Capítulo 3
Modelação Conceptual do Sistema

Versão provisória 2001-05-04

Conteúdos

Índice de Figuras

3.1 Modelos e Metodologias

Uma forma efectiva de preparar a construção de uma aplicação ou de um sistema passa por produzir inicialmente um modelo de todas as suas componentes mais relevantes. Esta é uma prática normal em todos os ramos de engenharia, que utilizam os modelos como instrumentos para comunicação objectiva, para teste de hipóteses ou para prever desempenhos. Como exemplos concretos de modelos podemos referir uma carta topográfica, a maqueta de um edifício e o seu projecto estrutural, o desenho de um circuito integrado, um conjunto de equações diferenciais ou uma sequência de bases para proteínas ou medicamentos a sintetizar.

Um modelo é sempre mais simples do que a realidade que pretende retratar, permitindo assim aos intervenientes na sua definição concentrarem-se nos aspectos mais relevantes do problema. Por exemplo, uma carta de um dado local pode focar aspectos topográficos, se se destinar a preparar o traçado de uma estrada, ou pode focar a constituição química dos terrenos se se destinar a preparar alterações às actividades agrícolas aí existentes.

Dependendo do problema em causa, o modelo poderá tornar-se mais simples ou mais complexo. A decisão relativa ao grau de complexidade requerida nos modelos, depende dos recursos existentes, técnicos e económicos, que pode por sua vez envolver outros tipos de modelos. A experiência existente em problemas semelhantes, associada a uma grande dose de bom senso, pode ajudar a controlar a complexidade dos modelos necessários. Comparativamente com outras disciplinas, a engenharia de suportes lógicos é recente e por isso a sua base de experiências muito mais reduzida. Este facto implica que o cuidado a ter no processo de modelação deve ser comparativamente maior, bem como o grau de bom senso.

Modelos Conceptuais

Este capítulo apresenta os principais modelos que são necessários para construir sistemas de informação, em particular sistemas que envolvem bases de dados. Sendo estes sistemas lógicos de natureza discreta, os modelos para preparar a sua construção terão de ajudar a fazer a passagem das nossas intuições relativas à realidade para implementações discretas. Para tal tem importância fundamental o conceito de entidade ou *objecto*, bem como toda a teoria do conhecimento que lhe está subjacente, e que apreendemos desde muito cedo. Esta teoria permite-nos por um lado organizar a realidade, analisar os objectos que a formam e relacioná-los uns com os outros, e por outro lado representá-la com diagramas, palavras e outras notações abstractas, isto é, recorrendo a uma linguagem de comunicação.

Os modelos de seguida apresentados, designados no seu conjunto por MOOT, *modelos orientados por objectos e tipos*, constituem assim uma linguagem, com regras próprias. O seu uso efectivo na construção de sistemas informáticos de qualidade, pressupõe a utilização de um método ou metodologia de trabalho, com fases e processos bem definidos, que sirva de referencial para organização do projecto. Esta metodologia é também designada por MOOT, *modelação orientada por objectos e tipos*.

