente pela primeira vez em Outubro de 1995. Em Novembro de 1997, a UML foi adoptada pelo OMG (Object Management Group) como uma linguagem de modelação padronizada e de livre utilização. Actualmente a UML está na versão 1.5 (OMG, 2003).

A UML recorre a uma notação padronizada, constituída por um conjunto limitado de elementos de modelação, que podem ser tipificados em diagramas, abstracções e relacionamentos.

Um modelo em UML é constituído por um conjunto de diagramas que representam aspectos complementares de um sistema de informação. Em cada um destes diagramas são utilizados símbolos que representam os elementos que estão a ser modelados (abstracções) e linhas que relacionam esses elementos. Os símbolos e as linhas têm significado específico e possuem formas distintas, constituindo uma forma de notação.

© FCA - EDITORA DE INFORMÁTICA

1.5 NOTAÇÃO

1.5.1 Diagramas

A UML disponibiliza o seguinte conjunto de diagramas:

Diagrama de Use Case - serve para identificar as fronteiras do sistema e descrever os serviços *(use cases)* que devem ser disponibilizados a cada um dos diversos utilizadores (actores). Capítulo 2.

Diagrama de Classes - através do qual descrevemos a estrutura de informação (classes e suas relações) que é utilizada no sistema. Capítulo 3.

Diagrama de Objectos - que pode ser utilizado para ilustrar um diagrama de classes com um exemplo concreto.

Diagrama de Sequência e Diagrama de Colaboração - servem para ilustrar como os objectos do sistema interagem para fornecer a funcionalidade do *use case*. Estes diagramas designam-se genericamente por Diagramas de Interacção. Capítulo 5.

Diagrama de Actividade - pode ser utilizado para descrever cada um dos *use cases*, realçando o encadeamento de actividades realizadas por cada um dos objectos do sistema, numa óptica de fluxo de trabalho (*work-flow*). Capítulo 4.

Diagrama de Estados - que é utilizado para modelar o comportamento dos objectos isto é, descrever alterações nos valores de atributos dos objectos em resultado da ocorrência de certos eventos. Capítulo 6.

Diagrama de Componentes - utilizado para descrever a arquitectura da aplicação informática em termos de componentes de *software*. Capítulo 8.

Diagrama de Instalação (*deployment*) - permite descrever a arquitectura de equipamento informático utilizado e distribuição dos componentes da aplicação pelos elementos da arquitectura.

1.5.2 Abstracções de modelação

As Abstracções podem ser tipificadas como estruturais, comportamentais, de agrupamento ou anotacionais. A figura 1.2 ilustra algumas destas abstracções.

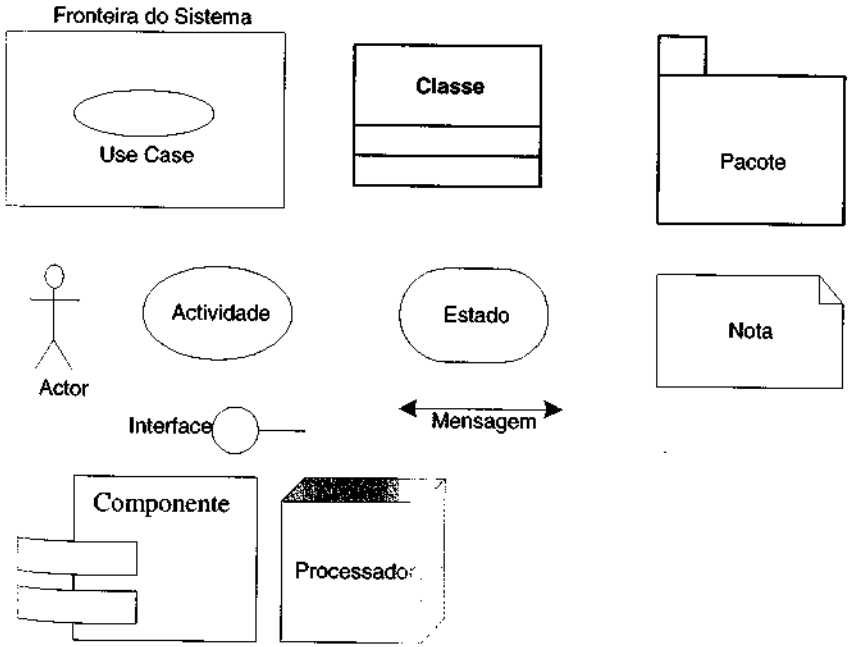


FIGURA 1.2 – ELEMENTOS ABSTRACTOS DE MODELAÇÃO DA UML

FCA - EDITORA DE INFORMÁTICA

Abstracções estruturais e comportamentais reflectem a orientação por objectos da UML, permitindo descrever a estrutura e o comportamento dos diversos elementos que constituem o sistema de informação. Abstracções de agrupamento são meramente conceptuais, podendo ser utilizadas para agrupar outros elementos estruturais, comportamentais ou mesmo de agrupamento. As notas são utilizadas para colocar comentários nos diagramas.

Os Relacionamentos são utilizados para realçar relações existentes entre os elementos abstractos de modelação (Fig. 1.3):

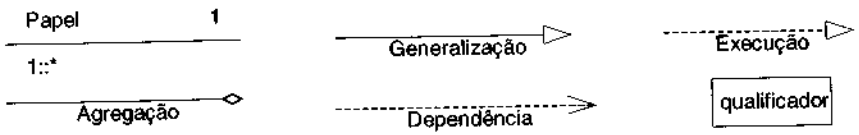


FIGURA 1.3 – RELACIONAMENTOS EM UML

Estereótipos, etiquetas e restrições são mecanismos comuns a UML que podem ser utilizados para reforçar a capacidade de expressão da linguagem (Fig. 1.4):

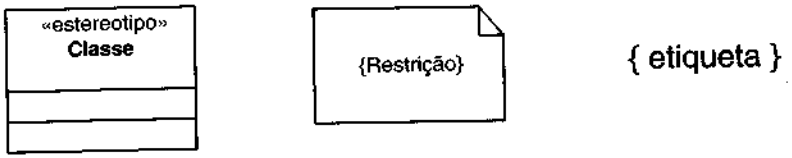


FIGURA 1.4 - MECANISMOS COMUNS DE MODELAÇÃO

O conceito de **estereótipo** permite estender a capacidade expressiva da linguagem, i.e., atribuir novos significados aos símbolos que são utilizados para a modelação de um determinado tipo de problema. As etiquetas servem para caracterizar elementos de modelação específicos. As restrições nos diferentes elementos podem ser também evidenciadas através de notas.

1.6 DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

O desenvolvimento de sistemas de informação segue um método que enquadra um conjunto de regras, etapas e actividades que devem ser satisfeitas.

A UML pode ser utilizada em diversas aproximações de desenvolvimento, desde o tradicional ciclo de vida sequencial em cascata, até às abordagens mais recentes utilizando protótipos. Todavia, as características da UML tornam-na particularmente adequada para um desenvolvimento iterativo e incremental.

Outro aspecto relevante no processo de desenvolvimento é a utilização de uma arquitectura de referência que permita enquadrar e potenciar as vantagens da linguagem UML.

1.6.1 Método iterativo e incremental

O desenvolvimento de sistemas de informação inclui a realização de tarefas de análise da organização, levantamento de requisitos de utilização, análise do sistema de informação, desenho, codificação, teste, instalação e manutenção. A nossa experiência indica-nos que para sistemas de informação complexos não é possível isolar e concluir completamente cada uma destas tarefas.

O desenvolvimento de um sistema de informação segundo uma abordagem iterativa e incremental pressupõe a existência de sucessivas iterações de refinamento, que se repetem ao longo do tempo, até se obter uma solução final. Os riscos técnicos são identificados e prioritizados inicialmente, sendo revistos em cada iteração.

O *UnifieaModelling Process* (UMP) é uma abordagem iterativa e incremental que sugere uma utilização efectiva da UML (Jacobson et al, 1999). Este processo propõe que um projecto seja estruturado numa dimensão temporal e numa dimensão processual.

© FCA - EDITORA DE INFORMÁTICA

Na dimensão temporal são identificadas 4 fases: **Início** (*Inception*) na qual se especifica a visão do projecto; **Elaboração** associada ao planeamento das actividades e recursos, bem como às características gerais da arquitectura; **Construção** durante a qual o sistema é construído de forma iterativa; **Transição,** na qual é disponibilizado o sistema aos utilizadores.

Na dimensão de processo são contempladas diversas actividades técnicas: análise e modelação do negócio, levantamento de requisitos, análise, desenho, programação, teste e instalação. Complementarmente devem ser asseguradas actividades de gestão de projecto e de preparação da instalação. Em cada uma destas actividades podem ser utilizados instrumentos específicos, designadamente os diagramas UML ou técnicas de gestão de projecto.

A figura l.5 mostra corno as fases e os componentes do processo se articulam.

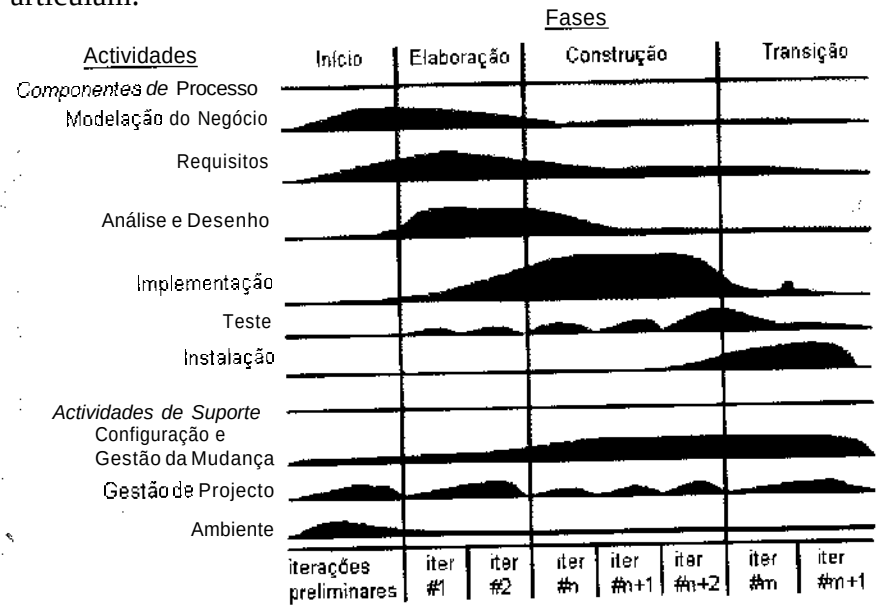


FIGURA 1.5 – PROCESSO DE DESENVOLVIMENTO
(ADAPTADA DE BOOCH ET AL, 1999)

Como se pode verificar pela figura, em cada uma das fases existe uma actividade que é predominante, mas não é exclusiva. Por exemplo, será possível fazer levantamento de requisitos e desenvolver um pequeno protótipo demonstrador na fase de Início. Todas as tarefas podem ser realizadas em cada fase de forma iterativa e incremental.

Esta aproximação tem claras vantagens, melhorando o detalhe e o âmbito de especificação. O facto de antecipadamente se limitar o número de iterações permite convergir para uma solução final de uma forma controlada, minimizando o risco de dilatar o prazo de conclusão do projecto.

1.6.2 Arquitectura

Uma arquitectura de referência constitui um elemento fundamental para a concepção, construção, evolução e gestão de um sistema de informação.

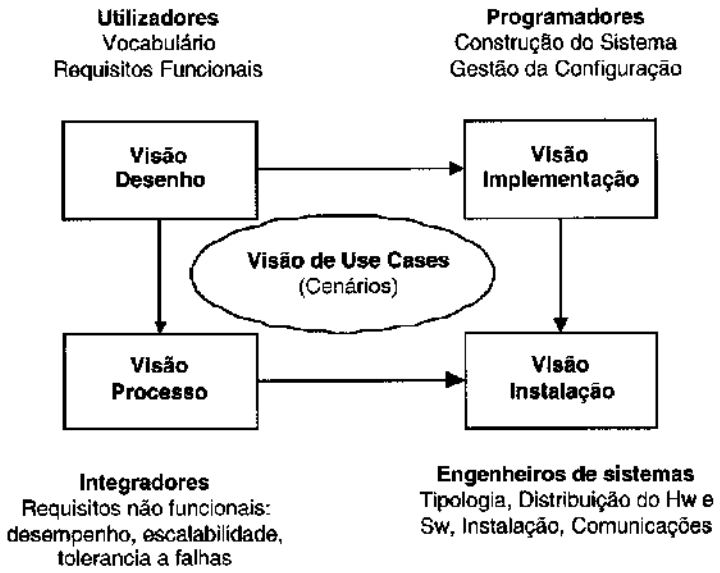


FIGURA 1.6 – ARQUITECTURA 4+1 DE MODELAÇÃO
(ADAPTADA DE BOOCH ET AL, 1999)

© FCA - EDITORA DE INFORMÁTICA

A figura 1.6 sugere uma arquitectura adaptada às características da UML, segundo a qual um sistema de informação deve ser desenvolvido segundo 4 visões, cada uma das quais aprofunda aspectos complementares do sistema, e que são centradas numa quinta, que permite validar e ilustrar as anteriores. A visão central é constituída por cenários que ilustram os requisitos mais importantes do sistema e que são descritos sob a forma de *use cases.* As restantes quatro visões são igualmente descritas através dos diagramas da UML. A arquitectura de modelação é abordada com mais detalhe no Capítulo 9.

1.7 DO EXEMPLO

Ao longo deste livro será utilizado um caso de estudo para facilitar a descrição da linguagem e exemplificar a sua utilização.

Este caso de estudo baseia-se no desenvolvimento de um sistema de informação para gerir uma empresa que possui uma rede integrada de lojas de produção e distribuição de refeições rápidas. O sistema de informação irá apoiar os serviços de atendimento, acompanhamento de clientes e encomendas. O sistema de informação permite que as encomendas possam ser recebidas por telefone, pela Internet ou nas lojas.

A PhonePizza constitui um exemplo bastante rico que permite descrever em profundidade as especificidades da linguagem. Simultaneamente, permite criar um modelo que pode ser facilmente adaptado a muitas organizações que pretendam desenvolver estratégias de negócio baseadas em práticas de comércio electrónico.

© FCA - EDITORA DE INFORMÁTICA

Diagrama de *Use Cases* 2

2,1 CONCEITO E APLICAÇÃO

Os *use* ***cases***, ou traduzindo à letra "casos de utilização", constituem a técnica em UML para representar o levantamento de requisitos de um sistema. Desde sempre que o correcto levantamento de requisitos no desenvolvimento de sistemas de informação tenta garantir que o sistema será útil para o utilizador final, estando de acordo com as suas necessidades.

O requisito num sistema é uma funcionalidade ou característica considerada relevante na óptica do utilizador. Normalmente, representa o comportamento esperado do sistema, que na prática consiste num serviço que deve ser disponibilizado a um utilizador (Booch, Rumbaugh e Jacobson, 1999).

Bennet, McRobb e Farmer (1999) identificam três categorias de requisitos:

Requisitos funcionais - descrevem o que um sistema faz ou é esperado que faça. Estes são os requisitos que inicialmente serão levantados, abrangendo a descrição de processamentos a efectuar pelo sistema, entradas (*inputs*) e saídas (*outputs*) de informação em papel ou no ecrã que derivam da interacção com pessoas e outros sistemas.

Requisitos não funcionais - relacionados com as características qualitativas do sistema, descrevendo a qualidade com que o sistema deverá fornecer os requisitos funcionais. Abrange medidas de desempenho como, por exemplo, tempos de resposta, volume de dados ou considerações de segurança.

* **Requisitos de facilidade de utilização (*usability*)** - garantem que existirá uma boa ligação entre o sistema desenvolvido, utilizadores do sistema e também as tarefas que desempenham apoiados pelo sistema.

Existem várias técnicas que podem ser utilizadas para efectuar o levantamento de requisitos. Estas técnicas abrangem a realização de reuniões participativas (*workshops*), entrevistas, questionários, observação directa, estudo e amostra de documentos e relatórios. Para efectuar um correcto levantamento, frequentemente os analistas combinam diversas destas técnicas.

Os diagramas de *use cases* são utilizados para a apresentação de requisitos e para assegurar que tanto o utilizador final como o perito numa determinada área ou o especialista informático, possuem um entendimento comum dos requisitos. O seu objectivo é mostrar o que um sistema deve efectuar e não como o vai fazer.

Estes diagramas utilizam as seguintes abstracções de modelação:

* Actores
* *Use cases*
« Relações (*Include*, *Extend* e Generalização)

O seguinte texto descreve um conjunto de requisitos para um novo sistema, que poderia ser obtido, por exemplo, em resultado de uma entrevista:

"Pretende-se desenvolver um sistema de informação de gestão para um grupo de pizzarias PhonePizza, que permita aos clientes efectuar encomendas na loja e através da Internet. Na loja, o cliente dirige-se ao empregado de balcão que introduzirá no sistema a encomenda pretendida.

Caso a encomenda seja efectuada através da Internet, o cliente terá que se identificar, através do seu nome de

utilizador e palavra-chave (controlo de acesso). O cliente pode então registar os artigos que pretende encomendar, podendo usufruir de um desconto no item, caso este esteja em promoção. O sistema deve ainda permitir que o Gestor da Pizzaria efectue as reservas de mesa, verificando se este tem autorização para o efectuar."

O resultado do levantamento de requisitos encontra-se representado pelo seguinte diagrama de *use cases*.

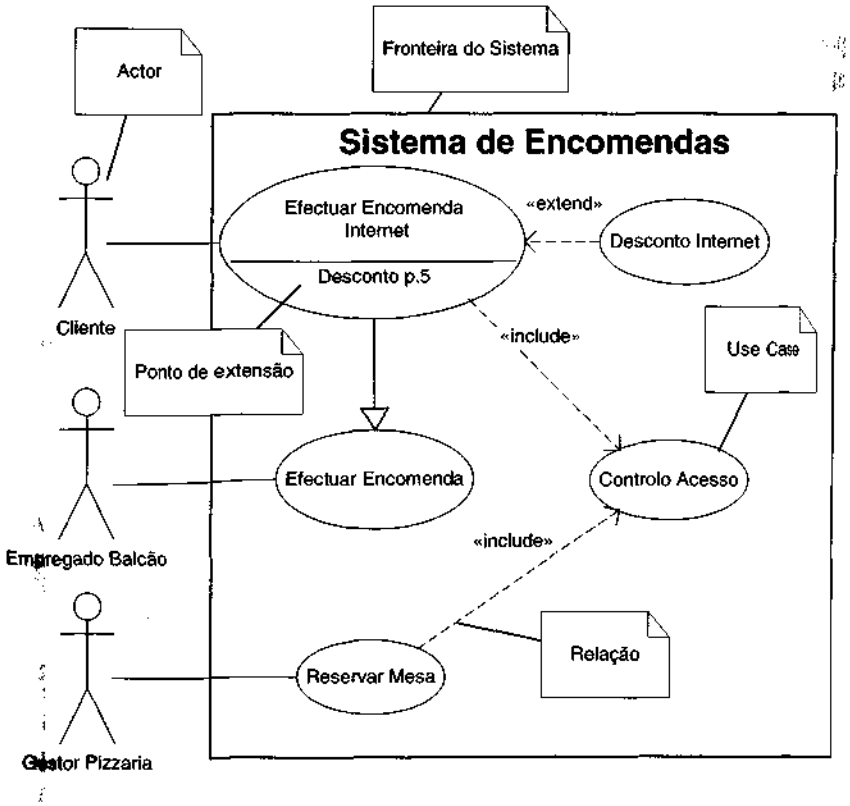


FIGURA 2.1 - DIAGRAMA DE *USE CASE*: SISTEMA DE ENCOMENDAS

© FCA - EDITORA DE INFORMÁTICA

Este diagrama é o resultado de um processo de construção. Este processo, bem como os principais conceitos, são explicados em seguida.

2.1.1 Âmbito

Para assegurar a compreensão do projecto a desenvolver por todas as partes envolvidas, é necessário definir à partida o seu âmbito e objectivo(s). Esta definição deve ser curta e concisa, evitando detalhes sobre os requisitos do sistema. Normalmente, não ultrapassa um parágrafo para sistemas pequenos ou algumas páginas para sistemas de maior dimensão. Para o exemplo apresentado inicialmente, teremos a seguinte descrição do seu âmbito.

Âmbito do sistema

> "Pretende-se desenvolver um sistema de informação de gestão para um grupo de pizzarias PhonePizza, que permita aos clientes efectuar encomendas na loja e através da Internet"

2.1.2 Actores

A primeira tarefa a desenvolver para construir um diagrama de *use cases é* a identificação dos actores do sistema. Um **actor** representa uma entidade externa que interage com o sistema.

Recorrendo ao nosso exemplo, são identificados os seguintes actores:

FIGURA 2.2 - EXEMPLOS DE ACTORES

Os actores Cliente e Gestor Pizzaria são as pessoas que irão interagir com o sistema. Apesar da representação humanizada, os actores podem não ser só pessoas, mas também outros sistemas físicos ou lógicos como, por exemplo, um módulo de Contabilidade.

Em geral, um actor pode invocar vários *use cases* e um *use case* pode ser invocado por vários actores. Por exemplo, um actor Funcionário pode também ser um actor Chefe de Loja.

Os actores devem ser caracterizados através de uma pequena descrição, de forma a assegurar uma correcta compreensão do significado do actor por todos os elementos da equipa envolvida na análise. Para os actores identificados anteriormente, teremos a seguinte descrição:

Descrição dos actores

> **Cliente** - Uma pessoa que encomenda produtos da PhonePizza pela Internet e nas pizzarias.
> **Empregado de Balcão** - Empregado que recebe as encomendas ao balcão da pizzaria.
> **Gestor Pizzaria** - Empregado que está encarregue de efectuar as reservas de mesa numa pizzaria.

2.1.3 *Use cases* de Negócio e de Sistema

Os *use cases* podem ser definidos numa perspectiva de Negócio ou de Sistema. Na primeira perspectiva, procura-se identificar a forma como se responde a um cliente ou evento em termos de processo de negócio. Na perspectiva do Sistema, procura-se caracterizar as funcionalidades que a aplicação a desenvolver (*software*) deve disponibilizar ao utilizador.

A principal razão para esta distinção (por vezes, pouco nítida em contextos altamente informatizados, como o caso PhonePizza que

© FCA - EDITORA DE INFORMÁTICA

nos serve de exemplo) prende-se com o facto de nem todos os *use cases* (processos) de negócio virem a ser suportados pelo sistema informático. Para além disso, devemos ter presente, que uma simplificação no processo de negócio poderá ser uma solução mais correcta (eficiente e eficaz) do que o desenvolvimento de uma aplicação informática com uma funcionalidade complexa.

Após a identificação dos actores, deve-se identificar para cada actor os *use cases* em que este interage com o sistema. No nosso exemplo, estes são identificados na tabela seguinte:

ACTOR	USE CASES
Cliente	» Efectuar Encomenda Internet » Controlo de Acesso
Empregado Balcão	* Efectuar Encomenda » Controlo de Acesso
Gestor Pizzaria	Reservar Mesa Controlo de Acesso

TABELA 2.1 - IDENTIFICAÇÃO DE *USE CASES POR* ACTOR

2.1.4 Comunicação entre actores e *use cases*

A comunicação entre um actor e os *use cases* pode ser representada uma simples linha recta ou uma seta cujas pontas indicam a direcção da comunicação.

- **Linha recta simples** - Os actores podem estar colocados em qualquer ponto do diagrama, com o pressuposto que existirá alguma comunicação de emissão ou recepção.
- **Seta unidireccional** - A seta indica o sentido preferencial da comunicação. Normalmente, neste caso é habitual a colocação dos actores emissores à esquerda da fronteira do sistema, e dos actores receptores à direita.

É normal utilizarmos tanto a primeira como a segunda alternativa. Neste livro os exemplos mais simples utilizarão a linha recta simples, enquanto que os exemplos um pouco mais complexos utilizarão a seta unidireccional. A fig. 2.3 ilustra estas duas alternativas.

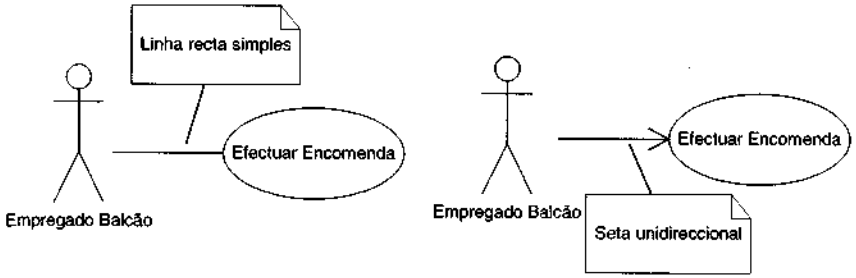


FIGURA 2,3 – REPRESENTAÇÃO DE COMUNICAÇÃO

2.1.5 Tempo

Na identificação dos *use cases* parte-se do princípio que todos são originados pelos actores. Contudo, em alguns sistemas existem *use cases* que são despoletados, automaticamente, de acordo com um processo temporal cíclico, onde num determinado intervalo de tempo o *use case é* executado.

Por exemplo, no caso em estudo poderia existir a necessidade de efectuar um cópia periódica dos dados das encomendas ou o envio mensal das promoções aos clientes registados. A fig. 2.4 mostra a representação destes *use cases*, repare na seta unidireccional:

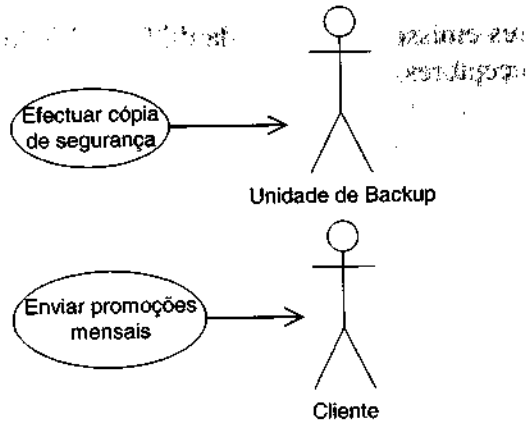


FIGURA 2.4 – EXEMPLO *USE CASES* CÍCLICOS

2.1.6 Cenário principal e Cenários Secundários

Cada um dos *use cases* identificados deve ser detalhado ou descrito em termos de cenários de utilização. Estes cenários são os possíveis caminhos seguidos dentro do *use case*, de forma a fornecer ao actor uma resposta (Shneider e Winters, 1999).

Esta descrição pode assumir a forma de texto livre ou estruturado segundo um conjunto de passos numerados, ficando esta decisão ao critério do analista. Complementarmente, a UML disponibiliza um conjunto de técnicas, designadas por diagramas de interacção, que permitem descrever de forma gráfica os diversos cenários (este tema é abordado no Capítulo 5).

Por exemplo, o detalhe do *use case* "Efectuar Encomenda Internet" seria:

Texto Livre

"Efectuar Encomenda Internet:
O cliente, após ter validado o seu acesso, selecciona a opção Encomendar, sendo mostrado em simultâneo com a sua encomenda o catálogo de produtos. Para adicionar um

produto, tem apenas que introduzir o código do mesmo, para que, automaticamente, o seu nome, descrição e preço sejam visualizados no respectivo item da encomenda. Ao mesmo tempo, é calculado o valor total da encomenda.

Através da opção Confirmar, o cliente confirma a sua encomenda e passa para a função de pagamento, onde após a introdução e confirmação dos dados do cartão de crédito é atribuído um número de identificação à encomenda, que posteriormente será entregue na morada do cliente."

Descrição Estruturada

Efectuar Encomenda Internet (Cenário Principal)	
Pré-condição	O cliente é um utilizador válido no sistema.
Descrição	1. O *use case* começa quando o cliente selecciona a opção de Encomendar. 2. Em simultâneo com a sua encomenda é mostrado o catálogo de produtos. 3. O cliente adiciona produtos à encomenda através da introdução do código do produto. 4. Automaticamente, o sistema mostra o nome, descrição e preço do produto. 5. De cada vez que é adicionado um produto, o valor total da encomenda é calculado. 6. O cliente confirma a sua encomenda através da opção Confirmar. 7. O sistema pede então os detalhes do cartão de crédito. 8. O sistema confirma os dados do pagamento e atribui um número de identificação à encomenda.
Pós-Condição	A encomenda será entregue na morada do cliente.

> Na nossa experiência, a descrição estruturada tem provado ser bastante eficaz.

No exemplo anterior, foram introduzidos os conceitos de **pré-condição e pós-condição** que indicam, respectivamente, o estado inicial e final do sistema aquando da realização do *use case.*

A pré-condição indica o que deve existir inicialmente para que o cenário descrito seja seguido com sucesso. No caso da pós-condição é demonstrado o que irá acontecer depois do cenário ser concluído.

Na descrição do *use case* pressupõe-se que estão reunidas todas as condições que garantem que tudo corre bem, sendo um cenário onde não surgem problemas, denominado como o **cenário principal.** Contudo, pode existir a necessidade de descrever situações (caminhos) alternativas, ou seja, **cenários secundários,** especialmente quando se pensa no que poderá correr mal no cenário. Por exemplo, no caso anterior, o cliente poderia ter introduzido um código de produto errado ou não ter um cartão de crédito válido.

Assim, ficaríamos com a seguinte descrição:

Efectuar Encomenda Internet (Cenários Secundários)	
Pré-condição	O cliente é um utilizador válido no sistema.
Descrição	1. O *use case* começa quando o cliente selecciona a opção de Encomendar. 2. Em simultâneo com a sua encomenda é mostrado o catálogo de produtos. 3. O cliente adiciona produtos à encomenda através da introdução do código do produto. a) Se um código é inválido o sistema avisa o cliente com uma mensagem. 4. Automaticamente o sistema mostra o nome, descrição e preço do produto.

	5. De cada vez que é adicionado um produto o valor total da encomenda é calculado. 6. O cliente confirma a sua encomenda através da opção Confirmar. 7. O sistema pede então os detalhes do cartão de crédito. 8. O sistema confirma os dados do pagamento e atribui um número de identificação à encomenda. a) Se o cartão for inválido, o sistema avisa o cliente através de uma mensagem, voltando em seguida para o passo 7.
Caminhos Alternativos	A qualquer momento, antes de efectuar o pagamento, o cliente pode cancelar a sua encomenda, pressionando no botão Cancelar.
Pós-Condição	A encomenda será entregue na morada do cliente.

As alternativas podem ser introduzidas directamente no texto da descrição ou quando mais complexas, na linha de Caminhos Alternativos.

A representação gráfica do cenário principal e dos cenários secundários pode ser efectuada em conjunto quando a sua complexidade é reduzida.

Ao serem definidos os actores e *use cases, é* também definida a fronteira do sistema, que separa os requisitos que estão fora ou dentro do sistema a desenvolver. Contudo, por vezes, é difícil distinguir claramente onde enquadrar determinados requisitos, ficando esta decisão ao critério do analista que deve procurar a resposta, analisando cuidadosamente a pertinência de cada requisito.

Nem todos os requisitos possuem a mesma importância para o sistema. Sendo assim, é necessário efectuar uma triagem dos mesmos, através de uma classificação por ordem de importância. É frequente a utilização da seguinte escala:

- **Obrigatório** - O requisito será incluído de certeza.
- **Desejável** - Não é garantida a sua inclusão, depende de outros factores como custos, risco ou recursos.
- **Adiado** - Será incluído numa segunda versão do sistema.

> A descrição de um *use case* deve incluir todos os detalhes encontrados na análise (actores, dados, processo) de forma a aumentar a informação disponível.

2.1.7 Relações de «include», «extend» e generalização

Os *use cases* podem estar relacionados entre si. As relações mais frequentes são «include», «extend» e generalização. A relação «include» significa que um determinado *use case* utiliza ou inclui a funcionalidade disponibilizada num outro *use case.*

No nosso exemplo foram utilizadas as seguintes relações:

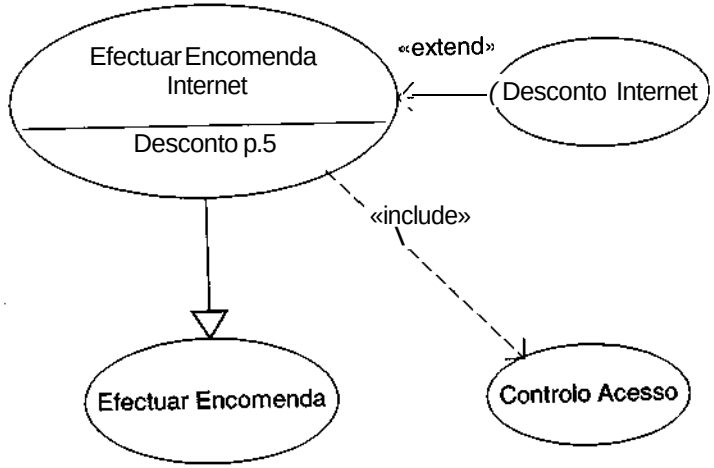


FIGURA 2.5 - EXEMPLO DE RELAÇÕES

Neste diagrama utiliza-se a relação «include» para demonstrar que a funcionalidade "Controlo Acesso" é utilizada quando uma encomenda é efectuada através da Internet. Esta relação também é útil quando existem *use cases* repetidos, pois evita a sua duplicação no diagrama.

> Alguns autores utilizam a relação «uses» em vez de «include».

Na descrição do *use case* "Efectuar Encomenda Internet" foi referido um pressuposto na pré-condição que o cliente era um utilizador válido do sistema, ou seja, já tinha passado pelo controlo de acesso. Caso não existisse este pressuposto, a relação «include» também teria que ser incluída na descrição, como é mostrado em seguida:

Descrição Estruturada *(Include)*

Efectuar Encomenda Internet (Cenário Principal)	
Pré-condição	
Descrição	1. Include: Controlo de Acesso. 2. O *use case* começa quando o cliente selecciona a opção de Encomendar. 3. Em simultâneo com a sua encomenda é mostrado o catálogo de produtos. 4. …

A relação «extend» ocorre quando existe um comportamento opcional que deve ser incluído num *use case.* Este comportamento é definido num segundo *use case* e invocado pelo *use case* base, através de um mecanismo de **pontos de extensão.**

O mecanismo de pontos de extensão permite definir no *use case* base onde o comportamento será incorporado, sem alterar a sua descrição. Também garante que o seu comportamento não seja

alterado caso o "Desconto Internet" deixe de existir. A sua descrição é efectuada da seguinte forma:

Descrição Estruturada (*extend*)

Efectuar Encomenda Internet (Cenário Principal)	
Pré-condição	
Descrição	4. ... 5. Para cada produto escolhido, o sistema verifica o seu preço, que é adicionado ao custo total da encomenda. **6. Se o produto está em promoção, existindo assim um desconto:** **a. Extend: Calcular Desconto.** 7. Em simultâneo com a sua encomenda é mostrado o catálogo de produtos. 8. ...

Na descrição do *use case* "Efectuar Encomenda Internet" é definido um ponto de extensão "Desconto p.6" que indica onde, será utilizado o comportamento do *use case* "Desconto Internet", neste caso no passo 6 (p.6). Desta forma, o cliente ao adicionar um novo produto à sua encomenda obterá um desconto caso o produto esteja em promoção.

A descrição do *use case* de extensão "Desconto Internet" é a seguinte:

Desconto Internet (Extensão)	
Pré-condição	O produto está em promoção na Internet.
Descrição	1. O sistema retorna a valor do desconto. 2. Mostra o desconto na encomenda. 3. Calcula o desconto subtraindo ao preço do produto.

A relação de **generalização** é utilizada quando existe um *use case* que é um caso particular de um outro *use case*. Por exemplo, na fig. 2.5 o comportamento do *use case* "Efectuar Encomenda internet" é semelhante ao *use case* "Efectuar Encomenda", existindo apenas pequenas variações específicas do meio onde é efectuada a encomenda.

A generalização usufrui das mesmas propriedades que uma relação pai/filho, onde o *use case* "filho" herda ou substitui por completo o comportamento do *use case* "pai". Por exemplo, imaginemos que o *use case* "Controlo de Acesso" pode ser realizado de 2 formas diferentes, conforme efectuado através da Internet ou na loja. Para representar estes requisitos, poderíamos utilizar a seguinte generalização (fig. 2.6):

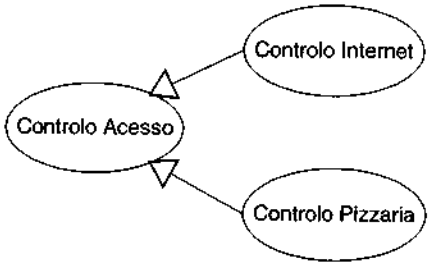


FIGURA 2,6 - EXEMPLO DE GENERALIZAÇÃO

Esta relação também pode ser utilizada entre actores, como é demonstrado pela fig. 2.7.

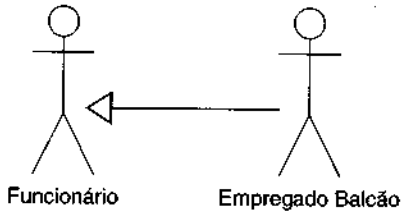


FIGURA 2.7 - EXEMPLO DE GENERALIZAÇÃO ENTRE ACTORES

No exemplo da figura anterior é estabelecida uma relação de generalização entre o actor Funcionário (caso geral) e o actor Empregado de Balcão (caso específico). Esta relação evita a duplicação de ligações quando ambos os actores partilham alguns *use cases.*

A fig. 2.8 ilustra a situação em que todos os funcionários têm que registar a hora de entrada/saída e apenas o empregado de balcão pode registar encomendas.

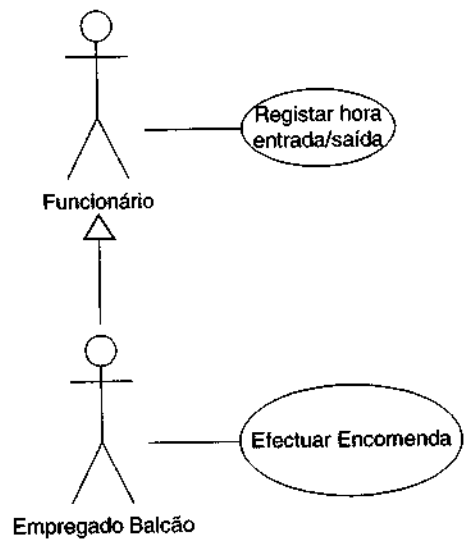


FIGURA 2.8 - EXEMPLO DE GENERALIZAÇÃO ENTRE ACTORES

Segundo Fowler (2000), as seguintes regras podem ser aplicadas para decidir sobre que relações utilizar:

* Utilizar «include» quando o mesmo *use case* pode ser utilizado em duas ou mais situações.
* Utilizar a generalização quando se descreve a variação do comportamento normal, pretendendo apenas efectuar uma descrição casual.

© FCA - EDITORA DE INFORMÁTICA

* Utilizar «extend» quando se descreve a variação do comportamento normal, mas de uma forma mais controlada, através de pontos de extensão no *use case* base.

Tendo definido os requisitos do sistema, é possível descrever os objectos que compõem o sistema e as suas relações, constituindo um resultado da análise dos requisitos. Esta descrição é efectuada através de um diagrama de classes, tema abordado no capítulo seguinte.

2,2 EXERCÍCIOS

2.2.1 Perguntas de Revisão

1: O que significa um *use casei*

2: Qual é a notação para um *use case*?

3: O que significa um actor?

4: Qual é a notação para um actor?

5: Para que servem os diagramas de *use case*?

6: Defina o conceito de requisito?

7: Que tipos de requisitos existem?

8: Que tipo de associação é possível entre um actor e um *use casei*

9: Que tipos de relação podem ser efectuadas entre *use cases*?

10: Qual é a diferença entre a relação de «include» e «extend»?

11: Que notação é utilizada para a relação de generalização?

12: O que significa um ponto de extensão?

2.2.2 Problemas Resolvidos

Efectue o levantamento de requisitos e desenhe o respectivo diagrama de *use cases*.

> Primeiro identifique os actores, em seguida, identifique os respectivos *use cases* e, por fim, desenhe o diagrama.

Biblioteca

Da entrevista com o responsável da biblioteca de uma universidade resultou a seguinte descrição para um novo sistema informático:

"Um das actividades principais da biblioteca é efectuar o empréstimo de publicações aos alunos da universidade. O empréstimo é registado pelos funcionários da biblioteca, que também consultam diariamente os empréstimos cujos prazos foram ultrapassados. Todo este processo é efectuado manualmente, sendo muito ineficiente. Espera-se que o novo sistema resolva esta situação.
Os alunos necessitam de pesquisar os livros existentes na biblioteca. Caso um livro esteja requisitado, é mostrada a data esperada de entrega."

Parque de Estacionamento

Considere os seguintes requisitos de um sistema informático para a gestão de um parque de estacionamento.

a) O controlo é efectuado com base na matrícula do veículo.

b) Na entrada do parque existirá um funcionário que introduz as matrículas no sistema, ficando de imediato registado a data e hora de início do estacionamento. O sistema tem que verificar se a matrícula existe.

c) Se a matrícula não for reconhecida pelo sistema, então o funcionário registará um novo veículo no sistema.

d) Na saída, um funcionário introduz novamente a matrícula, sendo que o sistema calcula o custo do estacionamento.

e) O Gestor do Parque precisa de consultar diariamente uma listagem dos estacionamentos. Em algumas situações, o gestor poderá desempenhar as funções de atendimento, no entanto, apenas o gestor poderá obter as listagens.

Solução Biblioteca

1° - Identificação dos actores

No texto da entrevista existem dois actores que interagem/utilizam o sistema:

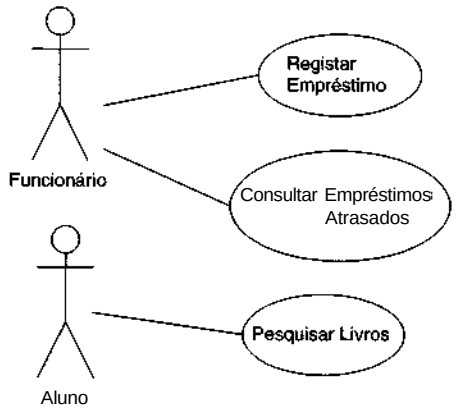
Funcionário - Pessoa responsável por registar o empréstimo e gerir os empréstimos em atraso.

Aluno - Necessita de pesquisar os livros existentes.

2° - Identificação dos *use cases* por actor

ACTOR	*USE CASES*
Funcionário	Registar Empréstimo
	Consultar Empréstimos Atrasados
Aluno	Pesquisar Livros

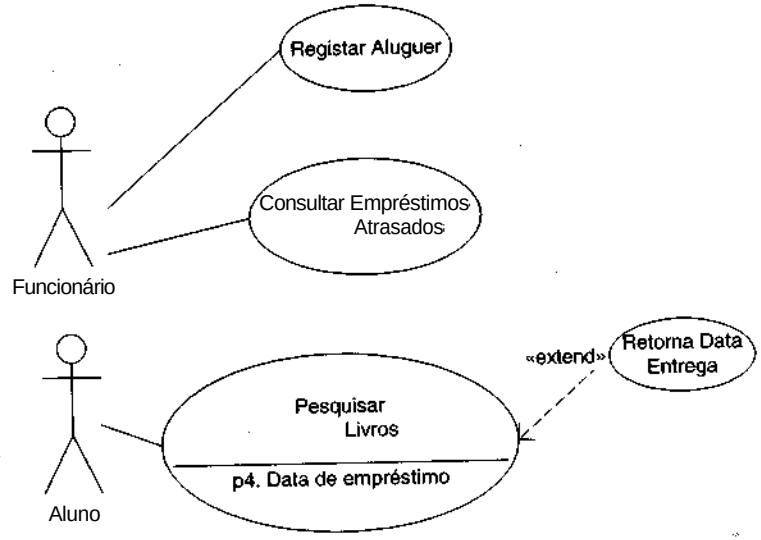
3° - Desenhar o diagrama de *use cases*



© FCA - EDITORA DE INFORMÁTICA

Solução Alternativa

A primeira solução não tem em consideração relações entre os *use cases* e também não demonstra que na pesquisa tem que ser mostrada a data de entrega quando um livro está emprestado. Sendo assim, o diagrama seguinte inclui este *use case*.



Possível descrição simplificada para o *use case* "Pesquisar Livro" (repare no extend):

1. O aluno introduz a sua expressão de pesquisa.
2. O sistema procura em cada livro a expressão.
3. Caso o livro possua a expressão de pesquisa, a sua referência é acrescentada à lista de livros a devolver.
4. Caso um livro esteja emprestado, acrescenta a data de entrega.
   a. Extend: Retorna data entrega.
5. A lista de referências é devolvida ao utilizador.

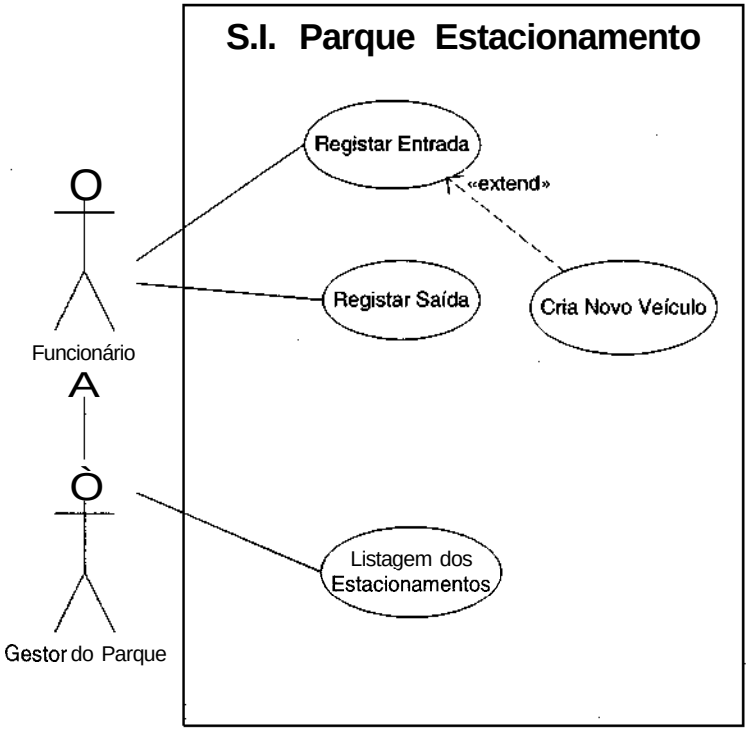
© FCA - EDITORA DE INFORMÁTICA

Solução Parque de Estacionamento

Identificação dos actores e *use cases*

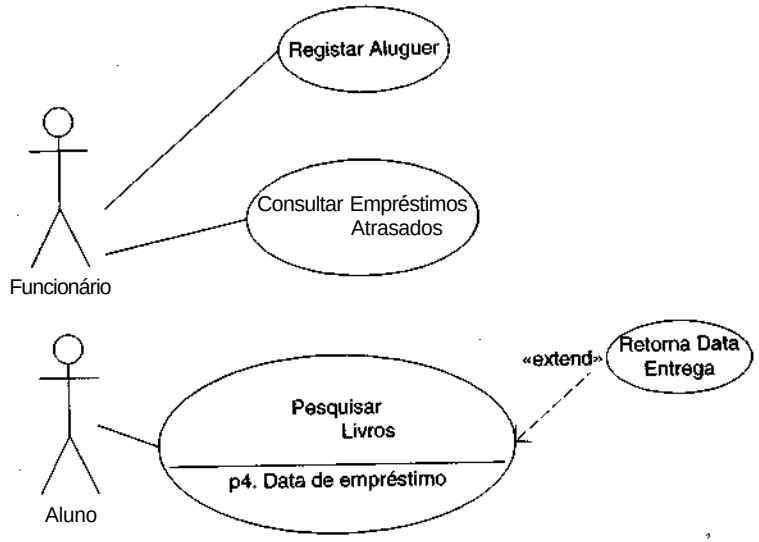
ACTOR	*USE CASES*
Funcionário	Registar Entrada (se matrícula não reconhecida, criar novo veículo) Registar Saída
Gestor do Parque	Listagem dos Estacionamentos

Diagrama de *use cases*



Solução Alternativa

A primeira solução não tem em consideração relações entre os *use cases* e também não demonstra que na pesquisa tem que ser mostrada a data de entrega quando um livro está emprestado. Sendo assim, o diagrama seguinte inclui este *use case.*



Possível descrição simplificada para o *use case* "Pesquisa: Livro" (repare no extend):

1. O aluno introduz a sua expressão de pesquisa.
2. O sistema procura em cada livro a expressão.
3. Caso o livro possua a expressão de pesquisa, a sua referência é acrescentada à lista de livros a devolver.
4. Caso um livro esteja emprestado, acrescenta a data de entrega.
   a. Extend: Retorna data entrega.
5. A lista de referências é devolvida ao utilizador.

© FCA - EDITORA DE INFORMÁTICA

Solução Parque de Estacionamento

Identificação dos actores e *use cases*

ACTOR	USE CASES
Funcionário	Registar Entrada (se matrícula não reconhecida, criar novo veículo) Registar Saída
Gestor do Parque	Listagem dos Estacionamentos

Diagrama de *use cases*

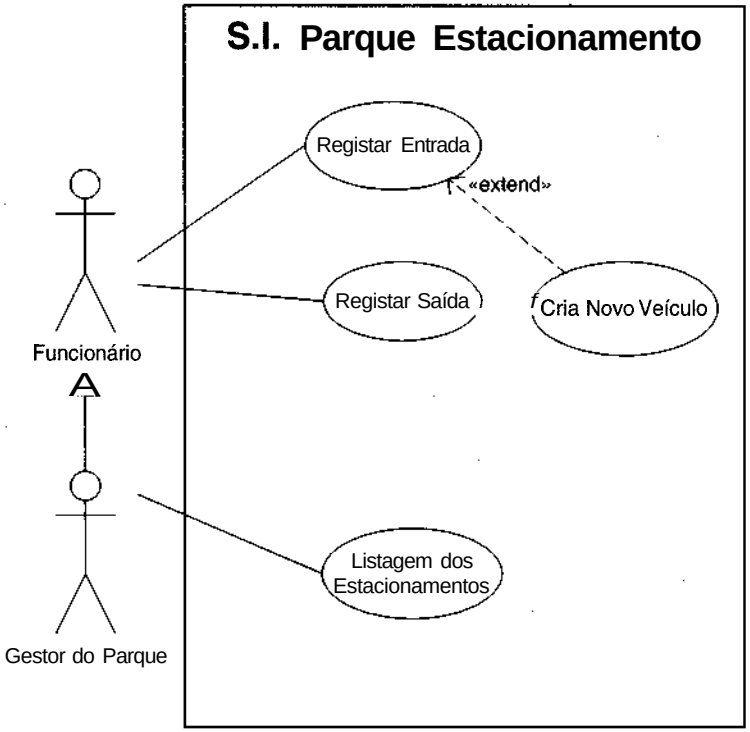


Diagrama de Classes 3

3.1 E APLICAÇÃO

A UML adoptou também o diagrama de classes, uma das técnicas mais utilizadas no desenvolvimento orientado por objectos. Este diagrama é uma descrição formal da estrutura de objectos num sistema. Para cada objecto descreve a sua identidade, os seus relacionamentos com os outros objectos, os seus atributos e as suas operações.

A criação de um modelo de classes resulta de um processo de abstracção através do qual se identificam os objectos (entidades e conceitos) relevantes no contexto que se pretende modelar e se procuram descrever características comuns em termos de propriedades (atributos) e de comportamento (operações). A essa descrição genérica designa-se por classe. Assim, as classes descrevem objectos com atributos e operações comuns, e servem dois propósitos: permitem compreender o mundo real naquilo que é relevante para o sistema de informação que se pretende desenvolver e fornecem uma base prática para a implementação em computador (Rumbaugh et al, 1991).

Os diagramas de classes descrevem o modelo geral de informação de um sistema. Os diagramas de objectos podem ser utilizados como exemplo para ajudar a compreender um diagrama de classes complexo.

Um diagrama de classes é composto pelos seguintes elementos abstractos de modelação:

© FCA - EDITORA DE INFORMÁTICA

« Classes de objectos
• Relações de Associação e Generalização
• Multiplicidade

A perspectiva estática fornecida pelo diagrama de classes tem como objectivo suportar os requisitos funcionais do sistema, que foram levantados previamente (tema abordado no Capítulo 2). Assim, o diagrama de classes é um resultado da análise de requisitos, fornecendo um modelo que mais tarde será utilizado na fase de desenho para a definição dos componentes da aplicação.

Tipicamente, o diagrama de classes é utilizado no seguinte conjunto interdependente de formas (Booch et al., 1999):

1. **Modelar o vocabulário de um sistema:** Envolve o decidir sobre que abstracções estruturais (aspectos mais importantes) fazem parte do sistema em estudo e quais estão fora das suas fronteiras.
2. **Modelar colaborações simples:** Visualizar o sistema como um todo constituído por classes e suas relações que, através do seu trabalho em conjunto, fornecem um comportamento cooperativo.
3. **Modelar o esquema lógico de uma base de dados (BD):** Desenhar a estrutura de dados para uma BD relacional ou orientada por objectos, de forma a guardar a informação do sistema.