Metodologia de Modelação

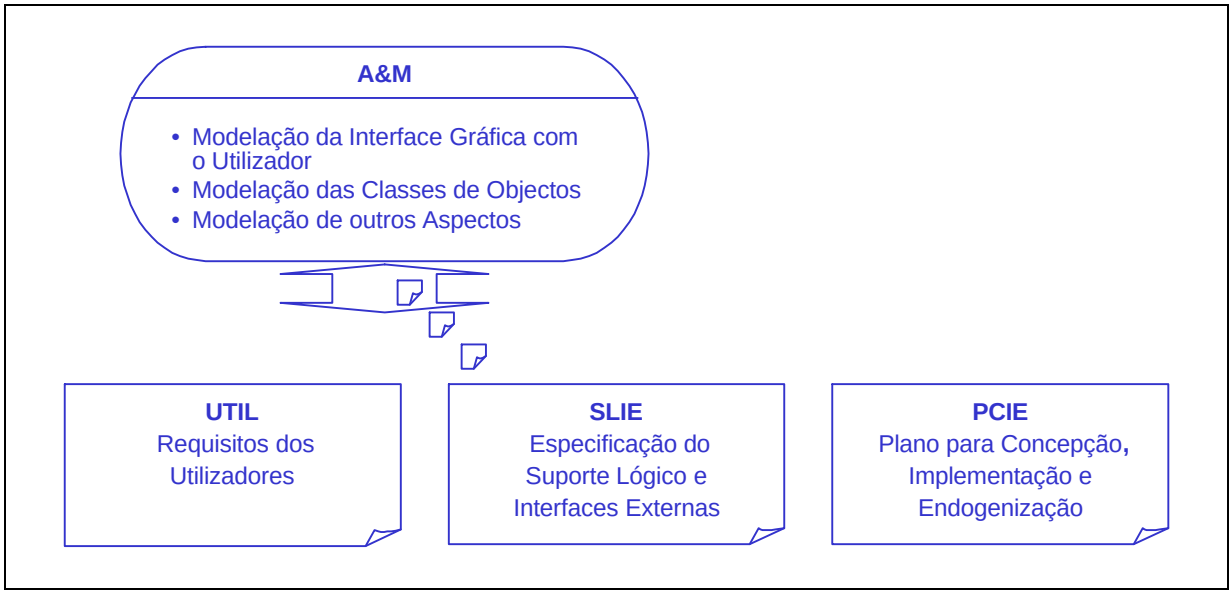


Figura 3.1 Processos de Modelação MOOT e Documentação Resultante. Neste capítulo tratamos essencialmente do processo de Modelação de Classes de Objectos, embora não seja um processo autónomo dos restantes.

Grande parte dos modelos aqui representados são semelhantes aos propostos pela UML, "Universal Modeling Language". No entanto, pelos motivos anteriormente referidos, dá-se particularmente ênfase ao modelo de entidades ou objectos, que se inspira no modelo idêntico OMT [Rumbaugh *et al* 1991] e indirectamente no modelo entidade-associação [Chen 1976]. A notação utilizada por este último modelo é brevemente apresentado em Anexo*, na versão de [Teorey *et al* 1986].

Uma outra versão da OMT é apresentada em [Blaha & Premerlani 1998], com um modelo de objectos recorrendo apenas a algumas noções de UML, e com notações dinâmicas e funcionais próprias.

Modelo dos Utilizadores e dos Casos de Utilização

Uma vez que os sistemas se destinam normalmente a muitos utilizadores concretos, é importante conseguir tipificar estes utilizadores em grandes grupos semelhantes e obter ou propor com o detalhe adequado descrições de casos de utilização. Esta tipificação vai permitir elaborar os documentos UTIL e SLIE, que por sua vez irão permitir melhorar o plano do projecto.

Por exemplo, num sistema de apoio à gestão de uma loja pode ser relevante identificar dois tipos de utilizadores: o gestor da loja, com acesso pleno a toda a informação e estatísticas particulares e globais, e o cliente da loja, com acesso a informação sobre produtos e suas existências. Cada um deste utilizadores terá exigências ou interesses próprios, que resultam por exemplo em requisitos, interfaces gráficas e algoritmos diferentes, eventualmente com alguns aspectos comuns.

* A disponibilizar em breve.

A criação ou identificação de situações ou cenários de utilização assume uma grande importância para que o processo de modelação conceptual de classes permita ser completo. A discussão prévia e detalhada de todos os casos de utilização, com eventual atribuição de prioridades ou grau de interesse pode também ajudar no processo de construção, permitindo planificar o projecto com base em julgamentos de prioridade versus esforço necessário. Obviamente as funcionalidades que os utilizadores julgam mais importantes e fáceis de implementar devem claramente estar suportadas no modelo conceptual e devem ser as primeiras a estar disponíveis nos protótipos.

Em UML existe o modelo de Casos de Utilização para tratar deste assunto. Aqui vamos assumir para já que é possível, recorrendo a um método informal, extrair ou propor casos de utilização. De seguida apresentam-se alguns exemplos de casos de utilização de um sistema para gestão de carteiras de produtos financeiros (adaptado de [Blaha & Premerlani 1998; 149]).

Exemplo 3.1: Casos de utilização para gestão de carteiras de produtos financeiros:

- Adicionar, eliminar e actualizar informação sobre uma pessoa.
- Adicionar, eliminar e actualizar informação sobre uma instituição financeira.
- Adicionar, eliminar e actualizar informação sobre uma carteira de títulos.
- Pesquisar todas as transacções sobre uma carteira de títulos.
- Calcular o valor de uma carteira de títulos em qualquer data.
- Calcular o valor de todas as comissões a cobrar pela gestão de uma carteira num dado intervalo de tempo.
- Estimar o valor de cada um dos títulos de uma carteira no prazo futuro definido.

O exemplo anterior não é de forma alguma exaustivo. Uma definição completa dos casos de utilização deveria contemplar todo o conhecimento relevante, obtido por exemplo a partir de documentação escrita existente sobre o problema, obtido através de conversas com peritos ou utilizadores actuais, e eventualmente comparando com outros sistemas semelhantes em funcionamento ou a que se tenha acesso. Um processo exaustivo de modelação com Casos de Utilização deveria identificar os *actores* externos ao sistema em estudo, podendo tratar-se de pessoas ou inclusivamente de outros sistemas, e descrever exaustivamente cada uma das possibilidades de utilização do sistema, incluindo aspectos de sequência temporal ([Schneider & Winters 1998] apresenta de uma forma guiada todo o processo de modelação UML de Casos de Utilização). Obviamente que se devem considerar casos de utilização inovadores, dadas as possibilidades futuras abertas por um novo sistema. O suporte à inovação é fundamental, mas a sua disponibilização num ambiente empresarial deve ser pensada com cuidado, em conjunto com os futuros utilizadores.

O Processo de Modelação Conceptual

O processo de modelação conceptual insere-se no âmbito da criação de suportes lógicos ou na concepção de sistemas com componentes informáticas. Este processo inicia-se normalmente no âmbito de um processo de reengenharia de uma organização ou de inovação empresarial.

Como vimos, a definição de um dado modelo passa naturalmente pela compreensão do problema em causa, o que requer um grande envolvimento com os seus especialistas, e o domínio dos seus dialectos técnicos. Passa também pelo conhecimento das notações de modelação, em particular do seu significado em termos das aplicações informáticas que vão ser construídas. No caso limite em que uma notação é definida como uma linguagem formal, passamos a ter também acesso ao suporte de uma sintaxe e semântica rigorosa, que têm de ser respeitadas.

A construção de um modelo conceptual de classes para o caso de um sistema concreto deve passar assim pelas seguintes fases:

Fase prévia: Identificar e estudar todas as fontes relevantes de conhecimento sobre o sistema, com o objectivo de tipificar os utilizadores e descrever exaustivamente os casos de utilização. Devem nesta fase também já ter sido elaboradas descrições do contexto e do problema, bem como maquetas de baixa resolução das principais interfaces gráficas com o utilizador.

Fase de Modelação de Classes ([Blaha & Premerlani 1998; 127]):

1. Identificar Classes (procurar nomes próprios e nomes comuns).
2. Identificar Associações (procurar verbos transitivos e preposições).
3. Acrescentar Atributos, detalhando as Classes e Associações (frases preposicionais possessivas).
4. Utilizar Generalizações para caracterizar semelhanças e diferenças entre objectos.
5. Testar caminhos de acesso e pesquisa de informação.
6. Iterar e refinar o modelo, acrescentando, eliminando ou alterando o detalhe ou nível de abstracção.
7. Organizar graficamente o modelo final.

Fase seguinte: Traduzir o modelo de classes para o modelo do sistema informático mais apropriado a utilizar, por exemplo, obtendo o esquema relacional no caso de se perspectivar a implementação com apoio num SGBDr.

Voltaremos a este tema mais em detalhe no final deste capítulo.

3.2 Modelo Estático

Como o nome indica, este modelo pretende capturar os requisitos de informação do sistema que são estáveis no tempo, ou que se deseja que não venham a sofrer alterações. Uma parte importante deste modelo poderá assim dar origem à declaração dos tipos de dados abstractos de um programa ou ao esquema de uma base de dados relacional.