Neste capítulo os diagramas são elaborados na perspectiva de modelar as classes do sistema e as suas relações. Aproveitando o exemplo apresentado no Capítulo 2, poderíamos acrescentar a seguinte descrição:

"Um cliente pode efectuar muitas encomendas, contendo cada encomenda diversos itens, numerados sequencialmente, que se referem a um determinado produto e respectiva quantidade encomendada. Os produtos vendidos pela

PhonePizza abrangem pizzas com diversos tamanhos (Pequena, Média e Grande), bebidas e saladas. O preço pode variar conforme o tamanho do produto bem como com as promoções existentes que têm uma data de início e de fim."

A fig. 3.1 apresenta um possível diagrama de classes simplificado para o problema.

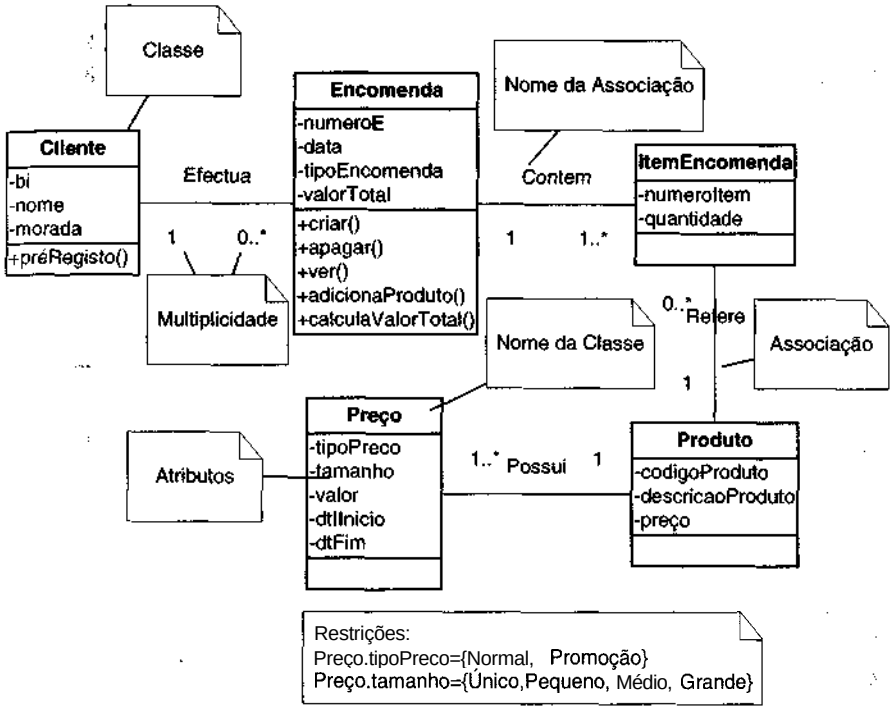


FIGURA 3.1 - EXEMPLO DIAGRAMA DE CLASSES

Existem pormenores como os diversos tipos de produto e associações que só são demonstrados nos tópicos avançados do diagrama de classes. Os principais conceitos são explicados em seguida.

© FCA - EDITORA DE INFORMÁTICA

3.1.1 O que é um Objecto

Um objecto é uma entidade ou conceito existente no contexto de modelação (mundo real). Que é relevante incorporar no modelo de informação. É caracterizado por um conjunto de Propriedades, um Comportamento e Identidade. As **Propriedades** são as características que definem o objecto, transpostas para um conjunto de atributos, cujos valores estabelecem o Estado do objecto. O **Comportamento** é definido como as operações que o objecto pode efectuar. A **Identidade** permite identificar um objecto em particular como único num conjunto de objectos semelhantes. Por exemplo, podemos ter os seguintes objectos:

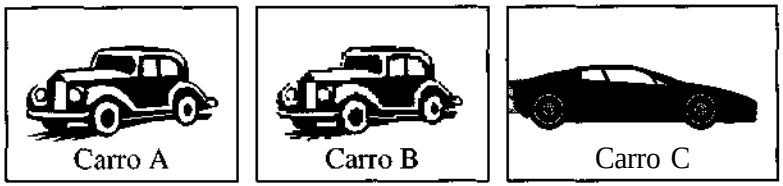


FIGURA 3,2 - EXEMPLO DE OBJECTOS CARRO

O carro A é diferente do carro B e C, contudo todos eles possuem um conjunto de atributos (Numero de Série, Cor, Data de Fabrico, etc.) que os definem (Estado) e realizam operações como de "Iniciar Marcha" ou "Acelerar" (Comportamento). E, para além de algumas semelhanças, possuem uma identidade própria que os torna únicos.

A mesma caracterização pode ser aplicada para os clientes de uma loja, por exemplo:

FIGURA 3,3 - EXEMPLO DE OBJECTOS CLIENTE

© FCA - EDITORA DE INFORMÁTICA

Os diversos objectos `cliente` também partilham as mesmas propriedades como o Nome, Morada, Data de Nascimento, Número Bilhete de Identidade, Sexo, etc. O mesmo para o comportamento, pois todos têm que se registar como clientes (`"registar"`) ou podem actualizar os seus dados pessoais (“`alterarDados`”).

3.1.2 O que é uma Classe

Representa uma abstracção sobre um conjunto de objectos que partilham a mesma estrutura e comportamento. Na prática, um objecto é um caso particular de uma classe, também referido como uma instância da classe.

No exemplo anterior poderíamos resumir os diferentes objectos num conjunto de propriedades e operações comuns (fig. 3.4):

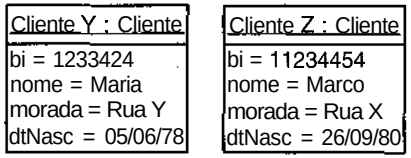


FIGURA 3,4 - RESUMO DE PROPRIEDADES E COMPORTAMENTO

Sendo descritos na seguinte classe `Cliente`:

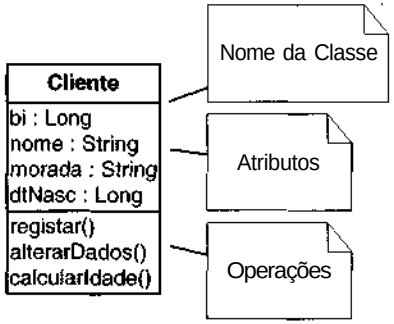


FIGURA 3.5 – CLASSE CLIENTE

As instâncias da classe (objectos) podem ser representadas da seguinte forma:

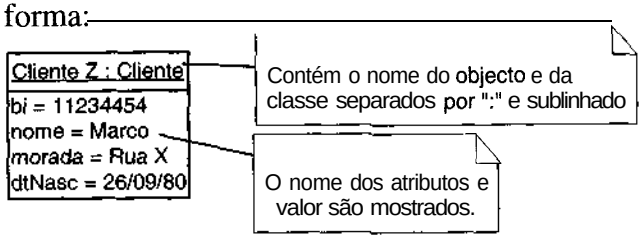


FIGURA 3.6 - INSTÂNCIA DE UMA CLASSE (OBJECTO)

Um **atributo** é uma característica que os objectos possuem e que é representada por um valor de dados. Por exemplo, o atributo Cor poderá ser igual a "Vermelho" ou "Azul". Os nomes dos atributos são únicos numa classe, não podendo existir na mesma classe dois atributos com o mesmo nome. Em classes diferentes podem existir atributos com nomes iguais.

Os objectos apenas comunicam entre si por **mensagens,** que na prática resulta na invocação de operações. Este conceito é explorado em detalhe no Capítulo 5. As **operações** são a representação lógica do comportamento de um objecto, consistindo em acções efectuadas por ou sobre um objecto. Em alternativa, também se pode definir operações como serviços disponibilizados por um objecto. Estes serviços serão invocados por outros objectos como parte integrante de uma colaboração (a modelação da colaboração é abordada no Capítulo 5).

Na classe `Cliente` pode-se definir a operação `calcularIdade()`. A UML utiliza parêntesis para simbolizar a existência ou não de parâmetros. Para a operação anterior poderia ser necessário fornecer a data actual, o que implicaria definir a operação como `calcularIdade(dtActual)`. Uma operação também pode possuir um valor de retorno, que no exemplo anterior corresponderia no retorno da idade do cliente. O nome, os parâmetros e valor de retorno da operação constituem a sua **assinatura** permitindo ao objecto distinguir as diversas operações.

© FCA - EDITORA DE INFORMÁTICA

Tanto os atributos como as operações podem ser visíveis ou não para outras classes. Esta propriedade denomina-se **visibilidade** e assume 3 níveis:

* **Público** - Qualquer classe tem acesso ao elemento. É representado através do prefixo `+`.
* **Protegido** - Qualquer descendente da classe pode utilizar o elemento. É representado através do prefixo `#`.
* **Privado** - Apenas a própria classe tem acesso ao elemento. É representado através do prefixo `-`.

Os atributos de uma classe são normalmente privados, sendo que o valor dos atributos só poderá ser alterado através da execução de uma operação (alterando assim o seu estado). Este facto, realça a propriedade de **encapsulamento** nos objectos que não é mais que "esconder" o conteúdo do objecto e apenas disponibilizar uma interface (operações) que fornece serviços a outros objectos, separando assim o que um objecto demonstra que faz, da forma como o faz. Este mecanismo permite que o conteúdo do objecto possa ser alterado sem afectar os outros objectos que estão dependentes da sua interface e também aumenta a sua capacidade de reutilização.

A figura 3.7 ilustra o conjunto de elementos públicos, privados e protegidos para a classe `Cliente`:

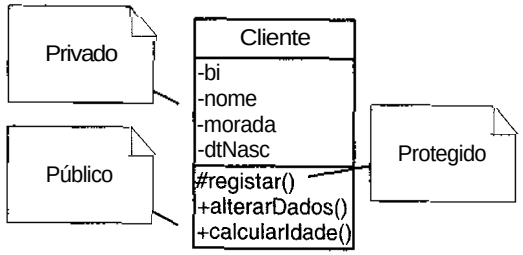


FIGURA 3.7 - VISIBILIDADE DE ATRIBUTOS E OPERAÇÕES

3.1.3 Tipos de dados básicos

Para cada atributo também pode ser identificado o seu tipo de dados, que caracteriza a informação que o atributo irá conter. Os tipos de dados disponíveis dependem directamente da linguagem de programação em que o sistema será desenvolvido. Contudo é possível restringir ao seguinte conjunto de tipos básicos:

- **Integer** - Representa um número inteiro.
- **Long** - Representa um número inteiro mas de maior dimensão.
- **Double** - Para números reais.
- **String** - Representa texto.
- **Date** - Para datas.
- **Boolean** - Valor lógico que representa Verdade ou Falso.

As operações também podem possuir um tipo de dados para os seus argumentos e para o resultado da operação. A figura seguinte (figura 3.8) ilustra os tipos de dados para a classe Cliente:

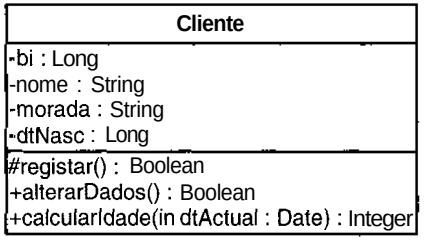


FIGURA 3,8 – EXEMPLO DE TIPOS DE DADOS

Para a operação calcularIdade é necessário fornecer a data actual (parâmetro dtActual), logo o seu tipo de dados é Date. A operação devolve a idade do cliente, logo o seu tipo de dados é um número inteiro (Integer).

3.1.4 Associações

No diagrama de classes, as associações representam as relações entre os objectos. No exemplo apresentado na figura 3.9, temos que um objecto da classe Pessoa pode trabalhar em muitas empresas e, por sua vez, uma empresa pode empregar muitas pessoas.

As associações são caracterizadas por possuir um nome e quando necessário podem também incluir o papel que os objectos têm na relação. Por exemplo:

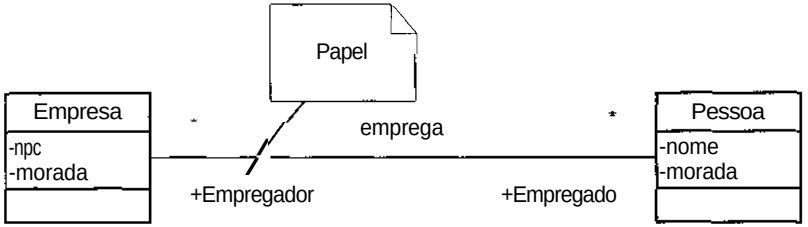


FIGURA 3,9 - EXEMPLO DE PAPEL NUMA RELAÇÃO

Na fig. 3.9 o papel de uma pessoa é ser o Empregado, enquanto que o papel de uma empresa é ser o Empregador.

> O nome das associações são normalmente verbos e de tamanho reduzido.

Uma classe pode possuir uma associação consigo própria, significando neste caso que um objecto da classe se relaciona com um ou vários objectos da mesma classe. Tipicamente, esta relação surge em situações de hierarquia como, por exemplo, o chefe de um conjunto de empregados é também um empregado. A figura seguinte (fig. 3.10) apresenta este exemplo:

© FCA - EDITORA DE INFORMÁTICA

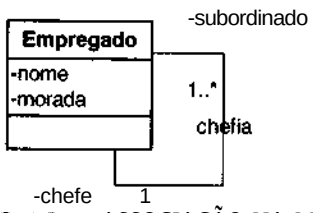


FIGURA 3.10 – ASSOCIAÇÃO NA MESMA CLASSE

3.1.5 Multiplicidade

As associações são também caracterizadas por possuir uma multiplicidade, que indica quantos objectos participam na relação. A multiplicidade pode assumir muitas formas, mas as mais comuns são:

- 0..1 - Opcional.
- 1..1 - Obrigatório existir um objecto, frequentemente representado por apenas 1.
- 1..10 - Um valor entre o intervalo estabelecido, neste caso de um a dez.
- **0..*** - Zero ou infinitos objectos da classe, também representado por apenas *.
- **1..*** - Um ou infinitos objectos da classe.

É possível efectuar várias combinações de multiplicidade numa associação. Por exemplo, na Figura 3.11 a relação "Um para Muitos" entre *a* Classe A e a Classe B significa que um objecto da Classe B está associado a um só objecto da Classe A e que um objecto da Classe A pode estar associado ou não (opcional) a muitos objectos da Classe B.

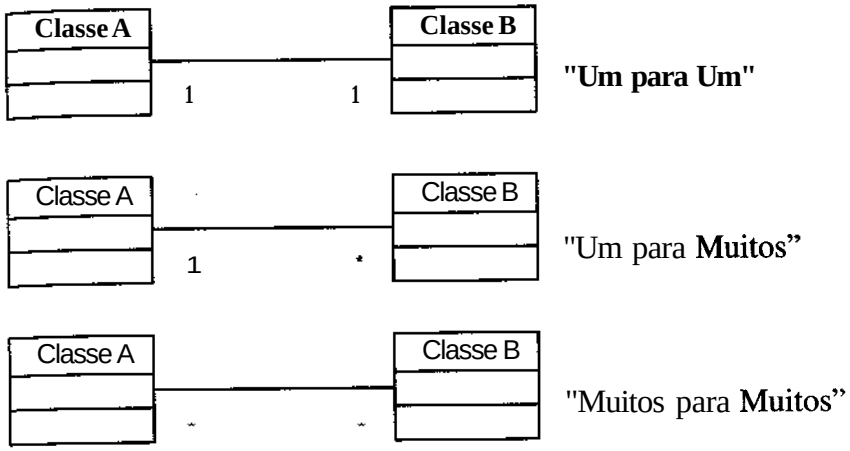


FIGURA 3,11 - MULTIPLICIDADE MAIS FREQUENTE

3.1.6 Identificação de classes

A identificação das classes não é um processo directo sendo, por vezes, necessárias diversas iterações e refinamentos até identificar correctamente todas as classes. Uma regra simples para começar o processo de identificação é sublinhando na descrição dos *use cases* os substantivos. Este procedimento é ilustrado em seguida para a descrição do *use case* Efectuar Encomenda Internet:

Efectuar Encomenda Internet (Cenário Principal)	
Pré-condição	O cliente é um utilizador válido no sistema.
Descrição	9. O *use case* começa quando o cliente selecciona a opção de Encomendar. 10. Em simultâneo com a sua encomenda é mostrado o catálogo de produtos. 11. O cliente adiciona produtos à encomenda através da introdução do código do produto. 12. Automaticamente, o sistema mostra o nome, descrição e preço do produto. 13. De cada vez que é adicionado um produto, o valor total da encomenda é calculado.

© FCA - EDITORA DE INFORMÁTICA

	14. O cliente confirma a sua encomenda através da opção Confirmar. 15. O sistema pede então os detalhes do cartão de crédito. 16. O sistema confirma os dados do pagamento e atribui um número de identificação à encomenda.
Pós-Condição	A encomenda será entregue na morada do cliente.

Com o procedimento anterior teríamos identificado as seguintes classes:

FIGURA 3.12 – IDENTIFICAÇÃO DE CLASSES

Nesta fase, apenas estamos preocupados em identificar as classes não sendo necessário identificar os respectivos atributos e operações. Como regra geral, os nomes das classes estão sempre no singular.

> Normalmente, na descrição dos *use cases* as classes e/ou objectos são identificados através dos substantivos.

Após a identificação preliminar é necessário efectuar uma pequena descrição para cada classe e ao mesmo tempo proceder a uma triagem com base nos seguintes critérios:

* **Fora do âmbito do sistema** - Algumas classes estão fora do domínio da aplicação (fronteira do sistema) e não devem ser consideradas. Não é fácil definir inicialmente os limites da fronteira, logo é natural que existam classes que só mais tarde

© FCA - EDITORA DE INFORMÁTICA

serão excluídas. Um engano comum é a inclusão de actores do sistema como classes. Normalmente, não devem ser incluídos a não ser que estes sejam utilizados em algum processo de negócio como, por exemplo, um cliente que efectua uma encomenda através da Internet.

Representa o sistema - Não é necessário representar o próprio sistema como uma classe.

Classes semelhantes - Podem existir classes que são sinónimos. Caso subsistam dúvidas, estas podem ser esclarecidas ao efectuar a descrição das classes.

Nível de detalhe - Eliminar as classes onde não é possível efectuar uma clara descrição (demasiado vagas) ou que são muito específicas quase como um objecto. Neste último caso, é preferível representar a classe do objecto.

Sendo assim, nas classes identificadas anteriormente seriam excluídas as classes `CartãoCrédito` e `CatálogoProdutos`, porque a primeira está fora do âmbito do sistema (os cartões apenas são usados para pagamento) e a segunda é muito vaga.

3.1.7 Identificação de atributos

Os atributos são características das classes que facilmente surgem na descrição dos *use cases*. Alguns atributos não são explicitamente referidos nas descrições, mas surgem do conhecimento do domínio como, por exemplo, o nome ou a morada de um cliente. A fig. 3.13 ilustra um diagrama de classes preliminar, baseado apenas no *use case* "Efectuar Encomendas Internet".

3.1.8 Identificação de associações e operações

As associações podem ser identificadas com base nas relações lógicas entre as classes. Também podem ser identificadas na descrição dos *use cases* através dos verbos como, por exemplo, "o cliente *efectua* encomendas". Contudo, só é possível compreender

na totalidade as associações através de uma análise das interacções entre os objectos das classes.

As operações ainda exigem um maior nível de detalhe, que numa fase inicial é difícil de atingir. Assim, à semelhança das associações, a sua definição só é possível após um estudo das interacções entre os objectos das classes. Este tema é abordado na Capítulo 5.

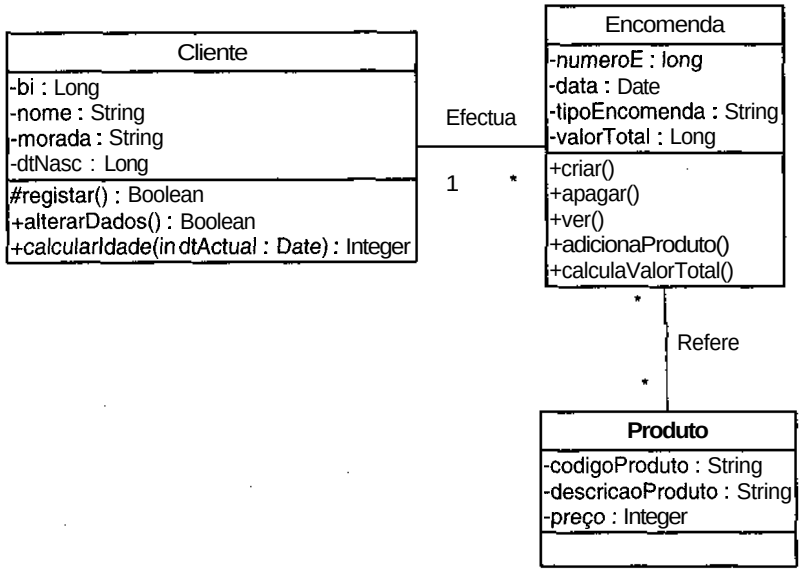


FIGURA 3.13 - DIAGRAMA DE CLASSES PRELIMINAR

Este diagrama simplifica a relação entre a encomenda e o produto, porque o conceito de composição só será abordado nos tópicos avançados deste capítulo (secção 3.2.3). Neste livro alguns exemplos utilizam esta simplificação por motivos de clareza de exposição.

3.1.9 Restrições

Para além das restrições impostas pelas associações no diagrama de classes, é também possível restringir o valor dos atributos das

© FCA - EDITORA DE INFORMÁTICA

classes. Por exemplo, na figura 3.14 o tipo de preço só pode ser do tipo "normal" ou "promoção". O mesmo para o tamanho que só permite os valores "pequeno", "médio", "grande".

As restrições no diagrama de classes, apenas devem ser representadas utilizando as chavetas "{}".

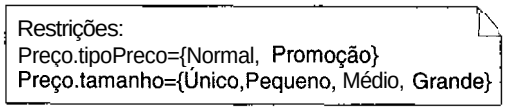


FIGURA 3,14 - EXEMPLO DE RESTRIÇÕES

Para além do nome do atributo, também se pode adicionar o nome da respectiva classe (`Preço.tipoPreco`).

3.2 TÓPICOS AVANÇADOS

3.2.1 Classes Associativas

Este tipo de classes surge da necessidade de reforçar o detalhe de informação de uma associação. É representado da seguinte forma:

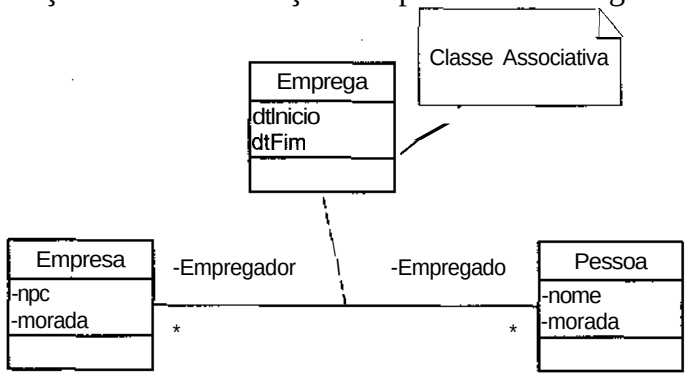


FIGURA 3.15 - EXEMPLO DE CLASSE ASSOCIATIVA

Neste caso, pretende-se especificar quando (data de início e data de fim) um empregado trabalhou para uma determinada empresa. Uma

classe associativa só existe em resultado da relação entre duas classes, sendo que por si só não terá significado.

> Normalmente, as classes associativas surgem nas relações de "Muitos para Muitos" e o nome da classe é dado pelo nome da associação.

3.2.2 Generalização e Herança

A generalização é um caso especial no diagrama de classes, que demonstra a noção de super-classe e subclasse na perspectiva de uma relação "pai e filho". Expandindo o exemplo apresentado inicialmente na fig.3.1, o produto seria representado da seguinte forma:

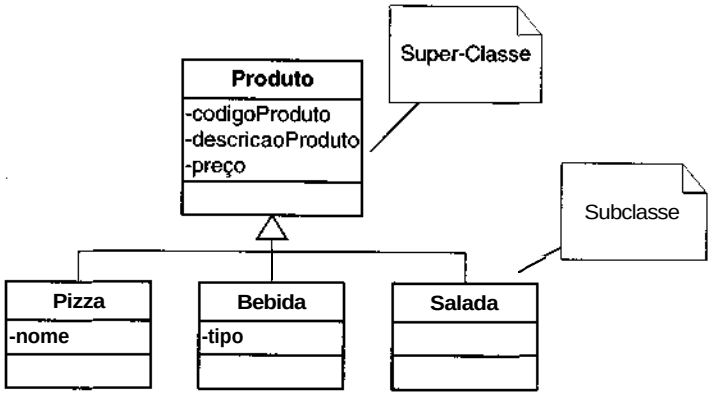


FIGURA 3.16 - EXEMPLO DE GENERALIZAÇÃO

No exemplo da fig. 3.16, através da utilização da generalização, o diagrama ilustra que existem 3 tipos de produtos Pizza, Bebida e Salada.

O conceito de **herança** está presente, pois as subclasses ("filhos") herdam da Super-Classe ("pai") a estrutura em termos de atributos e operações. Assim, está implícito que todas as subclasses possuem um código de produto (codigoProduto) e uma descrição (descricaoProduto).

3.2.3 Agregação e Composição

A agregação no diagrama de classes pretende demonstrar a o facto de que um todo é composto por partes. Por exemplo, podemos mostrar que um restaurante possui um conjunto de mesas com o seguinte diagrama na fig. 3.17:

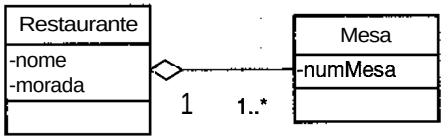


FIGURA 3.17 - EXEMPLOS DE AGREGAÇÃO

A composição é uma agregação com um significado mais forte existindo uma dependência directa entre as duas classes (se a parte deixar de existir, o todo também deixa de existir). Normalmente, a multiplicidade no lado do todo não ultrapassa o l, o que já não acontece com a agregação. O exemplo típico deste conceito é o caso de uma encomenda que é composta por itens:

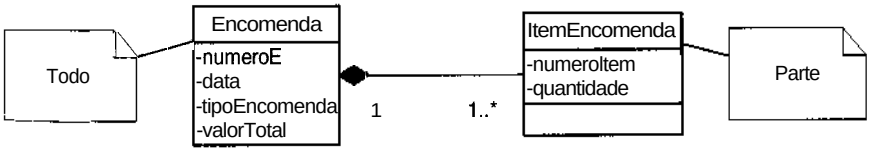


FIGURA 3,18 - EXEMPLO DE COMPOSIÇÃO

Neste caso, utiliza-se a composição, porque não faz sentido possuir uma encomenda sem itens ou itens sem uma encomenda.

> A distinção entre a agregação e a composição é ténue, ficando por vezes ao critério do analista.

É possível combinar a generalização com a agregação e composição. Por exemplo, podemos considerar no caso PhonePizza que para além dos produtos isolados, este podem ser agrupados em menus. Na prática teríamos o seguinte diagrama:

© FCA - EDITORA DE INFORMÁTICA

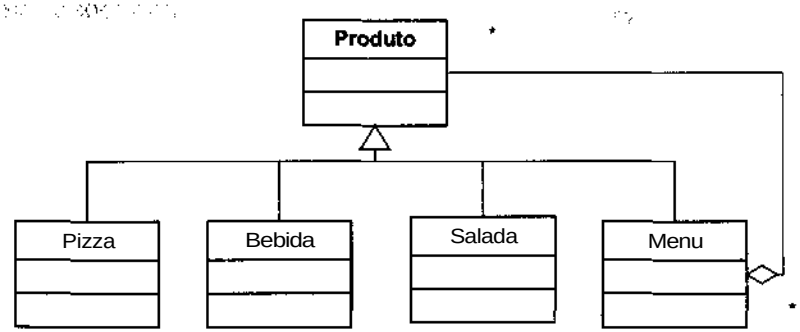


FIGURA 3.19 - COMBINAÇÃO ENTRE GENERALIZAÇÃO E AGREGAÇÃO

3.2.4 Diagrama de classes PhonePizza revisto

Em seguida, é apresentada, na fig. 3.20, uma actualização do diagrama de classes da PhonePizza utilizando os conceitos mais avançados.

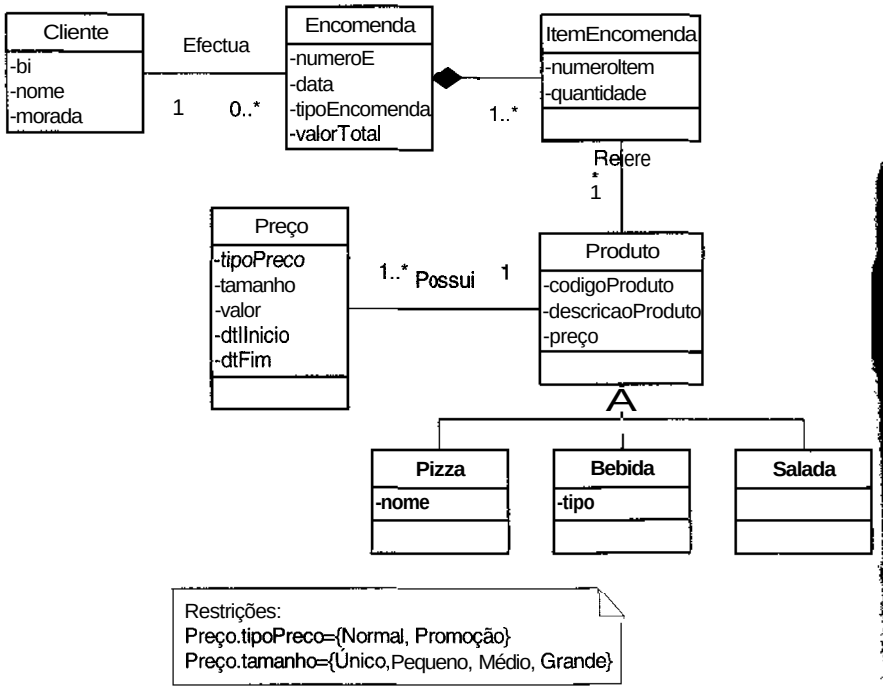


FIGURA 3.20 - DIAGRAMA DE CLASSES *PHONEPIZZA*

A principal diferença relativamente ao diagrama apresentado inicialmente é a utilização da composição entre a classe Encomenda e a classe `ItemEncomenda`, pois é uma relação muito forte, não fa/endo sentido existir uma encomenda sem itens e vice-versa. A generalização na classe Produto especifica os diversos produtos existentes nas subclasses Pizza, Bebida e Salada.

33 EXERCÍCIOS

3.3.1 Perguntas de Revisão

1: Qual é o objectivo de um diagrama de classes?

2: O que significa uma classe?

3: Qual é a notação para uma classe?

4: O que é um objecto?

5: Qual é a notação para um objecto?

6: Defina os conceitos de atributo e operações numa classe?

7: Em que consiste a visibilidade de um atributo?

8: Que tipos básicos podem assumir os atributos?

9: O que significa uma associação entre classes?

10: Defina o conceito de multiplicidade numa associação?

11: O que é uma classe associativa?

12: Qual a principal diferença entre a generalização, agregação e composição?

© FCA - EDITORA DE INFORMÁTICA

3.3.2 Problemas Resolvidos

Identifique classes e desenhe o respectivo diagrama.

> Primeiro identifique os vários objectos, em seguid agrupe-os em possíveis classes e, por fim, desenhe diagrama.

Biblioteca

Considere a seguinte informação adicional à descrição apresentada no exercício 2.2.2. Esta informação consiste num excerto da entrevista efectuada pelo consultor Paulo Bastos ao responsável da biblioteca João Almeida.

"**Paulo Bastos:** Como é que funciona o processo de empréstimo de publicações.

João Almeida: Bom, neste momento as publicações disponíveis aos alunos são os livros e as revistas que assinamos. Um aluno dirige-se com as publicações ao balcão de atendimento para preencher um ficha de empréstimo. Tem que efectuar uma ficha para cada publicação, preenchendo a cota e o título. Caso seja um livro, terá que escrever o(s) respectivo(s) autore(s).

P.B.: Existe alguma limitação no número de empréstimos?

J.A.: Sim, no máximo um aluno pode efectuar 3 empréstimos.

P.B.: Qual é o procedimento quando chega uma nova publicação?

J.A.: Bem... quando chega uma nova publicação esta é encaminhada para a responsável de catalogação, onde será analisada e definida a sua área de conhecimento. Existem várias áreas predefinidas como, por exemplo, Sociologia, Psicologia Informática, etc. Novas áreas de conhecimento podem se definidas.

P.B.: Existe alguma informação específica sobre cada uma da publicações?

© FCA - EDITORA DE INFORMÁTICA

J.A.: Para os livros temos que registar o seu número de identificação internacional, ISBN, e para as revistas registamos a suaperiodicidade."

parque de Estacionamento

Considere a seguinte informação adicional à descrição apresentada no exercício 2.2.2. Esta informação é um resumo das entrevistas conduzidas na empresa concessionária do parque de estacionamento.

- Em cada veículo apenas interessa guardar no sistema a respectiva matrícula.
- Um veículo pode efectuar vários estacionamentos no mesmo dia.
- Os veículos podem ser automóveis ou motorizadas.
- De início existe uma tarifa base que é aplicada a todos os veículos. Contudo, para veículos com um elevado número de estacionamentos, é possível criar tarifas específicas. Cada tarifa possui um custo por hora.
- O estacionamento possui um número de lugares limitado. Os lugares são caracterizados por um número, piso e um estado. Este estado só pode assumir os valores de Livre ou Ocupado.

Solução Biblioteca

1° - Identificação dos objectos

Os seguintes objectos são identificados:

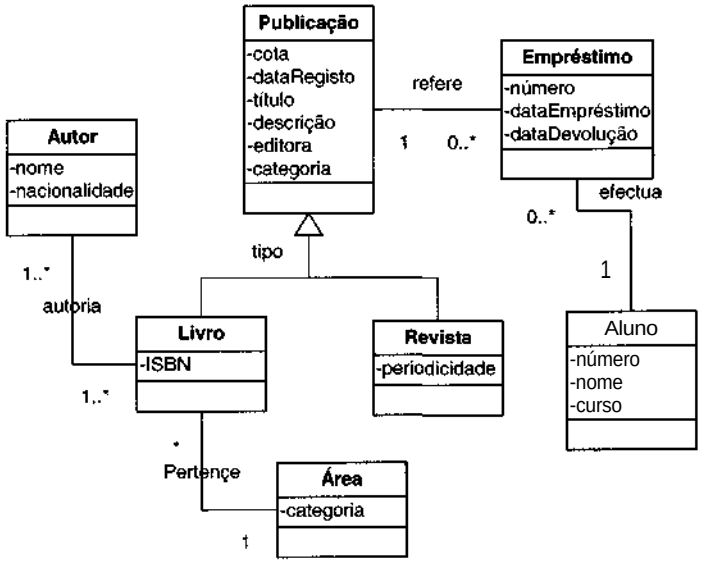
- Publicação
- Livro
- Revista
- Autor
- Área da publicação
- Ficha de Empréstimo

2° - Identificação das classes

- Publicação
- Livro

« Revista
* Autor
• Área
» Empréstimo

3° - Desenhar o diagrama de classes



Solução Parque de Estacionamento

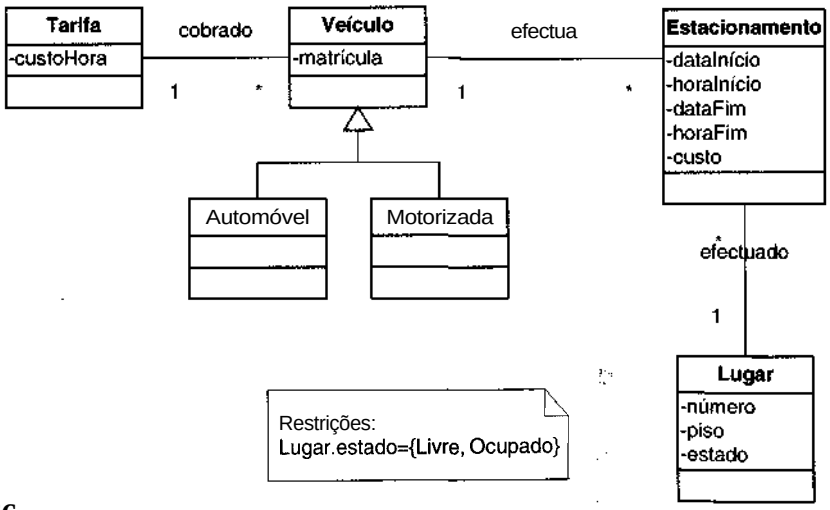


Diagrama de Actividades 4

4.1 E APLICAÇÃO

O diagrama de actividades constitui um elemento de modelação simples, mas eficaz para descrever fluxos de trabalho numa organização ou para detalhar operações de uma classe, incluindo comportamentos que possuam processamento paralelo.

No contexto dos sistemas de informação da gestão, define-se **processo de negócio** como um conjunto integrado de actividades de uma organização, que procura satisfazer um determinado objectivo e no qual participam um ou mais actores. Como já referimos anteriormente, um *use case* pode ser utilizado para identificar um processo de negócio de uma organização. O diagrama de actividades é assim particularmente útil quando se pretende detalhar um *use case* associado a um processo de negócio.

Um diagrama de actividades pode ainda ser utilizado na descrição de um fluxo de actividades mais alargado, envolvendo diversos *use cases.* No domínio da gestão das organizações, constitui aquilo que se pode designar por processo de negócio inter-funcional.

Uma outra característica interessante do diagrama de actividades é a capacidade de descrever conjuntos de actividades que se desenvolvem em paralelo. Esta capacidade pode ser utilizada, por exemplo, quando se descreve um projecto de desenvolvimento de software, no qual algumas das actividades podem ser realizadas em simultâneo por diversos actores.

No domínio das aplicações informáticas, um diagrama de actividades pode ser utilizado para descrever fluxos de controlo do

© FCA - EDITORA DE INFORMÁTICA

programa. Face aos fluxogramas tradicionais, o diagrama de actividades apresenta a vantagem de permitir descrever com rigor fluxos de processamento de actividades em paralelo, bem como de atribuir a uma classe responsabilidade pela execução de uma actividade.

Cada um dos diagramas propostos pela UML permite modelar um aspecto específico do sistema e deve ser utilizado em complemento com os outros diagramas. Esta nota serve para realçar utilizações por vezes indevidas do diagrama de actividade.

Assim, um diagrama de actividades não deve ser utilizado para demonstrar colaboração entre objectos. Neste caso um diagrama de sequência ou de colaboração é mais apropriado. Um diagrama de actividades também não deve ser utilizado para descrever o comportamento de um objecto ao longo do tempo, o que deve ser feito através de um diagrama de estado.

Estas utilizações indevidas são por vezes frequentes naqueles que possuem grande experiência em análise estruturada e que utilizam o diagrama de actividades para modelar decomposição funcional, sem o enquadrarem nos princípios da modelação orientada por objectos.

A utilização dos diagramas de actividades no contexto do paradigma dos objectos requer uma certa disciplina. Um dos princípios fundamentais dos objectos é a integração de atributos e comportamento. A utilização correcta de um diagrama de actividades exige que se identifique qual o objecto responsável pela realização de cada uma das actividades. Tal pode ser possível identificando junto de cada uma das actividades qual o objecto responsável. Uma forma alternativa de identificação de responsabilidade é a utilização de linhas verticais ("*swimlanes*") que enquadram as actividades que ficam associadas a cada objecto.

© FCA - EDITORA DE INFORMÁTICA

Considerando o exemplo da PhonePizza que temos vindo a utilizar pensemos no *use case* "Processar Encomenda na Pizzaria":

> "O cliente dirige-se ao balcão e pede ao funcionário um conjunto de produtos que pretende. O funcionário vai tomando nota do pedido, verificando se o produto está na lista de produtos comercializados e se existe em stock. No caso do produto não existir, informa o cliente. Se for detectada uma rotura de stock, é enviada uma mensagem ao Gestor de Loja para encomendar o produto em falta e o funcionário sugere um produto alternativo. Se o produto solicitado não pertencer à lista dos que são vendidos na pizzaria, o funcionário sugere igualmente um produto alternativo.
>
> Após o cliente ter concluído a sua encomenda, é determinado o valor da encomenda e solicitado o pagamento. Se o pagamento for válido, a encomenda é entregue ao cliente. Caso contrário, a encomenda é cancelada."

O diagrama de actividades que se apresenta na figura 4.1 descreve este *use case*. Neste diagrama são utilizados os seguintes elementos de modelação:

- Linhas verticais de responsabilidade
- Actividades de Início e de Fim
- Actividade intermédia
- Transição de actividade e símbolos de comportamento condicional

Cada um destes elementos é explicado detalhadamente nos pontos seguintes.

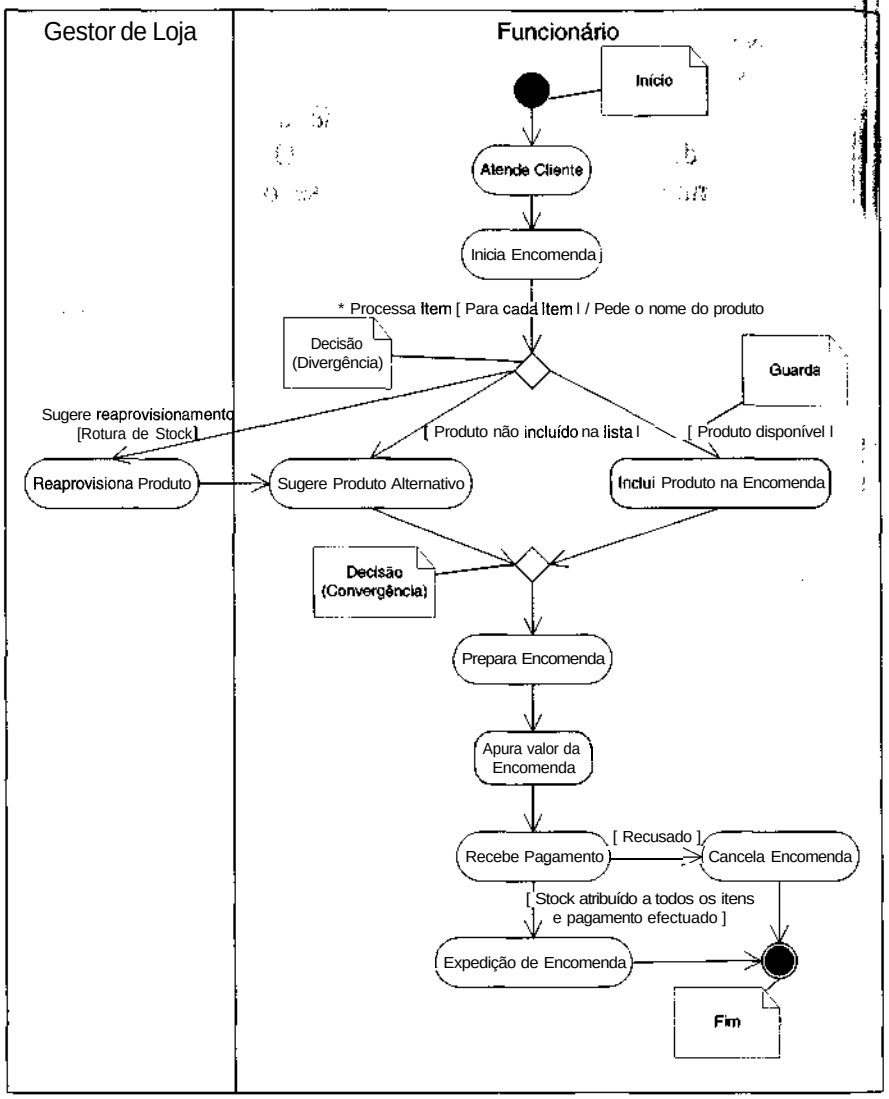


FIGURA 4.1 - EXEMPLO DIAGRAMA DE ACTIVIDADES

© FCA - EDITORA DE INFORMÁTICA

4.1.1 Linhas verticais de responsabilização

Através da utilização das linhas verticais de responsabilidade, é possível descrever quais são os objectos responsáveis por cada uma das actividades. No diagrama da fig. 4.1 podemos identificar que o Gestor de Loja é responsável pelo reaprovisionamento e o Funcionário pelas restantes actividades representadas no diagrama.

4.1.2 Actividades

Num diagrama de actividades é necessário identificar a **actividade inicial.** Esta actividade pode ser puramente virtual, definida para identificar o início do diagrama, ou corresponder a uma actividade operacional do sistema. Uma actividade inicial é descrita por um círculo preenchido a negro.

Uma **actividade operacional** é descrita graficamente por um rectângulo de lados arredondados com um identificador. Uma **actividade** permite descrever um conjunto de acções, que são realizadas quando a actividade se inicia, durante o seu decurso normal, e quando termina. Numa actividade podemos ainda descrever a ocorrência de eventos excepcionais.

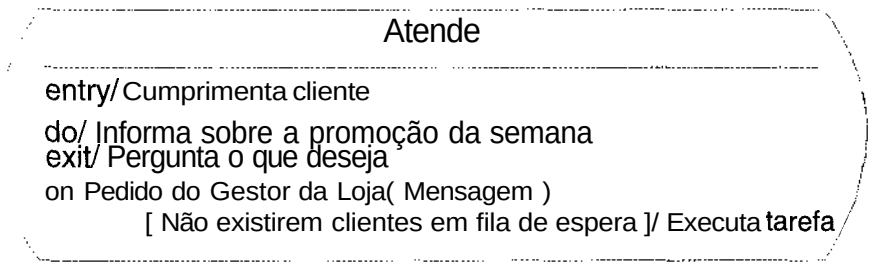


FIGURA 4,2 - ACTIVIDADE

Na figura 4.2 identificam-se as acções que são realizadas na actividade Atende Cliente, através da utilização dos descritores "entry/", "do/" e "exit/". Neste caso, o funcionário começa por saudar o cliente, em seguida informa-o sobre a promoção da semana e, por fim, pergunta-lhe o que deseja.

No decurso de uma actividade pode ser realizada uma acção em resposta a um evento exterior. Por exemplo, o Gestor da Pizzaria pode transmitir uma mensagem ao Funcionário para realizar uma determinada tarefa, desde que não existam clientes em fila de espera. Esse evento encontra-se descrito na figura associado ao descritor "on event/":

on Pedido do Gestor da Loja(Mensagem) [Não existem clientes em fila de espera]/Executa tarefa

Para identificar uma actividade terminal de um fluxo de trabalho utiliza-se um círculo a preto, limitado com uma circunferência. Num diagrama de actividades só existe uma actividade inicial, mas pode existir mais do que uma actividade terminal.

4.1.3 Transição entre actividades

Uma **transição** permite descrever a sequência pela qual as actividades se realizam (fig. 4.3):

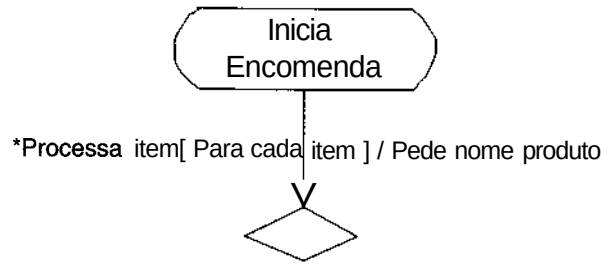


FIGURA 4,3 - TRANSIÇÃO

A transição entre actividades é representada por uma seta. Na transição podem ainda ser listados os eventos, acções e condições, com a seguinte sintaxe:

Evento (argumentos) [condição] / Acção ^alvo.algumEvento (args)

A transição Processa Item repete-se para cada um dos produtos que o cliente pretende adquirir. Esta funcionalidade designa-se por **concorrência dinâmica** e permite representar as iterações através do símbolo * , que aparece junto do identificador da transição, sem ter de construir um ciclo.

4.1.4 Comportamento condicional

Num fluxo de actividades podem existir caminhos alternativos. Para representar o fluxo de controlo num diagrama de actividades utilizam-se "guardas" e diamantes de decisão.

Guardas são expressões booleanas limitadas por parêntesis rectos [], que têm de ser verificadas para se realizar a transição para uma nova actividade.

Nos diagramas de actividade podem igualmente ser utilizados símbolos, em forma de diamante, para representar caminhos alternativos baseados numa expressão booleana (condição). Os **diamantes de decisão,** que são semanticamente equivalentes a múltiplas transições com guarda, devem ser utilizados para aumentar a legibilidade do diagrama. Estes símbolos podem ser utilizados para descrever uma **divergência** (*branch*) ou uma **convergência** (*merge*)no fluxo de controlo.

Um diamante de decisão que representa uma divergência no fluxo de controlo, possui uma transição de entrada e duas ou mais transições de saída. Um diamante que representa uma convergência possui uma ou várias transições de entrada e uma transição de saída. Os diamantes de decisão devem ser utilizados de forma

equilibrada. Se for utilizado um símbolo para representar um ponto de divergência, deverá existir um outro símbolo para representar a convergência no fluxo de controlo das actividades.

No diagrama da figura 4.4 utilizam-se guardas e símbolos de decisão para controlar quais as actividades que devem ocorrer, face à existência ou indisponibilidade do produto para satisfazer a encomenda.

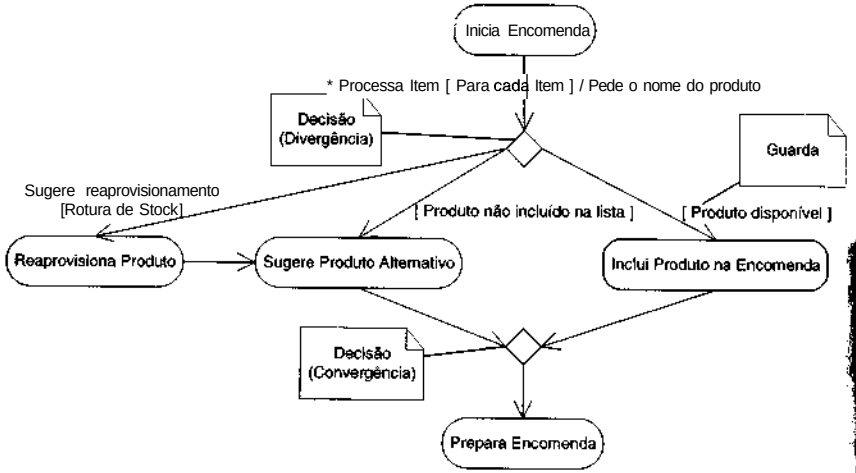


FIGURA 4,4 - DECISÃO E GUARDA

4.2 TÓPICOS AVANÇADOS

No âmbito dos diagramas de actividades poderemos identificar alguns aspectos que constituem tópicos avançados, designadamente:

- Agrupamento e decomposição de actividades
- Processamento paralelo
- Fluxo de objectos no diagrama de actividades

4.2.1 Agrupamento e decomposição de actividades

A UML permite que um conjunto de subactividades possam ser agrupadas numa superactividade ou que uma actividade possa ser decomposta num conjunto de subactividades.

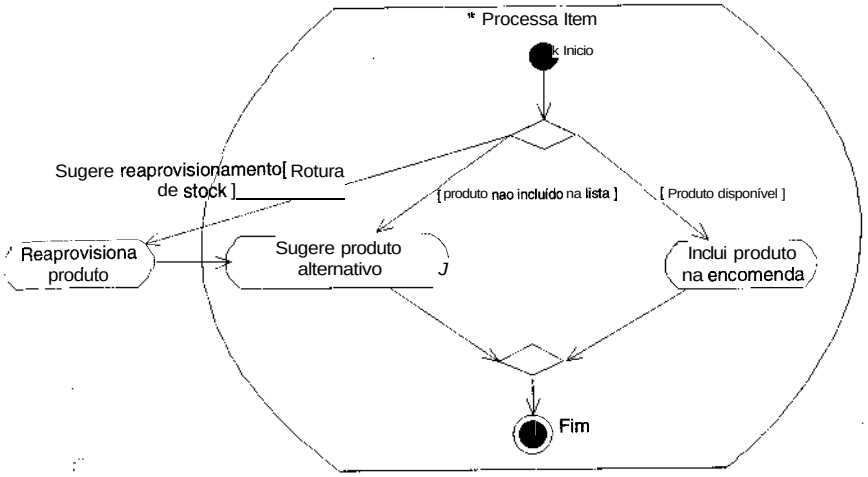


FIGURA 4.5 - ACTIVIDADES AGRUPADAS

A figura 4.5 apresenta um exemplo em que se optou por agrupar as actividades Sugere Produto Alternativo e Inclui Produto na Encomenda numa actividade designada por Processa Item. Nesta situação podemos representar no diagrama geral as **subactividades** incluídas na **superactividade** Processa Item, ou criar um diagrama específico para representar o detalhe da actividade Processa Item.

O identificador da actividade Processa Item vem antecedido de um * pelo facto de se repetir para cada um dos itens da encomenda. No diagrama que representa as subactividades, optamos por representar explicitamente uma actividade inicial e uma actividade final, permitindo, deste modo, que a actividade

Processa Item possa ser utilizada autonomamente noutros diagramas.

4.2.2 Processamento paralelo

Um aspecto relevante na capacidade de modelação dos diagramas de actividade é a possibilidade de representar fluxos de actividades que se desenvolvem em paralelo. Esta possibilidade é particularmente útil na descrição de processos organizacionais, porque ajuda a identificar oportunidades para aumentar a eficiência do processo, através da realização de actividades em paralelo.

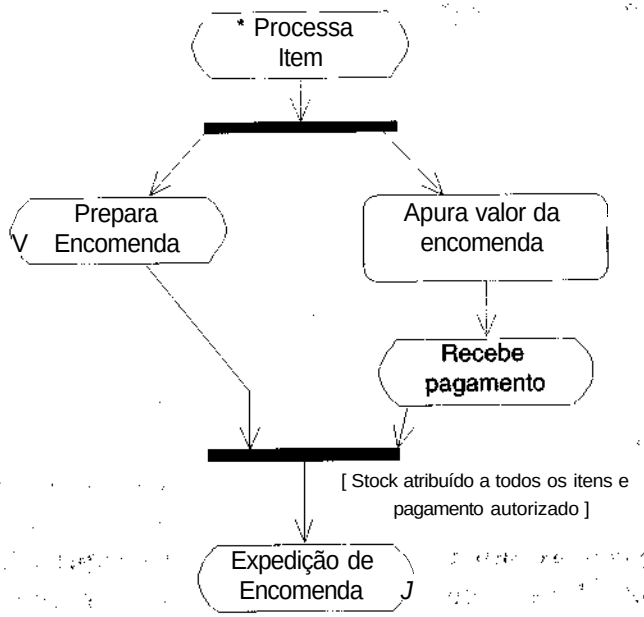


FIGURA 4,6 - PROCESSAMENTO PARALELO DE

Para descrever processamento paralelo são utilizadas barras horizontais. Estas podem assumir dois papéis: marcar um **ponto de divergência** (*fork*), a partir do qual duas ou mais tarefas se podem

© FCA - EDITORA DE INFORMÁTICA

iniciar em paralelo, ou permitir sincronizar (*join*) tarefas que têm de estar concluídas para que se inicie uma nova tarefa **(ponto de convergência)** . Num diagrama de actividades uma barra de divergência deve ser compensada com uma barra de convergência.

Na figura 4.6 representa-se o facto de se poder efectuar a preparação da encomenda em paralelo com o apuramento do valor e a concretização do pagamento.

Conceptualmente, esta representação descreve o facto de o conjunto de actividades pertencentes a cada um dos braços do fluxo poderem ser executadas em simultâneo, ou sequencialmente, ainda que não exista uma ordem estabelecida. Assim, poder-se-á proceder à preparação da encomenda e, posteriormente, ao apuramento do valor e pagamento ou vice-versa.

4.2.3 Fluxo de objectos

A intervenção de objectos para a realização das actividades pode ser representada colocando estes objectos nos diagramas e ligando--os à actividade através do símbolo de dependência. Na figura 4.7 exemplifica-se esta situação tomando como referência a actividade Prepara Encomenda que necessita do objecto e:Encomenda.

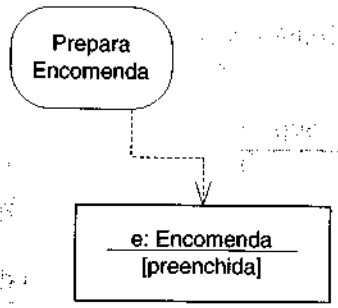


FIGURA 4.7 - FLUXO DE OBJECTOS NO FLUXO DE CONTROLO

4.2.4 Diagrama de actividades revisto

Em seguida, é apresentada, na figura 4.8, uma actualização do diagrama de actividades para o *use case* "Processar Encomenda na Pizzaria", utilizando os conceitos mais avançados.

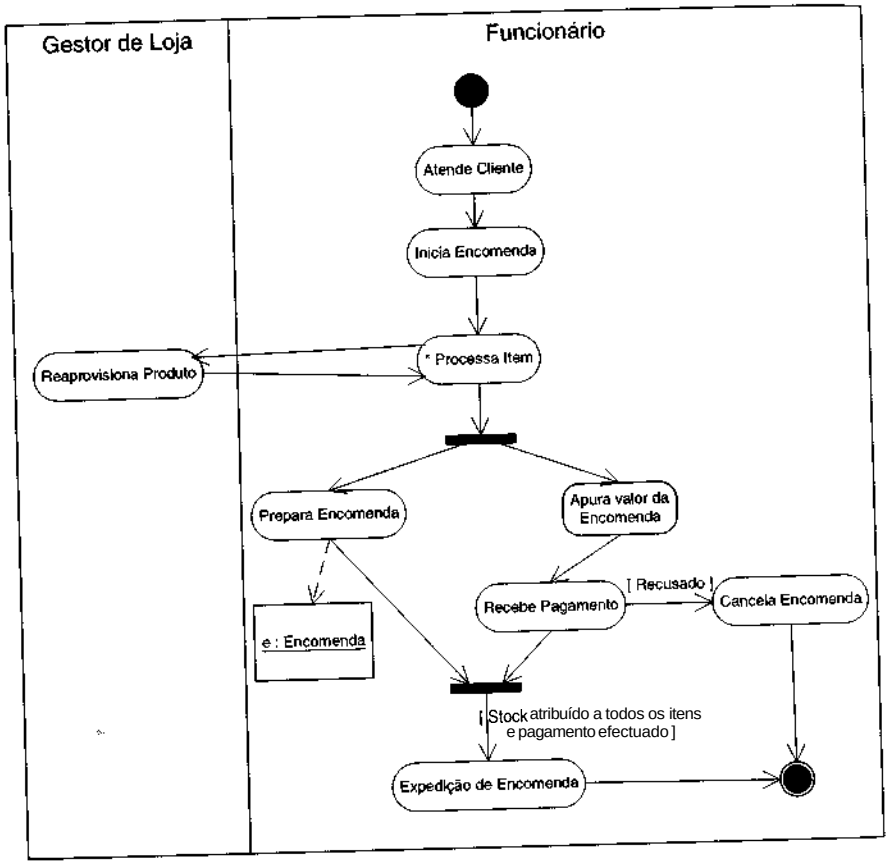


FIGURA 4,8 - EXEMPLO DIAGRAMA ACTIVIDADES AVANÇADO

A principal diferença entre este diagrama e o diagrama apresentado inicialmente (fig. 4.1) está no agrupamento de actividades na actividade `Processa Item`, utilização do processamento

paralelo e introdução do objecto `e:Encomenda` **na actividade** `prepara Encomenda`.

> Deve-se sempre começar com um diagrama de actividades simples para ter uma ideia geral do processo. Eventualmente, processos mais complexos podem ser repartidos por vários diagramas.

4.3 EXERCÍCIOS

4.3.1 Perguntas de Revisão

1: Qual a finalidade de um diagrama de actividades?

2: Identifique duas características distintivas dos diagramas de actividades como instrumentos de modelação.

3: Quais os elementos de modelação que são utilizados num diagrama de actividades?

4: Como se descreve num diagrama de actividades o princípio da responsabilidade de um objecto pela realização das operações?

5: Qual a notação utilizada para caracterizar uma actividade inicial?

6: Caracterize os diversos elementos que são utilizados para descrever uma actividade operacional.

7: Quantas actividades iniciais e terminais poderemos colocar num diagrama?

8: Que elementos podem ser listados numa transição entre actividades?

© FCA - EDITORA DE INFORMÁTICA

9: Que elementos de modelação podem ser utilizados num diagrama de actividade para descrever comportamento condicional?

10: É possível efectuar o agrupamento e a decomposição de actividades? Exemplifique de que forma o faria?

11: Que elementos são utilizados para modelar a execução de actividades em paralelo?

12: Como se representa a intervenção de objectos para a realização das actividades?

4.3.2 Problemas Resolvidos

Desenhe os respectivos diagramas de actividades de acordo com as descrições seguintes.

Biblioteca

Considere os seguintes requisitos para o processo de disponibilização de obras que foram definidos pelo responsável da biblioteca.

Os leitores, professores ou alunos, interessados na consulta de uma obra não disponível na Biblioteca podem apresentar uma sugestão de aquisição ao responsável.

Regularmente as listas com as publicações sugeridas são enviadas para os fornecedores com um pedido de proposta de fornecimento, que deve incluir prazo de entrega e preço.

As propostas dos fornecedores são analisadas e, em função dos preços e do orçamento disponível, serão seleccionadas as obras a adquirir. A biblioteca estabeleceu critérios que dão prioridade à aquisição de obras formativas, que façam parte da bibliografia das disciplinas do sistema de ensino. Após ter sido definida a lista de obras a adquirir, são enviadas notas de encomenda para os fornecedores seleccionados. As obras entregues pelos fornecedores

© FCA - EDITORA DE INFORMÁTICA

são verificadas no momento da recepção, sendo confrontadas as guias de remessa com as notas de encomenda, de modo a assegurar a consistência com a encomenda efectuada.

Após a catalogação e registo de cada obra no sistema de informação de gestão da biblioteca, é enviada uma notificação aos leitores que propuseram a sua aquisição. As novas obras são colocadas num expositor especial de divulgação, durante um período de 2 semanas, antes de serem arrumadas na respectiva prateleira. A partir desse momento a obra fica disponível para ser emprestada.

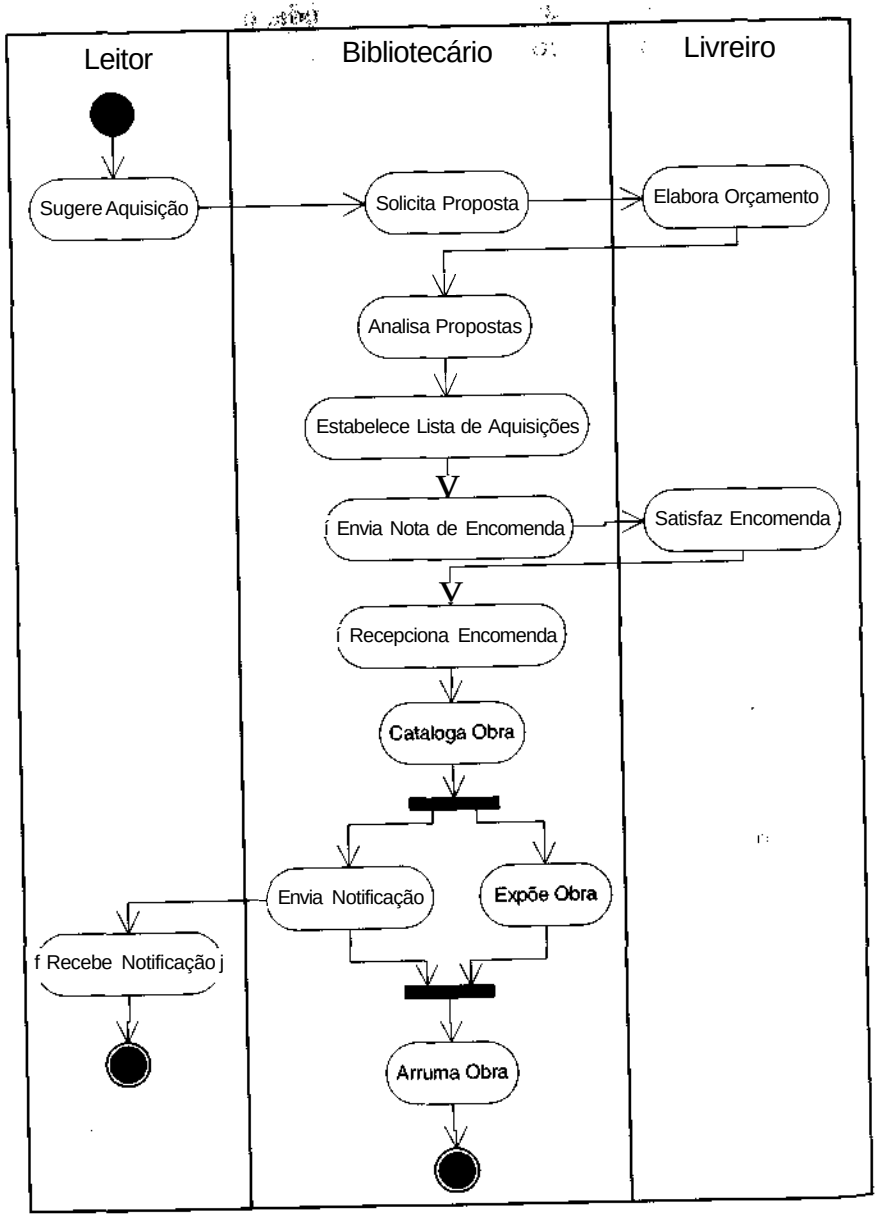
Parque de Estacionamento

Uma análise do controlo de entrada de viaturas no parque sugeriu algumas melhorias que poderiam ser introduzidas, no sentido de tornar o processo mais eficiente.

Um objectivo é simplificar a tarefa do funcionário que tem a responsabilidade de registar a matrícula dos veículos. A adopção de uma tecnologia de reconhecimento de imagens permite que seja o sistema de informação a visualizar, reconhecer e registar a matrícula de fornia automática. Assim, o processo passa a incluir as seguintes actividades:

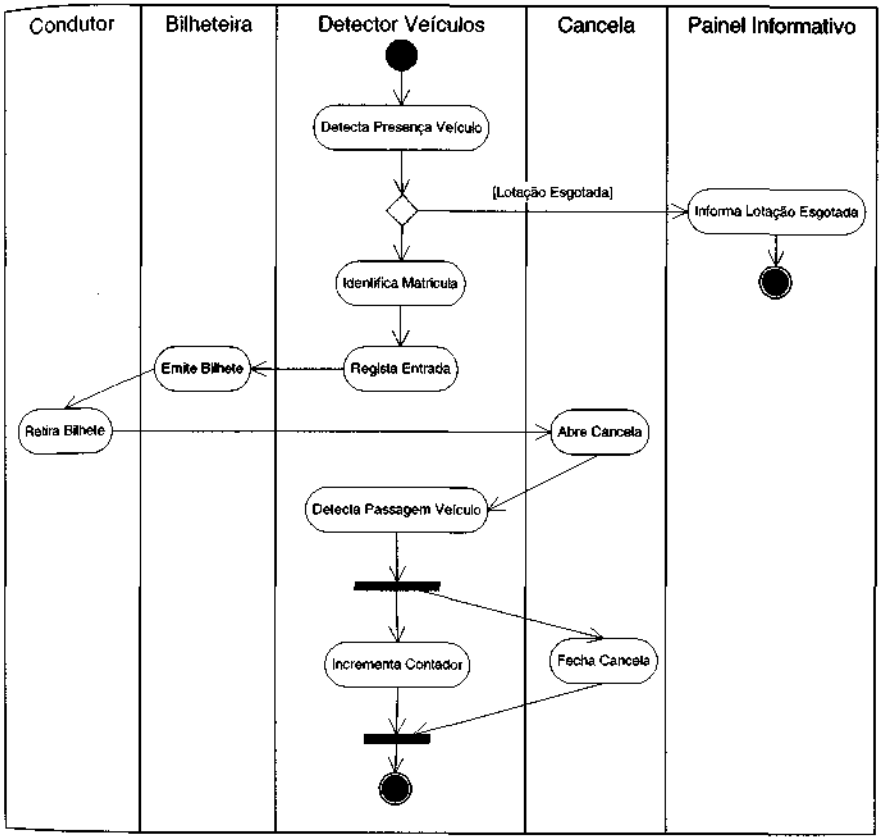
O sistema de informação detecta a presença do veículo junto à cancela de entrada. Se existir lugar vago no parque, identifica a matrícula do veículo, regista a entrada e emite o bilhete. Quando o condutor retira o bilhete, o sistema de informação abre a cancela e quando detecta a passagem do veículo, incrementa o contador de lotação e fecha a cancela.

Solução Biblioteca



© FCA - EDITORA DE INFORMÁTICA

Solução Parque de Estacionamento



Diagramas de Interacção 5

5,1 E APLICAÇÃO

Modelar a dinâmica de um sistema é fundamental para dominar a sua complexidade e compreender as suas particularidades. Os diagramas de interacção são utilizados na UML para modelar os aspectos dinâmicos do sistema em termos dos objectos e suas interacções, tendo como base as mensagens trocadas entre objectos. Booch et al (1999) definem uma **interacção** como um comportamento que consiste na troca de um conjunto de mensagens entre objectos, dentro de um contexto, para atingir um objectivo.

Os diagramas de interacção permitem definir e clarificar a colaboração entre as classes do sistema. Normalmente, são utilizados para ilustrar o comportamento do sistema num cenário de concretização de um *use case.*

É frequente utilizar diagramas de interacção em conjunto com a descrição textual dos *use cases,* pois facilitam a sua compreensão ao fornecer uma representação gráfica das interacções entre os objectos.

Diagrama de interacção é uma designação genérica que na UML se aplica a diagrama de sequência ou diagrama de colaboração. Um diagrama de sequência apresenta as interacções entre objectos a Partir do encadeamento temporal das mensagens. Um diagrama de colaboração descreve as mesmas interacções mas centradas nos objectos intervenientes. Estes diagramas podem ser desenhados com vários níveis de detalhe e ao longo das diversas etapas do Processo de desenvolvimento do sistema.

Um diagrama de interacção é composto pelos seguintes elementos abstractos de modelação:

« Objectos
* Ligações (*links*)
* Mensagens

Por exemplo, vejamos o seguinte descrição para o caso PhonePizza:

> "Para que o cibernauta possa efectuar encomendas através da Internet este terá que efectuar um pré-registo onde indicará o seu nome, morada, número de telefone, *username* e *password*. O pré-registo será confirmado através de um código de acesso que será enviado por correio electrónico. O código será utilizado uma única vez pelo cliente para activar os serviços de encomenda pela Internet."

Esta descrição caracteriza o *use case* "Pré-registo pela Internet". Um cenário de registo típico será utilizado para apresentar os diagramas de interacção.

Cada um dos diagramas é explicado detalhadamente em seguida.

5,2 DE SEQUÊNCIA

Na figura 5.1 é apresentado o diagrama de sequência para o *use case* "Pré-registo pela Internet".

O diagrama de sequência é um diagrama de interacção que realça a ordem cronológica das mensagens entre objectos.

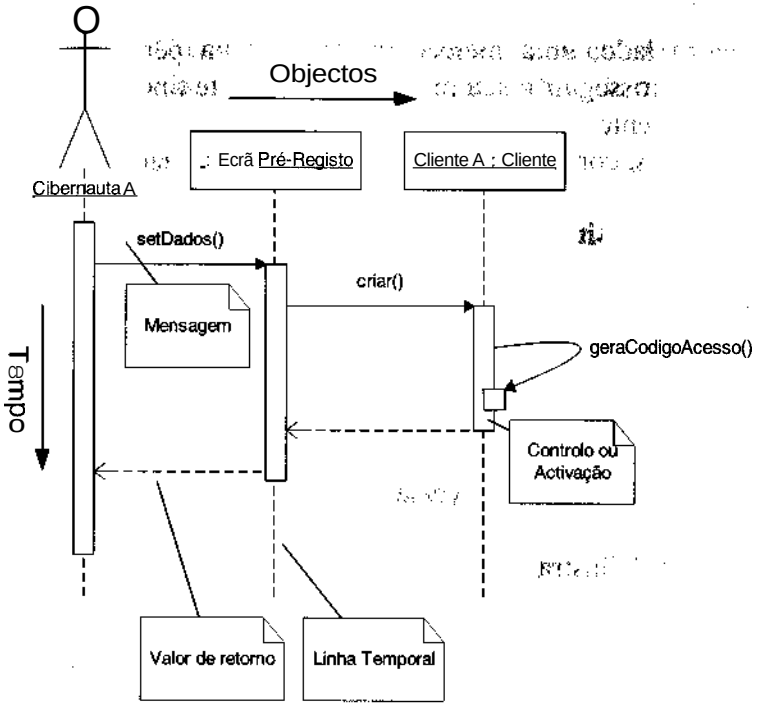


FIGURA 5,1 — DIAGRAMA DE SEQUÊNCIA: PRÉ-REGISTO

5.2.1 Mensagens

As **mensagens** trocadas entre objectos representam a invocação de um serviço (operação) disponibilizado por um objecto, com o objectivo de despoletar uma acção ou actividade. Uma definição mais formal descreve uma mensagem como a especificação da comunicação entre objectos.

O tipo de mensagens pode ser síncrono, assíncrono, simples ou de retorno. Uma **mensagem síncrona** significa que o objecto emissor fica suspenso à espera de uma resposta, retomando posteriormente o controlo. Utiliza-se esta mensagem quando o objecto emissor necessita de dados provenientes do objecto receptor, para continuar o seu processamento.

© FCA - EDITORA DE INFORMÁTICA

Por outro lado, uma **mensagem assíncrona** permite à operação emissora prosseguir o seu processamento. Este tipo de mensagens é particularmente útil para ilustrar sistemas com processos concorrentes, como *é* o caso do exemplo apresentado na fig. 5.4.

A mensagem simples utiliza-se quando ainda não está definido o tipo da mensagem ou este tipo não *é* relevante.

Por **fim, a mensagem de retorno** *é* utilizada para ilustrar o retorno da mensagem enviada que poderá ser um valor ou um sinal. Para mensagens simples ou síncronas está implícita a existência de um retorno, sendo a sua representação opcional. No entanto, para mensagens assíncronas deve-se representar a mensagem de retorno.

A figura 5.2 ilustra a notação gráfica para os diferentes tipos de mensagens.

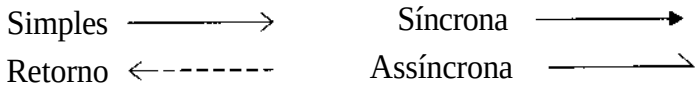


FIGURA 5,2 — TIPOS DE MENSAGENS

As mensagens podem despoletar vários tipos de acção no objecto receptor. A UML define os seguintes tipos de acção para as mensagens:

- ***Call*** - Invoca uma operação de um objecto. Este tipo de mensagem pode ser enviada ao próprio objecto.
- ***Return*** - Retorna um valor para o objecto emissor para mensagens síncronas ou um sinal para mensagens assíncronas.
- *Send* - Envia um sinal a um objecto.
- ***Create*** - Cria um objecto.
- ***Destroy*** - Destrói um objecto;

O tipo de acção mais frequente *é* o *Call*, utilizado em mensagens síncronas. O tipo *Send é* utilizado em mensagens assíncronas.

Apenas os tipos *Create* e *Destroy* são explicitamente ilustrados nas mensagens, todas as outras estão implícitas ao tipo de mensagem.

5.2.2 Linha temporal e controlo

No diagrama de sequência existe uma **linha temporal** que acompanha o ciclo de vida dos objectos, onde também pode ser representado o intervalo de tempo. Neste intervalo, o objecto tem o **controlo** para processamento e envio de mensagens. A figura 5.3 realça estes conceitos ilustrando um excerto do exemplo apresentado inicialmente na figura 5.1.

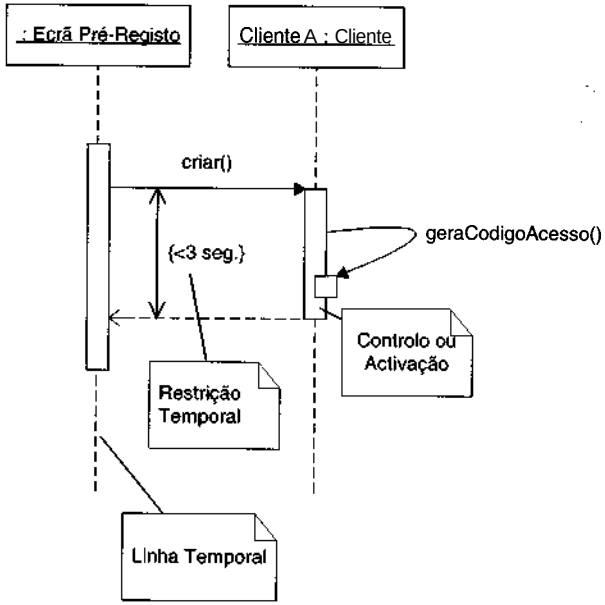


FIGURA 5.3 - LINHA TEMPORAL E CONTROLO

Sempre que um objecto envia uma mensagem, tem que possuir o controlo. Este poderá estar sempre presente ou existir apenas quando ocorre algum processamento no objecto. É possível impor urna **restrição temporal** entre o envio de uma mensagem e a respectiva resposta. Neste caso, não pode ultrapassar os *3* segundos.

No caso da fig. 5.3, o objecto de interface Ecrã `Pré-Registo` envia uma mensagem para o objecto `Cliente`, para que um novo cliente seja criado. O objecto `Cliente`, por sua vez, envia uma mensagem a si próprio para que seja gerado um código de acesso (repare na **linha dupla de controlo).** Findo este processamento, é enviado para o objecto de interface uma mensagem de retorno.

Na figura 5.4, o diagrama de sequência é utilizado para ilustrar o cenário em que um cliente cria uma encomenda, adiciona itens e confirma a encomenda. Neste caso, um objecto `Cliente` cria uma nova encomenda, usando a mensagem com o estereótipo `«create»`.

O processo de adicionar produtos à encomenda é repetido para cada produto a adicionar, por isso é utilizado um asterisco (*) como **símbolo de iteração.** Este processo requer o envio de uma mensagem `adicionaProduto()` ao objecto Encomenda, que por sua vez irá adicionar o produto à encomenda após a consulta do respectivo preço através da mensagem `devolvePreço()`.

Por fim, o cliente confirma a encomenda enviando a mensagem `confirmaEncomenda()` ao objecto Encomenda. Esta mensagem apenas será enviada se o número de itens/produtos adicionados à encomenda for superior a zero.

Por vezes, é necessário introduzir uma **condição** no diagrama. Para tal, especifica-se a condição entre parêntesis rectos perto da linha de controlo como, por exemplo, na figura 5.4 é utilizada a condição `[número itens>0]`, que significa que uma encomenda será registada apenas se contiver pelo menos um item/produto. De forma a demonstrar que um conjunto de mensagens são enviadas de acordo com uma condição, pode-se utilizar uma separação na linha de controlo. A figura 5.5 ilustra esta alternativa.

> Não se deve representar muitas condições no diagrama de sequência, pois corre-se o risco de o complicar.

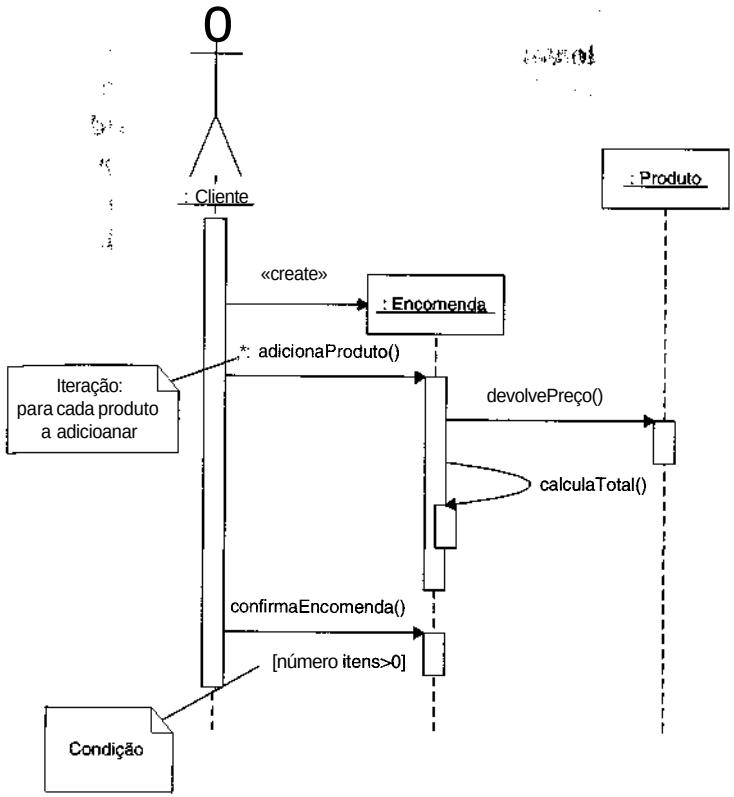


FIGURA 5,4 - DIAGRAMA DE SEQUÊNCIA: EFECTUAR ENCOMENDA

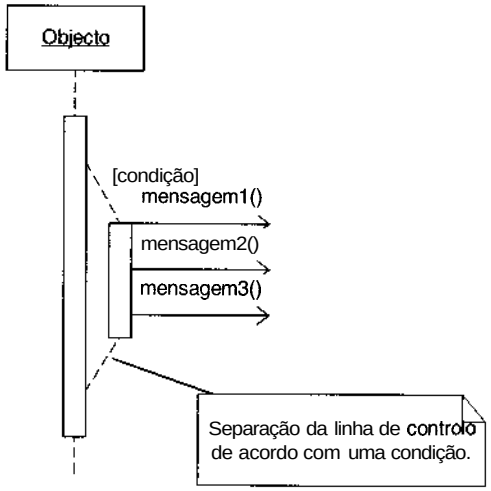


FIGURA 5.5 - SEPARAÇÃO DA LINHA DE CONTROLO

© FCA - EDITORA DE INFORMÁTICA

5.2.3 Processamento em paralelo

A utilização de mensagens assíncronas permite efectuar diagramas de sequência onde são enviadas mensagens para objectos distintos que estarão a correr em paralelo.

Por exemplo, no diagrama de sequência para o controlo de acessos (figura 5.6), o objecto `Controlo Acesso`, ao receber uma mensagem `verificaAcesso()`, cria dois outros objectos que irão correr em paralelo para verificar as permissões do utilizador nos objectos e operações do sistema. A mensagem que pede a verificação (`verifica()`) é assíncrona, sendo assim a operação emissora pode continuar o seu processamento.

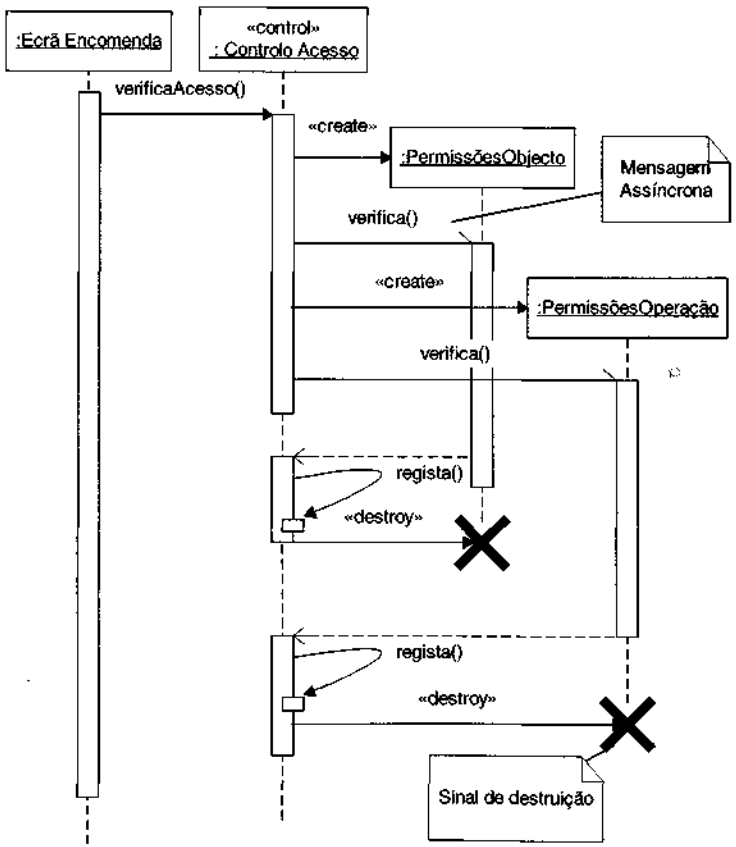


FIGURA 5,6 - DIAGRAMA DE SEQUÊNCIA: CONTROLO DE ACESSO

À medida que cada verificação de permissões vai terminando é enviado um sinal ao objecto `Controlo Acesso`, que regista a seu resultado individual e remove o respectivo objecto de permissões, sendo representado por um **sinal de destruição** (X), cuja representação é opcional (um objecto pode autodestruir-se). Quando o último objecto de permissões termina, o resultado global da verificação de acesso é retornado.

5.2.4 Interface com o utilizador

Nem todos os exemplos de diagramas de sequência que foram apresentados possuem um objecto de interface que faça de intermediário entre o utilizador e o sistema. De facto a sua representação é opcional, mas ao seguir um cenário de um *use case* está implícita a existência de uma interface. Estes objectos de interface de utilizador também devem ser identificados, especificados e representados. Contudo, numa actividade de análise apenas devemos estar preocupados em identificar a natureza da interface, em particular como é que um actor acede à funcionalidade e que informação necessita. A especificação detalhada da interface com o utilizador é efectuada, posteriormente, numa actividade tipicamente de desenho. O Capítulo 7 aborda este tema.

5.3 DE COLABORAÇÃO

Na figura 5.7 é apresentado o diagrama de colaboração que descreve o cenário de realização do *use case* introduzido no início do capítulo. O diagrama de colaboração realça a organização estrutural dos objectos que enviam e recebem mensagens. Possui uma equivalência semântica com o diagrama de sequência, no entanto representa a informação de uma forma diferente. Enquanto que o diagrama de sequência está rigidamente ligado à variável tempo, o diagrama de colaboração apenas demonstra a interacção entre os objectos.

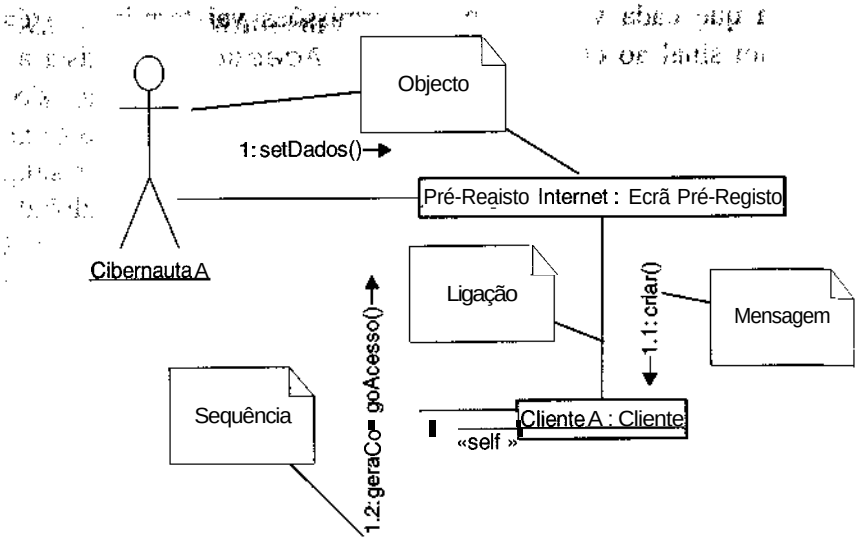


FIGURA 5,7 - DIAGRAMA DE COLABORAÇÃO: PRÉ-REGISTO

Os objectos podem possuir diferentes papéis *(roles)* nos vários cenários de colaboração. Aqui o conceito de **papel** significa que o nível da abstracção do diagrama pode ir de objectos específicos a objectos mais abstractos. Por exemplo, representar Cliente A como uma instância da classe Cliente, pode significar que Cliente A corresponde a um cliente real. No entanto, Cliente A num sentido mais abstracto poderá representar qualquer instância da classe Cliente.

5.3.1 Ordenação numérica

No diagrama de colaboração as mensagens trocadas entre os objectos são ordenadas numericamente, começando no 1 para a primeira mensagem e progredindo sequencialmente. Para representar agrupamentos de mensagens, utiliza-se a escala decimal onde, por exemplo, a primeira mensagem de um subgrupo de mensagens será a `1.1`. A profundidade não é limitada, permitindo assim representar casos mais complexos.

5,3.2 Repetições

Caso as mensagens requeiram repetições, como no exemplo do adicionar itens à encomenda, existe a possibilidade de adicionar um prefixo de repetição ao número da mensagem. Este prefixo pode ser simplesmente * para repetições não quantificadas ou um intervalo tipo `[i := 1..n]`, onde n será o limite superior.

5.3.3 Estereótipos

As ligações entre os objectos podem possuir estereótipos nos seus extremos que as classificam. No exemplo da figura 5.8, existe uma ligação no objecto Encomenda cuja extremidade está classificada com o estereótipo «`self`», que significa que é uma ligação ao próprio objecto, permitindo assim que o objecto envie mensagens a si mesmo. Outras classificações típicas são a «local», «global», que indicam se o objecto receptor é local ou não ao objecto emissor.

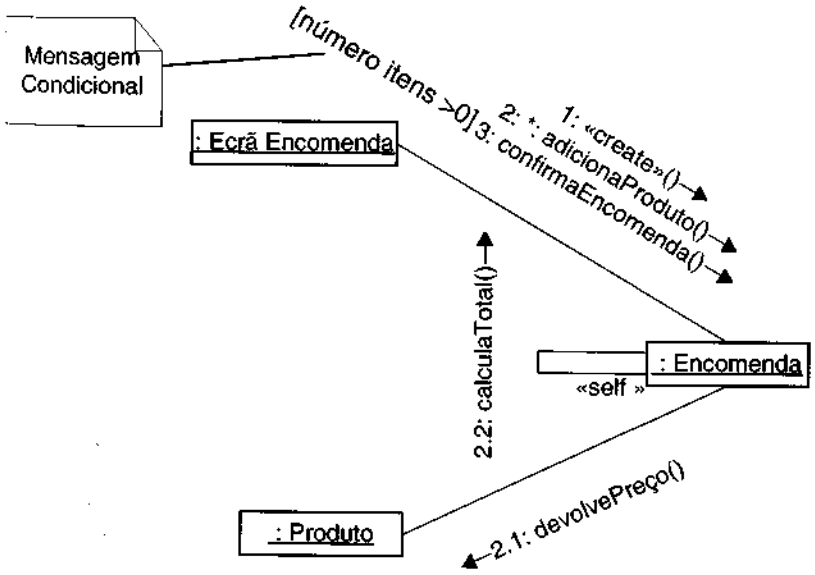


FIGURA 5.8 - DIAGRAMA DE COLABORAÇÃO: EFECTUAR ENCOMENDA

© FCA - EDITORA DE INFORMÁTICA

A figura 5.8 mostra o diagrama de colaboração para o *use c* "Efectuar Encomenda" referido anteriormente. Na prati existe uma equivalência directa com o diagrama de sequên (figura 5.4).

5.3.4 Mensagens condicionais

Estas mensagens só são enviadas quando uma determinada condição é validada. Esta condição é ilustrada entre parêntesis rectos `[]`. Por exemplo, na figura 5.8 a mensagem 3: `confirmaEncomenda()` só será enviada se a condição `[númeroItens>0]` for válida.

5.3.5 Sincronização

Para sincronizar processos concorrentes utiliza-se um prefixo (/) de sincronização que estabelece as mensagens que devem ser concluídas antes do envio da uma mensagem. Este mecanismo é demonstrado na figura 5.9.

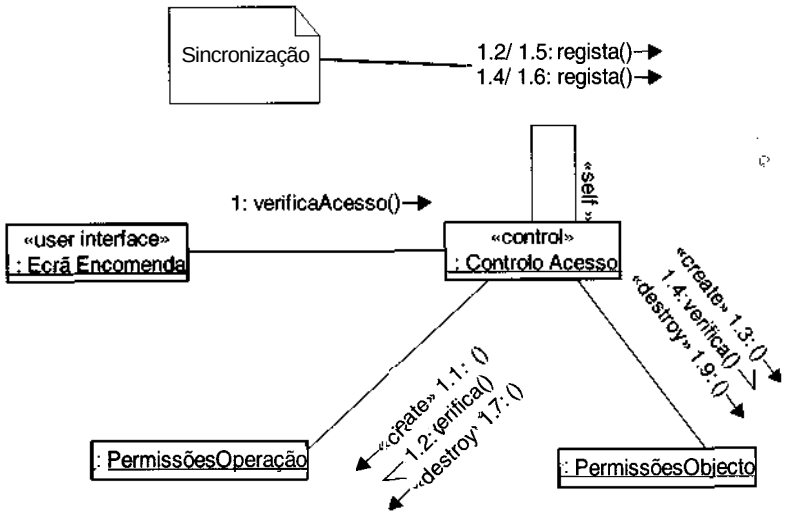


FIGURA 5.9 - DIAGRAMA DE COLABORAÇÃO: CONTROLO DE ACESSOS

Neste diagrama de colaboração a mensagem `1.5: regista()` só é enviada quando a `1.2: verifica()` terminar.

5.3.6 Objectos e ligações

O diagrama de colaboração também representa explicitamente a ligação entre os objectos que deriva das associações no diagrama de classes.

No diagrama de classes as associações representam a relação entre as diversas classes. Quando se concretizam as classes em objectos, a associação passa a ser representada por uma **ligação** que representa uma instância da associação.

Quando existe uma ligação entre dois objectos estes podem trocar mensagens entre si. A figura 5.10 exemplifica esta equivalência:

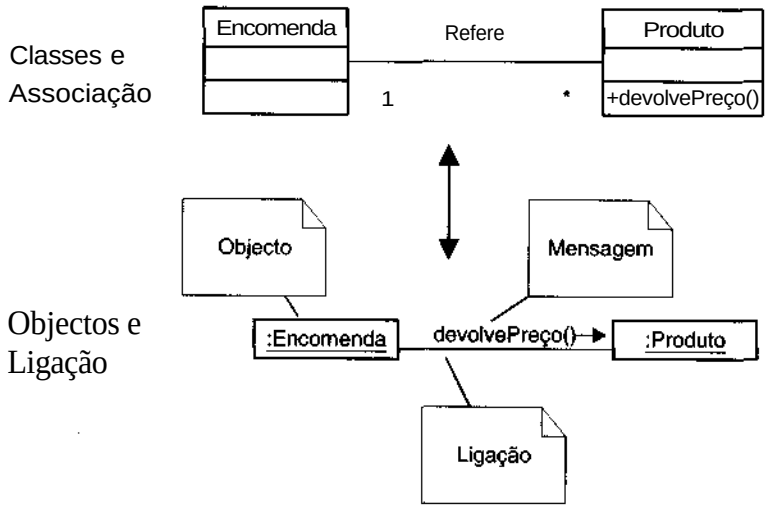


FIGURA 5.10 - TRANSIÇÃO PARA OBJECTOS E LIGAÇÕES

© FCA - EDITORA DE INFORMÁTICA

Nesta figura é efectuada a transposição do diagrama de classes num cenário de colaboração entre objectos, onde um objecto da classe Encomenda envia uma mensagem a um objecto da classe Produto.

5,4 CONSTRUÇÃO DE DIAGRAMAS DE INTERACÇÃO

O cenário principal de um *use case é* um bom ponto de partida para a construção de um diagrama de interacção. Deve-se estabelecer os objectos que participam com os seus serviços para alcançar a funcionalidade desejada no cenário.

Deve-se também tentar manter consistente a responsabilidade individual de cada objecto, aceitando apenas mensagens (fornecer serviços) que estejam directamente relacionadas com o seu âmbito.

Recomenda-se a utilização de uma ferramenta CASE que faça uma integração automática dos diagramas, permitindo associar uma mensagem para um objecto a uma operação que este possui, ou criar uma nova. Caso seja criada uma nova operação, esta é automaticamente reflectida na respectiva classe do objecto e, em consequência, na sua representação no diagrama de classes. Este mecanismo garante que existe consistência entre os diversos modelos.

5.5 EXERCÍCIOS

5.5.1 Perguntas de Revisão

1: Explique o conceito de interacção?

2: Quais são os elementos que compõem um diagrama de interacção?

3: Qual é a função principal de um diagrama de sequência?

4: Explique o conceito de mensagem?

5: Que tipos de mensagem podem ser representados num diagrama de sequência?

6: Explique o conceito de linha temporal e controlo?

7: Qual é a principal diferença entre um diagrama de colaboração e um diagrama de sequência?

8: Em que consiste uma ligação num diagrama de colaboração?

9: Como é indicada a repetição num diagrama de interacção?

10: Como é ilustrada a criação de um objecto num diagrama de sequência?

11: Como é ilustrada a destruição de um objecto num diagrama de sequência?

12: Como é representada uma mensagem condicional?

© FCA - EDITORA DE INFORMÁTICA

5.5.2 Problemas Resolvidos

Elabore o diagrama de sequência para os seguintes problemas.

Primeiro identifique os objectos e as suas interacções.

Biblioteca

Considere a seguinte descrição para o cenário principal do *use case* "Registar Empréstimo".

Registar Empréstimo (Cenário Principal)	
Pré-condição	
Descrição	9. O *use case* começa quando o funcionário selecciona a opção de Registar Empréstimo. 10. De acordo com a ficha de empréstimo, previamente preenchida pelo aluno, o funcionário começa por introduzir o número do aluno. 11. Selecciona a opção Criar Novo Empréstimo. O sistema só permite criar novos empréstimos se o aluno não excedeu o limite de 3 empréstimos. 12. O funcionário introduz a cota da publicação que será emprestada. 13. O empréstimo é formalizado e gravado na base de dados através da opção Confirma.
Caminhos Alternativos	
Pós-Condição	O sistema volta para o ecrã de registo de empréstimos.

Parque de Estacionamento

Considere a seguinte descrição para o cenário principal do *use case* "Registar Entrada".

Registar Entrada (Cenário Principal)	
Pré-condição	
Descrição	1. O *use case* começa quando o funcionário introduz uma matrícula no seu terminal. 2. Caso a matrícula não seja reconhecida, será efectuado o registo do novo veículo. a. Extends: Cria Novo Veículo 3. O funcionário confirma o estacionamento pressionando a tecla Enter. O sistema regista a data e a hora do início do estacionamento.
Caminhos Alternativos	
Pós-Condição	

Solução Biblioteca

1° Identificação dos objectos

Da descrição do cenário principal do *use case* "Registar Empréstimo" surgem os seguintes objectos:

Funcionário - Actor responsável por despoletar o *use case*. A sua representação é opcional.

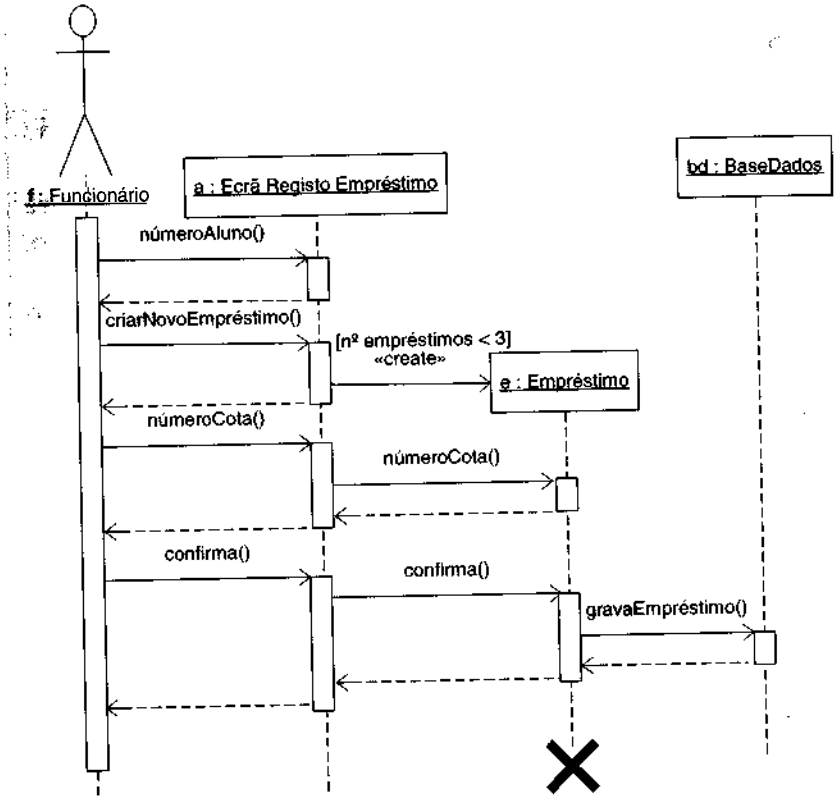
Ecrã Registo Empréstimo - Representa a interface com o actor, onde este pode registar empréstimos de publicações. A representação deste objecto permite identificar as classes responsáveis pela interacção com os utilizadores.

Empréstimo - Representa o empréstimo efectuado.

© FCA - EDITORA DE INFORMÁTICA

Base de Dados - Representa a base de dados onde será gravado o empréstimo.

2° Desenho do Diagrama de Sequência



Solução Parque de Estacionamento

1° Identificação dos objectos

Da descrição do cenário principal do *use case* "Registar Entrada" surgem os seguintes objectos:

Funcionário - Actor responsável por despoletar o *use case.* A sua representação é opcional.

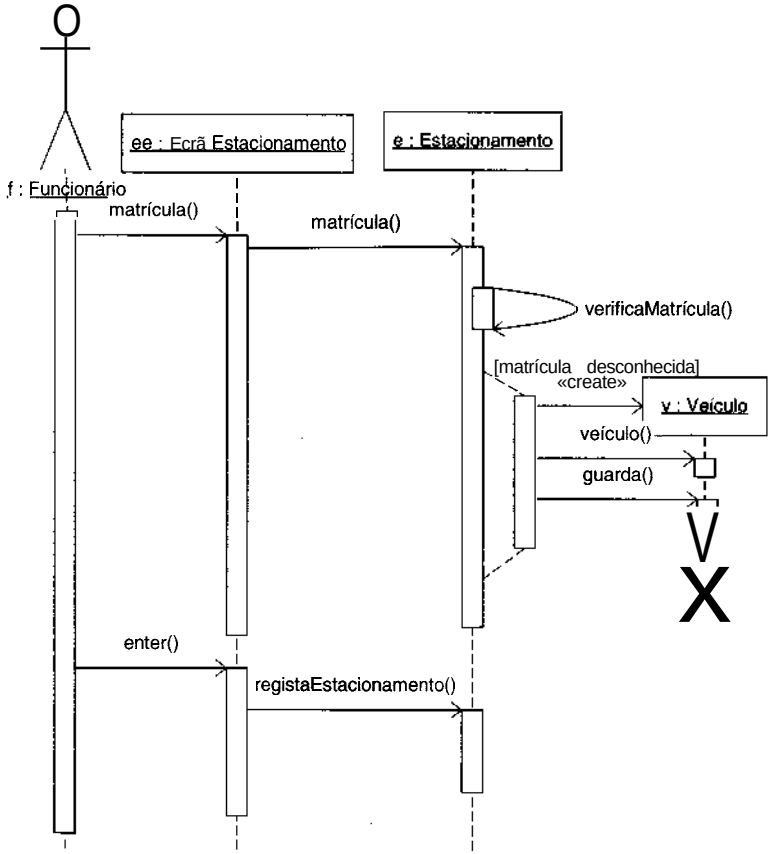
Ecrã Estacionamento - Representa a interface com o actor, onde este pode registar estacionamentos. Empréstimo - Representa o empréstimo efectuado.

Estacionamento - Representa um estacionamento que será registado.

Veículo - Representa um novo veículo que será adicionado ao sistema, caso este seja desconhecido.

Neste problema optou-se por não representar a base de dados (ou outro objecto semelhante) onde seriam guardados os diversos registos do sistema.

2° Desenho do Diagrama de Sequência



FCA - EDITORA DE INFORMÁTICA

Diagrama de Estados 6

6.1. CONCEITO E APLICAÇÃO

O diagrama de estados é utilizado para descrever o comportamento de um objecto. Um **estado** representa uma situação estável de um objecto que se prolonga durante um intervalo de tempo, durante o qual o objecto não sofre estímulos externos nem os atributos sofrem qualquer alteração de valor.

Na modelação de um sistema de informação deve-se criar um diagrama de estados somente para cada classe de objecto que tenha um comportamento dinâmico relevante como, por exemplo, os objectos de controlo ou objectos de interface.

O diagrama de estados é semelhante ao diagrama de actividades. A principal diferença entre ambos é que o diagrama de estados é centrado no objecto, enquanto um diagrama de actividade é centrado no processo, estando adequado à modelação de actividades nesse processo.

Consideremos o caso de estudo PhonePizza utilizado anteriormente. Neste caso existem classes de objectos com diversos graus de interacção dinâmica. Por exemplo, os objectos da classe Código Postal, ou mesmo a Pizzaria, são bastante estáveis: desde o momento em que são criados no sistema de informação praticamente não sofrem alterações no valor dos seus atributos.

Em contrapartida, temos objectos com ciclos de vida simultaneamente mais curtos e mais dinâmicos como, por exemplo, os objectos da classe Encomenda. Entre o pedido da encomenda e

FCA - EDITORA DE INFORMÁTICA

a sua entrega ao cliente decorrem somente alguns minutos, mas torna-se necessário controlar cada uma das diversas etapas intermédias do processo de satisfação dessa encomenda.

A figura 6.1 representa o diagrama de estados para um objecto da classe Encomenda.

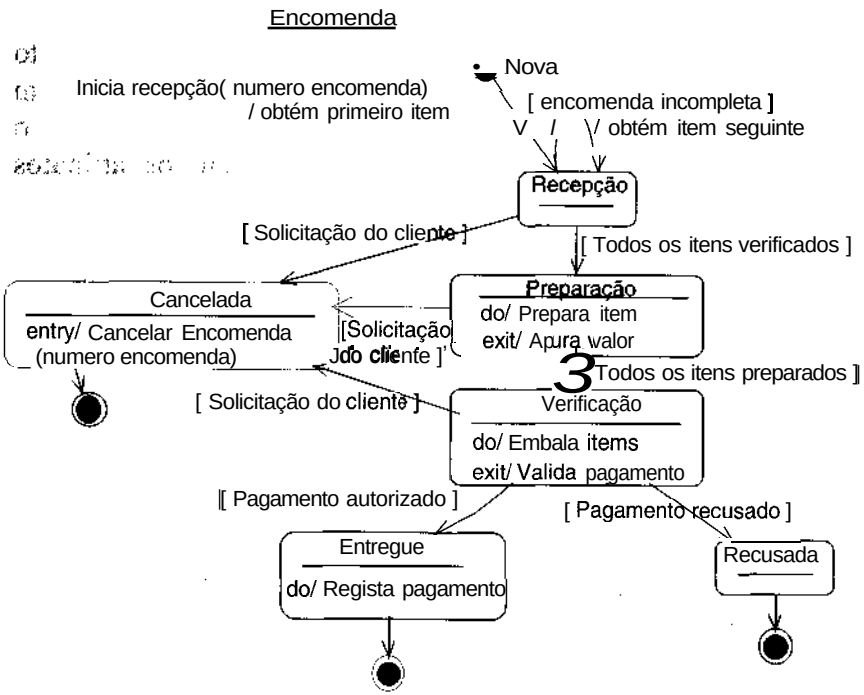


FIGURA 6.1 - DIAGRAMA DE ESTADO DE ENCOMENDA

Uma encomenda solicitada na loja começa por ser registada, sendo-lhe atribuído um número e ficando num estado "Nova". Em seguida, é registado o primeiro dos diversos itens que constituem a encomenda, colocando-a num estado de "Recepção". Após a recepção de todos os itens, inicia-se a preparação da encomenda, ficando esta no estado "Preparação". Posteriormente, a encomenda fica no estado "Verificação" e, por fim, passa ao

estado de "Entregue". A encomenda poderá ser Cancelada por solicitação do cliente ou ser Recusada pela falta do pagamento.

Os símbolos utilizados no diagrama de estados permitem descrever o estado que marca o início do diagrama (círculo), a transição entre estados (seta), os estados intermédios (rectângulo) e os estados finais (círculo negro e circunferência concêntricos). No diagrama de estados podem também ser utilizados guardas, símbolos de decisão e barras de sincronização.

6.1.1 Estado

O estado é representado por um rectângulo de cantos arredondados com um identificador e um compartimento para descrever as operações que são executadas nesse estado.

As operações associadas aos estados designam-se por actividades, já que demoram algum tempo a ser executadas e correspondem a actividades que podem ser identificadas num diagrama de actividades. As actividades podem ser activadas em quatro momentos distintos: no início do estado (`entry/`), durante o estado (`do/`), imediatamente antes da transição de estado (`exit/`) ou em resposta a um estímulo (on event). Neste último caso, a sintaxe utilizada é:

```
evento(args)[condição] :  /operação.
```

Por exemplo, após a transição para o estado "`Verificação`", é realizada a actividade "do/Embala itens" e, por fim, é realizada a actividade "`exit/Valida` Pagamento".

© FCA - EDITORA DE INFORMÁTICA

6.1.2 Transição entre estados

A transição entre dois estados acontece por via de estímulos externos (eventos) que estão associados à realização de acções (operações da classe). A transição entre estados é representada por uma seta que pode ter associada uma instrução com a seguinte sintaxe:

evento (argumentos) [condição] / acção estadoAlvo.evento (argumentos)

Por exemplo, a transição entre o estado "Nova" e "Recepção" acontece através de um evento designado por Inicia recepção (número de encomenda), associado a uma acção que é Obtém primeiro item, representado no diagrama com uma sintaxe simplificada.

A transição para um estado pode também estar sujeita à satisfação de uma condição representada entre parêntesis rectos [ ], designada por **guarda.** Em cada momento só poderá ocorrer uma transição de saída de um estado, pelo que as condições expressas nas guardas têm de ser mutuamente exclusivas. Por exemplo, a transição do estado "Verificação" para o estado "Recusada" exige que se verifique a condição [pagamento recusado]. Se o pagamento for autorizado, a transição far-se-á para o estado "Entregue".

No diagrama aparece representada uma transição que se inicia e termina no estado "Recepção" para descrever que a encomenda se mantém nesse estado até serem conhecidos todos os itens.

6,2. TÓPICOS AVANÇADOS

6.2.1 Agrupamento de Estados

pio exemplo que estamos a utilizar, a encomenda poderá passar para o estado "Cancelada" por solicitação do cliente. Essa transição pode-se realizar a partir de três estados distintos que são: "Recepção", "Preparação" e "Verificação". Este facto pode ser representado de uma forma mais simples, ou seja, através da utilização de um **superestado,** aumentando a legibilidade do diagrama.

Na figura 6.2 está representado um novo diagrama de estados da classe encomenda.

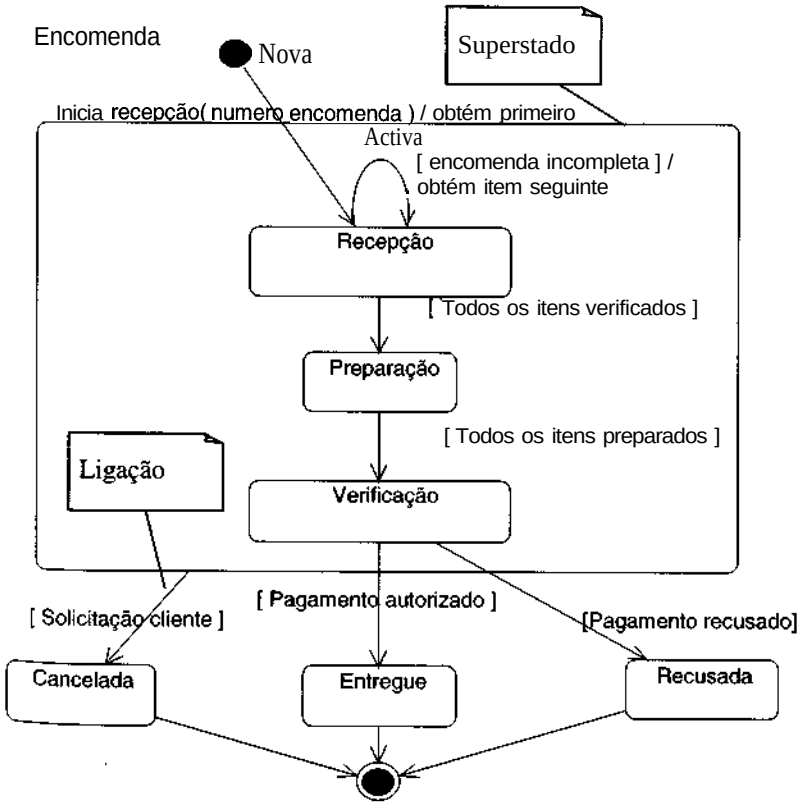


FIGURA 6.2 - SUPERESTADO E SUBESTADOS

© FCA - EDITORA DE INFORMÁTICA

Neste diagrama é utilizado um superestado designado por "Activa", que engloba os estados simples ou subestados "Recepção", "Preparação" e "Verificação". A partir de qualquer um destes subestados é possível a transição para o estado "Cancelada", o que se representa através da transição que tem origem no superestado "Activa". As transições para os estados "Entregue" e "Recusada" continuam a ter origem directamente no estado "Verificação".

6.2.2 Concorrência entre subestados

Existe uma relação muito próxima entre actividades e estados. Num sistema de informação, uma actividade encontra-se associada à execução de uma operação de uma classe de objectos, podendo delimitar um estado no ciclo de vida do objecto. Como foi referido no capítulo relativo aos diagramas de actividade, o UML permite descrever fluxos de actividades que podem ser executados em paralelo **(concorrência),** utilizando linhas de sincronização divergentes e convergentes.

A realização de actividades em paralelo tem impacto no diagrama de estados, sendo necessário reflectir o facto de um objecto poder estar em estados alternativos, dependendo do fluxo de controlo de actividades que está a ser realizado.

Consideremos o exemplo da Encomenda. No estado de Verificação têm de ser concluídas duas actividades - Embala Itens e Valida Pagamento - para se passar ao estado seguinte. Cada uma dessas actividades encontra-se associada a um estado específico, que importa controlar. A fig. 6.3 apresenta o diagrama de estados que descreve esta situação.

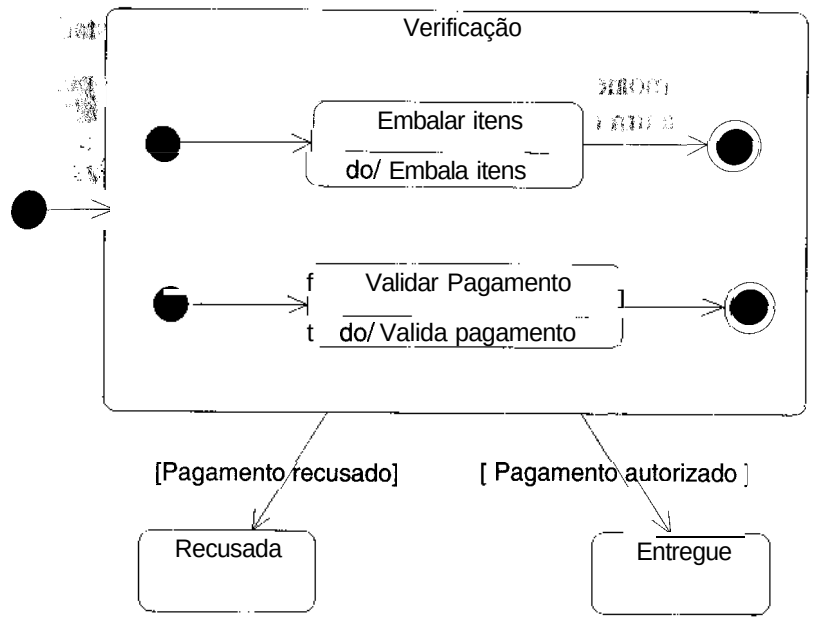


FIGURA 6,3 – CONCORRÊNCIA DE SUBESTADOS

> Convém efectuar um diagrama de estados para cada classe de objectos com um comportamento dinâmico relevante.

6,3 EXERCÍCIOS

6.3.1 Perguntas de Revisão

1: Qual a finalidade de um diagrama de estados?

2: O que é um estado?

3: Quantos diagramas de estado são necessários especificar num modelo de um sistema de informação?

4: Quais os elementos de modelação que constam num diagrama de estados?

© EDITORA DE INFORMÁTICA

5: Que símbolo utiliza para representar graficamente um estado?

6: Em que momentos podem ser executadas as operações associadas a um estado?

7: Como se representa graficamente a transição entre estados?

8: O que é um superestado?

9: Que relação pode existir entre as actividades e os estados?

10: Como se traduz num diagrama de estados o processamento paralelo de actividades numa classe de objectos?

6.3.2 Problemas Resolvidos

Desenhe os respectivos diagramas de estados de acordo com as descrições seguintes.

Biblioteca

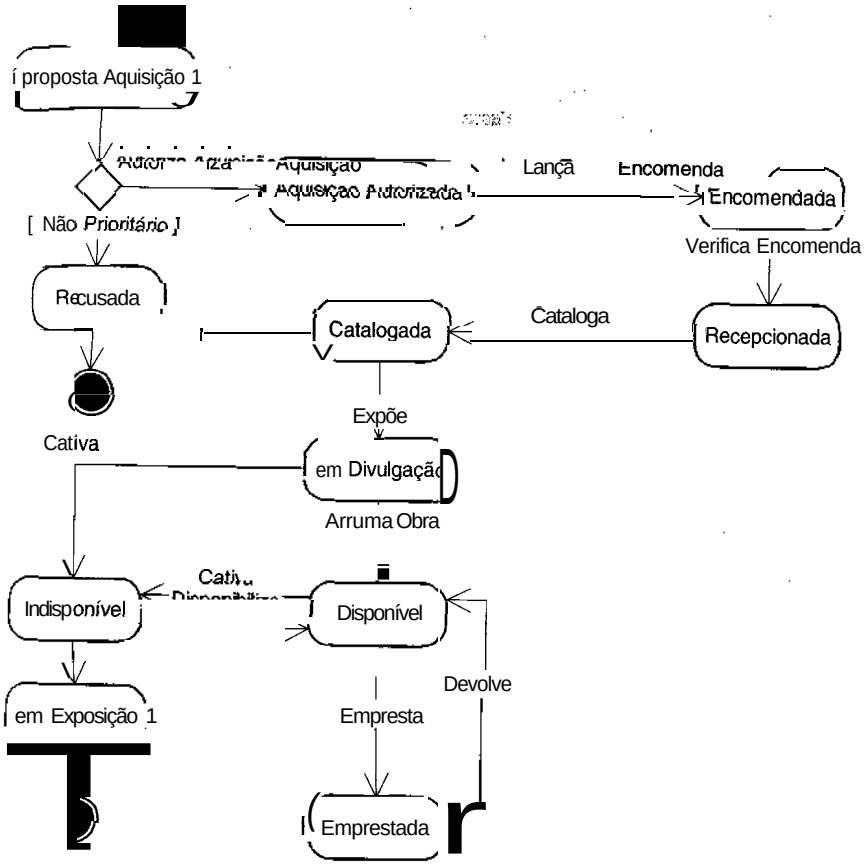
Considere o processo de disponibilização de obras descrito no capítulo 4. Além disso, considere que periodicamente as obras são analisadas e, em função do seu estado de conservação, podem ficar indisponíveis. As obras consideradas valiosas podem ser retiradas do circuito de empréstimo e colocadas em exposição.

Parque de Estacionamento

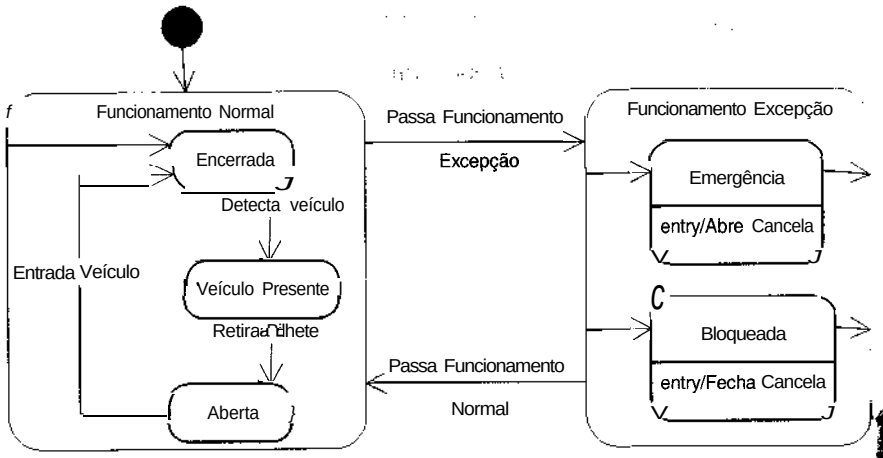
A cancela de entrada no parque de estacionamento possui diversos estados de funcionamento. Em utilização Normal o portão pode estar Aberta, Fechada ou ainda numa situação intermédia em que se detecta que o Veículo está Presente. Excepcionalmente, por motivos de segurança, a cancela pode ser Bloqueada ou ser colocada em Emergência permanecendo aberta.

© FCA - EDITORA DE INFORMÁTICA

Solução Biblioteca - Obra



Solução Parque de Estacionamento - Cancela de Entrada



Desenho do **Sistema** 7

7.1. CONCEITO E APLICAÇÃO

O desenho do sistema permite definir a organização das diversas partes do sistema. Nos capítulos anteriores os modelos produzidos centraram-se numa actividade de análise onde é identificado o que deve estar no sistema e as suas relações.

O desenho procura determinar como é que o sistema cumpre com os requisitos. Nesta actividade o diagrama de classes é refinado com vista a aumentar a possibilidade de reutilização dos componentes que mais tarde derivarão das classes.

A reutilização é um objectivo do desenvolvimento orientado por objectos, onde se procura produzir componentes reutilizáveis, que possam ser utilizados em diferentes aplicações, diminuindo assim os custos de desenvolvimento.

Este capítulo cobre algumas das estratégias possíveis, ao nível do desenho, para aumentar a possibilidade de reutilização. Um estudo mais completo pode ser encontrado em Bennet et al (1999).

Sendo assim, neste capítulo iremos abordar a introdução de interfaces para classes, camadas *(layers)* no diagrama de classes e divisão do sistema utilizando pacotes.

© FCA - EDITORA DE INFORMÁTICA

7,2 DE DE DESENHO

Para efectuar o diagrama de classes com uma perspectiva de desenho é necessário introduzir, previamente, um conjunto de conceitos e mecanismos da UML.

7.2.1 Estereótipos

O estereótipo na UML é um mecanismo de extensibilidade que permite aumentar a flexibilidade das classes, através de uma subclassificação. É possível criar qualquer tipo de estereótipo, sendo os mais utilizados os de interface e controlo. A representação destes estereótipos é efectuada da seguinte forma:

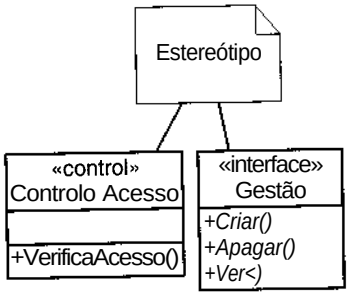


FIGURA 7.1 - EXEMPLO DE ESTEREÓTIPOS

O estereótipo de «interface» classifica as classes que apenas disponibilizam um conjunto de operações visíveis externamente (públicas). O conceito de interface é abordado com mais detalhe na secção 7.2.4.

Uma classe com o estereótipo de «control» representa uma classe cujo objectivo é conter um conjunto de regras que controlam determinadas operações do sistema e que coordenam as interacções com as outras classes.

Por vezes as classes que são classificadas através de um estereótipo também apresentam uma notação gráfica diferente, permitindo

assim a sua distinção imediata. Por exemplo, as classes da fig. 7.1 poderiam ser representadas por:

FIGURA 7,2 — REPRESENTAÇÃO ALTERNATIVA DE ESTEREÓTIPOS

7.2.2 Relação de Dependência

A relação de dependência surge quando uma classe recorre aos serviços disponibilizados por outra classe, numa relação de cliente/fornecedor de serviços. Por exemplo, quando um funcionário efectua ou consulta uma encomenda, este terá que aceder aos serviços de gestão disponibilizados pela classe Encomenda. Esta classe pode disponibilizar uma classe de interface abstracta[1] que agrupa os serviços necessários.

Na figura 7.3 a relação de dependência surge quando a classe Funcionário necessita dos serviços da classe Encomenda para que se possa gerir uma encomenda. Na prática, a interface Gestão disponibiliza as operações Criar(), Apagar() e Ver() da classe Encomenda. Caso os serviços disponibilizados sofram alguma alteração, então a classe que requer o serviço poderá também sofrer alterações, uma vez que é possível que esta seja confrontada com um novo comportamento.

© FCA - EDITORA DE INFORMÁTICA

[1] Classe que apenas indica as operações disponibilizadas, sendo a classe que disponibiliza a interface responsável por efectuar as operações.

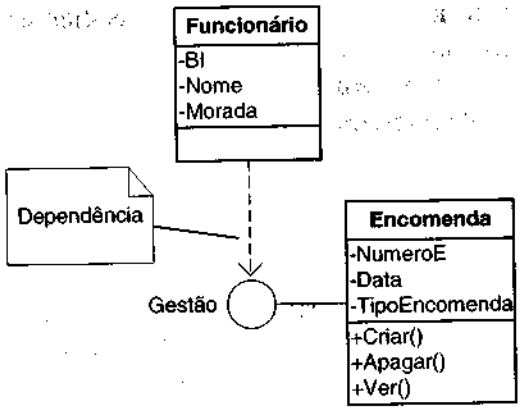


FIGURA 7,3 - RELAÇÃO DE DEPENDÊNCIA

> Normalmente utiliza-se a dependência quando queremos mostrar que um elemento de modelação utiliza os serviços de outro elemento.

7.2.3 Relação de Realização

Esta relação é quase uma mistura entre a relação de generalização e a de dependência, o que também se reflecte na sua notação (exemplo na fig. 7.4). O seu objectivo é mostrar que existe um contrato entre uma classe que especifica um serviço e uma outra classe que garante a realização do serviço.

Normalmente, a relação de realização será utilizada para especificar interfaces ou colaborações. Contudo, a utilização mais frequente é nas interfaces, um tema que é abordado em seguida.

7.2.4 Interfaces

As classes do sistema podem usufruir do conceito de **interface,** que permite separar o que é fornecido pela abstracção da classe da forma como é produzido. Na prática, uma interface é um grupo de

operações que são utilizadas para especificar um serviço. Este serviço funciona como um contracto entre a classe e os seus clientes que, por sua vez, constróem os seus serviços com base na interface estabelecida.

Ao efectuar esta separação, é possível efectuar alterações numa classe sem afectar as restantes, desde que não se altere a interface, reduzindo assim o impacto das alterações. Uma classe pode conter várias interfaces, fornecendo assim diferentes abstracções dos seus serviços, conforme o cliente.

A representação gráfica de uma interface consiste num círculo ou, em alternativa, numa classe, mas com o estereótipo de interface (`«interface»`). Ao contrário das classes normais, não é especificada a estrutura, não sendo representados os atributos, apenas as operações (fig. 7.4).

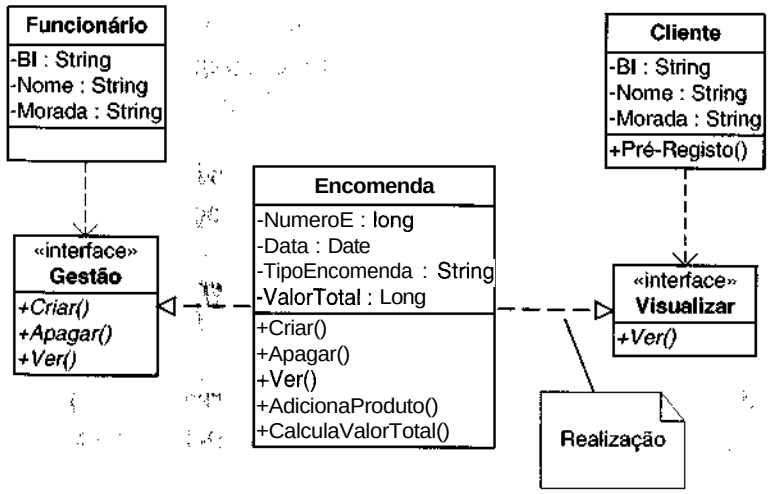


FIGURA 7.4 - EXEMPLO DE INTERFACES E REALIZAÇÃO

No exemplo da figura 7.4, a classe Encomenda possui duas interfaces `Gestão` e `Visualizar`. A classe `Funcionário` utiliza a interface `Gestão` que permite criar, apagar e consultar

© FCA - EDITORA DE INFORMÁTICA

encomendas. A classe `Cliente`, apenas pode ver as encomendas logo está dependente da interface Visualizar que se disponibiliza a operação `Ver()`.

O conceito de interface não é só aplicado a classes, mas também a componentes, um tema abordado no Capítulo 8. O comportamento esperado de uma interface pode ser detalhado através de diagramas de interacção (Capítulo 5), permitindo assim fornecer mais informação de forma a aumentar a sua compreensão e integração pelas classes cliente.

> Utiliza-se uma classe para representar uma interface quando se quer mostrar as suas operações e respectivas assinaturas (parâmetros e valor de retorno).

7.2.5 Diagrama de Classes com níveis

Um diagrama de classes com 3 níveis é um diagrama de classes que está dividido em 3 camadas de serviços:

1. **Serviços de Interface ou *"User services"*** - fornece a interface do utilizador para apresentação e recolha de dados.
2. **Serviços de Negócio ou *"Business services"*** - engloba as classes que possuem as regras fundamentais do negócio.
3. **Serviços de Dados ou *"Data services"*** - permitir manter, actualizar e aceder aos dados persistentes.

Numa arquitectura de 3 camadas, os serviços de Negocio respondem a pedidos dos serviços de Interface, ou de outros serviços de Negócio, para realizar uma tarefa de negócio. Para tal, é necessário executar uma operação específica sobre dados relevantes com base nas regras de negócio. Quando os dados residem num servidor de base de dados, os serviços de Negócio

asseguram o acesso aos serviços de Dados, isolando assim o programador do acesso directo à base de dados. Como as regras de negócio tendem a ser alteradas com relativa frequência, os serviços de Negócio são úteis para encapsular estas regras, separando assim a tarefa a desempenhar da forma como é desempenhada.

Ao isolar os serviços de Negócio dos serviços de Interface e Dados, este diagrama permite ir ao encontro do paradigma do desenvolvimento de aplicações cliente/servidor de larga escala, promovendo assim a reutilização, escalabilidade e manutenção dos componentes.

O diagrama na figura 7.5 demonstra esta abordagem para o caso da *PhonePizza*. Este diagrama define uma classe de interface de utilizador Ecrã `Pré-Registo` que necessita da classe `Cliente` nos serviços de negócio para efectuar o registo, invocando para tal a operação `Pré-Registo()`. Por sua vez, a classe `Cliente` necessita de guardar num suporte físico (base de dados ou ficheiro) os dados do cliente que está a efectuar o pré-registo, utilizando então os Serviços de Dados através da classe SD_Cliente.

Um procedimento semelhante é aplicado para as classes Ecrã Reservas e Ecrã Encomenda. Contudo, estas necessitam de verificar se o cliente possui permissão para efectuar as operações de reserva e encomenda, um serviço fornecido pela classe Controlo de Acesso.

5 A classe Controlo de Acesso possui um estereótipo de `«control»`, pois a sua função é apenas conter as regras de negócio que gerem o acesso ao sistema.

> As classes de interface com o utilizador possuem o esterótipo `«user interface»`.

© FCA - EDITORA DE INFORMÁTICA

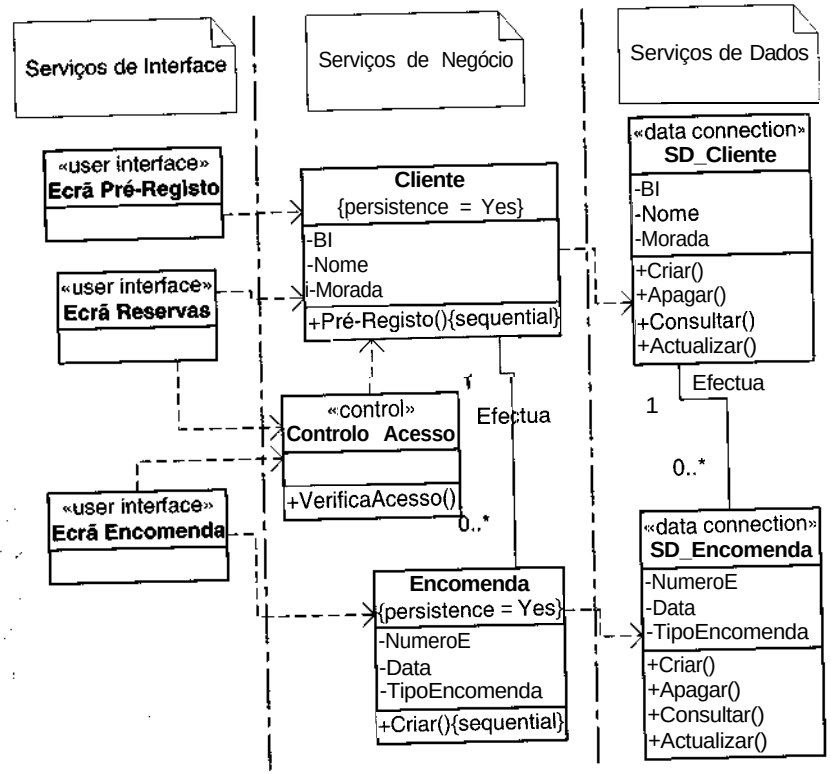


FIGURA 7,5 - EXEMPLO DIAGRAMA DE CLASSES COM 3 NÍVEIS

Uma das características das classes nos serviços de Negócio é a sua **persistência** em termos de informação. Assim, as classes "persistentes" (`persistence = Yes`) necessitam que os seus objectos sejam gravados fisicamente numa base de dados ou noutro formato. Ao contrário das outras duas classes presentes no diagrama, a classe `Controlo de Acesso` não necessita que a sua informação seja persistente, pois utiliza os serviços da classe `Cliente`. No entanto, caso exista a necessidade de manter um registo de acessos específico da classe, esta seria marcada como persistente e teria uma classe correspondente no Serviço de Dados.

A parte física dos dados dependerá do tipo de base de dados (relacional ou orientada por objectos) a ser utilizada. Caso seja relacional, é possível efectuar a transposição das classes dos Serviços de Dados para o modelo relacional através de um conjunto de regras. Estas regras encontram-se descritas no Anexo 1.

> Todas as classes de característica persistente nos serviços de Negócio terão uma ligação com uma ou mais classes nos serviços de Dados.

7.3 PACOTES

Até agora temos abordado diversos conceitos que rodeiam a UML, utilizando pequenos exemplos de leitura fácil. No entanto, no desenvolvimento de projectos de grande dimensão é muito difícil manter a simplicidade dos diagramas, pois a quantidade de informação a representar (por exemplo, o número de classes) inviabiliza qualquer tentativa de manter um único diagrama de cada tipo.

Os **pacotes** (*packages*)na UML permitem dividir a complexidade do sistema em partes mais pequenas para uma melhor gestão. Um pacote é um mecanismo que permite agrupar elementos de modelação UML (diagramas, classes, componentes, interfaces, etc). Um pacote é representado graficamente por uma pasta, contendo um nome. A fig. 7.6 demonstra esta representação:

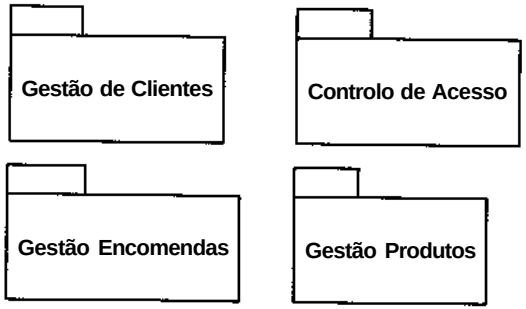


FIGURA 7,6 - REPRESENTAÇÃO DE PACOTES

© FCA - EDITORA DE INFORMÁTICA

Na figura 7.9 o pacote Encomendas Internet agrupa dois formulários (`FormEncomenda` e `FormCatálogo`) de acesso público, que permitem ao cliente introduzir a sua encomenda através da Internet. O elemento `Encomenda` é privado, sendo apenas visível dentro do pacote.

> Não exagerar nos níveis da hierarquia de pacotes. No máximo utilizar 3 níveis.

7.3.2 Representação do sistema em 3 níveis

Na secção 7.2.5 foi abordado o conceito de diagrama de classes de 3 níveis, onde é feita a distinção entre os serviços de interface, negócio e dados. Estas classes também poderiam ser agrupadas em pacotes. A figura 7.10 ilustra esta abordagem.

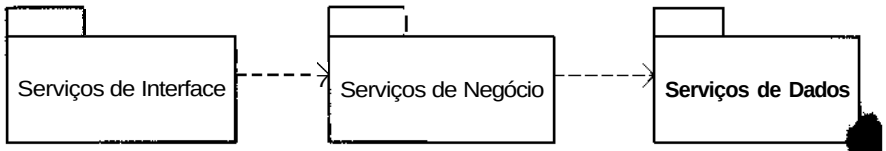


FIGURA 7.10 - DIAGRAMA DE PACOTES COM 3 NÍVEIS

7,4 EXERCÍCIOS

7.4.1 Perguntas de Revisão

1: Explique o conceito de reutilização?

2: Explique o conceito de estereótipo?

3: O que é, e como se representa, o estereótipo de «`interface`»?

4: O que é, e como se representa, o estereótipo de «`control`»?

5: Explique o conceito da relação de dependência?

© FCA - EDITORA DE INFORMÁTICA

6: Como é representada a relação de dependência?

7: Explique o conceito da relação de realização num diagrama de classes?

8: Como é representada a relação de realização?

9: Qual é o objectivo da criação de um Diagrama de Classes com vários níveis?

10: Numa arquitectura de 3 níveis, quais são os serviços utilizados?

11: Explique o conceito de pacote?

12: Como se estabelece a relação entre pacotes?

Diagramas Físicos 8

8.1. CONCEITO E APLICAÇÃO

A UML disponibiliza dois tipos de diagramas para descrever as características físicas de um sistema: o Diagrama de Componente e o Digrama de Instalação. Por características físicas entende-se a concretização da descrição lógica suportada com os diagramas de *use cases*, classes, actividades, estados e interacção, em componentes de *software* que constituem aplicações informáticas a serem processadas. Assim, os Diagramas de Componentes permitem descrever os diversos "pedaços" de *software* que são os programas fonte, bibliotecas ou programas executáveis.

O objectivo dos Diagramas de Instalação é descrever a arquitectura do sistema em termos de *hardware* e a sua relação com os diferentes componentes *(software).* Os componentes necessitam ser executados em algum recurso computacional que contenha memória e um processador. O Diagrama de Instalação define em que recursos os diferentes componentes estarão localizados.

O desenvolvimento de componentes bem estruturados e com interfaces bem definidas permite efectuar uma manutenção mais eficiente e, em último caso, reduzir ou mesmo eliminar o impacto negativo da substituição de componentes antigos por novos, desde que estes mantenham as mesmas interfaces.

Actualmente, as linguagens de programação orientadas por objectos (JAVA, Visual C++, etc.) incluem a noção de componente como parte integrante do seu ambiente de desenvolvimento. A UML não só permite a representação destes componentes, mas também possibilita uma integração com as ferramentas de

© FCA - EDITORA DE INFORMÁTICA

programação, permitindo, por exemplo, gerar numa determinada linguagem a estrutura das classes e componentes.

Recorrendo novamente ao exemplo PhonePizza, poderíamos ter a seguinte descrição dos requisitos, em termos de arquitectura do sistema.

> "O sistema de encomendas central deverá permitir a ligação de um servidor HTTP responsável pela recepção das encomendas via Internet. Deve também permitir a consolidação de todos os movimentos efectuados no servidor de encomendas de cada pizzaria.
>
> Por sua vez, o servidor na pizzaria terá ligações aos terminais com ecrã táctil disponíveis nas mesas e às máquinas de encomenda no balcão. A comunicação entre os diversos sistemas será efectuada com base em transacções em XML.
> A informação do sistema é guardada numa Base de Dados (BD) central, contudo cada pizzaria possuirá uma BD local."

A figura 8.1 combina o diagrama de componentes com o diagrama de instalação, ilustrando apenas o sistema central de encomendas e a sua ligação com a Internet. A combinação dos diagramas fornece uma visão mais abrangente do sistema, ilustrando a existência de dois servidores (HTTP e Encomendas Central), que comunicam entre si através de um protocolo TCP/IP-XML.

Optou-se por dividir os componentes do sistema com interface Internet para o servidor HTTP, para não sobrecarregar o servidor Encomendas Central. Os componentes são idênticos, apenas diferenciados pela interface fornecida, facto que é realçado pela ligação de dependência que os une. O servidor HTTP aloja as páginas HTML do *site* da PhonePizza, que também poderiam ser representadas como componentes

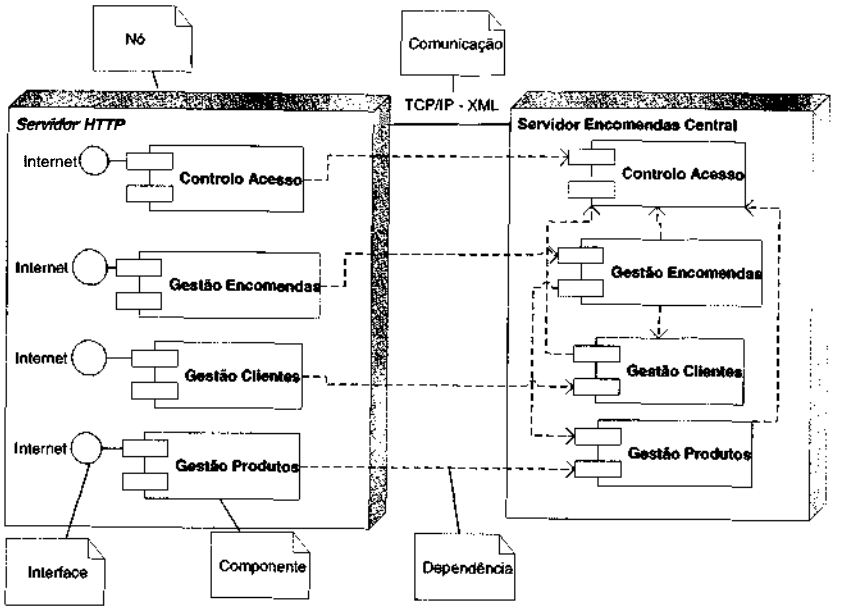


FIGURA 8.1 - DIAGRAMAS Físicos

8.2. DIAGRAMA DE COMPONENTES

Um diagrama de componentes mostra um conjunto de componentes e as suas relações. Para o exemplo fornecido anteriormente poderíamos ter o seguinte diagrama:

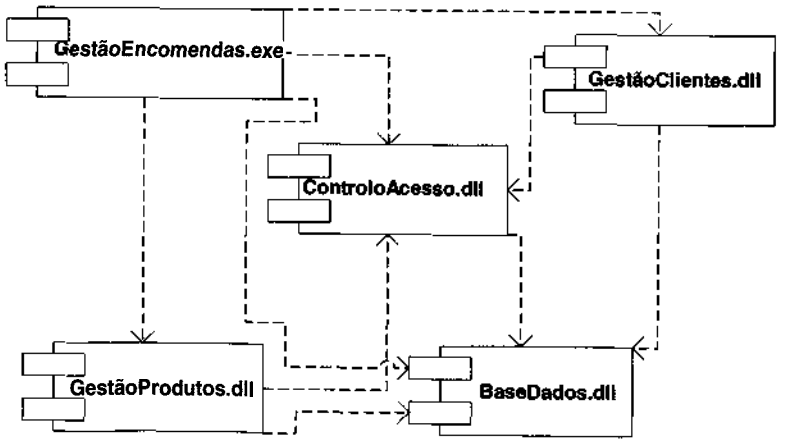


FIGURA 8.2 - DIAGRAMA DE COMPONENTES

© FCA - EDITORA DE INFORMÁTICA

O diagrama na figura 8.2 ilustra os diversos componentes que formam o sistema e suas relações de dependência. A divisão é efectuada de acordo com a sua natureza, existindo assim cinco componentes:

» **GestãoEncomendas.**exe - Responsável por todas as operações relacionadas com encomendas (criar, alterar, apagar, actualizar, etc.). Depende do componente ControloAcesso.dll para verificar se o utilizador possui permissões para executar as operações. Também depende do resto dos módulos, pois necessita de ter informações sobre produtos e clientes e de guardar a sua informação numa base de dados.

• **GestãoProdutos.dll** - Encarregue por todas as operações relativas à gestão de produtos. Depende do componente ControloAcesso.dll para verificar se o utilizador possui permissões para executar as operações. Também depende do módulo BaseDados . dll para guardar a sua informação.

* **GestãoClientes.dll** - Encarregue de todas as operações relativas à gestão de clientes. À semelhança da GestãoProdutos.dll, também depende do componente ControloAcesso . dll e do módulo BaseDados . dll.

* ControloAcesso **.dll** - Responsável por conter as regras e a política de acesso às operações e objectos do sistema. Apenas depende do componente BaseDados . dll para guardar a sua informação.

• **BaseDados.dll** - Responsável por conter as operações de acesso e manutenção da informação nas bases de dados, separando assim os outros componentes dos diferentes tipos de bases de dados e protocolos de acesso.

Esta divisão depende da sensibilidade do analista para o desenvolvimento do sistema, o que obriga a possuir algum conhecimento técnico ou ser auxiliado por um programador.

8.2.1. Componentes

Um componente representa um módulo físico de código, sendo o resultado do desenvolvimento numa linguagem de programação ou outra técnica. O desenvolvimento por componentes permite reforçar a reutilização como forma de diminuição de custos e possíveis erros, dado que podem ser previamente já testados.

Normalmente, os componentes encontram-se interligados por uma **relação de dependência,** que demonstra o impacto nos diversos componentes das alterações a um componente em particular. Esta relação pode possuir vários estereótipos conforme o tipo de componentes.

A fig. 8.3 ilustra a composição do componente ControloAcesso.dll, que depende das bibliotecas Utilizador.dll e PolíticasAcesso.dll

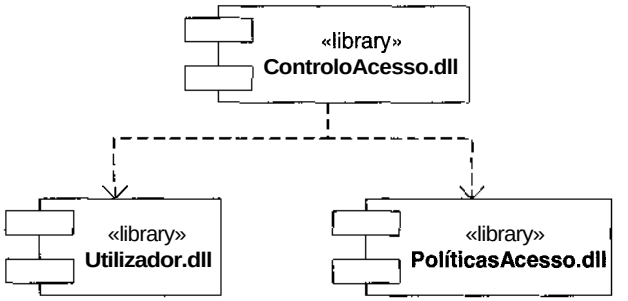


FIGURA 8,3 — DIAGRAMA DE COMPONENTES

Na UML o diagrama de componentes pode ser utilizado para modelar:

* **código fonte** - organização dos ficheiros de código fonte.
* **ficheiros binários** - organização dos ficheiros binários incluindo executáveis e bibliotecas.
» **base de dados** - modelação das tabelas de uma base de dados.

© FCA - EDITORA DE INFORMÁTICA

Esta flexibilidade é conseguida através da utilização de estereótipos como, por exemplo, «html», «library», «executable», «table», etc. O seguinte exemplo ilustra as dependências para código fonte em C++:

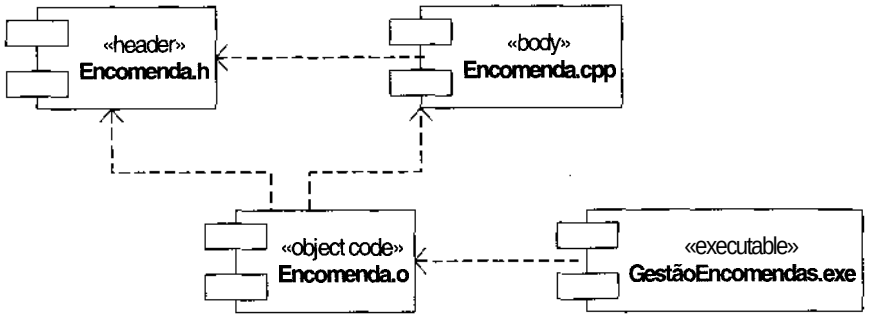


FIGURA 8.4 – DIAGRAMA DE COMPONENTES PARA C++

É também possível utilizar uma notação gráfica específica para cada estereótipo, permitindo assim distinguir visualmente os diversos componentes.

O exemplo na figura 8.5 demonstra os componentes de um site que permite efectuar encomendas através da Internet. O diagrama utiliza uma notação gráfica diferente para ilustrar as páginas HTML, que entre si partilham de uma relação de dependência com o estereótipo de «hyperlink». Este estereótipo estende o significado da dependência para a linguagem HTML, onde as ligações entre páginas são efectuadas no hipertexto.

Neste caso, a página Index.html é a página inicial possuindo uma ligação à página Login.html que efectua uma verificação de acesso ao utilizador. Caso seja um utilizador válido, é fornecido acesso à página Menu.html que disponibiliza as opções de encomendar ou consultar o catálogo de produtos. As páginas Login.html, Encomendas.html e Catalogo.html requerem serviços aos diferentes componentes do sistema. Estes

componentes disponibilizam para o efeito uma interface Internet que agrupa os diversos serviços requeridos.

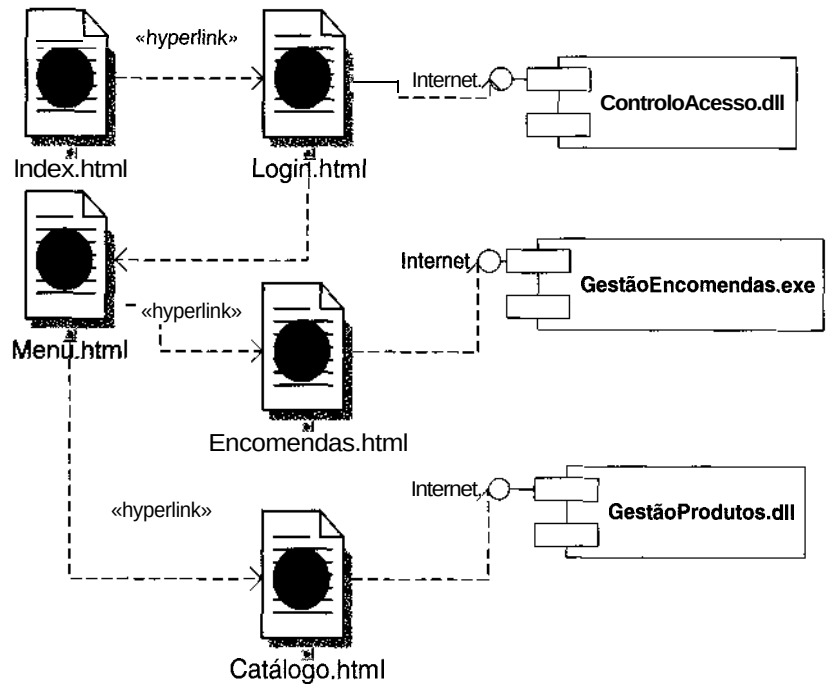


FIGURA 8.5 - DIAGRAMA DE COMPONENTES PARA HTML

8.2.2. Interfaces

Os componentes podem disponibilizar diferentes interfaces conforme a necessidade dos subsistemas aos quais prestam serviços. Na fig. 8.5, cada componente possui uma interface denominada Internet, que representa o conjunto de serviços disponibilizados pelo componente, adaptados às necessidades e formato particular da Internet. Esta interface pode ser representada utilizando uma classe com o estereótipo de «interface» ou com a representação mais abstracta. Por exemplo, na figura 8.6, a interface do componente Controlo de Acesso é representada

© FCA - EDITORA DE INFORMÁTICA

como uma classe e, em alternativa, de uma forma mais abstracta através de um pequeno círculo com uma ligação ao componente.

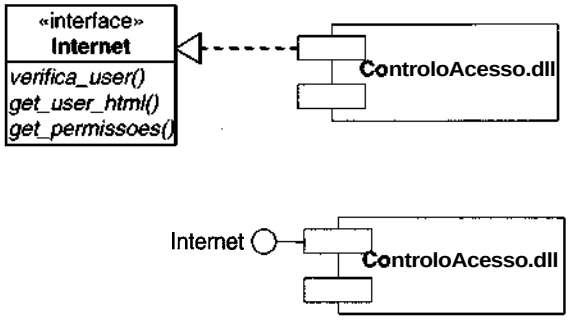


FIGURA 8.6 - INTERFACE PARA UM COMPONENTE

8.3. DIAGRAMA DE INSTALAÇÃO

Este diagrama ilustra a arquitectura do sistema em termos de nós (*nodes*) que efectuam o processamento de componentes. Na prática, permite demonstrar como o *hardware* estará organizado e como os componentes (*software*) estarão distribuídos, estabelecendo assim a sua relação física.

A fig. 8.7 contém um possível diagrama de instalação para o exemplo apresentado no início deste capítulo. Este diagrama de instalação define seis componentes que comunicam entre si. É de realçar que existe urna separação entre o Servidor de Base de Dados e o Servidor de Encomendas Central por razões de fiabilidade e rapidez. No entanto, na loja optou-se por incluir uma base de dados local dentro do Servidor Encomendas Loja.

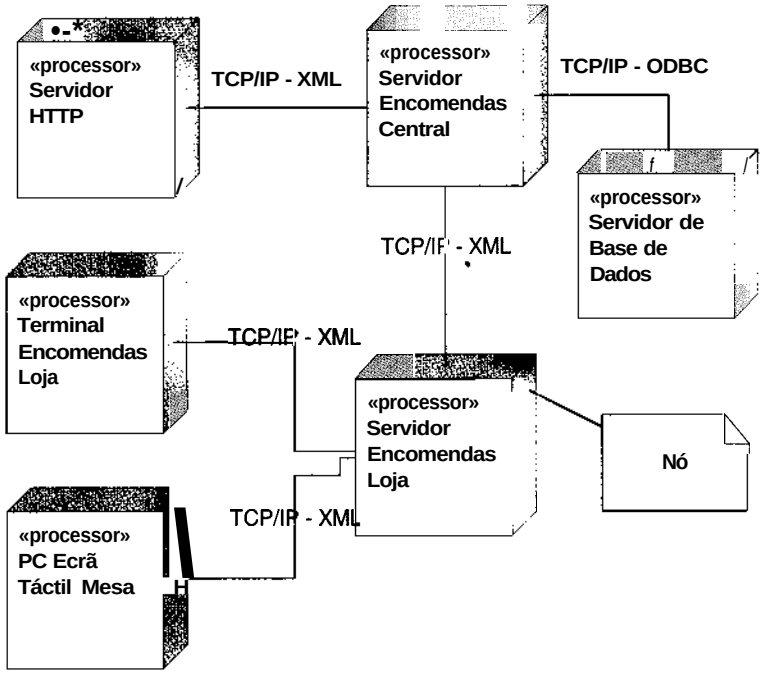


FIGURA 8,7 - DIAGRAMA DE INSTALAÇÃO

8.3.1. Nós

Representa um recurso físico onde são executados os componentes do sistema. A sua representação gráfica pode ser alterada de forma a representar canonicamente os diversos elementos físicos, utilizando ícones especiais que representam computadores, servidores ou terminais. A figura 8.8 ilustra esta alternativa.

FCA - EDITORA DE INFORMATICA

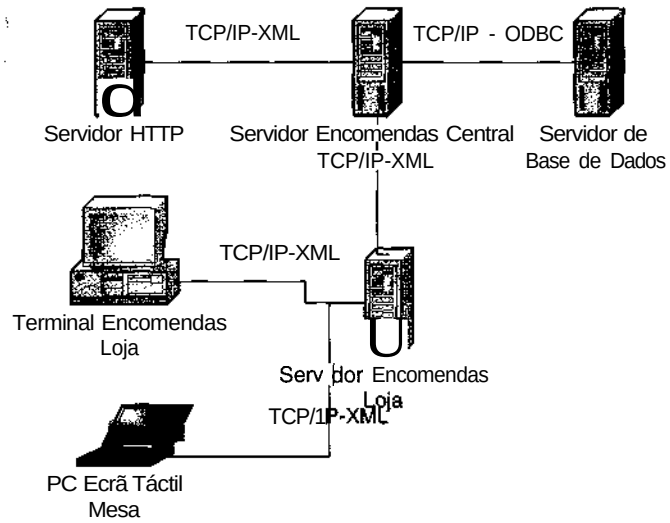


FIGURA 8,8 - REPRESENTAÇÃO ALTERNATIVA DE Nós

Os nós podem ser classificados através de estereótipos, para representar com mais detalhe os recursos físicos. Na fig. 8.9, o nó Terminal Encomendas Loja utiliza o estereótipo «processor» para representar um recurso físico com um processador e o estereótipo de «device» para um dispositivo leitor de código de barras.

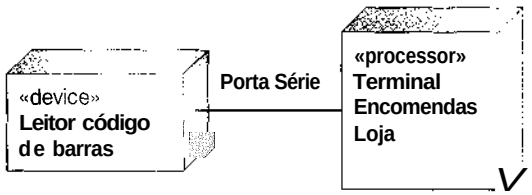


FIGURA 8,9 — Nós COM ESTEREÓTIPOS

Os nós, através da utilização de estereótipos, podem representar qualquer tipo de recurso físico como ecrãs, impressoras, terminais, etc.

8.3.2. Comunicação

Normalmente, os nós terão que comunicar entre si. Para tal, a UML liga os diversos elementos através de uma **linha de comunicação,** que representa a comunicação num determinado protocolo. Esta linha de comunicação também pode albergar estereótipos como, por exemplo, «internet» para ilustrar uma ligação através da Internet. Este cenário é ilustrado na fig. 8.10.

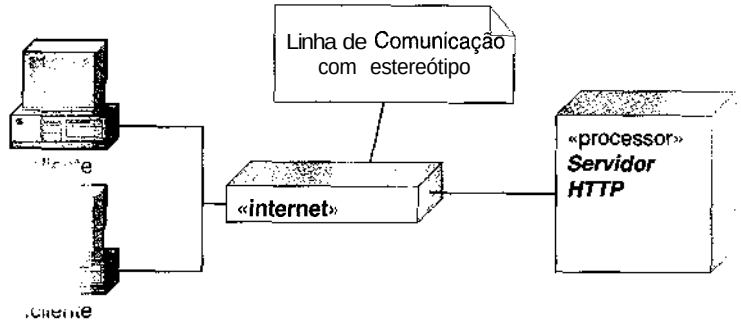


FIGURA 8,10 - LINHA DE COMUNICAÇÃO

8.3.3 Nós e componentes

Os diagramas de componentes podem ser mostrados dentro de nós, como forma de ilustrar o seu ambiente de execução. A figura 8.11 exemplifica os componentes existentes no servidor de encomendas central e servidor de base de dados.

O componente Gestão de Encomendas está localizado no Servidor Encomendas Central. Este componente está dependente do componente ODBC, que contem as rotinas de acesso à base de dados VendasDB. A comunicação entre os nós é efectuada através do protocolo TCP/IP, onde também está a ligação ODBC.

O Servidor de Base de **Dados conterá a base de dados**

VendasDB e o SGBD (Sistema de Gestão de Base de Dados).

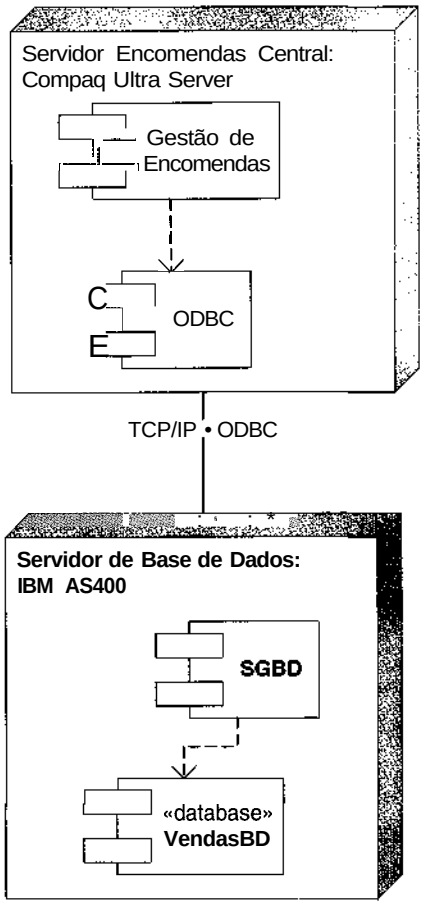


FIGURA 8,11 - Nós E COMPONENTES

8.4 EXERCÍCIOS

8.4.1 Perguntas de Revisão

1: O que é um componente?

2: Qual é a notação para um componente?

3: Defina o conceito de nó?

4: Qual é a notação para um nó?

5: Que relação pode ser estabelecida entre os componentes?

6: Qual é o objectivo do diagrama de componentes?

7: Qual é o objectivo do diagrama de instalação?

8: Defina o conceito de interface num componente?

9: O que significa a linha de comunicação?

10: Que tipo de informação pode ser modelada pelo diagrama de componentes?

11: Qual é o objectivo da utilização de estereótipos num nó?

12: Qual é o objectivo de utilizar nós e componentes no mesmo diagrama?

8.4.2 Problemas Resolvidos

Desenhe os respectivos diagramas físicos de acordo com as descrições seguintes.

Biblioteca

Considere os seguintes requisitos para o sistema da biblioteca. Estes requisitos foram definidos pelo chefe da equipa de desenvolvimento.

- O sistema seguirá uma arquitectura cliente/servidor através de uma rede local e no protocolo TCP/IP.

© FCA - EDITORA DE INFORMÁTICA

- A biblioteca terá apenas um servidor, que será responsável por efectuar a gestão de empréstimos e publicações.

- Existirá um computador na recepção da biblioteca, que será utilizado pelos seus funcionários para efectuarem o atendimento.

- A biblioteca disponibilizará um conjunto de computadores para consulta e pesquisa de publicações.

- De forma a maximizar o desempenho do sistema, será utilizado um servidor dedicado apenas à base de dados da biblioteca.

- O sistema será dividido em 4 módulos:

a) **Módulo Gestão Empréstimos** - responsável pelas operações de gestão dos empréstimos da biblioteca.

b) **Módulo Gestão de Publicações** - responsável pelas operações de gestão das publicações da biblioteca.

c) **Módulo Atendimento** - responsável pelas operações de atendimento ao público, nomeadamente a criação de novos empréstimos.

d) **Módulo Pesquisa** - responsável pelas operações de pesquisa e consulta na biblioteca.

Parque de Estacionamento

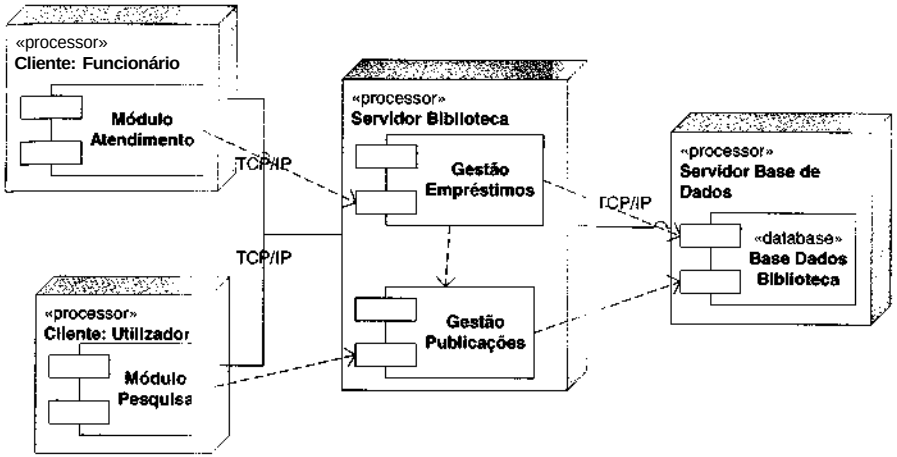
O sistema a instalar no parque de estacionamento será constituído pelos seguintes elementos:

- Servidor: computador responsável por conter o módulo de Gestão dos Estacionamentos e o módulo de Gestão de Veículos. Também conterá a base de dados que suporta ao sistema e estará ligado a um servidor central, que agrega a gestão de diversos parques.

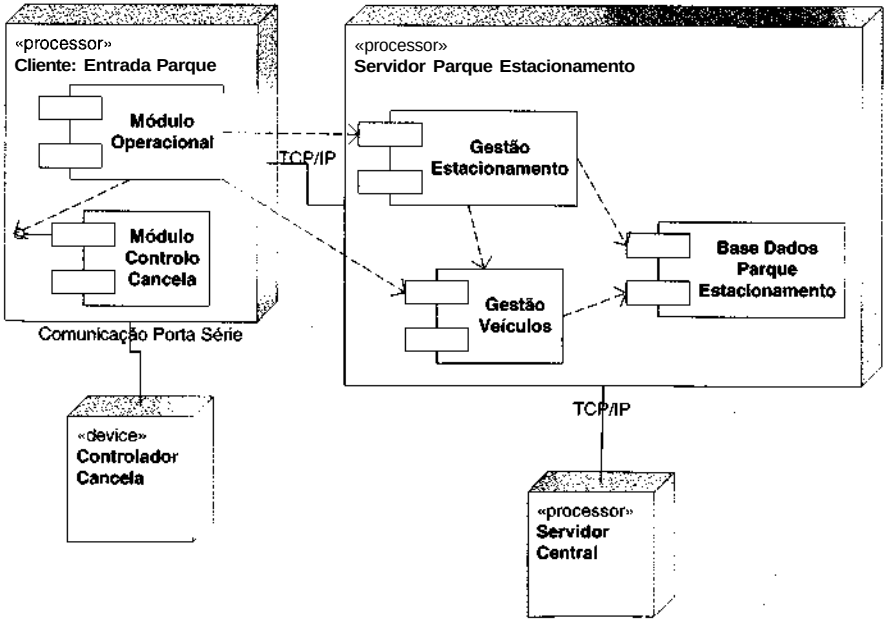
- Cliente: Situado na entrada do parque, este computador conterá o módulo Operacional (regista entradas e saídas) e também o módulo de Controlo de Cancela (interligados através de uma interface específica). Na porta série do computador estará ligado o dispositivo de controlo da cancela.

O Servidor estará ligado a um servidor central que agrega a gestão de diversos parques.

Solução Biblioteca



Solução Parque de Estacionamento



© FCA - EDITORA DE INFORMÁTICA

Processo de **Modelação** 9

9.1 CONCEITO E APLICAÇÃO

Neste capítulo enquadramos a utilização da UML numa metodologia que potencie as capacidades da linguagem e realçamos a importância do uso de ferramentas informáticas de apoio à modelação com UML. Este capítulo conclui-se com a apresentação da MDA - Model Driven Architecture, que constitui uma importante linha de evolução da UML.

As abordagens ao ciclo de vida de sistemas de informação são propostas metodológicas que enquadram as etapas e actividades necessárias ao seu desenvolvimento. As diversas aproximações procuram reduzir o tempo de desenvolvimento do projecto e consequentemente o seu custo. Simultaneamente, procuram contribuir para a produção de sistemas de elevada qualidade e com funcionalidade acrescida, capazes de evoluir continuamente de modo a satisfazer requisitos de utilizadores cada vez mais exigentes.

A UML é uma linguagem aberta e muito rica do ponto de vista semântico, que pode ser utilizada em diferentes enquadramentos metodológicos. No entanto, a necessidade de reforçar a eficiência e eficácia do processo de desenvolvimento aconselha a utilização da UML em conjunto com métodos que aproveitem toda a potencialidade do paradigma dos objectos.

© FCA - EDITORA DE INFORMÁTICA

9.1.1 Orientações para o desenvolvimento

O processo de desenvolvimento de sistemas informáticos de média e grande dimensão deve ter em consideração o seguinte conjunto de orientações:

- Deve ser **incremental,** de modo a que seja possível dominar gradualmente o conhecimento do domínio de aplicação e a funcionalidade exigida, bem como deve comportar a inclusão de novas funcionalidades, numa lógica de melhoria contínua.
- **Deve ser iterativo,** para permitir o desenvolvimento em ciclos sucessivos, disponibilizando versões intercalares do sistema com as quais os utilizadores podem trabalhar e que vão respondendo à satisfação de conjuntos acrescidos de requisitos.
- Deve ser baseado numa **arquitectura de modelação,** que permita caracterizar a estrutura e comportamento do sistema de informação, a sua funcionalidade, o seu nível de desempenho, as interfaces com utilizadores e outros sistemas, as restrições tecnológicas e económicas. Deve permitir também enquadrar os contributos complementares dos diversos participantes no projecto: utilizadores, analistas, programadores, integradores de sistemas, gestores, etc.
- **Deve ser centrado nos Use Cases,** de modo a realçar as funções que o sistema deve proporcionar a um conjunto identificado de potenciais utilizadores (actores).
- Deve permitir o desenvolvimento de **componentes** que possam ser programados e testados de forma autónoma, e reutilizados em diversos sistemas.
- Deve permitir a **gestão de equipas** de dimensão adequada atribuindo responsabilidades por tarefas que possam ser realizadas em paralelo, de modo a reduzir o ciclo temporal do desenvolvimento.
- Deve facilitar a elaboração de **documentação** de utilização e de administração do sistema.

9,2 PROCESSO DE MODELAÇÃO UNIFICADO

9.2.1 Actividades

O desenvolvimento de um sistema de informação exige a concretização de um conjunto de actividades:

- **Modelação de negócio,** descreve a estrutura e a dinâmica da organização, servindo de enquadramento ao sistema de informação.
- **Levantamento de requisitos,** descreve as características, comportamentos ou propriedades desejadas para o sistema pelos potenciais utilizadores.
- **Análise,** descreve o que o sistema deve fazer, com rigor, mas sem restrições quanto à natureza técnica da solução que venha a ser adoptada.
- **Desenho,** descreve a arquitectura do sistema, identificando com elevado detalhe o modo como os requisitos devem ser satisfeitos do ponto de vista técnico.
- **Codificação**, correspondente ao desenvolvimento dos programas e teste unitário.
- **Integração e Teste,** efectua a integração dos diversos módulos de *hardware* e componentes de *software*, e avalia a robustez do sistema recorrendo a métricas de detecção de erros.
- **Instalação,** disponibiliza uma versão operacional do sistema.
- **Gestão da configuração,** inclui as tarefas de manutenção correctiva e evolutiva.

Complementarmente, o sucesso do desenvolvimento de um sistema exige ainda que seja assegurada a realização de actividades de apoio que incluem a **Gestão do projecto, a Gestão da mudança e a Instalação da infra-estrutura.**

FCA - EDITORA DE INFORMÁTICA

9.1.1 Orientações para o desenvolvimento

O processo de desenvolvimento de sistemas informáticos de média e grande dimensão deve ter em consideração o seguinte conjunto de orientações:

» Deve ser **incremental,** de modo a que seja possível dominar gradualmente o conhecimento do domínio de aplicação e a funcionalidade exigida, bem como deve comportar a inclusão de novas funcionalidades, numa lógica de melhoria contínua.

* Deve ser **iterativo,** para permitir o desenvolvimento em ciclos sucessivos, disponibilizando versões intercalares do sistema com as quais os utilizadores podem trabalhar e que vão respondendo à satisfação de conjuntos acrescidos de requisitos.

* Deve ser baseado numa **arquitectura de modelação,** que permita caracterizar a estrutura e comportamento do sistema de informação, a sua funcionalidade, o seu nível de desempenho, as interfaces com utilizadores e outros sistemas, as restrições tecnológicas e económicas. Deve permitir também enquadrar os contributos complementares dos diversos participantes no projecto: utilizadores, analistas, programadores, integradores de sistemas, gestores, etc.

* Deve ser **centrado nos Use Cases,** de modo a realçar as funções que o sistema deve proporcionar a um conjunto identificado de potenciais utilizadores (actores).

* Deve permitir o desenvolvimento de **componentes** que possam ser programados e testados de forma autónoma, e reutilizados em diversos sistemas.

* Deve permitir a **gestão de equipas** de dimensão adequada, atribuindo responsabilidades por tarefas que possam ser realizadas em paralelo, de modo a reduzir o ciclo temporal de desenvolvimento.

* Deve facilitar a elaboração de **documentação** de utilização e de administração do sistema.

9.2 PROCESSO DE MODELAÇÃO UNIFICADO

9.2.1 Actividades

O desenvolvimento de um sistema de informação exige a concretização de um conjunto de actividades:

* **Modelação de negócio,** descreve a estrutura e a dinâmica da organização, servindo de enquadramento ao sistema de informação.

* **Levantamento de requisitos,** descreve as características, comportamentos ou propriedades desejadas para o sistema pelos potenciais utilizadores.

* **Análise,** descreve o que o sistema deve fazer, com rigor, mas sem restrições quanto à natureza técnica da solução que venha a ser adoptada.

* **Desenho,** descreve a arquitectura do sistema, identificando com elevado detalhe o modo como os requisitos devem ser satisfeitos do ponto de vista técnico.

* **Codificação**, correspondente ao desenvolvimento dos programas e teste unitário.

* **Integração e Teste,** efectua a integração dos diversos módulos de *hardware* e componentes de *software*, e avalia a robustez do sistema recorrendo a métricas de detecção de erros.

* **Instalação,** disponibiliza uma versão operacional do sistema.

* **Gestão da configuração,** inclui as tarefas de manutenção correctiva e evolutiva.

Complementarmente, o sucesso do desenvolvimento de um sistema exige ainda que seja assegurada a realização de actividades de apoio que incluem a **Gestão do projecto, a Gestão da mudança e a Instalação da infra-estrutura.**

FCA EDITORA DE INFORMÁTICA

9.2.2 Fases

As abordagens originais ao ciclo de vida, começaram por propor que as actividades de desenvolvimento de sistemas de informação fossem realizadas de um modo predominantemente sequencial, existindo por vezes a possibilidade de iterações entre actividades consecutivas. O mecanismo de controlo de conclusão de uma actividade seria em geral um documento que teria de ser validado antes do projecto transitar para a fase seguinte.

Esta aproximação linear não satisfaz os objectivos de aumento de eficiência e eficácia anteriormente enunciados. Uma aproximação diversa é aquela proposta no *Unified Modelling Process* (Jacobson, 1999), que sugere um modelo de desenvolvimento adequado à UML, no qual a dimensão funcional das actividades se integra ortogonalmente com a dimensão temporal das fases do projecto.

Este processo identifica 4 fases do desenvolvimento:

* **Início,** estabelece o caso de negócio e limita o âmbito do projecto, incluindo critérios de avaliação de sucesso e de risco, estimativa de recursos necessários e um plano de trabalho com as principais etapas, actividades e pontos de controlo. Nesta fase, procede-se ao levantamento de requisitos *(use cases)* e pode-se desenvolver um protótipo simplificado que apoia a tomada de decisão de avançar, ou não, com o projecto.
* **Elaboração,** procura analisar em detalhe o domínio do problema, estabelecer uma arquitectura, desenvolver um plano de projecto e eliminar os factores de risco. Para consolidar o trabalho realizado nesta fase, deve ser desenvolvido um protótipo que suporte os principais *use cases*, que servirá para apoiar a decisão de avançar para a fase de construção.
* **Construção,** procura desenvolver de forma iterativa e incremental um produto que será disponibilizado aos utilizadores. Isto implica detalhar os restantes *use cases* e

critérios de aceitação, refinar o desenho, e completar a codificação e teste aplicação.

* **Transição,** disponibiliza uma versão de teste final da aplicação aos utilizadores finais e procede a algumas afinações de pormenor, de modo a obter a versão de produção do sistema. Após a aceitação e entrada em funcionamento do sistema, deve-se proceder a uma reflexão crítica para avaliação do projecto, de modo a melhorar os métodos utilizados. No caso de se identificar a necessidade de obter uma nova versão do sistema, dá-se início a uma nova iteração do ciclo de desenvolvimento.

A figura 9.1 representa este modelo integrado de actividades e fases.

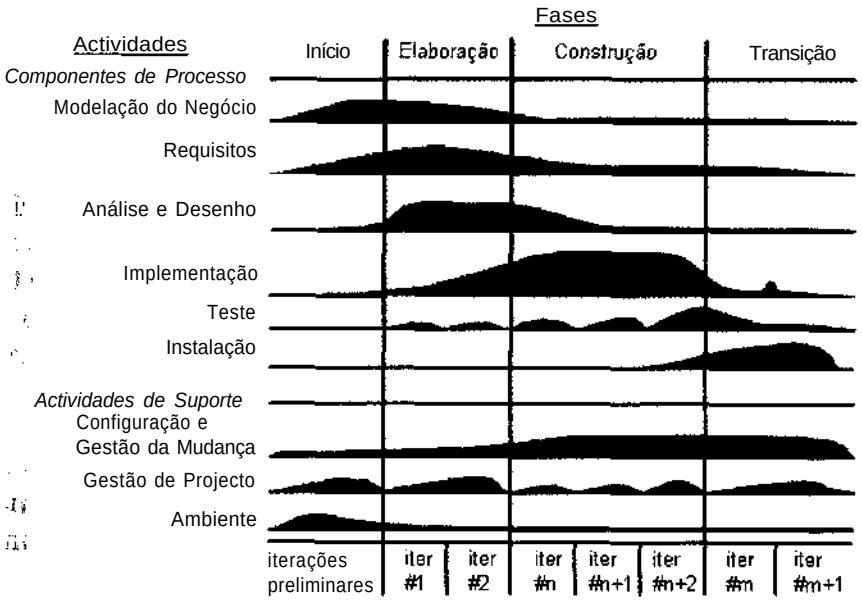


FIGURA 9.1 - PROCESSO DE DESENVOLVIMENTO
(ADAPTADA DE BOOCH ET AL, 1999)

Esta abordagem permite que o processo de desenvolvimento seja incremental. Em cada uma das fases admite-se a ocorrência simultânea de várias actividades, ainda que exista uma que é preponderante. Assim, por exemplo, na fase de Início a actividade principal é a modelação do negócio, seguida do levantamento de requisitos, que se prolongam pela fase de Elaboração. Nesta fase de Elaboração dá-se o reforço das actividades de análise e desenho.

A articulação entre o Método de Modelação Unificado e a UML concretiza-se de um modo prático pela utilização das diversas técnicas diagramáticas da linguagem no âmbito das actividades propostas ao longo do ciclo de vida do desenvolvimento de software, que se inicia com a modelação do negócio e se conclui com a elaboração de documentação de administração do sistema. Estas técnicas são o diagramas de *use cases*, sequência e colaboração, classes, actividades, estados, componentes e instalação.

9.2.3 Arquitectura de modelação

O *Unified Modelling Process* propõe uma forma de organização destes modelos, de acordo com as perspectivas complementares dos diversos intervenientes no processo de desenvolvimento. A esta forma de organização designa-se por arquitectura de modelação e contribui para harmonizar as diversas perspectivas bem como gerir o desenvolvimento do sistema de forma iterativa e incremental.

A figura 9.2 apresenta esta arquitectura de modelação que integra 5 visões ou perspectivas complementares. Cada visão representa uma projecção na organização e estrutura do sistema, centrada num aspecto particular desse sistema.

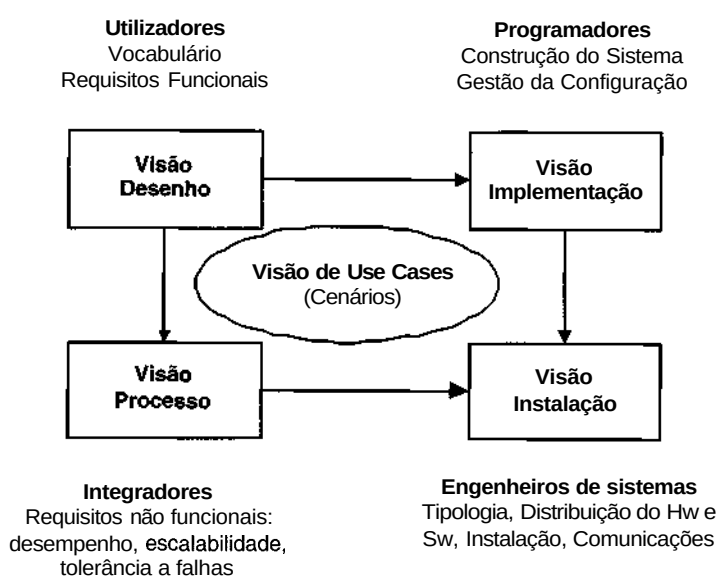


FIGURA 9,2 – ARQUITECTURA 4+1 DE MODELAÇÃO
(ADAPTADA DE BOOCH ET AL, 1999)

- **Visão de *use cases*,** inclui os *use cases* que descrevem o comportamento do sistema de acordo com os seus utilizadores, analistas e avaliadores. Os aspectos estáticos desta visão são descritos através de diagramas de *use case;* os aspectos dinâmicos são detalhados através de diagramas de interacção e de diagramas de actividade.
- **A visão de desenho** define os requisitos funcionais do sistema, isto é, os serviços que serão disponibilizados aos utilizadores. Para tal especificam-se as classes, interfaces e colaborações que constituem o vocabulário do problema. Os aspectos estáticos desta visão são descritos através de diagramas de classes e diagramas de objectos; os aspectos dinâmicos são detalhados através de diagramas de interacção, estado e actividade.
- **A visão do processo** inclui a definição dos fluxos e processos que suportam os mecanismos de concorrência e sincronização.

FCA - EDITORA DE INFORMÁTICA

Endereça ainda as decisões relacionadas com o desempenho e escalabilidade do sistema.

* **A visão de implementação** centra-se na gestão da configuração das diversas versões do sistema, com base em componentes e ficheiros que podem ser integrados para dar origem ao sistema. Os aspectos estáticos desta visão são descritos com diagramas de componentes; os aspectos dinâmicos são descritos através de diagramas de interacção, diagramas de estados e diagramas de actividades.
* **A visão de instalação** define a topologia do equipamento (processadores e terminais) utilizados pelo sistema, incluindo as características e a forma de integração física dos diversos nós, bem como os componentes que são executados nos processadores, associados a processos e fluxos de controlo. Os aspectos estáticos desta visão são descritos através de diagramas de instalação; os aspectos dinâmicos são descritos através de diagramas de interacção, diagramas de estados e diagramas de actividades.

Estas 5 visões interagem entre si. Por exemplo, um processador da visão de instalação suporta diversos programas executáveis e ficheiros fonte que são identificados como componentes na visão de implementação. Estes componentes são a concretização física de classes de objectos identificados na visão de desenho e na visão de processo.

9.2.4 Resultado da Modelação

Do processo de desenvolvimento resulta a criação de um conjunto de modelos:

* **Modelo de Negócio,** estabelece uma representação da organização.
* **Modelo de Domínio,** estabelece o contexto do sistema.

* **Modelo de Use Case,** especifica os requisitos funcionais do sistema.
* **Modelo de Análise (opcional),** define uma concepção geral.
* **Modelo de Desenho,** especifica o vocabulário do sistema e a solução proposta para a arquitectura do sistema.
* **Modelo de processo (opcional),** define os mecanismos de concorrência e sincronização.
* **Modelo de Implementação,** especifica os componentes que constituem o sistema.
* **Modelo de Instalação,** define a topologia do equipamento (*hardware*).
* **Modelo de Teste,** define os critérios para validação e verificação do sistema.

Estes modelos, designados por artefactos, vão sendo gradualmente refinados ao longo das diversas fases do ciclo de vida, permitindo representar, visualizar, especificar, construir e documentar o sistema.

9.3 AO DESENVOLVIMENTO

Tendo presente o ciclo de desenvolvimento proposto que integra actividades e fases, e tomando como referência a arquitectura anteriormente descrita, poderemos sugerir um conjunto de orientações práticas para desenvolver os modelos da UML que descrevem o sistema de informação.

1. Identificar o problema no contexto da organização e caracterizá-lo ainda que sumariamente.
2. Constituir uma equipa de projecto que inclua todos os actores que são relevantes para a resolução do problema: utilizadores, decisores, clientes, fornecedores, analistas, programadores, etc.

Nomear um director de projecto e atribuir responsabilidades aos seus membros.

3. Criar um modelo de negócio onde se descrevam os processos e actividades que são desenvolvidos pela organização através de diagramas de *use case*, diagramas de interacção e diagramas de actividade. Se for conveniente, pode ser desenvolvido um modelo de classes para descrever a informação utilizada na execução desses processos.
4. Validar o modelo de negócio com os membros da equipa de projecto.
5. Identificar aqueles processos de negócio e actividades que irão ser suportados pelo sistema de informação a desenvolver e descrevê-los em pormenor utilizando diagramas de interacção e de actividade. Elaborar um diagrama de classes detalhado, identificando com mais rigor os atributos e as operações. Para classes que possuam comportamento dinâmico relevante, utilizar diagramas de estado para descrever essa dinâmica.
6. Utilizar os modelos obtidos para desenvolver um protótipo que possa ser validado pelos membros da equipa de projecto, no qual esteja patente a interface de utilizador e a funcionalidade dos *use cases* mais relevantes.
7. Refinar os modelos de *use case*, interacção, classes e estados, incorporando as reacções, comentários e sugestões dos membros da equipa ao protótipo apresentado.
8. Elaborar um novo protótipo que incorpore os requisitos dos utilizadores e que se aproxime de uma versão final beta do sistema.

Esta aproximação prática é iterativa, incremental, baseada em *use cases* e coerente com a arquitectura de modelação apresentada neste capítulo. Devemos ter presente que o sucesso de um sistema depende do envolvimento de todos os participantes. Por esse motivo, deve ser promovida a realização de reuniões de trabalho

alargadas, complementadas com entrevistas individuais. A utilização da UML e do *UnifiedModelling Process* asseguram a possibilidade de o sistema poder vir a suportar nova funcionalidade no futuro. Tal facto reduz a pressão para que todos os requisitos tenham de ser identificados como condição prévia de desenvolvimento do sistema.

Uma forma de controlar melhor o processo de desenvolvimento e de reforçar a participação de todos os actores envolvidos passa por uma gestão rigorosa dos requisitos dos utilizadores e dos modelos que representam o sistema, negociando antecipadamente o número de versões dos diagramas e protótipos que devem ser asseguradas em cada uma das fases.

Utilizar reuniões participativas envolvendo os vários intervenientes para levantar requisitos.

9.4 FERRAMENTAS DE MODELAÇÃO COM UML

Estando na posse de uma linguagem de modelação e de um método que enquadre a sua utilização, o aumento da produtividade passa pela utilização de ferramentas informáticas de apoio ao processo de desenvolvimento. Estas ferramentas designam-se por C.A.S.E., que significa *Computer Aided Systems Engineering* ou *Computer Aided Software Engineering.* Nesta classificação poderemos enquadrar um vasto conjunto de aplicações informáticas que são utilizadas com o objectivo de automatizar, tanto quanto possível, as tarefas de desenvolvimento: editores de ficheiros, compiladores e *debuggers;* utilitários de definição e manipulação de dados; geradores de ecrãs e de relatórios; aplicações de gestão de projectos; aplicações de gestão de versões de software; aplicações de teste de software; ambientes integrados de desenvolvimento *(Rapid Application Development*); etc.

© FCA - EDITORA DE INFORMÁTICA

No contexto específico desta publicação, importa referir um tipo particular destas aplicações que são editores gráficos especializados que apoiam o processo de modelação visual em particular aqueles que utilizam a UML. Estas ferramentas de apoio à criação dos diagramas de modelação devem apresentar um conjunto de características distintivas, das quais se destacam:

* Disponibilizar meios para desenhar os diagramas da UML e facilitar as operações de consulta e navegação nos diagramas do modelo.
* Assegurar a integridade da informação através da utilização de um repositório único de dados onde se registam os objectos, modelos, documentos ou quaisquer outros artefactos associados ao modelo.
* Verificar a consistência de notação e dos nomes dos elementos de modelação utilizados nos diversos diagramas.
* Possibilidade de utilizar representações gráficas distintas para os diversos estereótipos.
* Suportar o controlo de versões dos diagramas e facilidades de rastreio das alterações efectuadas.
* Possibilidade de geração de código das aplicações utilizando diversas linguagens de programação (VB, C++, Java, etc.) .
* Apoio à criação de modelos a partir do código fonte das aplicações *(reverse engineering)*.
* Apoio à elaboração de documentação de projecto que descreva os modelos.
* Permitir a integração de modelos criados por vários elementos da equipa de desenvolvimento.
* Disponibilizar formatos normalizados de ficheiros para troca de informação com outras aplicações do mesmo tipo.

* Facilidade de geração de documentação dos modelo em diversos formatos de ficheiro (html, word, etc.) e formatos gráficos.

São inúmeras as vantagens resultantes da utilização destas ferramentas:

* Uniformizar os métodos e práticas de concepção e desenvolvimento de sistemas de informação, alargando-os a todos os membros de uma organização.
* Controlo acrescido dos diagramas produzidos e aumento da possibilidade de reutilização.
* Melhor gestão da informação produzidas no projecto.
* Redução do ciclo de desenvolvimento a par de uma melhoria da qualidade do processo.
* Facilidade de formação dos utilizadores nas regras da linguagem e nos métodos de desenvolvimento utilizados.
* Reforço da comunicação entre os diversos elementos da equipa de projecto.

Não nos podemos esquecer que, até ao momento, o desenvolvimento de sistemas de informação computacionais é uma actividade que utiliza intensivamente as capacidades intelectuais do ser humano. Assim nunca é demais realçar que a utilização de uma ferramenta não substitui as pessoas envolvidas no processo, mas somente potência a sua competência, experiência e empenho.

Seleccionamos duas destas ferramentas para uma breve apresentação: o Rose 2000 da Rational e o Visio 2000 da Microsoft. A razão desta escolha prende-se com o facto de o Rose ser considerado uma referência entre as ferramentas C.A.S.E. compatíveis com a UML. Para além disso o Rose é desenvolvido Pela Rational, a empresa que foi o berço da UML e onde

© FCA - EDITORA DE INFORMÁTICA

actualmente trabalham Booch, Jacobson e Rumbaugh. A escolha do Visio prende-se com a disponibilidade desta ferramenta e a excelente capacidade de edição gráfica. No entanto urna pesquisa na Internet permite identificar inúmeras ferramentas C.A.S.E. compatíveis com a UML, que oferecem um conjunto de funcionalidades muito interessante por um preço acessível.

9.4.1 Rose 2000

O Rose é um produto desenvolvido pela Rational Software Corporation com o objectivo de apoiar o desenvolvimento de soluções eficientes e robustas para arquitecturas cliente/servidor, sistemas distribuídos e sistemas de processamento em tempo real. Uma versão de demonstração deste sistema pode ser obtido através do *site* da empresa em www.rational.com.

Para facilitar a criação de um novo modelo a ferramenta apresenta vários moldes *(templates)*, cada um dos quais disponibiliza um conjunto de elementos necessários para a modelação de um certo tipo de sistema. Alguns destes moldes estão orientados para o desenvolvimento de aplicações em diversas plataformas tecnológicas ou utilizando linguagens de programação específicas: Oracle, VB5, VB6, VC6, JDK12, Jenterprise, etc. Existe um molde para o desenvolvimento de modelo segundo o método UMP.

A figura 9.3 apresenta o ecrã disponibilizado pelo Rose para a criação de modelos. No topo deste ecrã identificamos uma barra horizontal com menus e botões que dão acesso às funções da aplicação. Na zona superior esquerda existe uma janela com diversos pacotes associados às várias visões propostas pelo processo de desenvolvimento UMP, e nos quais deverão ser incluídos os diversos diagramas que fazem parte do modelo a desenvolver. Na zona inferior esquerda do ecrã existe uma janela de diálogo que permite ao utilizador visualizar e editar a descrição do elemento de modelação que estiver seleccionado. Numa barra vertical apresentam-se os símbolos gráficos dos elementos de

modelação, específicos de cada tipo de diagrama, que irão ser editados na janela presente na zona direita do ecrã.

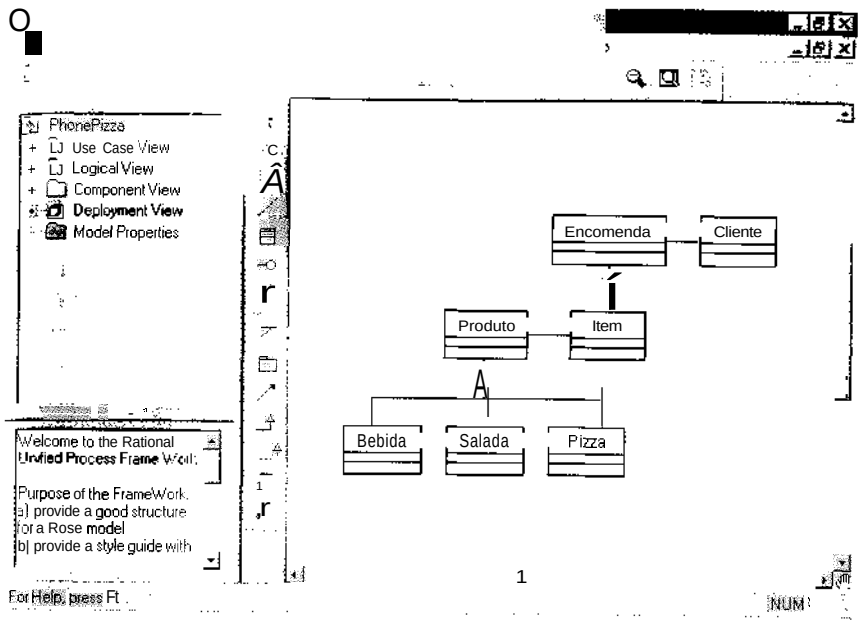


FIGURA 9,3 - RATIONAL ROSE

Na opção de menu designada por *Tools* encontram-se agrupadas um conjunto de funções de grande utilidade num contexto de aplicação mais avançado que incluem nomeadamente, o controlo de versões, a integração de diversos (sub) modelos, a geração de código em diversas linguagens ou a engenharia inversa.

A ferramenta possui facilidades de apoio interactivo (*helponline*) ao utilizador que descrevem detalhadamente as funções disponibilizadas em cada ecrã, bem como noções aprofundadas da UML e dos princípios de modelação.

© FCA - EDITORA DE INFORMÁTICA

9.4.2 Visio 2000

A versão Visio 2000 Enterprise Edition constitui uma ferramenta de desenho, de uso geral e utilização individual, que disponibiliza facilidades para a criação de diagramas UML. Na figura 9.4 apresenta-se a interface de trabalho disponibilizada pelo Visio.

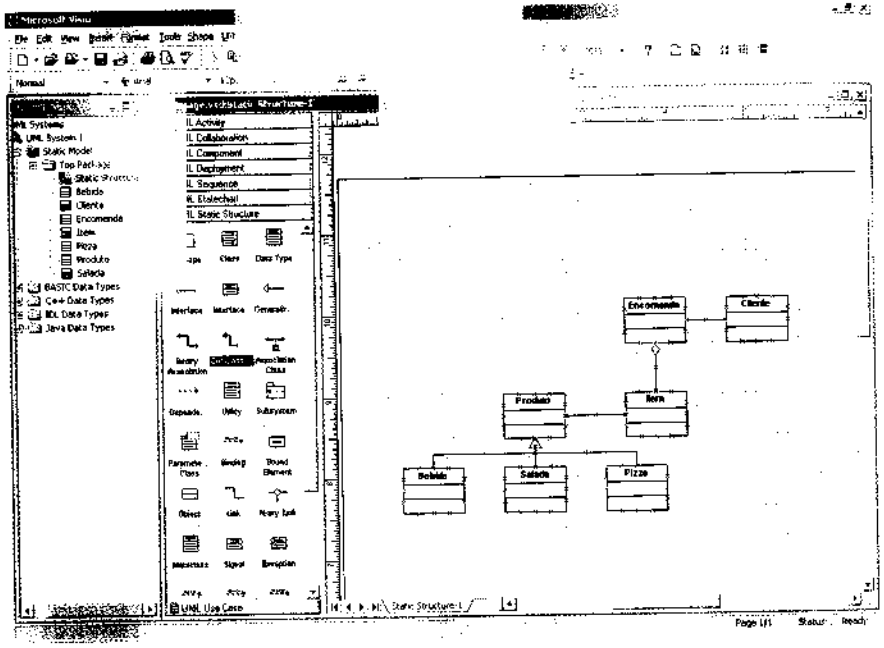


FIGURA 9.4 – VISIO 2000

Um aspecto relevante desta ferramenta é a facilidade de edição dos diagramas. O utilizador pode definir o tipo de diagrama que pretende construir e a aplicação apresenta-lhe uma palete (*stencil*) com os elementos de modelação que pode utilizar nesse contexto.

Outras funcionalidades incluem a criação de modelo a partir da engenharia inversa de bases de dados, a integração de múltiplos diagramas num único repositório e a geração automática do esquema da base de dados. Os utilizadores podem documentar o processo de desenvolvimento de software através dos tipos de

diagramas definidos pelo UML 1.2, criar diagramas de classes através da engenharia inversa de programas em Visual Basic, Visual C++ e Java da Microsoft. Permite também gerar a estrutura de codificação para Visual Basic, Visual C++ e Java da Microsoft a partir do modelo UML e gerar relatórios a partir dos diagramas.

9.5 - ARCHITECTURE

A UML é uma linguagem que permite a visualização, especificação, construção e documentação de sistemas de informação orientados por objectos. Pode ser utilizada com todos os tipos de processos, ao longo de todo o ciclo de desenvolvimento, em diferentes tecnologias de implementação.

O modelo de 3 camadas com a estruturação dos componentes do sistema em torno de serviços de interface, de serviços de negócio e de serviços de dados facilita o processo de adaptação de um sistema para múltiplas interfaces de utilização (janelas, navegador Internet, etc.) ou múltiplos repositórios de dados (ficheiros, sistema de gestão de bases de dados relacional ou orientado por objectos, etc.).

Actualmente, uma área em grande evolução é a dos sistemas de informação distribuídos, associados à crescente adopção das tecnologias Internet e dos serviços que lhe estão associados, designadamente a WWW (World Wide Web). Encontram-se actualmente disponíveis diversas tecnologias que facilitam o desenvolvimento de sistemas distribuídos, como as plataformas ("*middleware*") CORBA, J2EE ou .NET, ou linguagens de estruturação de dados como o XML. O desenvolvimento deste tipo de sistemas constitui um desafio acrescido para todos aqueles que pretendem desenvolver soluções que sejam capazes de vir a incorporar os avanços tecnológicos que entretanto vão surgindo. A MDA - Model Driven Architecture é uma proposta desenvolvida no seio do OMG, que surge como uma resposta a este problema (MDA, 2001).

A MDA constitui um quadro de referência, que inclui um método e uma proposta de arquitectura para especificação e desenvolvimento de sistemas de informação que incorporem um conjunto de serviços distribuídos, que sejam facilmente adaptáveis a diversas plataformas tecnológicas e direccionados para diversos domínios de aplicação (figura 9.5)

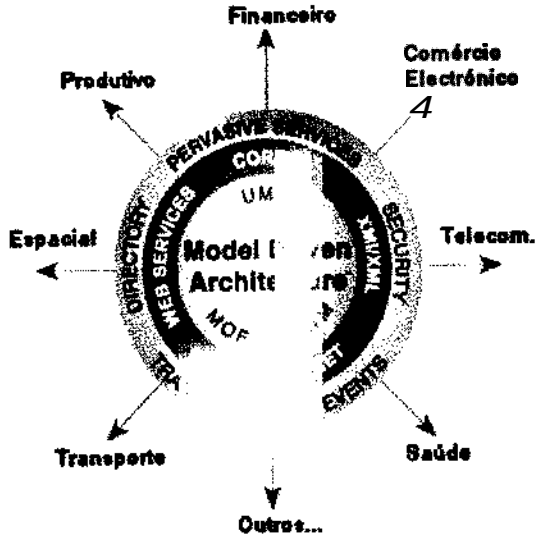


FIGURA 9,5 - MODELO CONCEPTUAL DA MDA
(ADAPTADA DE MDA, 2001)

A abordagem proposta para a estruturação de um sistema de informação baseia-se no desenvolvimento de uma arquitectura em torno de dois grandes grupos de modelos: modelos que descrevem o domínio do problema designados por PIM (Platform Independente Models) e modelos que estabelecem a interface para a plataforma que irá ser utilizada para instalar o sistema, designados por PSM (Platform Specific Models).

A figura 9.6 descreve a estrutura básica desta arquitectura, realçando a multiplicidade de plataformas tecnológicas que poderão ser suportadas.

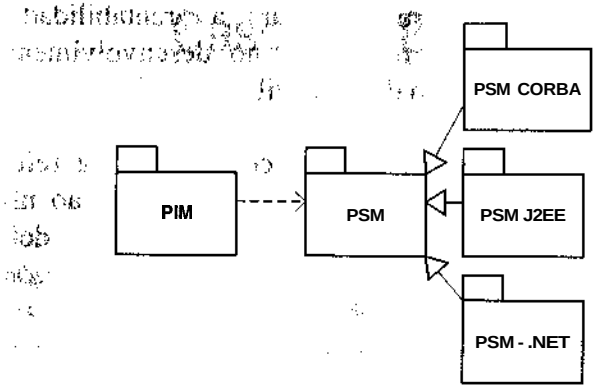


FIGURA 9,6 - MDA (ARQUITECTURA BASEDA EM MODELOS)

Cada um destes modelos é constituído exclusivamente por diagramas UML, complementados por declarações e restrições formais expressas em OCL (Object Constraint Language).

Benefícios resultantes desta aproximação incluem: possibilitar a validação do modelo de negócio expresso no PIM, eliminando a complexidade de aspectos técnicos que são específicos da plataforma; permitir que o mesmo sistema, com uma estrutura e um comportamento bem determinados, seja produzido e implementado em diversas plataformas de uma forma mais simples; facilitar a integração de aplicações legadas e a inter-operacionalidade de sistemas, através da adopção de termos expressos no PEVI que são independentes da plataforma; facilitar a realização de testes de conformidade e melhorar a qualidade dos sistemas informáticos produzidos.

Devemos ter presente que os objectos de informação e as regras de funcionamento de uma organização, descritos no PIM, têm um ritmo de evolução que é normalmente mais lento do que as alterações tecnológicas. A separação entre o domínio de aplicação e as plataformas de desenvolvimento específicas permite assegurar uma evolução gradual do sistema, aumentando a capacidade de este incorporar novas soluções tecnológicas que venham a surgir no

A MDA constitui um quadro de referência, que inclui um método e uma proposta de arquitectura para especificação e desenvolvimento de sistemas de informação que incorporem um conjunto de serviços distribuídos, que sejam facilmente adaptáveis a diversas plataformas tecnológicas e direccionados para diversos domínios de aplicação (figura 9.5)

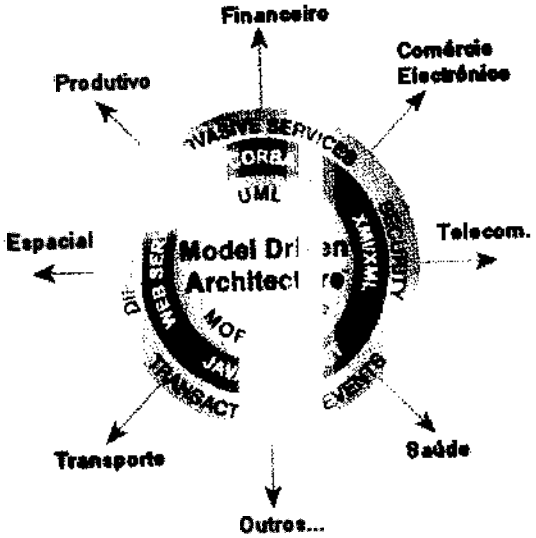


FIGURA 9,5 - MODELO CONCEPTUAL DA MDA
(ADAPTADA DE MDA, 2001)

A abordagem proposta para a estruturação de um sistema de informação baseia-se no desenvolvimento de uma arquitectura em torno de dois grandes grupos de modelos: modelos que descrevem o domínio do problema designados por PIM (Platform Independente Models) e modelos que estabelecem a interface para a plataforma que irá ser utilizada para instalar o sistema, designados por PSM (Platform Specific Models).

A figura 9.6 descreve a estrutura básica desta arquitectura realçando a multiplicidade de plataformas tecnológicas que poderão ser suportadas.

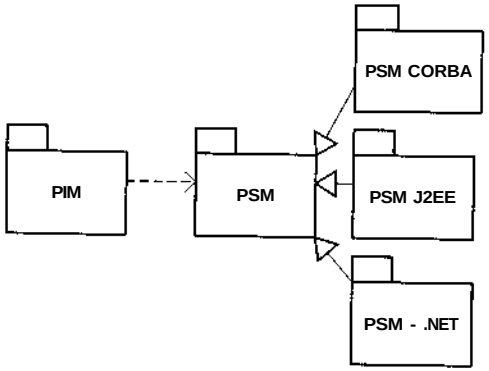


FIGURA 9.6 - MDA (ARQUITECTURA EM MODELOS)

Cada um destes modelos é constituído exclusivamente por diagramas UML, complementados por declarações e restrições formais expressas em OCL (Object Constraint Language).

Benefícios resultantes desta aproximação incluem: possibilitar a validação do modelo de negócio expresso no PIM, eliminando a complexidade de aspectos técnicos que são específicos da plataforma; permitir que o mesmo sistema, com uma estrutura e um comportamento bem determinados, seja produzido e implementado em diversas plataformas de uma forma mais simples; facilitar a integração de aplicações legadas e a inter-operacionalidade de sistemas, através da adopção de termos expressos no PIM que são independentes da plataforma; facilitar a realização de testes de conformidade e melhorar a qualidade dos sistemas informáticos produzidos.

Devemos ter presente que os objectos de informação e as regras de funcionamento de uma organização, descritos no PIM, têm um ritmo de evolução que é normalmente mais lento do que as alterações tecnológicas. A separação entre o domínio de aplicação e as plataformas de desenvolvimento específicas permite assegurar uma evolução gradual do sistema, aumentando a capacidade de este incorporar novas soluções tecnológicas que venham a surgir no

futuro. Assim, consegue-se reforçar a rentabilidade dos significativos investimentos realizados no desenvolvimento dos sistemas e alargar o seu tempo de vida útil.

Complementarmente, a MDA procura contribuir para reforçar o grau de eficiência do processo de desenvolvimento, ao nível da automatização da tarefa de geração de código, definindo orientações que podem ser adoptadas por ferramentas integradas de desenvolvimento.

A MDA constitui um contributo concreto para a resolução de um conjunto fundamental de problemas, que desde sempre tem preocupado toda a comunidade que se dedica ao desenvolvimento de sistemas de informação: que o desenvolvimento seja eficiente, permitindo a redução de custos e o cumprimento de prazos; que o processo seja cada vez mais eficaz, assegurando que as características dos sistemas reflectem os requisitos colocados pelos seus utilizadores; que os sistemas sejam construídos de forma flexível para poderem incorporar as alterações que a evolução irá naturalmente impor, quer no contexto da organização quer no da tecnologia;

Estes têm sido temas recorrentes ao longo da curta história dos sistemas de informação, todavia reforçado pelo facto de o ritmo da mudança tecnológica e organizacional ser contínuo e cada vez mais acentuado.

Integrada no contexto de evolução da modelação visual orientada por objectos, que inclui a UML e o Processo de Modelação Unificado, a MDA representa o mais recente contributo para responder aos desafios que se colocam ao desenvolvimento de sistemas de informação. Igualmente promissora, a MDA deve ser acompanhada com particular atenção.

© FCA - EDITORA DE INFORMÁTICA

Casos de Estudos 10

Neste capítulo apresentamos dois casos de estudo. Com o caso PhonePizza, procuramos sistematizar de fornia integrada os diversos exemplos que foram sendo apresentados ao longo do livro. O caso SIUniversitas surge num domínio de aplicação reconhecido por grande parte dos leitores a quem esta obra se destina, o que simplifica a compreensão do problema, facilita a identificação de novos requisitos e permite a exploração de soluções alternativas.

Os diagramas que se apresentam são simplificados. Não foram explorados todos os aspectos que seriam requeridos para o desenvolvimento completo destes sistemas de informação, que apresentam um razoável nível de complexidade. O grau de detalhe apresentado tem apenas em consideração os objectivos de natureza pedagógica que se pretendem atingir, estando sujeitos ao formato da publicação que limita a representação de diagramas mais abrangentes.

Devemos realçar que este capítulo apresenta uma proposta de resolução possível para os problemas enunciados. Outras soluções alternativas são igualmente válidas. Os diagramas apresentados são ainda passíveis de serem refinados através de sucessivas iterações.

10.1 PhonePizza

Considere que se pretende desenvolver um sistema de informação que tem como objectivo apoiar a gestão de encomendas de um grupo de pizzarias, a PhonePizza. Este sistema irá prestar serviços de atendimento, acompanhamento de clientes e encomendas. Para tal o grupo PhonePizza decidiu constituir uma equipa dentro do seu

departamento de sistemas de informação para concretizar o projecto.

10.1.1 Modelo Negócio

A nova organização da PhonePizza será constituída por 4 unidades organizativas: Central, Centro de Chamadas, Internet e Pizzaria.

Na figura 10.1 utiliza-se um diagrama de objectos para representar esta estrutura organizacional.

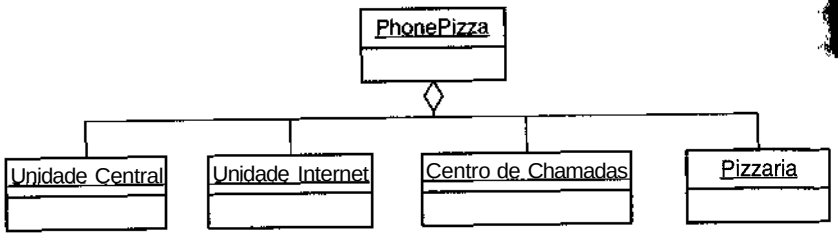


FIGURA 10.1 – ESTRUTURA ORGANIZATIVA DA PHONEPIZZA

Unidade Central

Esta unidade é responsável pelo controlo de gestão da PhonePizza, recebendo regularmente os dados (encomendas) gerados pelas restantes unidades.

Unidade Internet

Disponibiliza serviços de encomenda e consulta de produtos pela Internet. Para encomendar, os clientes efectuam um pré-registo onde indicam alguns dados pessoais.

Centro de Chamadas

Disponibiliza serviços de encomenda e consulta de produtos pelo telefone.

Pizzaria

Efectua o atendimento ao público e satisfaz as encomendas recebidas localmente, pela Internet ou pelo telefone.

Uma pizzaria é composta pelas seguintes áreas: Sala Restaurante (onde estão as mesas), Balcão, Entregas, Cozinha e Armazém. Um restaurante possui um número variável de mesas numeradas que se podem encontrar num dos seguintes estados: LIVRE, OCUPADA, INDISPONÍVEL.

As mesas podem ser reservadas por vários clientes, sendo a reserva referenciada por um número e data. Um cliente pode reservar várias mesas.

Os funcionários distribuem-se pelas seguintes categorias: Gestor de Loja, Empregado de Mesa, Empregado de Balcão, Gestor de Encomendas e Estafeta. Todos os empregados são identificados por um número e podem trabalhar em várias lojas, sendo importante saber a data e a duração do contrato de trabalho em cada uma das lojas.

As diversas pizzarias/lojas do grupo são caracterizadas através de um código, nome, descrição e zona de influência.

Catálogo de Produtos

O catálogo consiste numa listagem de produtos por código, nome, descrição e preço. O catálogo contém também as promoções do mês.

Código	Nome	Descrição	Preço
0001	Pizza Marguerita	Pizza Base	3 € (P) 4,8 € (M) 6 € (F)
0002	Pizza Barbeque	Pizza Base + Molho Barbeque + Mozzarella + Frango + Bacon	3,75 € (P) 5,5 € (M) 7 € (F)
0003	Mexicana	Pizza Base + Molho	3,75 € (P)

© FCA - EDITORA DE INFORMÁTICA

		Mexicano + Mozzarella + Carne + Cebola	5,5 € (M) 7 € (F)
0004	Tropical	Pizza Base + Ananás + Extra Queijo + Fiambre	4 € (P) 6 € (M) 7,5 € (F)
0005	Pizza Composta	Pizza Base + Ingredientes à escolha	Preço da Pizza Base mais preço ingredientes.
0006	Ingrediente	Preço único, variando apenas no tamanho da pizza.	0,5 € (P) 0,75 € (M) 0,95 € (F)

Encomenda

O procedimento de satisfação de uma encomenda é semelhante quando é efectuada na Internet, por telefone ou na pizzaria, só variando a fornia como a informação é apresentada ao cliente (papel ou formato electrónico).

Os produtos disponíveis são pizzas, bebidas, saladas e ingredientes, podendo ser vendidos individualmente ou agrupados num menu. Uma pizza é constituída por uma pizza base (Marguerita) com 3 tamanhos (Pequena, Média, Familiar), onde podem ser adicionados até 10 ingredientes. Existem também pizzas pré-definidas (isto é Mexicana, Barbeque, etc.) que contêm determinados ingredientes. Todos os produtos possuem um código, nome, descrição e preço.

Uma encomenda contém vários produtos em diferentes quantidades. Assim que é inserida no sistema (sendo identificada por um número e tipo: "Código da Pizzaria"; "Internet"; "Telefone") é desencadeado o seguinte procedimento:

1. A loja da área do cliente recebe a encomenda no seu terminal (quando não efectuada na pizzaria), que é adicionada à lista de encomendas a satisfazer. O estado inicial de uma encomenda é NOVA.

2. Assim que a loja começa a satisfazer a encomenda (confeccionar a pizza, juntar o menu, etc.), o estado da encomenda passa a PROCESSO.
3. Quando os itens da encomenda estão todos prontos para serem entregues, dependendo a entrega do tipo de encomenda, esta será entregue por estafeta ou pelo empregado de mesa. O empregado que estiver livre no momento encarrega-se de efectuar a sua entrega, passando o estado da encomenda a CAMINHO.
4. Caso seja uma entrega ao domicílio, é gerada a factura correspondente à encomenda, caso contrário, a factura só é gerada quando o cliente a solicitar. Uma factura é caracterizada por possuir um número, data, itens de produto com valor unitário e valor total.
5. Por fim, após a entrega da encomenda é registado que esta foi entregue (estado ENTREGUE).

Este mecanismo de estados permite aos clientes saberem num dado momento o estado da sua encomenda. O gestor de encomendas consulta as encomendas, sendo a sua pesquisa orientada para a satisfação das seguintes necessidades:

* Saber a quantidade de encomendas efectuadas por área, incluindo o telefone e morada do cliente.
* Conhecer as encomendas efectuadas por cliente.

10.1.2 Modelo de Domínio

Seguindo a orientação do modelo de negócio da PhonePizza organizámos o sistema de informação em 4 subsistemas: Central, Telefone, Internet e Pizzaria (figura 10.2).

Com esta arquitectura pretende-se que cada um dos subsistemas funcione com um elevado grau de autonomia, podendo continuar a

		Mexicano + Mozzarella + Carne + Cebola	5,5 € (M) 7 € (F)
0004	Tropical	Pizza Base + Ananás + Extra Queijo + Fiambre	4 € (P) 6 € (M) 7,5 € (F)
0005	Pizza Composta	Pizza Base + Ingredientes à escolha	Preço da Pizza Base mais preço ingredientes.
0006	Ingrediente	Preço único, variando apenas no tamanho da pizza.	0,5 € (P) 0,75 € (M) 0,95 € (F)

Encomenda

O procedimento de satisfação de uma encomenda é semelhante quando é efectuada na Internet, por telefone ou na pizzaria, só variando a forma como a informação é apresentada ao cliente (papel ou formato electrónico).

Os produtos disponíveis são pizzas, bebidas, saladas e ingredientes, podendo ser vendidos individualmente ou agrupados num menu. Uma pizza é constituída por uma pizza base (Marguerita) com 3 tamanhos (Pequena, Média, Familiar), onde podem ser adicionados até 10 ingredientes. Existem também pizzas pré-definidas (isto é Mexicana, Barbeque, etc.) que contêm determinados ingredientes. Todos os produtos possuem um código, nome, descrição e preço.

Uma encomenda contém vários produtos em diferentes quantidades. Assim que é inserida no sistema (sendo identificada por um número e tipo: "Código da Pizzaria"; "Internet"; "Telefone") é desencadeado o seguinte procedimento:

1. A loja da área do cliente recebe a encomenda no seu terminal (quando não efectuada na pizzaria), que é adicionada à lista de encomendas a satisfazer. O estado inicial de uma encomenda é NOVA.

2. Assim que a loja começa a satisfazer a encomenda (confeccionar a pizza, juntar o menu, etc.), o estado da encomenda passa a PROCESSO.
3. Quando os itens da encomenda estão todos prontos para serem entregues, dependendo a entrega do tipo de encomenda, esta será entregue por estafeta ou pelo empregado de mesa. O empregado que estiver livre no momento encarrega-se de efectuar a sua entrega, passando o estado da encomenda a CAMINHO.
4. Caso seja uma entrega ao domicílio, é gerada a factura correspondente à encomenda, caso contrário, a factura só é gerada quando o cliente a solicitar. Uma factura é caracterizada por possuir um número, data, itens de produto com valor unitário e valor total.
5. Por fim, após a entrega da encomenda é registado que esta foi entregue (estado ENTREGUE).

Este mecanismo de estados permite aos clientes saberem num dado momento o estado da sua encomenda. O gestor de encomendas consulta as encomendas, sendo a sua pesquisa orientada para a satisfação das seguintes necessidades:

* Saber a quantidade de encomendas efectuadas por área, incluindo o telefone e morada do cliente.
» Conhecer as encomendas efectuadas por cliente.

10.1.2 Modelo de Domínio

Seguindo a orientação do modelo de negócio da PhonePizza organizámos o sistema de informação em 4 subsistemas: Central, Telefone, Internet e Pizzaria (figura 10.2).

Com esta arquitectura pretende-se que cada um dos subsistemas funcione com um elevado grau de autonomia, podendo continuar a

© FCA - EDITORA INFORMÁTICA

assegurar um número significativo de serviços mesmo que se interrompa a comunicação com os restantes subsistemas.

Cada um dos subsistemas terá uma base de dados própria, que contém a informação replicada necessária à realização das operações locais. Esta arquitectura exige a comunicação através de mensagens (transacções em XML) para a realização dos *use case* que exigem a participação dos vários subsistemas.

É de referir que a redundância de informação é plenamente justificada pelo aumento de eficiência na operação de consulta efectuada localmente em cada um dos subsistemas, e pela possibilidade de funcionamento autónomo de cada um desses subsistemas.

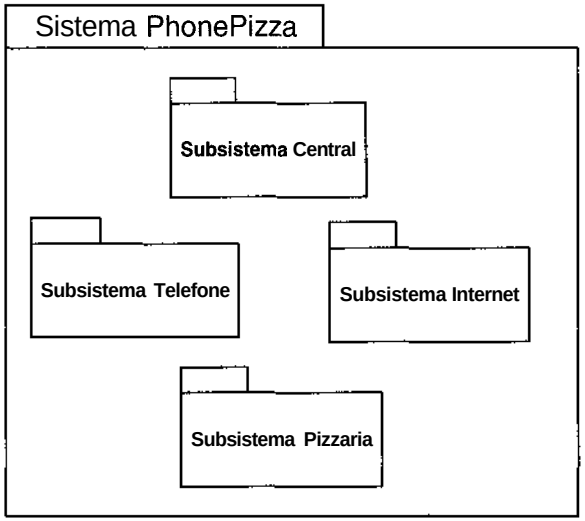


FIGURA 10.2 - MODELO DE DOMÍNIO PHONEPIZZA

Subsistema Central

A função principal deste subsistema é centralizar e gerir toda a informação gerada no processo de encomenda. Logo, deve manter informação actualizada sobre produtos, clientes, pizzarias,

funcionários e encomendas, interagindo com os outros subsistemas através de um mecanismo de actualização regular por troca de mensagens.

Subsistema Centro de Chamadas

Nas encomendas telefónicas, o cliente tem que se identificar através da utilização do número de telefone e morada. Em seguida, é verificado se existe alguma loja que distribua para aquela área; caso contrário, não se aceita o pedido. Uma loja distribui exclusivamente para uma área, pelo que só existe uma loja por área.

A área é caracterizada como a zona de influência de uma pizzaria, que neste caso é identificada pelos vários códigos postal das localidades onde distribui. Admite-se que os colaboradores do Centro de Chamadas confirmam se um dado cliente pertence à zona de distribuição.

Subsistema Internet

No caso de encomendas através da Internet, o utilizador tem que efectuar um pré-registo (indicando imediatamente um Username, Password, Telefone e Morada), sendo confirmado através de um código de acesso que será enviado através de uma mensagem de correio electrónico. O código de acesso será utilizado para activar os serviços disponibilizados na Internet. Após a activação dos serviços, o código de acesso não será jamais utilizado.

Assim que o cliente recebe o código de acesso, poderá efectuar encomendas através do seu *Username* e *Password*. No sistema todos os clientes serão identificados pelo seu número de telefone.

© FCA - EDITORA DE INFORMÁTICA

Subsistema Loja

A encomenda na *pizzaria* abrange todos os pedidos efectuados pessoalmente pelo cliente, nos serviços de balcão e mesa. No serviço ao balcão, o cliente efectua o seu pedido a um empregado do atendimento que se encarregará de introduzir o pedido no sistema.

Para efectuar o pedido na mesa, é disponibilizado um terminal do sistema com um ecrã táctil em cada mesa, onde o cliente terá acesso ao catálogo de produtos e ao conteúdo da sua encomenda.

O sistema dará apoio ao acompanhamento do procedimento de entrega ao domicílio.

10.1.3 Modelo de *Use Cases*

Actores

Existem os seguintes actores que interagem com o sistema de informação do grupo PhonePizza:

- **Cibernauta** -todo aquele que, através da Internet, consulta as páginas do grupo PhonePizza;
- **Cliente** - é a pessoa que encomenda produtos e/ou reserva mesas;
- **Colaborador do Centro de Chamadas** - é o funcionário da PhonePizza que atende as chamadas telefónicas dos clientes no centro de chamadas;
- **Estafeta** - é o funcionário da PhonePizza que se desloca a casa dos cibernautas e clientes para entregar os produtos;
- **Empregado** - é o funcionário da PhonePizza que atende os clientes na loja, quer no balcão: Empregado de Balcão, ou na mesa: Empregado de Mesa;
- **Gestor de Loja** - é o funcionário da PhonePizza responsável por gerir uma loja do grupo.

- **Gestor de Encomendas** - é o funcionário da PhonePizza responsável por gerir o processamento das encomendas;

Para além destes actores humanos identifica-se ainda um outro tipo de actores, que são os diversos subsistemas que constituem o sistema PhonePizza.

- **Subsistema Central**
- **Subsistema Internet**
- **Subsistema Centro de Chamadas**
- **Subsistema Pizzaria**

Use cases

Tomando como referencia cada um dos actores identificam-se os *use cases* em que participam:

Do Cibernauta:
- Consultar Catálogo de Produtos
- Consultar Promoções
- Efectuar pré-registo por Internet

Do Cliente:
- Efectuar Encomenda
- Efectuar Encomenda por Telefone
- Efectuar Encomenda por Internet
- Efectuar Encomenda na Pizzaria
- Efectuar Encomenda na Pizzaria--Balcão
- Efectuar Encomenda na Pizzaria--Mesa
- Consultar Catálogo de Produtos
- Consultar Promoções
- Activar Serviço Internet
- Consultar Conteúdo de Encomenda
- Consultar Estado da Encomenda
- Reservar Mesa
- Emitir Factura

Do Colaborador do Centro de Chamadas:
- Consultar Catálogo de Produtos
- Consultar Promoções

Do Empregado (Mesa ou Balcão):
- ***Registar Entrega Completa***

Do Empregado de Balcão:
- Consultar Catálogo de Produtos
- Consultar Promoções

Do Gestor de Loja:
- Consultar Encomendas
- Consultar Encomendas por Área
- Consultar Encomendas por Cliente

Do Gestor de Encomendas:
- Alterar Estado Encomenda

© FCA - EDITORA DE INFORMÁTICA

Subsistema Loja

A encomenda na pizzaria abrange todos os pedidos efectuados pessoalmente pelo cliente, nos serviços de balcão e mesa. No serviço ao balcão, o cliente efectua o seu pedido a um empregado do atendimento que se encarregará de introduzir o pedido no sistema.

Para efectuar o pedido na mesa, é disponibilizado um terminal do sistema com um ecrã táctil em cada mesa, onde o cliente tem acesso ao catálogo de produtos e ao conteúdo da sua encomenda.

O sistema dará apoio ao acompanhamento do procedimento de entrega ao domicílio.

10.1.3 Modelo de *Use Cases*

Actores

Existem os seguintes actores que interagem com o sistema de informação do grupo PhonePizza:

* **Cibernauta** -todo aquele que, através da Internet, consulta as páginas do grupo PhonePizza;
* **Cliente** - é a pessoa que encomenda produtos e/ou reserva mesas;
* **Colaborador do Centro de Chamadas** - é o funcionário da PhonePizza que atende as chamadas telefónicas dos clientes no centro de chamadas;
* **Estafeta** - é o funcionário da PhonePizza que se desloca a casa dos cibernautas e clientes para entregar os produtos;
* **Empregado** - é o funcionário da PhonePizza que atende os clientes na loja, quer no balcão: Empregado de Balcão, ou na mesa: Empregado de Mesa;
* **Gestor de Loja** - é o funcionário da PhonePizza responsável por gerir uma loja do grupo.

* **Gestor de Encomendas** - é o funcionário da PhonePizza responsável por gerir o processamento das encomendas;

Para além destes actores humanos identifica-se ainda um outro tipo de actores, que são os diversos subsistemas que constituem o sistema PhonePizza.

* **Subsistema Central**
* **Subsistema Internet**
* **Subsistema Centro de Chamadas**
* **Subsistema Pizzaria**

Use cases

Tomando como referencia cada um dos actores identificam-se os *use cases* em que participam:

Do Cibernauta:
* Consultar Catálogo de Produtos
* Consultar Promoções
* Efectuar pré-registo por Internet

Do Cliente:
* Efectuar Encomenda
* Efectuar Encomenda por Telefone
* Efectuar Encomenda por Internet
* Efectuar Encomenda na Pizzaria
* Efectuar Encomenda na Pizzaria--Balcão
* Efectuar Encomenda na Pizzaria--Mesa
* Consultar Catálogo de Produtos
* Consultar Promoções
* Activar Serviço Internet
* Consultar Conteúdo de Encomenda
* Consultar Estado da Encomenda
* Reservar Mesa
* Emitir Factura

Do Colaborador do Centro de Chamadas:
* Consultar Catálogo de Produtos
* Consultar Promoções

Do Empregado (Mesa ou Balcão):
* ***Registar Entrega Completa***

Do Empregado de Balcão:
* Consultar Catálogo de Produtos
* Consultar Promoções

Do Gestor de Loja:
* Consultar Encomendas
* Consultar Encomendas por Área
* Consultar Encomendas por Cliente

Do Gestor de Encomendas:
* Alterar Estado Encomenda

© FCA - EDITORA DE INFORMÁTICA

Do subsistema Central
* Actualização de Dados

Do subsistema Internet
* Efectuar pré-registo
* Activar serviços central
* Enviar encomenda pizzaria
* Consultar Catálogo de Produtos

Do subsistema Telefone
* Enviar encomenda pizzaria

Do subsistema Pizzaria/Loja
* Satisfazer encomenda
* Enviar encomendas do dia
* Consultar Catálogo de Produtos

Diagrama de *use cases*

Nesta apresentação optamos por elaborar diagramas de *use cases* para cada um dos subsistemas. É de notar que nos diagramas de *use case* aparecem representados como actores os restantes subsistemas com os quais o subsistema comunica.

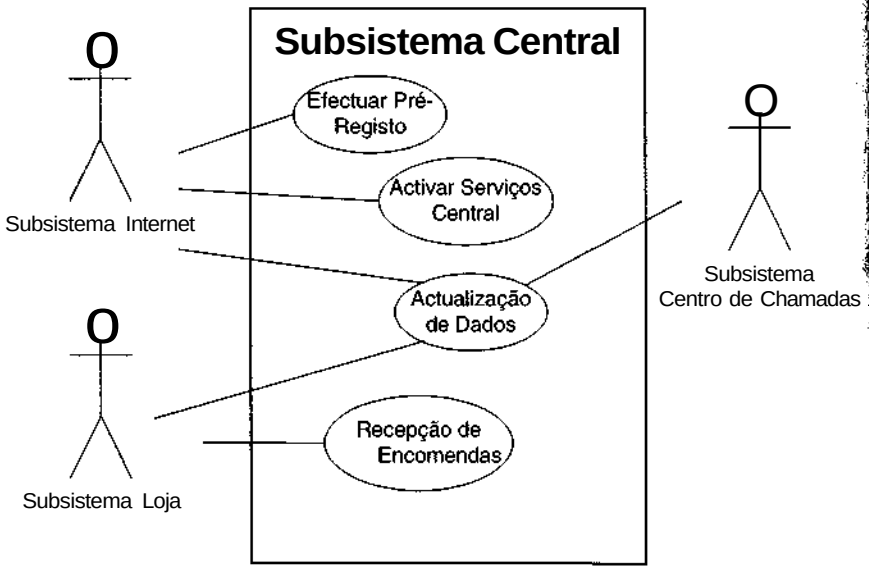


FIGURA 10.3 - DIAGRAMA *USE CASE* SUBSISTEMA CENTRAL

© FCA - EDITORA DE INFORMÁTICA

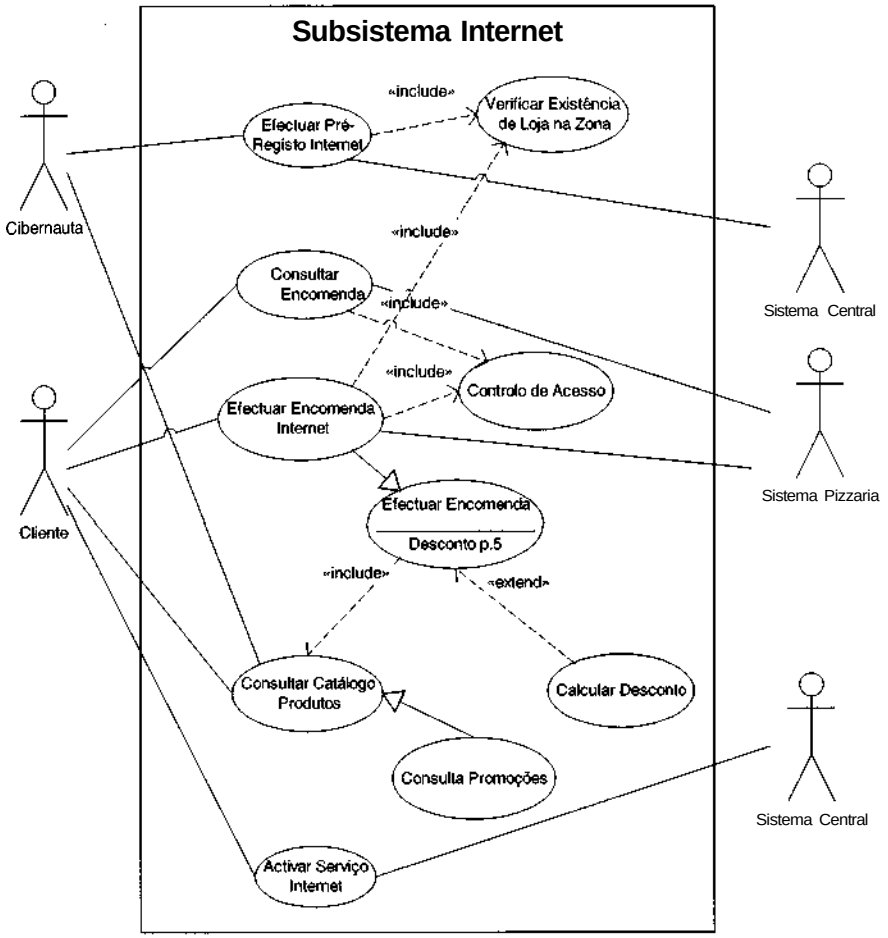


FIGURA 10,4 - SUBSISTEMA INTERNET

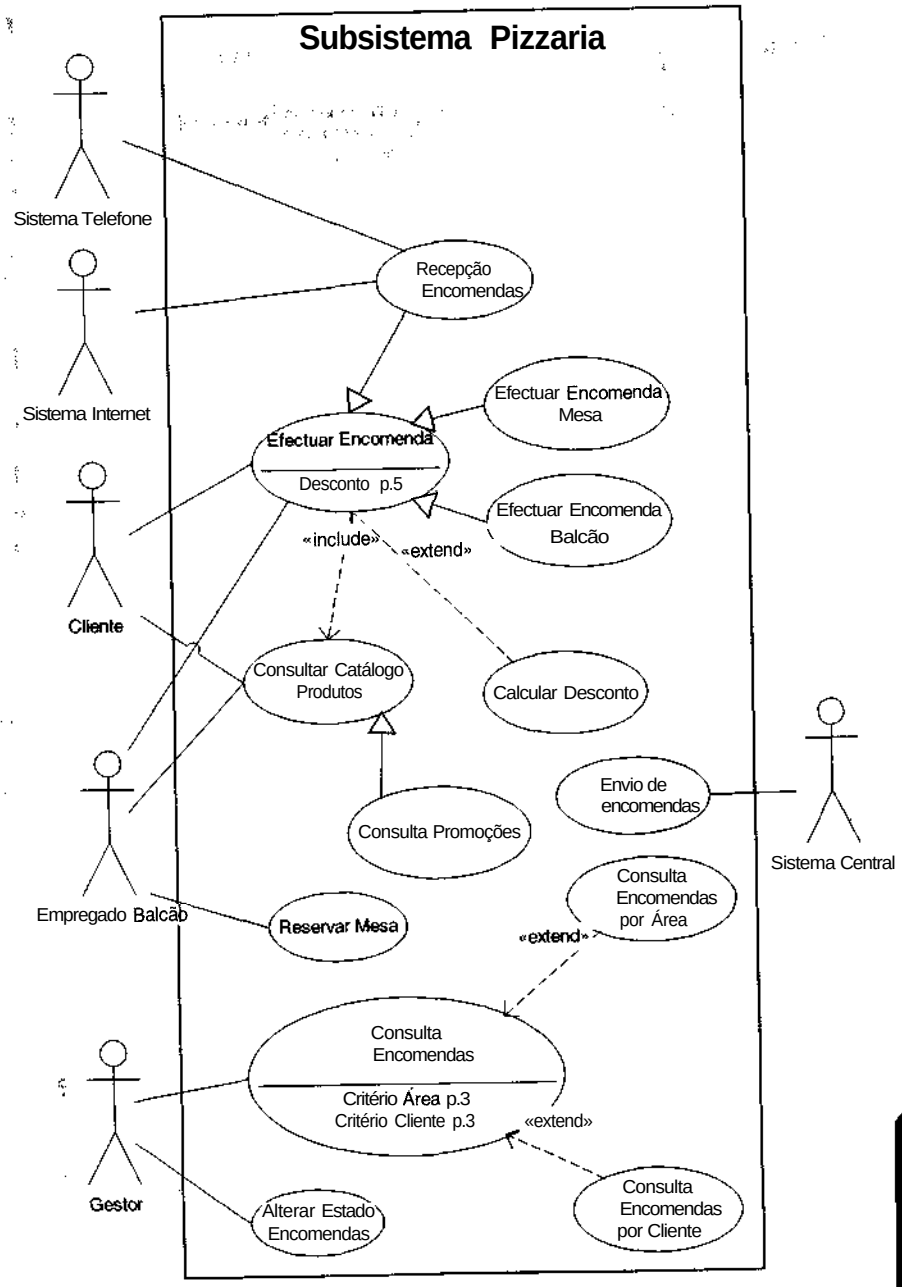


FIGURA 10,5 - SUBSISTEMA PIZZARIA

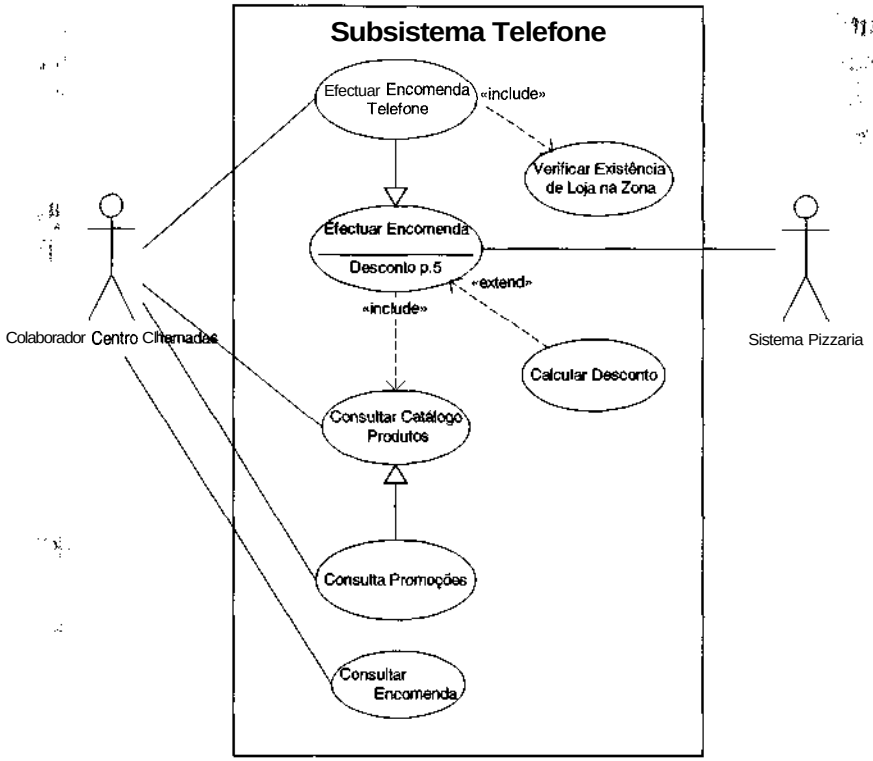


FIGURA 10,6 - SUBSISTEMA TELEFONE

Descrição dos *use cases*

Nesta secção, são descritos alguns dos *use cases* que serão posteriormente detalhados através de diagramas de sequência e actividade. As restantes descrições encontram-se no Anexo II.

Alguns dos use cases são internos a cada um dos subsistemas. Por exemplo a consulta do catálogo de produtos pode ser feita através de um ecrã de utilizador disponibilizado pelo subsistema da Internet, e utilizando informação residente na base de dados local sobre produtos e promoções. Existe um *use case* semelhante para permitir a consulta do catálogo na pizzaria através do terminal, mas

© FCA - EDITORA DE INFORMÁTICA

utilizando informação residente na base de dados local do subsistema da pizzaria. Por este motivo a descrição destes dois *use cases* será muito semelhante, sugerindo a possibilidade de reaproveitamento de diagramas.

Existe um outro conjunto de *use cases* que implicam comunicação entre 2 ou mais subsistemas. Nesta situação encontra-se o pré-registo que envolve comunicação entre o subsistema Internet e o subsistema central. A aceitação de uma encomenda pela Internet é um *use case* que exige o envio de uma mensagem do subsistema Internet para o subsistema da Pizzaria. Por esse motivo estes *use cases* encontram-se identificados em ambos os subsistemas, ainda que a sua descrição detalhada e representação utilizando um diagrama de sequência sejam distintas.
A título de exemplo apresentamos a descrição de 3 *use cases* associados ao subsistema de Internet.

- Consultar catálogo de produtos na Internet
- Efectuar pré-registo pela Internet
- Efectuar Encomenda na Internet

Estes *use cases* serão posteriormente representados através de diagramas de sequência.

Use Case: ***Consultar catálogo de produtos na Internet***

1. O Cibernauta, o Cliente, o Colaborador do Centro de Chamadas e o Empregado de Balcão utilizam o sistema de informação para consultar o catálogo de produtos.
2. O catálogo de produtos deverá ser apresentado sob a forma de uma listagem de produtos com a seguinte informação: código, nome, descrição e preço.
3. Caso seja pretendida a consulta das promoções do mês, então é utilizado o **caso específico:** *Consultar Promoções*

Use Case: ***Efectuar pré-registo por Internet***

1. O Cibernauta utiliza o sistema de informação através da página Internet da *PhonePizza* para efectuar o pré-registo, requisitando um código de acesso.
2. O Cibernauta tem que indicar: *username*, *password*, telefone e morada.
3. **Includes** *Verificar a Existência de Loja numa Zona*

Pós-condição: É gerado um código de acesso que será enviado por correio electrónico.

Use Case: ***Efectuar Encomenda na Internet***

1. O Cliente utiliza o sistema de informação através da página Internet da *PhonePizza* para efectuar a encomenda.
2. O cliente fornece os seus dados identificativos e estes são validados.
3. Se pretender, o Cliente pode consultar:
   a. o catálogo de produtos: **Includes** *Consultar Catálogo de Produtos*
   b. e/ou as promoções do mês: **Includes** *Consultar Promoções*
4. O Cliente escolhe o(s) produto(s) que pretende, (indicando o código ou o nome do produto e a quantidade).
5. Para cada produto escolhido, o sistema verifica o seu preço e é adicionado ao custo total da encomenda.
6. Se o produto está em promoção, existindo assim um desconto:
   a. Extends *Calcular Desconto.*
7. O produto é adicionado aos itens da encomenda.
8. O Cliente confirma a Encomenda
9. A Encomenda é transmitida para a Pizzaria da zona da morada do cliente.

© FCA - EDITORA DE INFORMÁTICA

Diagramas de Sequência

Apresentamos nesta secção os diagramas de sequência dos *use cases* descritos anteriormente.

Na figura 10.5 apresenta-se o diagrama de sequência que descreve a Consulta de Catálogo. No diagrama de sequência encontram-se representados os objectos de interface, os objectos de negócio e os objectos de dados. Ainda que a maioria dos objectos de negócio de um sistema de informação sejam persistentes, em geral omitimos a representação dos correspondentes objectos de dados para evitar sobrecarregar os diagramas.

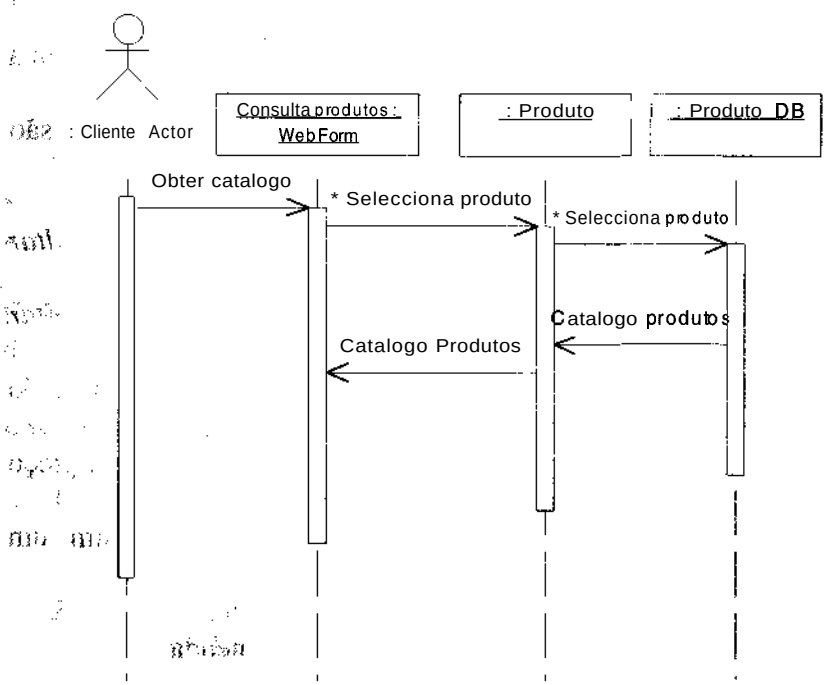


FIGURA 10.7 – CONSULTA CATÁLOGO

Em *use cases* onde têm de ser tomadas decisões complexas utilizamos um objecto de controlo que assegura o acompanhamento

do fluxo de execução e valida as regras de negócio relevantes. Por exemplo, na descrição de Efectuar Encomenda pela Internet surge um objecto designado por Gestor de Encomendas que desempenha essa tarefa.

Para descrever um *use case* onde seja necessário assegurar comunicação com os outros subsistemas através do envio de mensagens utilizamos um objecto de controlo, designado por Gestor de Transacções.

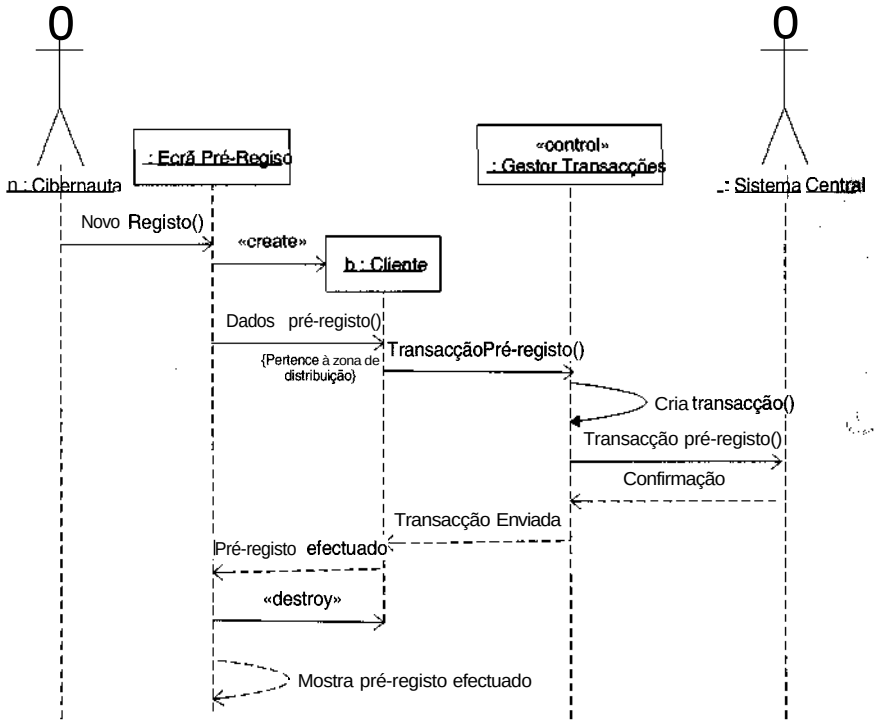


FIGURA 10,8 – PRÉ-REGISTO CLIENTE

Para facilitar a representação do diagrama de sequência que descreve a satisfação de uma encomenda, optamos por decompô-lo em dois subdiagramas: um que descreve a recepção da informação

da encomenda e um outro que descreve a operação de envio para o subsistema Central.

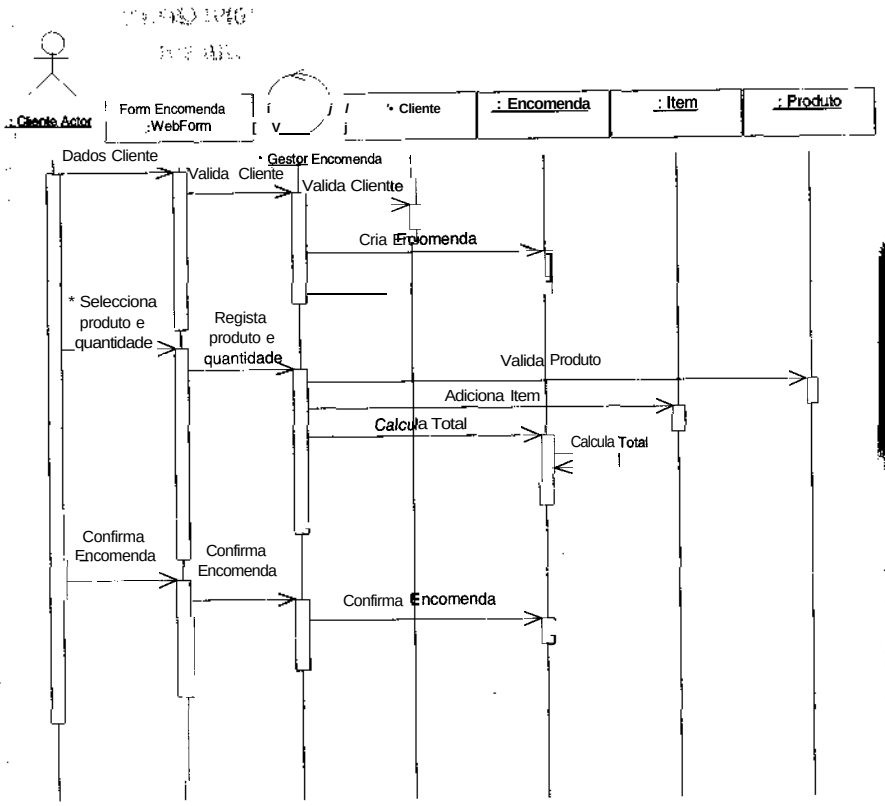


FIGURA 10,9 - RECEPÇÃO ENCOMENDA

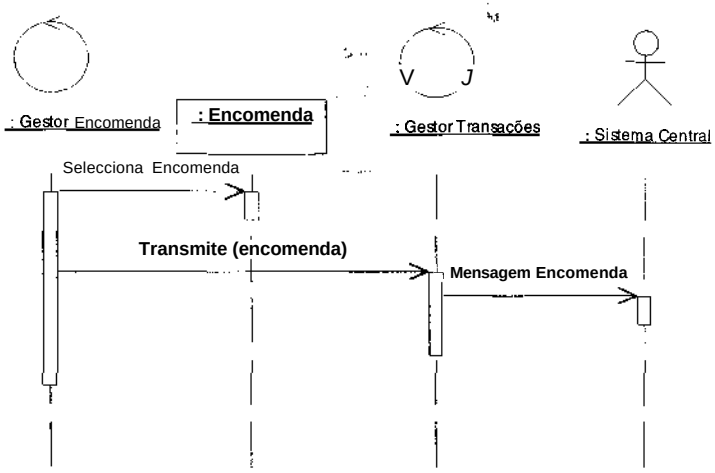


FIGURA 10.10 - TRANSMITIR ENCOMENDA

10.1.4 Modelo de Desenho

Diagrama de Classes

Na figura 10.11 apresenta-se um diagrama das classes da camada de negócio do sistema.

Este é um diagrama geral que inclui todas as classes que suportam informação necessária ao funcionamento global da PhonePizza. Para cada um dos subsistemas existirá um (sub)modelo que inclui algumas das classes deste modelo geral. Por exemplo no caso do subsistema da Internet não se justifica que sejam consideradas classes que descrevem a estrutura organizativa da Pizzaria (Restaurante, Balcão, Cozinha, etc.) ou aquelas associadas aos colaboradores (Funcionário). Para simplificar o diagrama omitimos também a representação das classes de controlo identificadas nos diagramas de sequência.

© FCA - EDITORA DE INFORMÁTICA

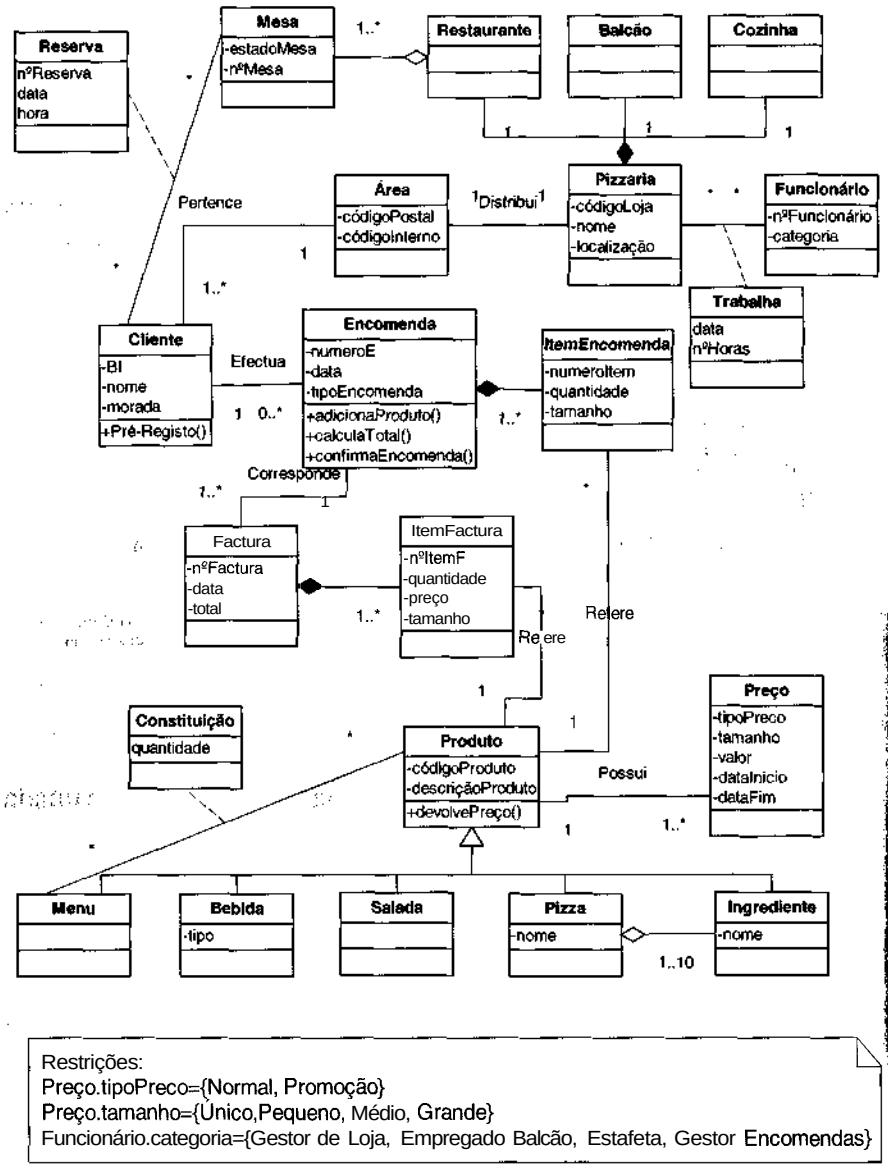


FIGURA 10.11 – DIAGRAMA DE CLASSES PHONEPIZZA

Descrição das classes

Todas as classes são descritas em pormenor no Anexo II.

Diagrama de Estados

A Encomenda constitui a classe com um comportamento dinâmico mais relevante neste sistema. No Capítulo 6 encontra-se descrito o diagrama de estados desta classe.

10.1.5 Modelo de Implementação

Na figura 10.12 apresenta-se, como exemplo, o diagrama de componentes para o subsistema Internet.

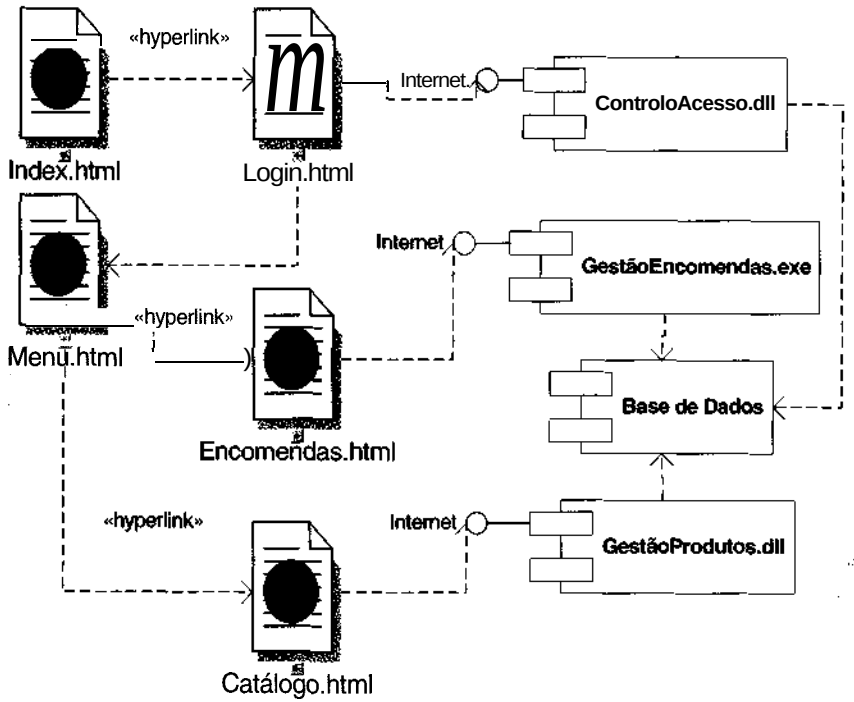


FIGURA 10.12 - DIAGRAMA DE COMPONENTES

© FCA - EDITORA DE INFORMATICA

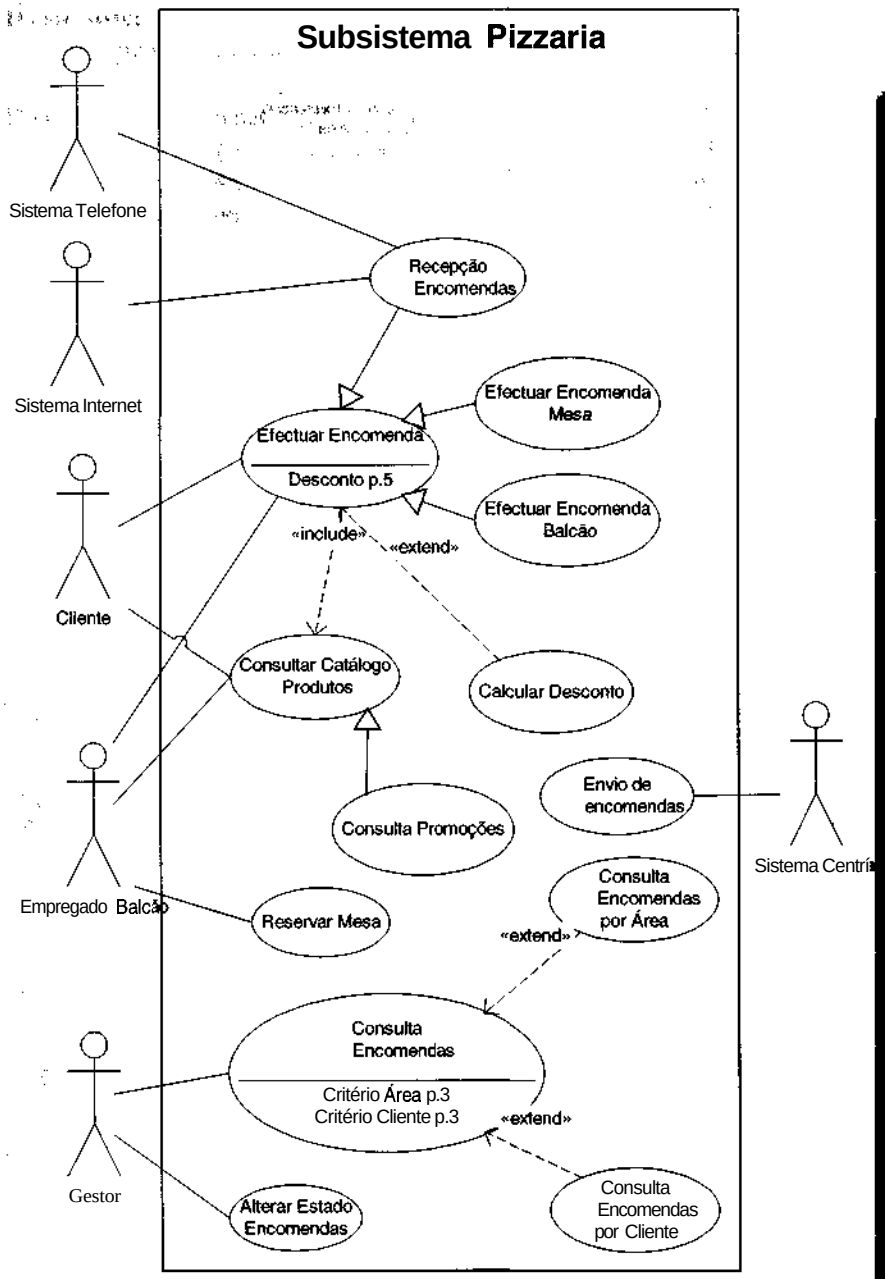


FIGURA 10.5 - SUBSISTEMA PIZZARIA

© FCA - EDITORA DE INFORMÁTICA

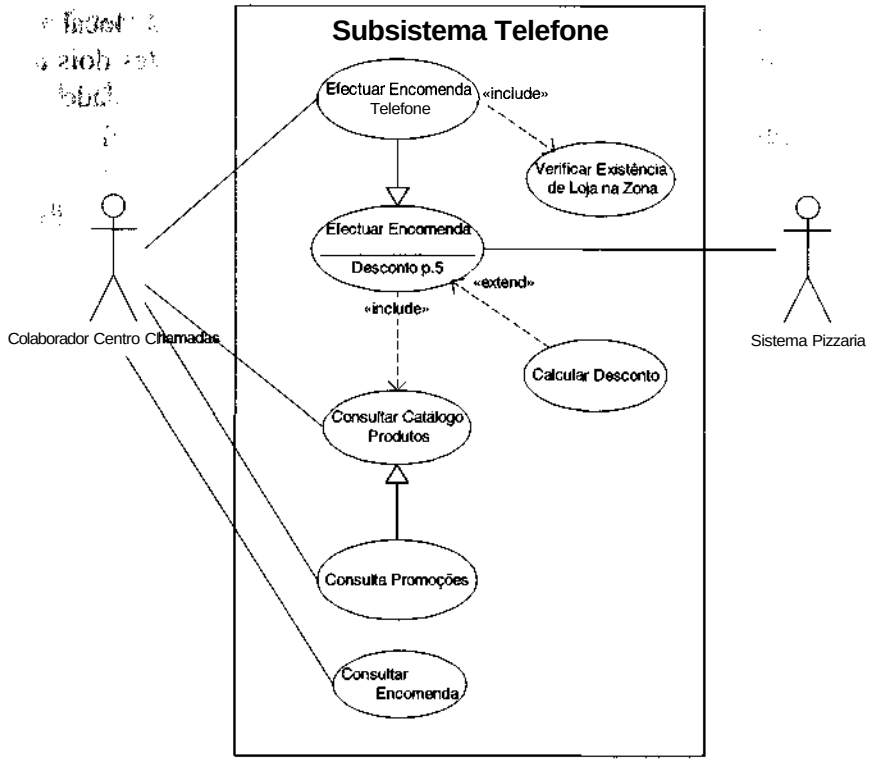


FIGURA 10,6 — SUBSISTEMA TELEFONE

Descrição dos *use cases*

Nesta secção, são descritos alguns dos *use cases* que serão posteriormente detalhados através de diagramas de sequência e actividade. As restantes descrições encontram-se no Anexo II.

Alguns dos use cases são internos a cada um dos subsistemas. Por exemplo a consulta do catálogo de produtos pode ser feita através de um ecrã de utilizador disponibilizado pelo subsistema da Internet, e utilizando informação residente na base de dados local sobre produtos e promoções. Existe um *use case* semelhante para permitir a consulta do catálogo na pizzaria através do terminal, mas

utilizando informação residente na base de dados local do subsistema da pizzaria. Por este motivo a descrição destes dois *use cases* será muito semelhante, sugerindo a possibilidade de reaproveitamento de diagramas.

Existe um outro conjunto de *use cases* que implicam comunicação entre 2 ou mais subsistemas. Nesta situação encontra-se o pré-registo que envolve comunicação entre o subsistema Internet e o subsistema central. A aceitação de uma encomenda pela Internet é um *use case* que exige o envio de uma mensagem do subsistema Internet para o subsistema da Pizzaria. Por esse motivo estes *use cases* encontram-se identificados em ambos os subsistemas, ainda que a sua descrição detalhada e representação utilizando um diagrama de sequência sejam distintas.
A título de exemplo apresentamos a descrição de 3 *use cases* associados ao subsistema de Internet.

* Consultar catálogo de produtos na Internet
* Efectuar pré-registo pela Internet
* Efectuar Encomenda na Internet

Estes *use cases* serão posteriormente representados através de diagramas de sequência.

Use Case: ***Consultar catálogo de produtos na Internet***

1. O Cibernauta, o Cliente, o Colaborador do Centro de Chamadas e o Empregado de Balcão utilizam o sistema de informação para consultar o catálogo de produtos.
2. O catálogo de produtos deverá ser apresentado sob a forma de uma listagem de produtos com a seguinte informação: código, nome, descrição e preço.
3. Caso seja pretendida a consulta das promoções do mês, então é utilizado o **caso específico**: *Consultar Promoções*

© FCA - EDITORA DE INFORMÁTICA

Use Case: ***Efectuar pré-registo por Internet***

1. O Cibernauta utiliza o sistema de informação através da página Internet da *PhonePizza* para efectuar o pré-registo, requisitando um código de acesso.
2. O Cibernauta tem que indicar: *username*, *password*, telefone e morada.
3. **Includes** *Verificar a Existência de Loja numa Zona*

Pós-condição: É gerado um código de acesso que será enviado por correio electrónico.

Use Case: ***Efectuar Encomenda na Internet***

1. O Cliente utiliza o sistema de informação através da página Internet da *PhonePizza* para efectuar a encomenda.
2. O cliente fornece os seus dados identificativos e estes são validados.
3. Se pretender, o Cliente pode consultar:
   a. o catálogo de produtos: **Includes** *Consultar Catálogo de Produtos*
   b. e/ou as promoções do mês: **Includes** *Consultar Promoções*
4. O Cliente escolhe o(s) produto(s) que pretende, (indicando o código ou o nome do produto e a quantidade).
5. Para cada produto escolhido, o sistema verifica o seu preço e é adicionado ao custo total da encomenda.
6. Se o produto está em promoção, existindo assim um desconto:
   a. Extends *Calcular Desconto.*
7. O produto é adicionado aos itens da encomenda.
8. O Cliente confirma a Encomenda
9. A Encomenda é transmitida para a Pizzaria da zona da morada do cliente.

Diagramas de Sequência

Apresentamos nesta secção os diagramas de sequência dos *use cases* descritos anteriormente.

Na figura 10.5 apresenta-se o diagrama de sequência que descreve a Consulta de Catálogo. No diagrama de sequência encontram-se representados os objectos de interface, os objectos de negócio e os objectos de dados. Ainda que a maioria dos objectos de negócio de um sistema de informação sejam persistentes, em geral omitimos a representação dos correspondentes objectos de dados para evitar sobrecarregar os diagramas.

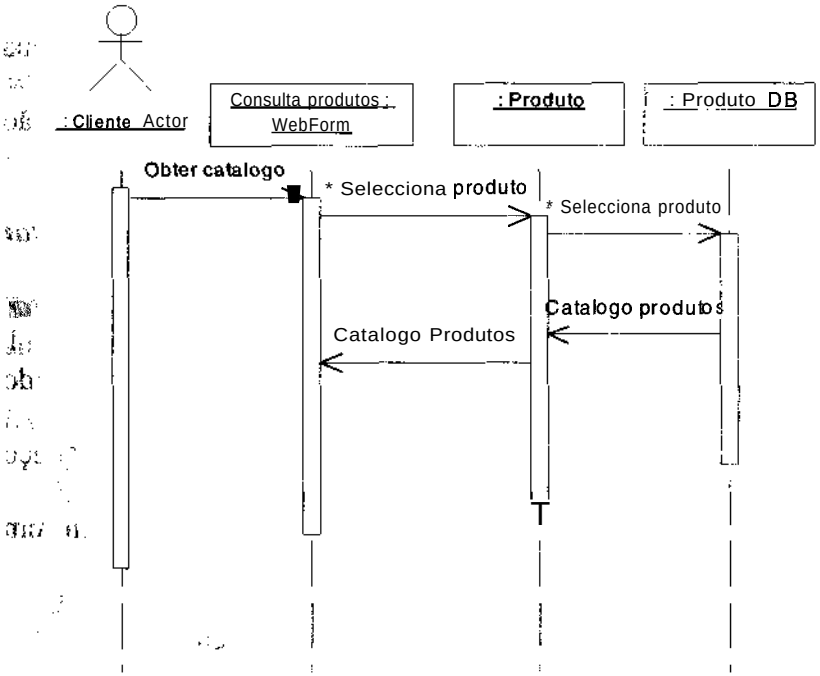


FIGURA 10.7 - CONSULTA CATÁLOGO

Em *use cases* onde têm de ser tomadas decisões complexas utilizamos um objecto de controlo que assegura o acompanhamento

do fluxo de execução e valida as regras de negócio relevantes. Por exemplo, na descrição de Efectuar Encomenda pela Internet surge um objecto designado por Gestor de Encomendas que desempenha essa tarefa.

Para descrever um *use case* onde seja necessário assegurar comunicação com os outros subsistemas através do envio de mensagens utilizamos um objecto de controlo, designado por Gestor de Transacções.

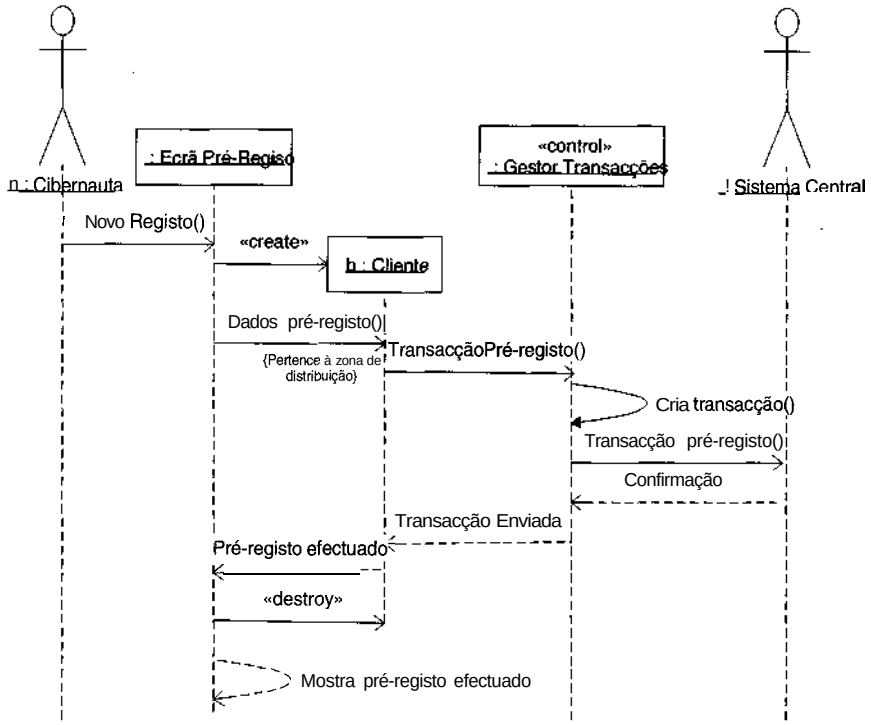


FIGURA 10.8 – PRÉ-REGISTO CLIENTE

Para facilitar a representação do diagrama de sequência que descreve a satisfação de uma encomenda, optamos por decompô-lo em dois subdiagramas: um que descreve a recepção da informação

© FCA - EDITORA DE INFORMÁTICA

da encomenda e um outro que descreve a operação de envio para o subsistema Central.

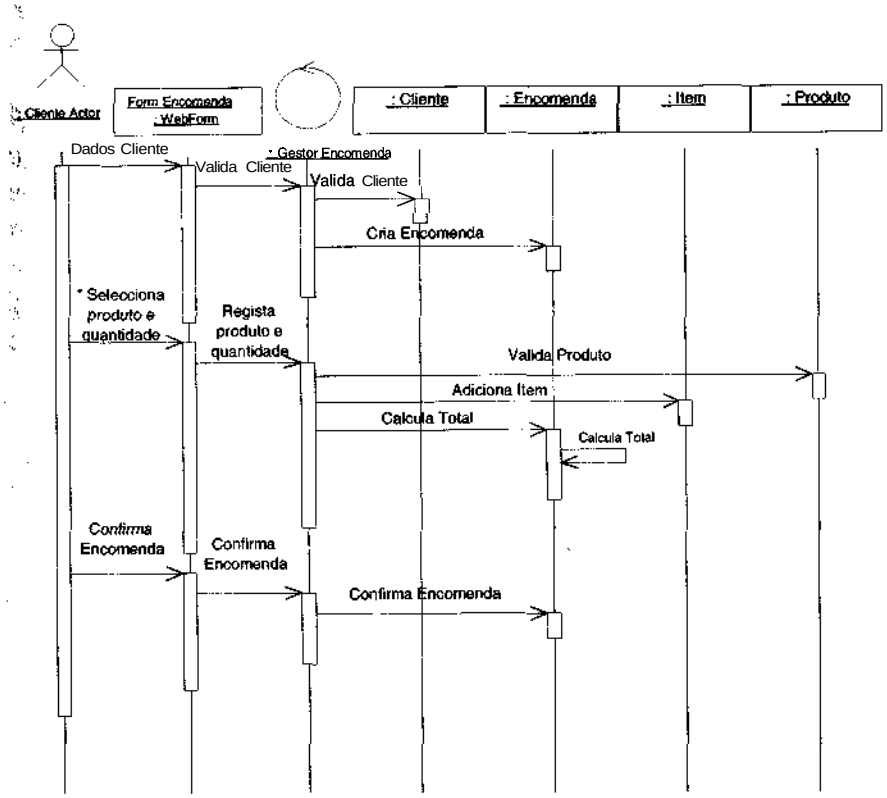


FIGURA 10.9 - RECEPÇÃO ENCOMENDA

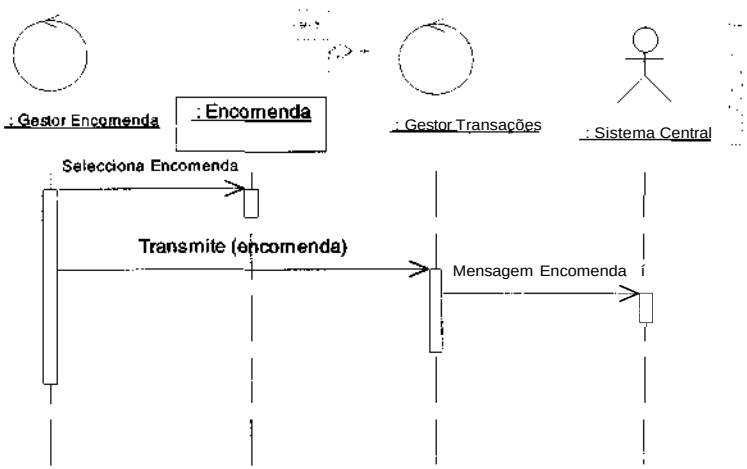


FIGURA 10,10 - TRANSMITIR ENCOMENDA

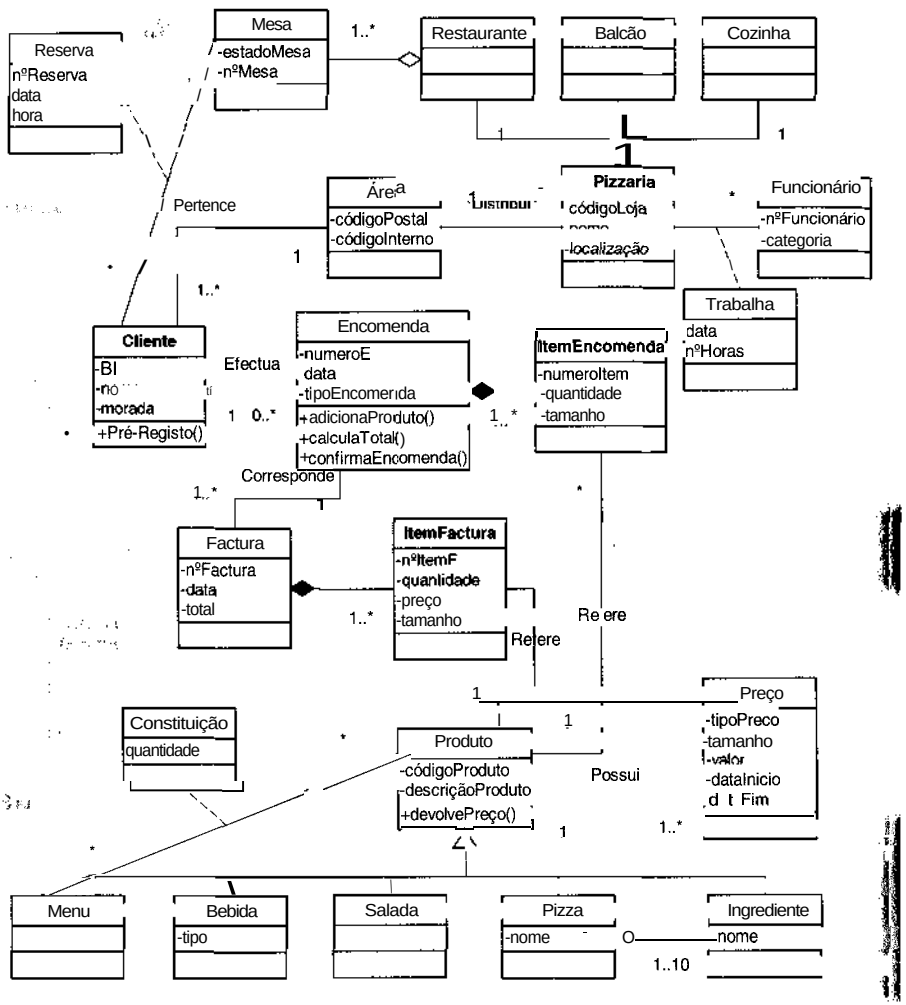
10.1.4 Modelo de Desenho

Diagrama de Classes

Na figura 10.11 apresenta-se um diagrama das classes da camada de negócio do sistema.

Este é um diagrama geral que inclui todas as classes que suportam informação necessária ao funcionamento global da PhonePizza. Para cada um dos subsistemas existirá um (sub)modelo que inclui algumas das classes deste modelo geral. Por exemplo no caso do subsistema da Internet não se justifica que sejam consideradas classes que descrevem a estrutura organizativa da Pizzaria (Restaurante, Balcão, Cozinha, etc.) ou aquelas associadas aos colaboradores (Funcionário). Para simplificar o diagrama omitimos também a representação das classes de controlo identificadas nos diagramas de sequência.

© FCA - EDITORA DE INFORMÁTICA



Restrições:
Preço.tipoPreco={Normal, Promoção}
Preço.tamanho={Único,Pequeno, Médio, Grande}
Funcionário.categoria={Gestor de Loja, Empregado Balcão, Estafeta, Gestor Encomendas}

FIGURA 10,11 - DIAGRAMA DE PHONEPIZZA

Descrição das classes

Todas as classes são descritas em pormenor no Anexo II.

Diagrama de Estados

A Encomenda constitui a classe com um comportamento dinâmico mais relevante neste sistema. No Capítulo 6 encontra-se descrito o diagrama de estados desta classe.

10.1.5 Modelo de Implementação

Na figura 10.12 apresenta-se, como exemplo, o diagrama de componentes para o subsistema Internet.

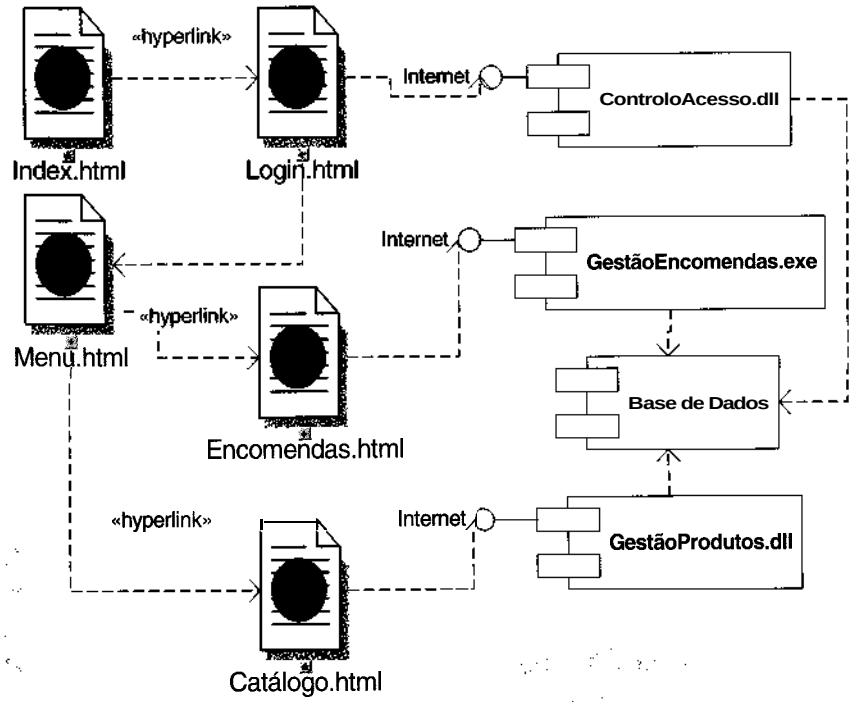


FIGURA 10.12 - DIAGRAMA DE COMPONENTES

© FCA - EDITORA DE INFORMÁTICA

10.1.6 Modelo de Instalação

Na figura 10.13 apresenta-se o diagrama de instalação do sistema PhonePizza.

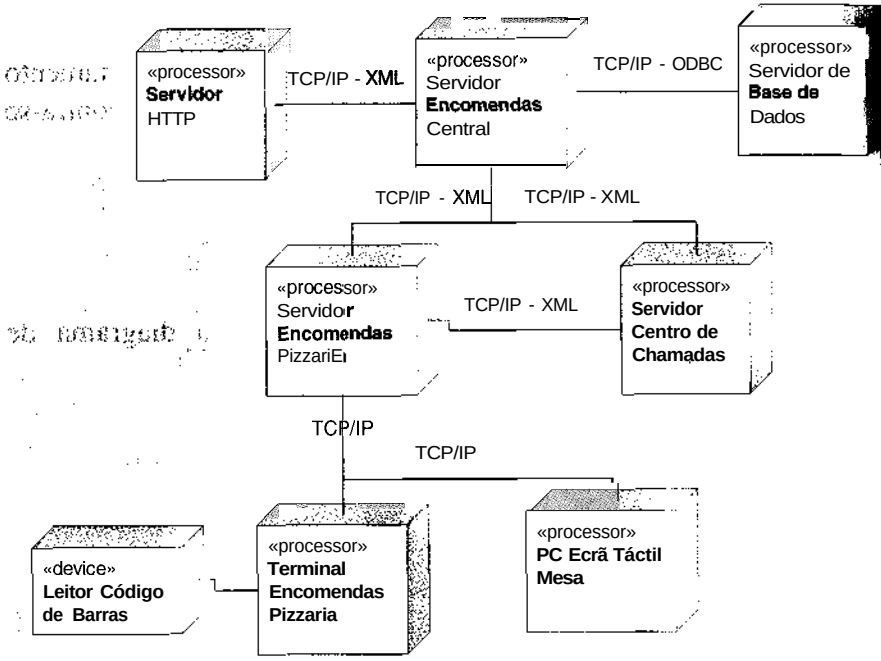


FIGURA 10.13 – DIAGRAMA GERAL DE INSTALAÇÃO

10.2 SIUNIVERSITAS®

Considere que se pretende desenvolver um sistema de informação integrado para a gestão de uma Universidade. Este sistema será conhecido por SIUniversitas® e procura racionalizar recursos e aumentar a eficiência dos processos de gestão.

Identificam-se seguidamente os principais requisitos recolhidos após reuniões com os elementos da direcção da Universidade, pessoal administrativo, docentes e representantes de alunos.

O sistema terá como ponto de acesso preferencial a Internet, disponibilizando conteúdos de acesso público e de acesso restrito. Através de um protocolo com uma instituição bancária, todos os funcionários e alunos terão um cartão magnético de identificação. O acesso restrito será validado através da introdução do número do cartão e respectivo código PIN. A operação de verificação não pode demorar mais que 1 segundo. Após 3 tentativas inválidas, o cartão magnético é bloqueado. Esta informação sobre o cartão magnético é mantida pelo sistema.

Os recursos humanos da Universidade são geridos de acordo com o seu tipo. Assim, os funcionários da Universidade são divididos em Docente e Não Docente. Todos os funcionários possuem um número de identificação interno, nome, morada, número de BI, NIF e estado civil. Estes registos são mantidos pela Secção de Pessoal, pelo que o sistema deverá permitir a gestão dos funcionários.

Os funcionários não docentes têm que registar diariamente as horas de trabalho efectuadas. Este registo é feito através de uma máquina PontoMatic situada na entrada principal da Universidade, onde cada funcionário terá que passar o seu cartão magnético à entrada e à saída.
Depois da passagem do cartão, a máquina PontoMatic envia um pedido de registo de entrada/saída para o SlUniversitas®. O sistema registará para o dia de trabalho a hora de entrada e saída. Um dia de

© FCA - EDITORA DE INFORMÁTICA

trabalho poderá ser classificado como Normal, Tolerância de Ponto, Ponte. É da responsabilidade da Secção de Pessoal classificar cada dia.

Os docentes possuem um número de gabinete, número de cacifo e podem ter dedicação exclusiva, parcial ou serem convidados. Possuem uma categoria (Assistente Estagiário, Assistente, Professor Auxiliar, Professor Associado ou Professor Catedrático) e podem leccionar várias disciplinas num determinado semestre de um ano lectivo. Um docente poderá ser o responsável pela coordenação de várias disciplinas.

Uma disciplina possui um código, um nome, endereço da página oficial na Internet e poderá ser teórica, prática ou teórico-prática. Terá também um programa que será constituído pelos objectivos pedagógicos, tópicos a abordar e bibliografia recomendada. O docente coordenador da disciplina é responsável por introduzir e alterar o programa. É também responsável por registar os diversos elementos de avaliação da disciplina[2]. Um elemento de avaliação possui uma data, descrição e pode ser classificado como trabalho de grupo, teste, frequência, avaliação contínua, exame ou exame de 2ª época.

Para cada elemento de avaliação, o docente publicará uma pauta com os respectivos resultados. Uma pauta possui um código, descrição do elemento de avaliação, turma, estado (não oficial, oficial e lançada) e é constituída pelos diversos resultados. Cada resultado diz respeito a um aluno e possui uma nota de O a 20. Inicialmente, as pautas são criadas com um estado não oficial, sendo permitido aos docentes efectuar alterações. Passados 10 dias úteis, o sistema altera o estado para oficial, permitindo assim que a Secção Académica proceda ao registo oficial das notas (ver procedimento da inscrição). O docente é avisado da alteração através do envio de uma mensagem de correio electrónico.

[2] Para um semestre de um ano lectivo.

No decorrer do ano lectivo, o docente terá que introduzir, diariamente, no sistema as aulas que leccionou. Após validar o seu acesso (serviço restrito), o docente introduz o código da disciplina e o código da turma. Em seguida, introduz o número de alunos presentes e o sumário. O sistema calcula automaticamente o número da aula.

A Universidade organiza os seus cursos em quatro tipos: Licenciatura, Mestrado, Pós-Graduação e Doutoramento. Um curso é caracterizado por possuir um código, nome e duração (anos curriculares). Assim, cada curso terá um ou mais anos curriculares, onde serão leccionadas várias disciplinas no 1° ou 2° Semestre. Um ano curricular é caracterizado por possuir um número de ano e tipo (normal, projecto, estágio ou tese). A Universidade utiliza um sistema de créditos por disciplina, cujo valor poderá ser diferente por curso.

Os alunos são agrupados em turmas, sendo que cada turma terá no máximo 50 alunos e pertence a um ano curricular Um turma é caracterizada por possuir uma sigla, ano lectivo em que foi formada e o regime horário (diurno ou nocturno). Num ano lectivo o aluno só pode pertencer a uma turma. A cada turma é atribuído um horário. Um número limitado de turmas é criado no início do ano lectivo pela Secção Académica, contudo no decorrer de uma inscrição poderá ser necessário a criação de uma nova turma.

O horário é introduzido pelos funcionários da Secção Logística de acordo com o seguinte processo: primeiro selecciona a turma a que o novo horário se destina e modifica a versão de mesmo (a, b, c, d, e ... z). Depois introduz os diversos períodos lectivos que compõem o horário. Cada período lectivo está associado a uma disciplina e possui um dia de semana (Segunda a Sábado), hora de início, hora de fim e sala.

No início de cada ano lectivo, os funcionários da Secção Académica registam no sistema as inscrições dos alunos. Caso esta

seja a primeira inscrição, o procedimento de registo exige a criação de uma nova ficha de aluno com os seguintes dados: nome, morada, telefone, sexo, nota de candidatura e número de aluno (o número é atribuído automaticamente). Nos dados da inscrição deve constar o ano lectivo, a época da inscrição (1ª Fase, 2ª Fase ou Especial), ano curricular do curso e a turma[3]. Um aluno efectua uma inscrição por ano lectivo, para um curso e num conjunto de disciplinas (máximo 10), sendo que deve ficar registado para cada disciplina a classificação obtida (nota e época[4]). A ficha de aluno poderá ser consultada e modificada pelo aluno através da Internet (acesso restrito).

O sistema disponibiliza aos alunos outros serviços de acesso restrito. Assim, o aluno poderá, por exemplo, consultar o resultado da sua avaliação oficial nas diversas disciplinas ou pedir o certificado de habilitações. Poderá também inscrever-se nos exames de 2ª época, para tal o aluno terá que seleccionar o código da disciplina, seguido da indicação se é melhoria ou exame. Por fim será apresentado ao aluno o montante devido, 10€ se a inscrição está dentro do prazo. Caso a inscrição seja fora do prazo, o sistema terá que calcular o montante da multa com base no agravamento de 1% por dia de atraso. Só se aceitam inscrições até 15 dias do prazo. O pagamento é efectuado através do débito directo no saldo do cartão. Esta operação só é possível através do envio de uma mensagem para o Sistema Bancário da Universidade.

A Universidade deseja a criação de um serviço de alertas SMS (*Short Message Service*). Este serviço de acesso restrito permite aos alunos definir até *3* alertas. Existem *3* tipos de alertas: Pauta, Nota Oficial e Elemento de Avaliação. Todos possuem um estado (activo ou inactivo) e para os alertas Pauta e Nota Oficial o sistema enviará uma mensagem SMS quando uma pauta é publicada por um docente ou uma nota oficial é lançada pela Secção Académica. No caso do alerta Elemento de Avaliação, o aluno tem que definir o

[3] É mostrado automaticamente o número de vagas disponíveis na turma.

[4] Normal ou 2ª época.

número de dias de antecedência, da data de avaliação, em que deseja receber o aviso. O envio de mensagens SMS é efectuado através do sistema GatewaySMS.

De forma a uniformizar a gestão de grupos de trabalho, o sistema gere os grupos de alunos para cada disciplina. Cada grupo possui um número, data de formação e é constituído no mínimo por 2 elementos. Cada elemento corresponde a um aluno, pelo que possui um número de aluno. O registo dos grupos será efectuado pelos alunos (acesso restrito). Antes de registar um grupo, o sistema verifica se cada elemento já está inscrito. A entrega dos trabalhos ao longo do semestre é registada no sistema por um funcionário da recepção (acesso restrito). Ao receber o trabalho, o funcionário introduz o número do grupo e efectua o registo da entrega. Um grupo pode efectuar várias entregas, ficando registado o número da entrega, data e hora. Os docentes podem extrair uma listagem com a constituição dos grupos e também com as entregas efectuadas.

O sistema disponibiliza diversos serviços de acesso público. Estes serviços abrangem a consulta de turmas, pautas, horários e programa de disciplinas.

10.2.1 Modelo de *Use Cases*

Actores

Existem os seguintes actores que interagem com o sistema de informação SIUniversitas®:

- **Aluno** - representa os alunos da Universidade;
- **Cibernauta** - todo aquele que, através da Internet, consulta as páginas sistema;
- **Docente** - representa um funcionário que lecciona aulas na Universidade;
- **Docente Coordenador** - representa um docente que coordena uma disciplina;

© FCA - EDITORA DE INFORMÁTICA

- **Funcionário** - representa todos os funcionários da Universidade;
- **Funcionário da Recepção** - é o funcionário que efectua atendimento ao público na recepção da Universidade;
- **GatewaySMS** - sistema responsável pelo envio de mensagens SMS;
- **PontoMatic** - é a máquina que regista diariamente as horas de trabalho dos funcionários não docentes;
- **Secção Académica** - representa todos os funcionários que pertencem à Secção Académica;
- **Secção Logística** - representa todos os funcionários que pertencem à Secção Logística;
- **Secção Pessoal** - representa todos os funcionários que pertencem à Secção Pessoal;
- **Sistema Bancário da Universidade** - representa o sistema bancário que controla os débitos no cartão magnético da Universidade.

Use cases

Tomando como referência cada um dos actores, identificam-se os *use cases* em que participam:

Do Aluno:
- Consultar Ficha
- Alterar Ficha
- Consultar Avaliação Final
- Definir Alerta
- Pedir Certificado de Habilitações
- Inscrever Exame 2ª Época
- Registar Grupo
- Validar Acesso

Do Cibernauta:
- Consultar Turmas
- Consultar Pautas
- Consultar Horários
- Consultar Programa de Disciplinas

Do Docente:
- Publicar Pauta
- Listar Constituição de Grupos
- Listar Entregas Efectuadas
- Registar Aula
- Aviso Alteração Estado da Pauta

Do Docente Coordenador:
- Introduzir Programa
- Alterar Programa

Do Funcionário:
- Valida Acesso

Do Funcionário da Recepção:
- Regista Entrega de Trabalhos

Para GatewaySMS:
- Enviar Alerta SMS

Do PontoMatic:
- Registar movimento

Da Secção Académica:
- Inscrever Aluno
- Criar Turma
- Registar Nota

Da Secção Logística:
- Introduzir Horário

Da Secção Pessoal:
- Gerir Funcionários
- Classificar Dia Trabalho

Diagrama de *use cases*

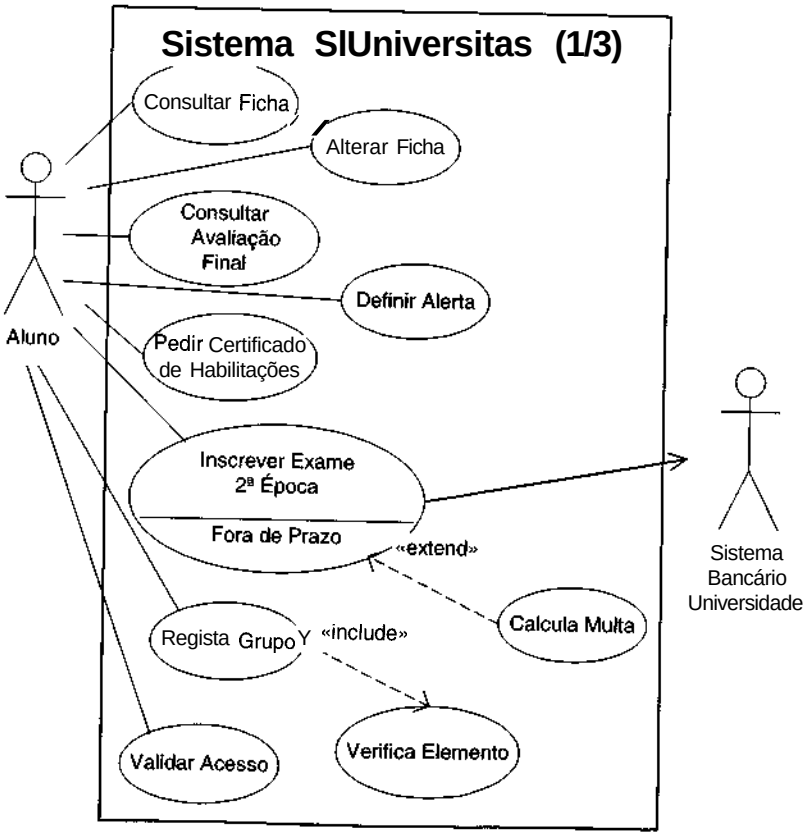


FIGURA 10,14 – DIAG, *USE* SIUNIVERSITAS® (1/3)

© FCA - EDITORA DE INFORMÁTICA

- **Funcionário** - representa todos os funcionários da Universidade;
- **Funcionário da Recepção** - é o funcionário que efectua atendimento ao público na recepção da Universidade;
- **GatewaySMS** - sistema responsável pelo envio de mensagens SMS;
- **PontoMatic** - é a máquina que regista diariamente as horas de trabalho dos funcionários não docentes;
- **Secção Académica** - representa todos os funcionários que pertencem à Secção Académica;
- **Secção Logística** - representa todos os funcionários que pertencem à Secção Logística;
- **Secção Pessoal** - representa todos os funcionários que pertencem à Secção Pessoal;
- **Sistema Bancário da Universidade** - representa o sistema bancário que controla os débitos no cartão magnético da Universidade.

Use cases

Tomando como referência cada um dos actores, identificam-se os *use cases* em que participam:

Do Aluno:
- Consultar Ficha
- Alterar Ficha
- Consultar Avaliação Final
- Definir Alerta
- Pedir Certificado de Habilitações
- Inscrever Exame 2ª Época
- Registar Grupo
- Validar Acesso

Do Cibernauta:
- Consultar Turmas
- Consultar Pautas
- Consultar Horários
- Consultar Programa de Disciplinas

Do Docente:
- Publicar Pauta
- Listar Constituição de Grupos
- Listar Entregas Efectuadas
- Registar Aula
- Aviso Alteração Estado da Pauta

Do Docente Coordenador:
- Introduzir Programa
- Alterar Programa

Do Funcionário:
- Valida Acesso

Do Funcionário da Recepção:
- Regista Entrega de Trabalhos

Para GatewaySMS:
- Enviar Alerta SMS

Do PontoMatic:
- Registar movimento

Da Secção Académica:
- Inscrever Aluno
- Criar Turma
- Registar Nota

Da Secção Logística:
- Introduzir Horário

Da Secção Pessoal:
- Gerir Funcionários
- Classificar Dia Trabalho

Diagrama de *use cases*

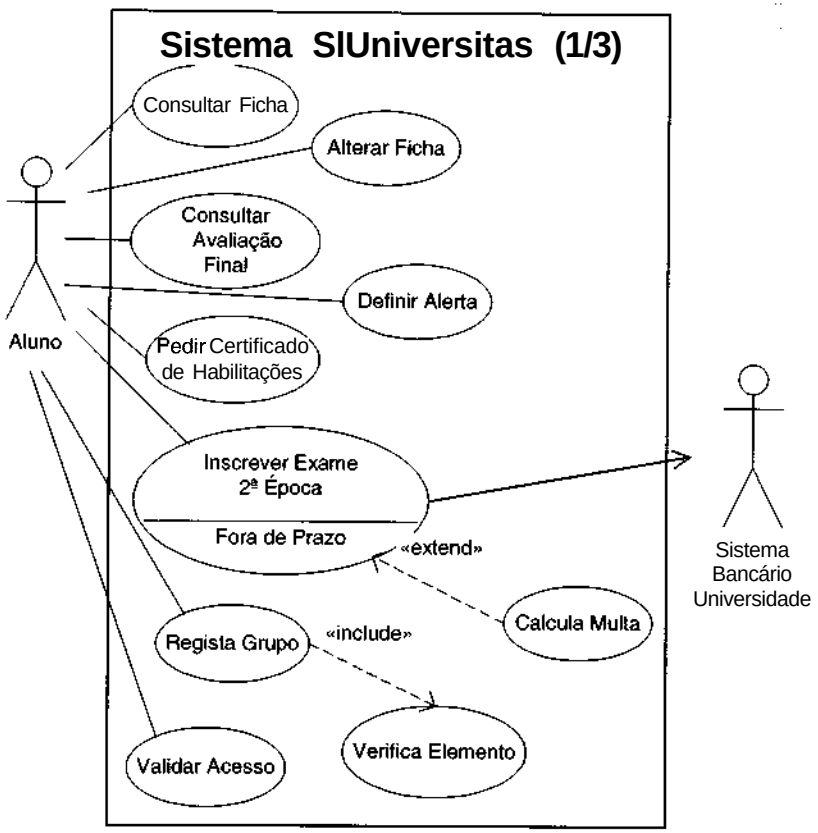


FIGURA 10.14 – DIAG. *USE CASES* SIUNIVERSITAS®(1/3)

© FCA - EDIÇÕES INFORMÁTICA

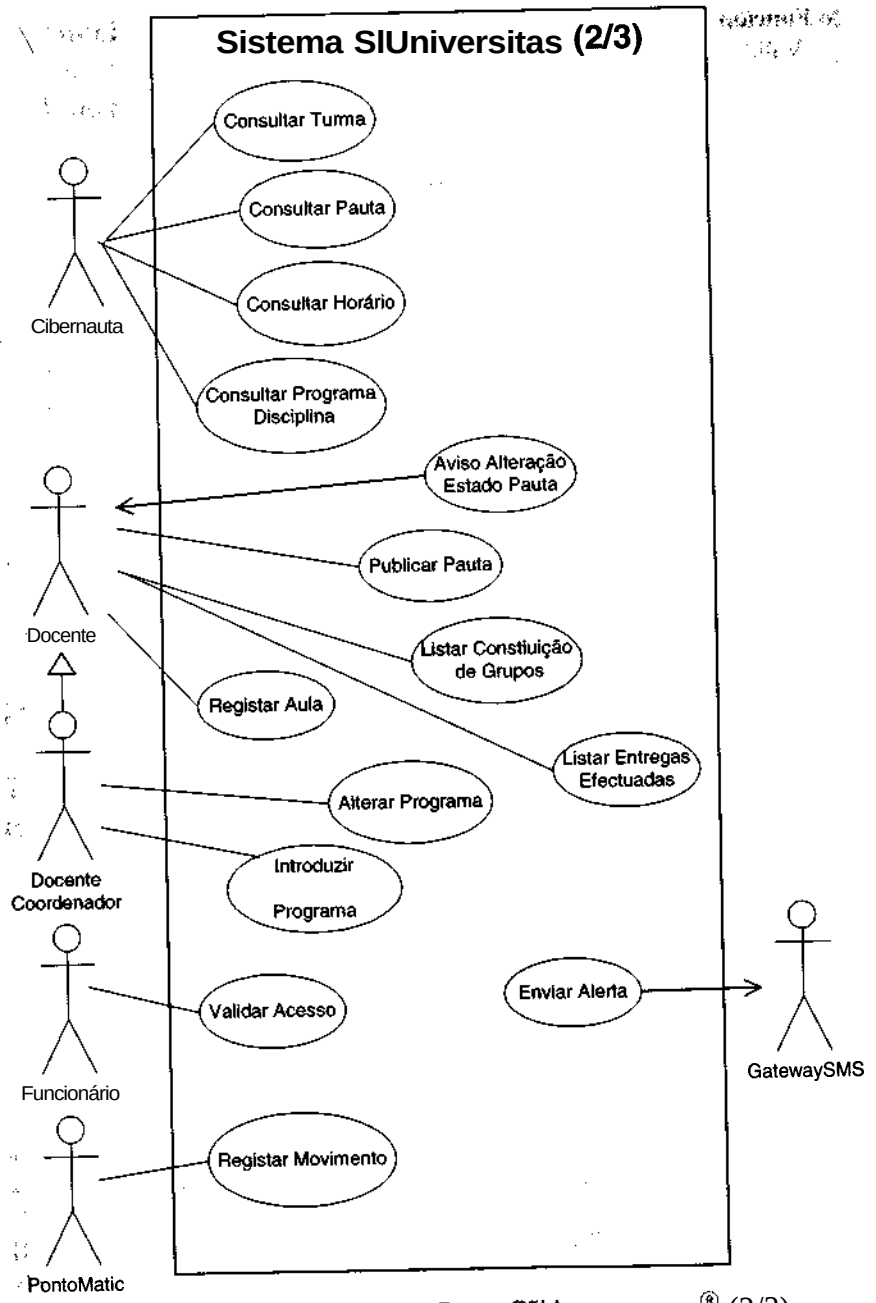


FIGURA 10,15 - DIÂG. *USE CASES* SIUNIVERSITAS® (2/3)

© FCA - EDITORA DE INFORMÁTICA

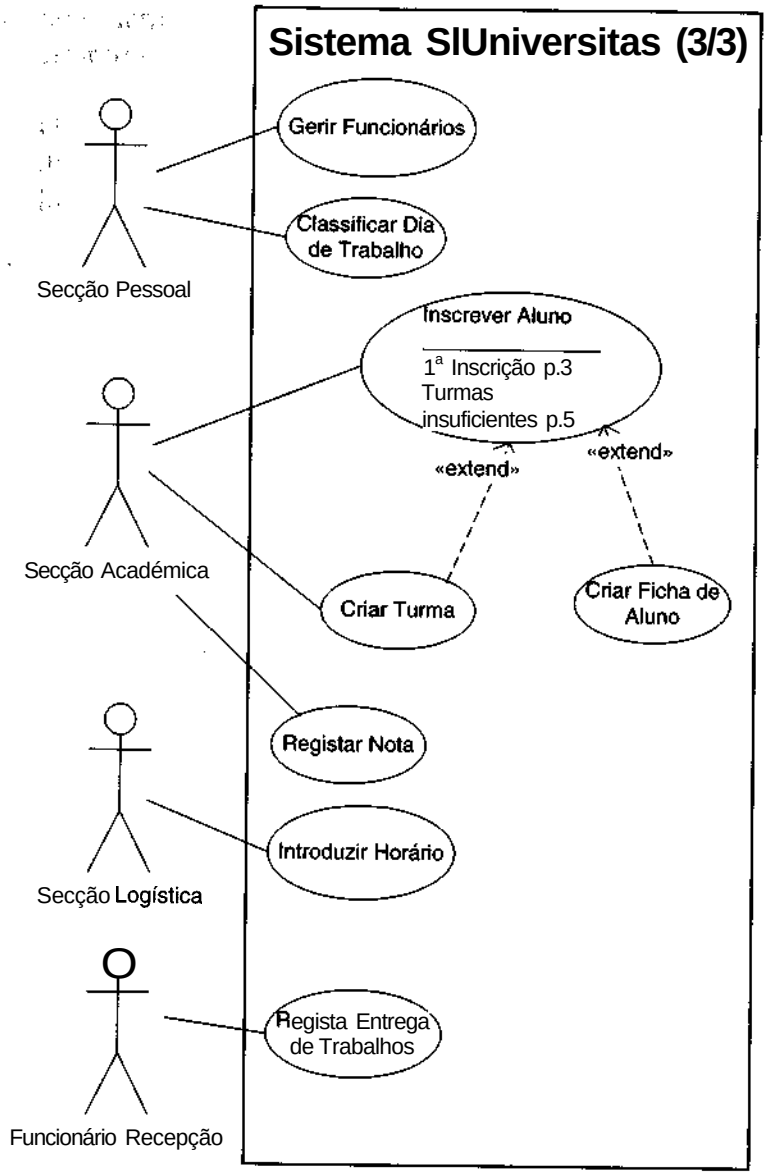
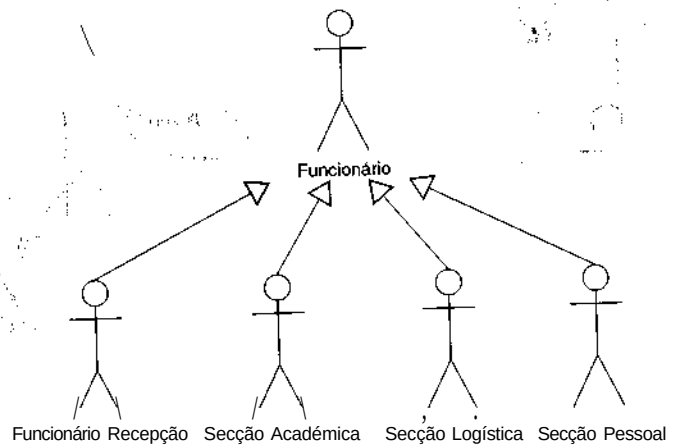


FIGURA 10.16 - DIAG. *USE CASES* SIUNIVERSITAS® (3/3)



10.17 - RELAÇÃO ENTRE ACTORES

Descrição dos *use cases*

Nesta secção são descritos alguns dos *use cases* que serão posteriormente detalhados através de diagramas de sequência ou actividade.

Use Case: ***Inscrever Aluno***

Pré-Condição: O funcionário validou o seu acesso previamente.

1. Os funcionários da Secção Académica utilizam o sistema para efectuar a inscrição de alunos.
2. Após seleccionar a opção Registar Nova Inscrição o funcionário terá que introduzir o número de aluno que está a inscrever.
3. Caso seja a primeira inscrição do aluno, será criada uma nova ficha de aluno, onde será atribuído um número de aluno.
   a. **Extends** *Criar Ficha de Aluno.*
4. Em seguida, o funcionário indica o ano lectivo, a época da inscrição (1ª Fase, 2ª Fase ou Especial), ano curricular e a turma.

5. Se o número de turmas é insuficiente, o funcionário poderá criar uma nova turma.
   a. **Extends** *Criar Turma.*
6. O funcionário introduz o código do curso e o código das disciplinas em que é feita a inscrição. Só será possível introduzir no máximo 10 disciplinas.

Use Case: ***Inscrever Exame de 2ª Época***

Pré-Condição: O aluno validou o seu acesso previamente.

1. Os alunos utilizam o sistema para efectuar a inscrição nos exames de 2ª época.
2. Após seleccionar a opção Inscrever Exame 2ª Época o aluno terá que seleccionar o código da disciplina e indicar se é para melhoria ou exame.
3. O sistema calcula o montante devido pelo aluno, sendo 10€ se a inscrição estiver dentro do prazo. Se a inscrição estiver fora do prazo, será cobrada uma multa.
   a. **Extends** *Calcular Multa.*
4. Só será possível efectuar inscrições se não decorreram 15 dias do prazo.
5. É enviada uma instrução de pagamento para o Sistema Bancário da Universidade.

Use Case: ***Validar Acesso***

1. Para aceder aos serviços de acesso restrito, o funcionário ou o aluno tem que validar o seu acesso.
2. Assim que é seleccionado um serviço com acesso restrito, é mostrado ao utilizador um ecrã de validação do acesso.
3. O funcionário/aluno introduz o número do seu cartão magnético e respectivo código secreto (PIN).
4. O sistema verifica se a informação fornecida está correcta. Esta operação não pode demorar mais que 1 segundo.
5. O funcionário/aluno dispõe de 3 tentativas. À 3ª tentativa falhada o sistema bloqueia o cartão magnético.

© FCA - EDITORA DE INFORMÁTICA

›iagrama de Classes Conceptual

ıpresentamos nesta secção o diagrama de classes conceptual. Este iagrama baseia-se nos requisitos identificados previamente.

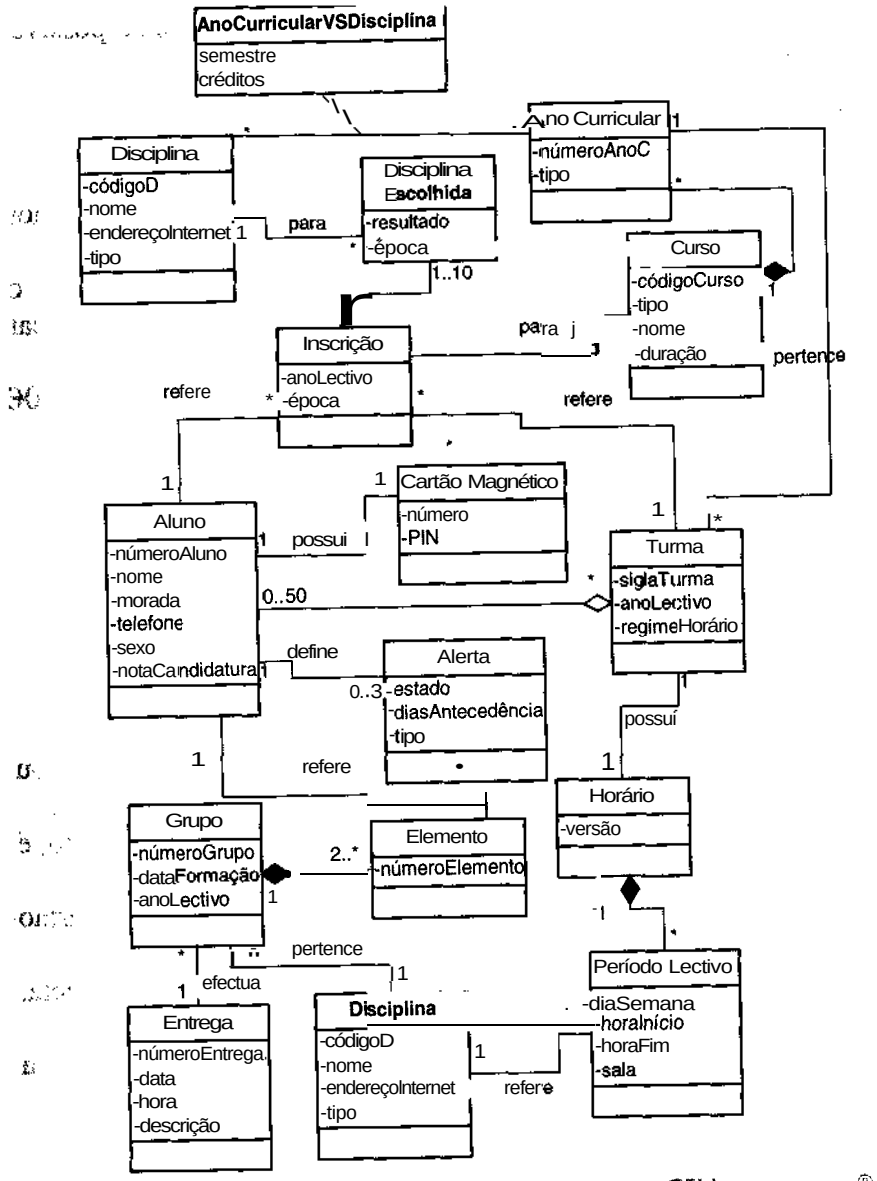


FIGURA 10,18 - DIAG. CLASSES CONCEPTUAL SIUNIVERSITAS®

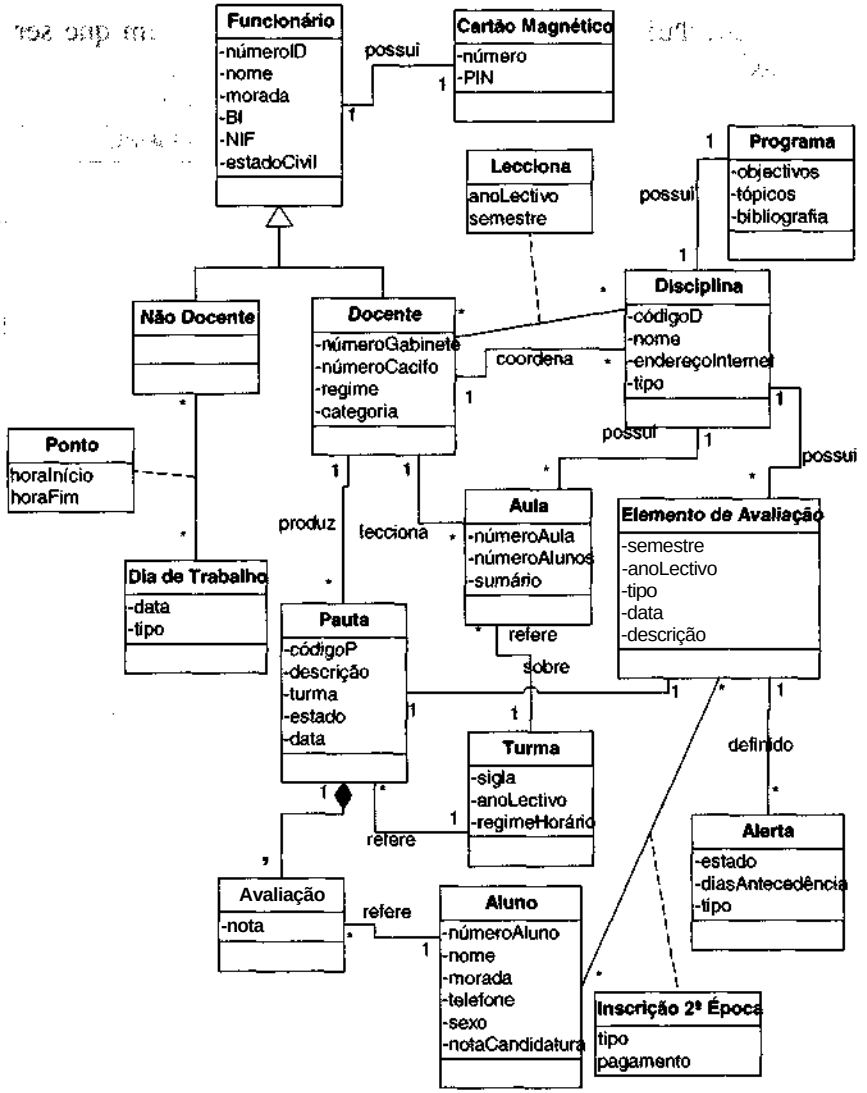


FIGURA 10,19 - DIAG, CLASSES CONCEPTUAL (CONT.)

O diagrama de classes conceptual teve que ser dividido em dois diagramas devido ao espaço disponível. De fornia a manter a coerência entre os diagramas, as classes Aluno, Cartão

© FCA - EDITORA DE INFORMÁTICA

Magnético, Turma, Alerta e Disciplina **tiveram que ser** duplicadas.
Foram também identificadas as seguintes restrições:

Restrições
Alerta.estado={Activo, Inactivo}
Alerta.tipo={Alerta Pauta, Alerta Nota Oficial, Alerta Avaliação}
Ano Curricular.tipo={normal, projecto, tese}
Avaliação.época={Normal, 2- Época}
Curso.tipo={Licenciatura, Mestrado, Pós-Graduação, Doutoramento}
Dia de Trabalho.tipo={Normal, Tolerância de Ponto, Ponte}
Disciplina.tipo={Teórica, Prática, Teórico-Prática}
Docente.categoria={Assistente Estagiário, Assistente, Professor Auxiliar, Professor Associado, Professor Catedrático}
Docente.regime={Dedicação Exclusiva, Dedicação Parcial, Convidado}
Elemento de Avaliação.tipo={trabalho de grupo, teste, frequência, avaliação contínua, exame, exame 2- época}
Funcionário.estadoCivil={Solteiro, Casado}
Inscrição 2- Época.tipo={Exame, Melhoria}
Inscrição.época={1ª Fase, 2- Fase, Especial}
Pauta.Estado={Não Oficial, Oficial, Lançada}
Turma.regimeHorário={diurno, nocturno}

FIGURA 10.20 – [illegible] AÇÃO DE CLASSES CONCEPTUAL

10.2.2 Modelo de Desenho

Diagramas de Sequência

Apresentamos nesta secção os diagramas de sequência dos *use cases* descritos anteriormente. Estes diagramas possuem um elevado nível de detalhe, muito aproximado de uma possível implementação. De forma a simplificar o desenho dos diagramas, foi omitida a representação dos objectos que acedem à base de dados.

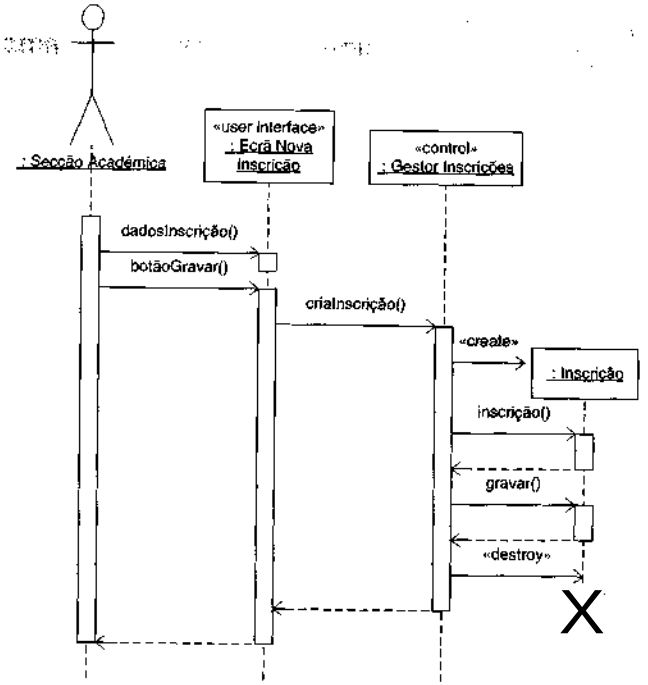


FIGURA 10.21 - SEQUÊNCIA "INSCREVER ALUNO"

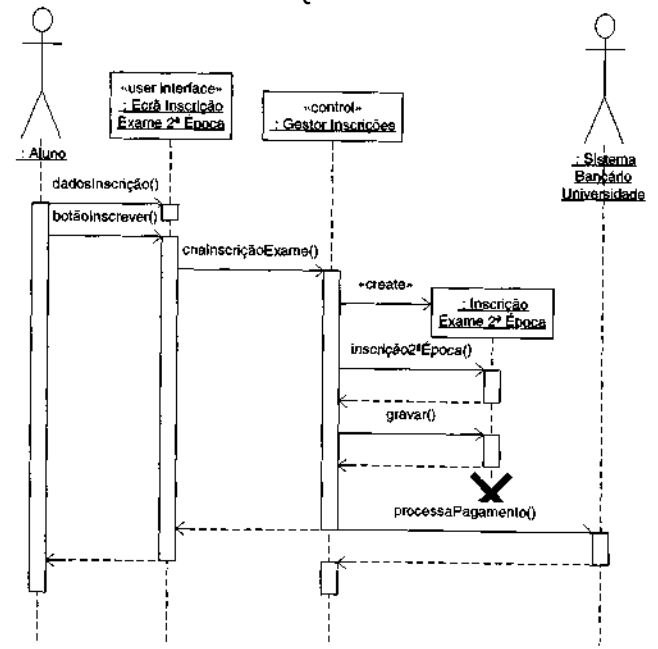


FIGURA 10.22 – [illegible] 2ª ÉPOCA"

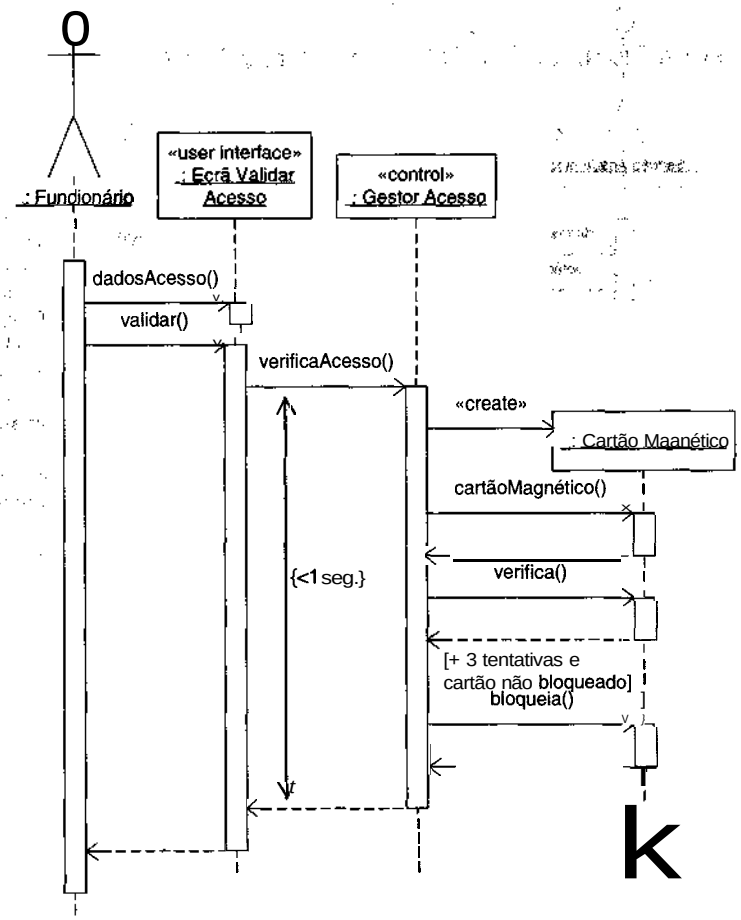


FIGURA 10,23 – DIAG. SEQUÊNCIA "VERIFICA ACESSO"

Diagrama de Classes de Desenho

O Diagrama de Classes apresentado nesta secção possui um nível de detalhe elevado, muito aproximado de uma possível implementação. Baseia-se principalmente no Diagrama de Classes Conceptual e nos Diagramas de Sequência, e procura realçar as dependências entre as classes. Por limitações de espaço, o diagrama contém apenas as classes utilizadas nos diagramas de sequências.

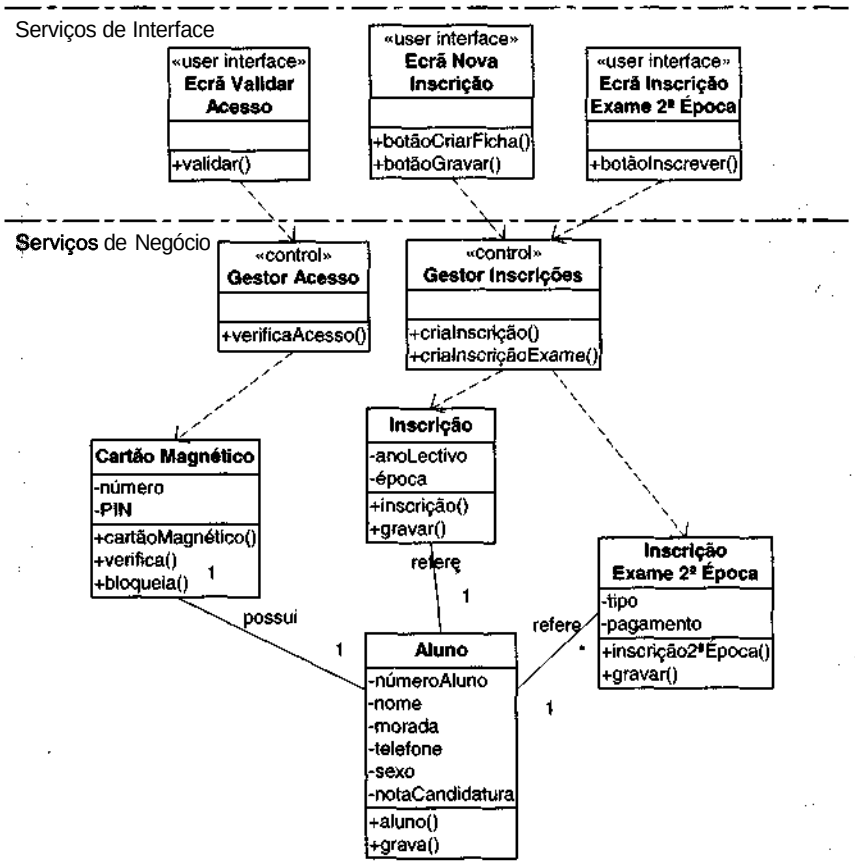


FIGURA 10,24 - DIAG. DE DESENHO

© FCA - EDITORA DE INFORMÁTICA

10.2.3 Modelo de Implementação

Na figura 10.25 apresenta-se, como exemplo, um excerto do diagrama de componentes para o sistema SlUniversitas®. O sistema será desenvolvido na linguagem de programação JAVA. O sistema será totalmente baseado em páginas dinâmicas através da tecnologia JSPs (Java Server Pages).

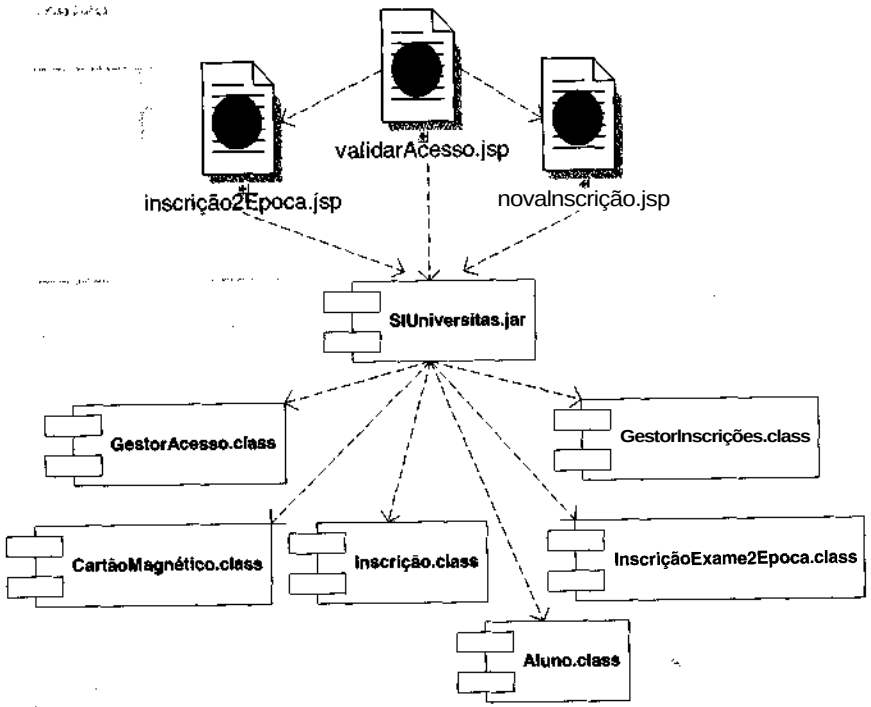


FIGURA 10.25 - DIAGRAMA DE COMPONENTES SIUNIVERSITAS®

10.2.4 Modelo de Instalação

O sistema SlUniversitas® seguirá uma arquitectura cliente/servidor, onde os clientes apenas necessitam de um programa que permita aceder às páginas Internet disponibilizadas pelo servidor. O Diagrama de Instalação apresentado na figura 10.26 ilustra também a localização dos principais componentes do sistema.

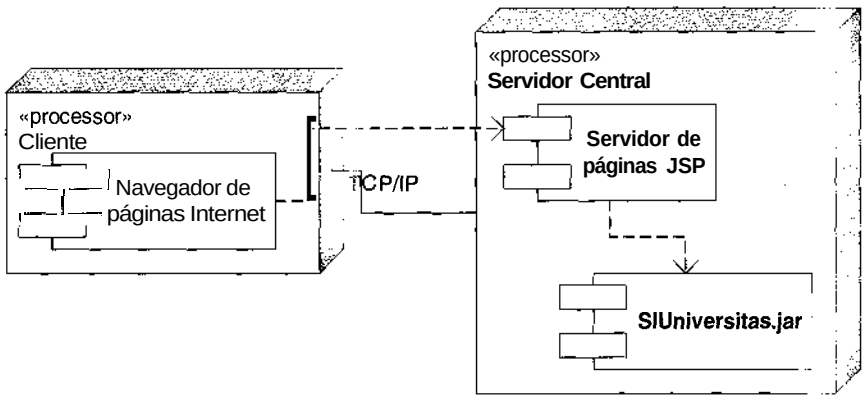


FIGURA 10.26 - DE INSTALAÇÃO SIUNIVERSITAS®

© FCA - EDITORA DE INFORMÁTICA

Anexo
Regras de transposição

I.1 CONCEITOS E APLICAÇÃO

As regras de transposição permitem efectuar a transição do diagrama de classes para o modelo relacional, garantindo que não exista perda de informação no processo.

Para que esta transição ocorra sem problemas, o diagrama de classes deve ser elaborado com a perspectiva de modelação da estrutura de uma base de dados.

O modelo relacional possui uma série de conceitos essenciais para a sua compreensão, estes conceitos são brevemente explicados em seguida.

I.1.1 Conceitos básicos

Tabela - Elemento estrutural do modelo relacional que contém dados agrupados/relacionados pela natureza do seu domínio, como por exemplo, a tabela `Cliente` pode possuir um conjunto de dados como o nome, morada ou telefone. Este conceito também é conhecido como relação.

Colunas/Atributos - Também denominados como campos de dados, consistem na unidade básica de armazenamento de informação de uma tabela, estabelecendo o esquema relacional da base de dados. Como, por exemplo, o nome, morada ou telefone para a tabela `Cliente`.

© FCA - EDITORA DE INFORMÁTICA

Tuplos – São as linhas de uma tabela, isto é, os valores que são preenchidos nos campos de dados.

Chave primária – Associação de um ou mais atributos que identificam univocamente cada tuplo de uma tabela.

Chave estrangeira – Conjunto de um ou mais atributos que são chave numa outra relação/tabela.

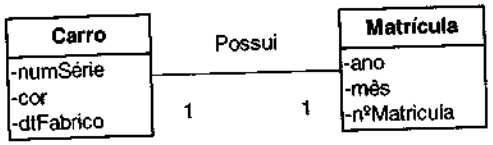
I.2 Regras

As seguintes regras derivam de Bennet et al. (1999).

Regra 1: Chave primária – Todas as tabelas devem possuir uma chave primária. Caso não existam atributos que satisfaçam esta condição, então um atributo deve ser criado para o efeito, por exemplo, um atributo "cp".

Regra 2: Origem das tabelas – As tabelas derivam somente das classes do diagrama e das associações de "muitos para muitos", incluindo os seus atributos.

Regra 3: Associação de "Um para Um" – Uma das tabelas deverá receber como chave estrangeira a chave primária da outra tabela. Neste caso, recebe a tabela em que faça mais sentido possuir o atributo. O seguinte exemplo ilustra esta regra:

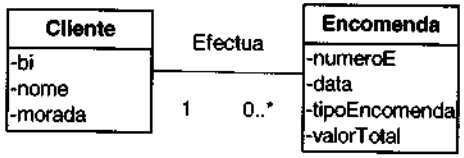


Aplicando a regra 3 teremos as seguintes tabelas:

Carro (<u>numSérie</u>, Cor, dtFabrico, *nºMatrícula*)
Matrícula (<u>nºMatrícula</u>, ano, mês)

A chave primária é representada em <u>sublinhado</u> e a chave estrangeira é representada em *itálico*. Chaves primárias e estrangeiras simultaneamente são representadas em <u>*sublinhado e itálico*</u>.

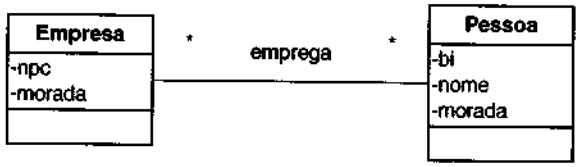
Regra 4: Associação de "Um para Muitos" – A tabela em que a informação será repetida é que recebe a chave estrangeira. Na prática consiste na regra informal em que a parte do "muitos" é que recebe a chave estrangeira. Por exemplo:



Corresponde às seguintes tabelas no modelo relacional:

Cliente (<u>bi</u>, nome, morada)
Encomenda (<u>numeroE</u>, data, tipoEncomenda, *bi*)

Regra 5: Associação de "Muitos para Muitos" – A transição dá origem a uma terceira tabela que representa a associação cuja chave primária é composta pelas chaves das tabelas associadas. Por exemplo:

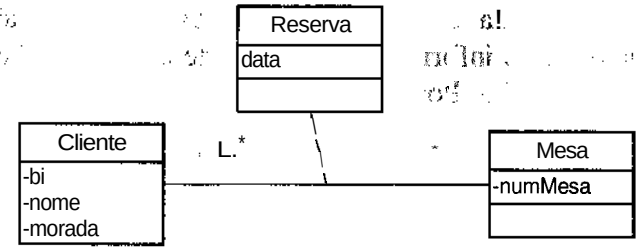


Modelo relacional:

Pessoa (<u>bi</u>, nome, morada)
Empresa (<u>npc</u>, Morada)
Emprego (<u>*bi*</u>, <u>*npc*</u>)

Regra 6: Transposição de classes associativas - Utiliza-se uma das regras correspondentes à associação, com a ressalva de que os atributos da classe associativa são recebidos pela tabela que recebe as chaves. Por exemplo:

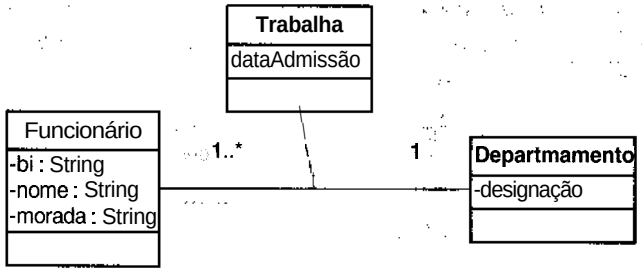
Classe associativa numa relação de "muitos para muitos"



Modelo relacional:

Cliente (bi» nome, morada)
Mesa (numMesa)
Reserva (*bi*, *numMesa*, data)

Classe associativa numa relação de "um para muitos"



Modelo relacional:

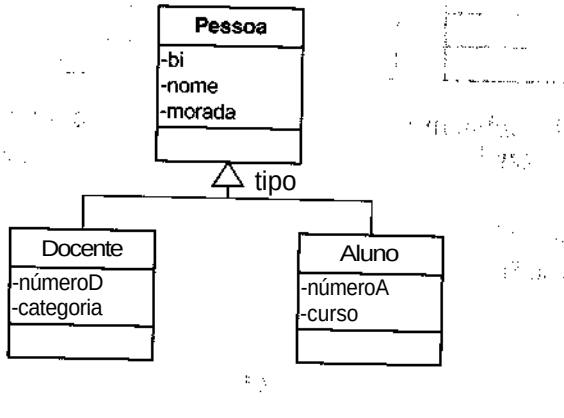
Funcionário (bi, nome, morada, *idDep*, dataAdmissão)
Departamento (idDep, designação)

© FCA - EDITORA DE INFORMÁTICA

Regra 7: Transposição de generalizações - Esta transposição varia conforme a natureza da identidade das subclasses. Assim, temos:

a) As subclasses possuem identidade própria independente da super classe:

A chave das tabelas que correspondem às subclasses é obtida exclusivamente com base nos próprios atributos. Será mantida a ligação com a tabela correspondente à superclasse, através da sua chave primária, sendo também criado um atributo que discriminará a que tabela de subclasse se refere um registo na tabela de super-classe (discriminante). Por exemplo:

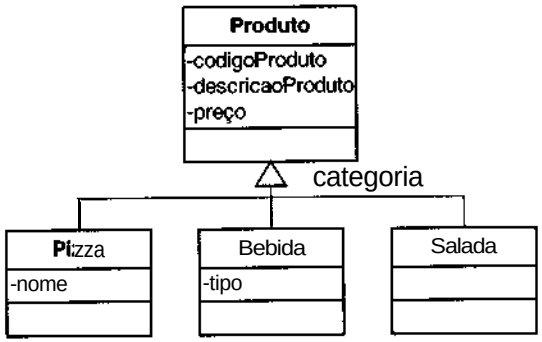


Modelo relacional:

Pessoa (bi, nome, morada, tipo)
Docente (númeroD, categoria, *bi)*
Aluno (númeroA, curso, *bi)*

b) As subclasses só têm identidade própria quando associadas à super classe:

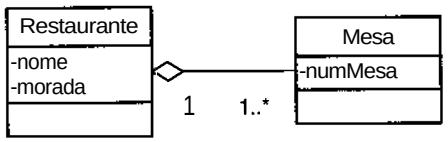
As tabelas das subclasses herdam a chave primária da tabela da super classe e também será criado o atributo discriminante. Por exemplo:



Modelo relacional:

Produto (codigoProduto, descricaoProduto, categoria={Pizza, Bebida, Salada})
Pizza (*codigoProduto*, nome)
Bebida (*codigoProduto*, tipo)
Salada (*codigoProduto*)

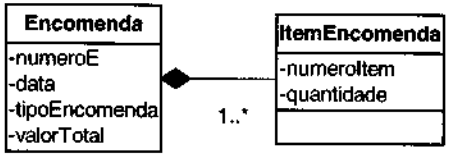
Regra 8: Transposição de Agregações - Esta transposição utiliza as mesmas regras de transposição para associações com a mesma multiplicidade. Por exemplo:



Modelo relacional:

Restaurante (idRestaurante. nome, morada)
Mesa (numMesa, *idRestaurante)*

Regra 9: Transposição de Composições - A tabela que equivale à classe de composição fica com a chave estrangeira. Esta chave também fará parte da sua chave primária. Por exemplo:



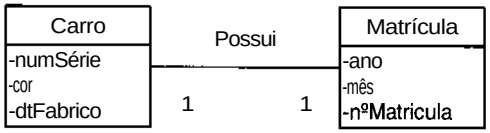
Modelo relacional:

Encomenda (numeroE, data, tipoEncomenda, valorTotal)
ItemEncomenda (numeroItem, *numeroE*, quantidade)

I.3 OPTIMIZAÇÃO DO MODELO RELACIONAL

A aplicação directa de algumas regras de transposição gera um modelo relacional pouco eficiente. Neste caso, é possível efectuar algumas optimizações ao modelo. As mais comuns são explicadas em seguida.

Optimização de associações "Um para Um" - A transposição deste tipo de associação, normalmente, pode ser optimizada através da remoção de uma das tabelas e inclusão dos seus atributos na tabela remanescente.



© FCA - EDITORA DE INFORMÁTICA

Modelo Relacional:

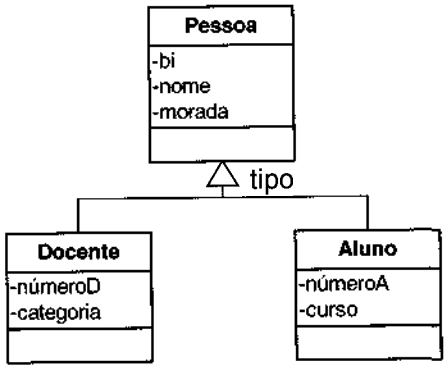
Carro (numSérie, Cor, dtFabrico, *nºMatrícula*)
Matrícula (nºMatrícula, ano, mês)

Optimização:

Carro (numSérie, Cor, dtFabrico, nºMatrícula, ano, mês)
Matrícula ~~(nºMatrícula, ano, mês)~~

Optimização de generalizações - A aplicação da regra cria um conjunto de tabelas que tornam o processo de consulta ineficiente.

a) As subclasses possuem identidade própria independente da superclasse:



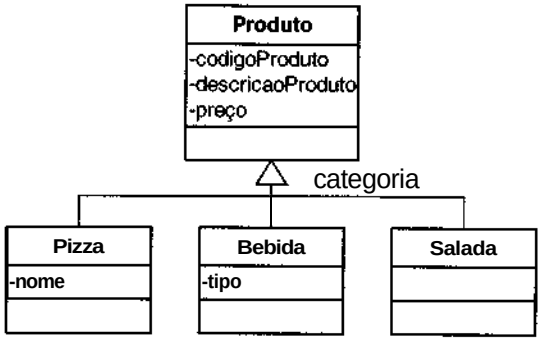
Modelo relacional:

Pessoa (bi, nome, morada, tipo={Docente, Aluno})
Docente (númeroD, categoria, *bi*)
Aluno (númeroA, curso, *bi)*

Optimização:

Pessoa (bi, nome, morada, tipo= [Docente, ~~Aluno})~~
Docente (númeroD, categoria, bi, nome, morada)
Aluno (númeroA, curso, bi, nome, morada)

b) As subclasses só têm identidade própria quando associadas à superclasse:



Modelo relacional:

Produto (codigoProduto, descricaoProduto, categoria={Pizza, Bebida, Salada})
Pizza (*codigoProduto*, nome)
Bebida (*codigoProduto*, tipo)
Salada (*codigoProduto*)

Optimização:

Produto (codigoProduto, descricaoProduto, categoria={Pizza, Bebida, Salada}, nome, tipo)
Pizza ~~(*codigoProduto*,nome)~~
Bebida ~~(*codigoProduto*~~, tipo)
Salada ~~(*codigoProduto*)~~

Optimização da Chave Primária - A chave primária de uma tabela pode ser composta por vários atributos. Quando os atributos são em excesso (mais do que 3), pode provocar ineficiências no modelo relacional. Neste caso, a optimização é efectuada através da alteração da chave primária para apenas 1 atributo (criado para o efeito, i.e. id), passando os atributos da chave inicial a atributos normais.

Anexo II
Descrições do **caso** PhonePizza

II.1 DESCRIÇÃO DE *USE CASES*

Use Case: ***Efectuar Encomenda por Telefone***

1. O Cliente telefona para a *PhonePizza* para efectuar uma encomenda.
2. O Colaborador do Centro de Chamadas recebe a chamada telefónica e utiliza o sistema de informação para registar a encomenda do Cliente.
3. O Cliente tem de se identificar, fornecendo o seu número de telefone e morada de entrega.
4. **Includes** *Verificar a Existência de Loja numa Zona*
5. Utiliza o mesmo comportamento do caso geral *Efectuar Encomenda.*

Use Case: ***Efectuar Encomenda na Pizzaria-Balcão***

1. O Cliente dirige-se ao balcão de uma loja *PhonePizza* para efectuar uma encomenda.
2. O Empregado de Balcão utiliza o sistema de informação para registar a encomenda do Cliente.
3. Se pretender, o Cliente solicita ao Empregado de Balcão informação sobre o(s) produto(s) disponíveis e o Empregado de Balcão, se necessitar pode consultar:
    a. o catálogo de produtos: **Includes** *Consultar Catálogo de Produtos*
    b. e/ou as promoções do mês: **Includes** *Consultar Promoções*

© FCA - EDITORA DE INFORMÁTICA

4. O Cliente escolhe o(s) produto(s) que pretende, (indicand o código ou o nome do produto e a quantidade) e Empregado de Balcão regista a(s) escolha(s) do Cliente.
5. Utiliza o mesmo comportamento do caso geral *Efectua Encomenda.*

Use Case: ***Efectuar Encomenda na Pizzaria-Mesa***

1. O Cliente senta-se numa mesa de uma loja *PhonePizza* e utiliza o sistema de informação da *PhonePizza* disponibilizado através de um terminal que se encontra em cada mesa para efectuar uma encomenda.
2. Se pretender, o Cliente pode consultar:
   a. o catálogo de produtos: **Includes** *Consultar Catálogo de Produtos*
   b. e/ou as promoções do mês: **Includes** *Consultar Promoções*
3. O Cliente escolhe o(s) produto(s) que pretende, (indicando o código ou o nome do produto e a quantidade).
4. Utiliza o mesmo comportamento do caso geral *Efectuar Encomenda.*

Use Case: ***Consultar Catálogo de Produtos***

1. O Cibernauta, o Cliente, o Colaborador do Centro de Chamadas e o Empregado de Balcão utilizam o sistema de informação para consultar o catálogo de produtos.
2. O catálogo de produtos deverá ser apresentado sob a forma de uma listagem de produtos com a seguinte informação: código, nome, descrição e preço.
3. Caso seja pretendida a consulta das promoções do mês, então é utilizado o **caso específico:** *Consultar Promoções*

Use Case: ***Consultar Promoções***

1. O Cibernauta, o Cliente, o Colaborador do Centro de Chamadas e o Empregado de Balcão utilizam o sistema de informação para consultar as promoções do mês.

2. As promoções do mês deverão ser apresentadas sob a forma de uma listagem de produtos que estão em promoção com a seguinte informação: código, nome, descrição e preço.

Use Case: ***Verificar a Existência de Loja numa Zona***

1. O sistema de informação verifica se existe alguma loja que distribua para uma determinada morada de entrega.
   a. Se existir alguma loja que distribua para aquela zona, o pedido é aceite.
   b. Caso contrário, o pedido fica sem efeito.
2. O sistema de informação retorna o resultado da verificação de existência de loja que distribua para a zona aonde pertence a morada.

Use Case: ***Controlo de Acesso***

1. O Cliente é solicitado a introduzir o *username* e a *password* de acesso.
2. Se o *username* e a *password* estiverem correctos, isto é, de acordo com o pré-registo, o controlo de acesso permite o acesso aos serviços.
3. Caso contrário, não é permitido o acesso aos serviços que requerem controlo.

Use Case: ***Efectuar pré-registo (subsistema Central)***

1. O subsistema Internet envia uma transacção com os dados do cliente a registar: *username, password,* telefone e morada e código de acesso.
2. O subsistema Central efectua o pré-registo do cliente.

Use Case: ***Activar serviços Central***

1. O subsistema Internet envia uma transacção de activação dos serviços no sistema central.
2. O subsistema Central confirma a activação através de uma transacção.

© FCA - EDITORA DE INFORMÁTICA

Use Case: Recepção de Encomendas

1. O subsistema loja envia uma transacção de contendo uma lista das encomendas satisfeita. Este procedimento é efectuado periodicamente.
2. O subsistema Central regista a lista das encomendas, sincronizando a sua base de dados.

Use Case: Actualização de Dados

1. O subsistema central envia para os restantes subsistemas uma transacção contendo uma actualização dos dados contidos na base de dados central (preços, códigos de produto, dados de clientes, etc).

II.2 DESCRIÇÃO DAS CLASSES

Cliente - Representa um cliente da *PhonePizza* que efectuou o pré-registo (através da Internet ou do telefone) de modo a usufruir dos serviços da pizzaria. O **Cliente** é caracterizado por n° de telefone, o qual o identifica univocamente perante o sistema, morada, código de acesso, o qual é utilizado para activação dos serviços da pizzaria, *username* e *password*. Um **Cliente** pertence a uma só área.

Encomenda - Representa uma encomenda efectuada quer na pizzaria, pela Internet ou pelo telefone. A **Encomenda** é caracterizada por um tipo (o qual define o modo como esta foi requerida), n° de encomenda, e por um estado (este atributo define em que fase do processamento a encomenda se encontra). A cada encomenda corresponderá pelo menos uma factura.

ItemEncomenda - Representa a referência a um produto de uma determinada encomenda. Cada **ItemEncomenda** caracteriza-se por uma quantidade, preço unitário e tamanho.

Factura - Representa uma factura referente a uma dada encomenda. A **Factura** é caracterizada por um n° de factura, data e total.

ItemFactura - Representa a referência a um produto de uma determinada factura, que por sua vez corresponde a um item de uma determinada encomenda. Cada **ItemFactura** caracteriza-se por uma quantidade e um preço de aquisição.

Pizzaria - Representa uma pizzaria do grupo *PhonePizza*. A **Pizzaria/Loja** caracteriza-se por um código de loja, um nome e uma descrição. Uma **Pizzaria** distribui apenas numa determinada área e é constituída por: sala, zona de entregas, balcão, cozinha e zona de entregas.

Sala - Representa a sala do restaurante de uma pizzaria do grupo *PhonePizza* e é constituída por mesas.

Entregas - Representa a zona das entregas de uma pizzaria do grupo *PhonePizza*.

Balcão - Representa o Balcão de uma pizzaria do grupo *PhonePizza*.

Cozinha - Representa a Cozinha de uma *pizzaria* do grupo *PhonePizza*.

Armazém - Representa a zona de Armazém de uma pizzaria do grupo *PhonePizza*.

Mesa - Representa o conjunto de mesa e cadeiras existente na sala do restaurante de uma pizzaria do grupo *PhonePizza*. A **Mesa** caracteriza-se por um n° e ainda por um indicador do seu estado (podendo este conter os seguintes estados: livre, ocupada e reserva).

Area - Representa a zona de influência de uma pizzaria do grupo *PhonePizza*. A **Area** é caracterizada por código postal e código interno que representa a zona de influência

Reserva - A Reserva representa uma ordem de reserva de um dado cliente em relação a uma dada mesa, numa determinada pizzaria do grupo *PhonePizza*. A **Reserva** é caracterizada por um n° e pela data da respectiva reserva.

© FCA - EDITORA DE INFORMÁTICA

Funcionário - Representa a classe de funcionários que trabalha nas pizzarias do grupo *PhonePizza*. O **Funcionário** é caracterizado por uma dada categoria (Gestor de Loja, Chefe de Mesa, Empregado de Mesa, Empregado de Balcão, Cozinheiro, Gestor de Encomendas e Estafeta) e ainda por um n° de funcionário.

Trabalha - Representa o acto de um Funcionário trabalhar durante um determinado período de tempo numa dada Pizzaria do grupo *PhonePizza*. A classe associativa **Trabalha** é caracterizada pelo número total de horas e pela data.

Produto - Representa a classe de produtos disponíveis aos clientes do grupo *PhonePizza*. Um **Produto** é caracterizado por um código de produto e por uma descrição do mesmo.

Preço - Representa a gama de preços que um determinado Produto pode assumir. O **Preço** é caracterizado por tipo - o qual indica se é um preço normal ou uma promoção; tamanho - indicando a variação do preço consoante o tamanho do produto; valor; data de início e de fim apenas aplicável se o tipo indicar promoção.

Bebida - É um tipo de **Produto,** e representa a classe das bebidas comercializadas nas pizzarias do grupo *PhonePizza*.

Salada - É um tipo de **Produto** e representa a classe das saladas comercializadas nas pizzarias do grupo *PhonePizza*.

Pizza - *É* um tipo de **Produto** e representa a classe das pizzas comercializadas nas pizzarias do grupo *PhonePizza*.

Ingrediente - É um tipo de Produto e representa a classe dos ingredientes disponíveis para compor as pizzas. O **Ingrediente** é caracterizado por um nome.

Menu - É um tipo de Produto e representa um conjunto de outros produtos, ou seja, um menu é um produto composto por produtos.

Constituição - Representa a constituição de um determinado Menu, ou seja, define quais as quantidades e os produtos que compõem o Menu.

Glossário

Abstracção: uma representação simplificada que contem aquelas características de uma entidade que a distinguem de todas as outras e que são relevantes para um fim específico.

Acção: a especificação de uma instrução executável que constitui a abstracção de um procedimento computacional. Uma acção resulta numa alteração no estado de um sistema e pode ser concretizada através do envio de uma mensagem para um objecto ou alterando o valor de um atributo.

Actividade: comportamento que persiste durante a duração de um estado

Actor: um actor é uma entidade externa de qualquer tipo que interage com o sistema. Os actores podem ser dispositivos físicos, pessoas ou sistemas de informação

Agregação: representa uma associação entre um objecto que é o todo e os objectos que são as suas partes.

Análise: a actividade do processo de desenvolvimento de um sistema de informação que procura determinar o que deve ser feito. Para tal deve formular um modelo do domínio do problema.

Arquitectura: caracteriza a estrutura e o comportamento de um sistema. Uma arquitectura pode ser construída a partir de classes, componentes e subsistemas interagindo através de interfaces, ligados através de relações e unidos por constrangimentos.

Assinatura de operação: é determinada pelo nome da operação, número e tipo dos parâmetros e tipo do valor devolvido.

Associação: uma ligação lógica entre classes que descreve ligações entre objectos e pode corresponder a uma relação lógica no domínio da aplicação ou a mensagens no fluxo de controlo dos programas.

© FCA - EDITORA DE INFORMÁTICA

Atributo: é uma característica que em conjunto com as operações descreve uma classe. Também define o conjunto de valores que as instancias da classe podem assumir nesse classificador.

Camada: permite organizar as classes ou componentes com um grau de abstracção idêntico.

Característica: uma propriedade como uma operação ou um atributo que é encapsulada num classificador, que pode ser uma classe, um interface ou um tipo de dados.

Cenário: uma sequência específica de acções que ilustra um comportamento.

Ciclo de vida: as fases que decorrem desde o ideia inicial até à instalação e operação de um sistema.

Classe abstracta: uma classe que não pode ser instanciada; uma super classe que serve de modelo genérico para as suas subclasses instanciadas.

Classe associativa: uma classe que é modelada para fornecer espaço de definição de atributos e operações que pertencem a uma associação entre classes.

Classe concreta: uma classe que pode ter instancias.

Classe: a descrição de um conjunto de objectos que partilham os mesmos atributos, operações, métodos, relacionamentos e semântica.

Colaboração: descreve como um conjunto de objectos interagem para satisfazer uma determinada função.

Comentário: uma anotação associada a um elemento ou colecção de elementos.

Componente: um módulo (fonte, binários ou executável) de aplicação com interface e identidade bem definidos.

Composição: uma forma de agregação forte com uma dependência permanente entre o todo e as suas partes.

Condição de guarda: uma expressão booleana que tem de ser verificada para permitir que uma transição ocorra.

Dependência: uma relação entre dois objectos de modelação na qual um objecto (fornecedor) presta um serviço a outros objectos (cliente).

© FCA - EDITORA DE INFORMÁTICA

Desenho: a actividade no desenvolvimento de sistema de informação que define como o sistema será implementado para satisfazer os requisitos funcionais e de qualidade pretendidos.

Desenvolvimento incremental: envolve uma análise inicial do âmbito do problema e a identificação dos principais requisitos. Os requisitos são priorizados *e* aqueles mais importantes são primeiramente satisfeitos e uma versão do sistema é entregue, passando-se em seguida a uma nova iteração.

Diagrama de actividade: uma variante do diagrama de estados que se centra no fluxo de actividade originada pelo processamento interno dentro de um objecto. Num diagrama de actividades a maioria dos estados são estados de acção (também designados actividades), onde cada um dos quais representa a execução de uma operação.

Diagrama de classes: um diagrama da UML que apresenta classes, tipos, os seus conteúdos e relacionamentos.

Diagrama de colaboração: um diagrama de colaboração descreve uma interacção entre objectos e o contexto da interacção através de ligações entre objectos. São utilizados para descrever cenários de realização de *use cases.*

Diagrama de componentes: um diagrama que representa a organização e as dependências entre componentes.

Diagrama de estados: um diagrama que especifica a sequência de estados que um objecto percorre ao longo da sua vida em resposta eventos, em conjunto com a sua resposta e acções

Diagrama de instalação: um diagrama que descreve a configuração dos nós de processamento e os componentes, processos e objectos neles instalados.

Diagrama de interacção: uma caracterização genérica que se aplica aos diagramas de sequência e de colaboração que descrevem interacções entre objectos.

Diagrama de objectos: diagrama semelhante ao diagrama de classes mas que contem instancias de objectos em vez de classes, ligações em vez de associações e apresenta valores de atributos.

Diagrama de objectos: um diagrama que descreve os objectos e as suas relações num determinado momento. Um diagrama de objectos pode ser considerado um caso especial de diagrama de classes ou diagrama de colaboração.

Diagrama de sequência: diagrama que apresenta interacções entre objectos numa sequência temporal.

Diagrama de *use case:* um diagrama que apresenta as relações entre actores e *use cases* no sistema.

Diagrama: representação gráfica de um conjunto de elementos de modelação representados como um grafo de nós (outros elementos de modelação) ligados por arcos (relações).

Encapsulamento: restringir o conhecimento de detalhes internos de uma classe (ou de um subsistema) a outras classes de modo a poder ser modificada sem que isso afecte o funcionamento de outras partes do sistema.

Estado: descreve uma situação durante a vida de um objecto durante a qual satisfaz uma condição, realiza uma actividade ou espera por um evento.

Estados concorrentes: se um objecto puder estar em mais do que um estado simultaneamente então esses estados são concorrentes.

Estereótipo: mecanismo de extensão do metamodelo.

Evento: uma ocorrência localizada no tempo e no espaço que é significativa para o sistema de informação.

Generalização: uma relação de taxinomia entre um elemento mais geral (super classe) e um elemento mais específico (subclasse). O elemento mais específico é consistente com o elemento mais geral e contem características adicionais.

Herança múltipla: uma variação semântica de generalização na qual um tipo (exemplo subclasse) pode ter mais do que um supertipo (exemplo superclasse).

Herança: mecanismo pelo qual um elemento mais específico incorpora a estrutura e comportamento de um elemento mais geral, concretizando uma relação de especialização e generalização entre classes.

Implementação: a definição de como algo é construído.

© 5T - EDITORA DE INFORMÁTICA

Instância de objecto: concretização de um objecto único que é membro de uma determinada classe.

Interface: um conjunto de operações que caracteriza o comportamento público de uma classe ou componente

Linha de vida de um objecto: uma linha num diagrama e sequência que representa a existência de um objecto ao longo de um período de tempo.

Mensagem: mecanismo utilizado para solicitar um serviço de um determinado objecto, que se encontra associado à invocação de uma operação.

Método: a implementação de uma operação.

Metodologia: aproximação ao desenvolvimento de sistemas de informação que inclui o uso de técnicas, notações, uma aproximação ao ciclo de vida e um conjunto de procedimentos e regras de trabalho.

Modelo: uma representação abstracta de um sistema físico

Multiplicidade: uma restrição numa associação que especifica o número de objectos num extremo de uma associação que se podem relacionar com um objecto no outro extremo da relação.

Nó: um recurso computacional com capacidade de processamento e memória.

Objecto persistente: um objecto que permanece para além do processo ou fluxo que o criou.

Objecto: uma entidade com uma identificação e fronteira bem definida que encapsula estado e comportamento. O estado é representado pelos atributos e relacionamentos; o comportamento é representado por operações e transições de estados. Um objecto é uma instancia de uma classe.

Operação: descreve um aspecto do comportamento de um objecto de uma classe; um serviço que é disponibilizado pela classe.

Pacote: é um mecanismo utilizado para agrupar elementos de modelação, geralmente classes ou componentes. Os pacotes podem ser incluídos noutros pacotes.

Papel: o nome de um comportamento específico de uma entidade num determinado contexto.

Parâmetro: a especificação de uma variável que pode ser passada, alterada e devolvida em mensagens, operações e eventos.

Pós-condição: uma restrição que se tem de verificar após a conclusão de uma operação.

Pré-condição: uma restrição que se tem de verificar quando a operação é invocada.

Processo de desenvolvimento: um conjunto de passos ordenados realizados com um determinado objectivo durante o desenvolvimento do sistema, tais como construir e implementar os modelos.

Protótipo: sistema parcialmente completo desenvolvido rapidamente para explorar requisitos específicos.

Regra de negócio: uma directiva que expressa restrições da organização e que se pode traduzir por exemplo, em restrições de multiplicidade na ligação entre classes.

Relação "extends": uma relação entre um *use case* estendido e um *use case* base que especifica como o comportamento definido para o *use case* extensão alarga (sujeito a condições especificadas na extensão) o comportamento definido para o *use case* base.

Relação "includes": uma relação entre um *use case* base e um *use case* de inclusão, que especifica como o comportamento de um *use case* base contem o comportamento do *use case* incluso.

Relação: uma ligação semântica entre dois elementos do modelo.

Requisito funcional: um requisito que especifica a funcionalidade exigia pelo utilizador.

Requisito: uma propriedade, comportamento ou característica pretendida para o sistema.

Responsabilidade: um contrato ou obrigação de uma classe.

Restrição: é um constrangimento ou uma condição semântica.

Serviço: uma função útil que é levada a cabo por um objecto ou subsistema a pedido de um outro objecto.

Tipo de dado: um descritor de um conjunto de valores sem identidade e cujas operações não tem efeitos laterais. Tipos predefinidos incluem números, cadeias de caracteres e datas.

© FCA - EDITORA DE INFORMÁTICA

Tipo primitivo: um tipo de dados básico pré-definido, sem qualquer subestrutura, tal como um inteiro ou uma cadeia de caracteres.

Transição: movimento de um estado para um outro despoletado por um evento.

***Use** case:* a especificação de uma sequência de acções (incluindo variantes) que um sistema pode realizar interagindo com actores do sistema.

Visibilidade: uma enumeração cujo valor (público, protegido ou privado) define como o elemento de modelação que lhe está associado é reconhecido fora do seu espaço de referência.

Bibliografia

Alhir, S. - *UML in a Nutshell.* O'Reilly, 1998.

Bennet, S., McRobb, S. e Farmer, R. - *Object-Oriented Systems Analysis and Design using UML.* McGraw-Hill, 1999.

Bennet, S., Skelton, J. e Lunn, K. - *Shaum's Outlines of UML.* McGraw-Hill, 2001.

Booch, G., Jacobson, L, Rumbaugh, J. - *The UnifiedModeling Language User Guide*. Addison Wesley, 1999.

Booch, Grady - *Object-oriented Analysis and Design with Applications.* 2ª Edição. Redwood City: Benjamin Cummings, 1994. ISBN 0-8053-5340-2

Coad, P. e Yourdon, E. - *Object Oriented Analysis.* Yourdon Press, 1991.

Conallen, J. - *Building Web Applications with UML.* Addisson-Wesley, 2000.

Eriksson, H. e Penker, M. - *UML Toolkit*. Wiley, 1998.

Fowler, M. e Scott, K. - *UML Distilled: A brief guide to the Standard object modelling language.* 2ª Edição. Addison-Wesley, 1999.

Gamma, E., Helm, R., Johnson, R. e Vlissides, J. - *Design Patterns: Elements of Reusable Object Oriented Software.* Addison-Wesley, 1995.

© FCA - EDITORA DE INFORMÁTICA

acobson L, Ericsson, M. e Jacobson A. - *The Object Advantage: Business Process Reengineering with Object Technology.* Addison-Wesley Object Technology Series, 1995.

acobson, L, Booch, G., Rumbaugh, J. - *Unified Software Development Process.* Addison Wesley, 1999.

Iagnus, P. e Hans-Erik - *Business Modeling With UML: Business Patterns at Work.* John Wiley & Sons, 2000.

Iarshall, C. - *Enterprise Modeling with UML: Designing Successful Software through Business Analysis.* Addison-Wesley Pub Co, 1999.

MDA - OMG - *Model Driven Architecture*, Document number ormsc/2001-07-01, 2001. http://www.omg.org/mda.

Meyer, B. - *Object Oriented Software Construction.* Prentice Hall, [997.

3MG - Object Management Group - *Unified Modeling Language Specificationv1.5.* 2003.
http://www.rational.com/uml/resources/documentation/index.jsp

Quatrani, T. - *Visual Modeling with Rational Rose 2000 and UML.* Addison-Wesley, 2000.

Rumbaugh, J., Blaha, M., Premeriam, W., Eddy, F. e Lorensen, W. - *Object-Oriented Modeling and Design.* Prentice-Hall r, International, 1991.

Shlaer S. e Mellor S. - *Object-Oriented Systems Analysis : Modeling the World in Data,* Yourdon Press, 1989.

Shneider, G. e Winters, Jason P. - *Applying use cases - A practical guide.* Addison-Wesley, 1999.

Wirfs-Brock, R., Wilkerson, B. e Wiener, L. - *Designing Object-Oriented Software.* Prentice Hall, 1990.

índice Remissivo

-A-

Actividade, 61
  acções, 62
  agrupamento, 65
Actividade inicial, 61
Actividade operacional, 61
Actor, 16
  conceito, 16
  descrição, 17
  generalização, 27
Agregação, 51
  âmbito do sistema, 16
Associação, 43
Atributo
  conceito, 40

-C-

Cenário, 20
Classe, 39
  conceito, 39
  controlo, 106
  interface, 106
  persistência, 112
  visibilidade, 41
Classes Associativas, 49
Componente, 123
  estereótipos, 124
  interface, 125
Composição, 51
Concorrência, 100
Condição, 80

-D-

Dependência, 107, 123
Diagrama de Actividades, 57
  actividade, 61
  actividade inicial, 61
  actividade operacional, 61
  comportamento condicional, 63
  conceito, 57
  convergência, 67
  divergência, 66
  fluxo de objectos, 67
  guardas, 63
  linhas verticais de responsabilização, 61
  transição, 62
Diagrama de Classes, 35
  agregação, 51
  aplicações, 36
  associações, 43
  classes, 39
  classes associativas, 49
  composição, 51
  conceito, 35
  generalização, 50
  multiplicidade, 44
  objectos, 38
  restrições, 48
Diagrama de Colaboração, 83
  conceito, 83
  condições, 86
  estereótipos, 85
  ligação, 87

© FCA - EDITORA DE INFORMATICA

ordenação, 84
repetições, 85
sincronização, 86
)iagrama de Componentes, 121
componente, 123
conceito, 121
dependência, 123
)iagrama de Estados, 95
conceito, 95
concorrência, 100
guarda, 98
superestado, 99
transição, 98
)iagrama de Instalação, 126
conceito, 126
linha de comunicação, 129
Nó, 127
)iagrama de Interacção
diagrama de colaboração, 83
diagrama de sequência, 76
)iagrama de Sequência, 76
conceito, 76
condição, 80
controlo, 79
iteração, 80
linha temporal, 79
mensagem assíncrona, 78
mensagem de retorno, 78
mensagem simples, 78
mensagem síncrona, 77
mensagens, 77
restrição temporal, 79
sinal de destruição, 83
)iagrama de *use cases*, 14
actor, 16
âmbito do sistema, 16
comunicação, 18
conceito, 14
relações, 24
use case, 17
Diagramas de Interacção, 75
Diagramas Físicos
diagrama de componentes, 121
diagrama de instalação, 126

-E-

Estado, 38, 41, 95
Estereótipo, 106
conceito, 8
Estereótipos, 124

-f-

Ferramentas CASE, 145
Rational Rose 2000, 148
Visio 2000, 150

-G-

Generalização, 50
herança, 50
Guarda, 63, 98

-I-

Interacção, 75
Interface, 108, 125
Iteração, 80

-l-

Ligação. *Ver Objecto*
Linha de Comunicação, 129
Linha temporal, 79

-M-

Mensagem, 77
síncrona, 77
Mensagem Assíncrona, 78
Mensagem de retorno, 78

Mensagem Simples, 78
Multiplicidade, 44

-N-

Nó, 127

-O-

Objecto, 38
comportamento, 38
conceito, 38
encapsulamento, 41
estado, 38
identidade, 38
ligação, 87
mensagens, 41
papel, 84
Operação
assinatura, 40
conceito, 40

-P-

Pacote, 113
hierarquia, 114
visibilidade, 114
Persistência, 112
Processo de Modelação
Unificado, 137
actividades, 137
fases de desenvolvimento, 138
Processo de negócio, 57

-R-

Rational Rose 2000, 148
Realização, 108
Requisito, 13
categorias, 13
conceito, 13
Restrição temporal, 79
Reutilização, 105

-S-

Sinal de Destruição, 83
Subactividade, 65
Superactividade, 65
Superestado, 99

-U-

UML, 3
arquitectura, 11
definição, 3
história, 4
notação, 6
UMP, 9
UMP, 9, 140
Use cases, 13
cenário principal, 20
cenários secundários, 20
comunicação, 18
conceito, 13
generalização, 27
negócio, 17
pós-condição, 22
pré-condição, 22
relações, 24
sistema, 17
tempo, 19

-V-

Visio 2000, 150

© FCA - Gr. Tecn. DE INFORMÁTICA