O modelo estático pode ser definido com uma abordagem do geral para o particular ou inversamente, procurando enriquecer os primeiros pormenores identificados.

Exemplo 3.2: Descrição geral da empresa OnTime:

> "A empresa OnTime presta aos seus clientes um serviço de transporte de pacotes de média e grande dimensão. Para tal conta com um grupo dedicado de motoristas e com viaturas de vários tipos".

Exemplo 3.3: Descrições particulares da empresa OnTime:

> "A empresa OnTime dispõe de uma frota própria de viaturas de carga, ligeiras e pesadas, incluindo reboques e atrelados. Dispõe ainda de viaturas que pertencem a clientes".
>
> "Cada serviço contratado é realizado num dado percurso, formado por uma sequência ligada de trajectos. Cada percurso tem origem e destino em depósitos de viaturas, podendo estes depósitos ser diferentes. Cada trajecto, com um nome único, é caracterizado por uma origem, um destino e uma distância em quilómetros. Por exemplo, o serviço OTN3456, previsto e realizado no percurso de Porto-Serralves para Braga-Centro, utilizou a viatura 34-98-DH e foi assegurado pelo Senhor Jorge Sampaio. Este percurso foi formado pela sequência ligada dos seguintes 3 trajectos: Porto-Leixões/Porto-Serralves, Porto-Serralves/Braga-Centro e Braga-Centro/Porto-Leixões. O sistema de planeamento operacional GistRv sugeriu que fossem utilizados os trajectos indicados para uma viatura do depósito Porto-Leixões".

Com base numa descrição da situação vamos ter de identificar e caracterizar os tipos de entidades presentes. Nas descrições anteriores temos por exemplo as entidades "cliente" "motorista" e "viatura". No modelo estático este processo passa numa primeira fase por definir as *classes* dos *objectos*, incluindo os seus *atributos*, e numa segunda fase por identificar e tipificar os *métodos*, isto é, as operações que podem modificar os valores dos atributos de objectos.

De seguida temos de perceber como as várias entidades se relacionam entre si. São particularmente importante as relações de *composição* ou *constituição* de objectos, como é o caso dos percursos constituídos por trajectos do Exemplo 3.3. Também é igualmente relevante numa primeira fase procurar identificar grupos de entidades que parece natural terem tipos semelhantes. No exemplo referido, podemos classificar as viaturas de carga, ligeiras e pesadas, reboques e atrelados como um grupo gnérico de viaturas. No modelo estático este processo passa pela definição de *ligações* estruturais de *associação* e de *agregação* entre classes de objectos e pela definição de hierarquias de *generalização* ou *particularização*.

Embora os conceitos que vão ser apresentados de seguida sejam difíceis de definir, devido à multiplicidade de usos a que se prestam, é fácil aceitarmos intuitivamente que a realidade é composta por entidades ou objectos. Também aceitamos dar nomes aos objectos, para facilitar a comunicação, e os agrupamos em classes, tipos ou conjuntos, talvez por economia de pensamento. Este processo de classificação passa por atribuir propriedades e valores, e por relacionar e analisar os objectos, de acordo com a perspectiva do problema que se está a resolver. Podemos pensar na actividade de um especialista em botânica, ao classificar e agrupar as espécies vegetais, ou na actividade de um físico ou químico que tem de analisar a constituição da matéria. Esta actividade de classificação tem por objectivo capturar regras simples que possibilitem descrever ou explicar o sistema em análise, e mais tarde melhorá-lo ou modificá-lo.

O Processo de Modelação Estática

A definição de um modelo estático está integrada no processo global de modelação, e deve seguir o método estabelecido. Há contudo regras próprias que sugerem que se deve iniciar o processo com a identificação das classes, e em seguida das associações. Este passo deve ser iterado acrescentando detalhe e alterações, por exemplo através de particularizações com novos tipos de classes, ou mesmo pela transformação de atributos ou associações complexas em novas classes. É importante ter acesso a vários pontos de vista sobre o problema, em particular e como vimos, às descrições dos peritos sobre o problema, a todos os relatórios existentes na empresa que o sistema vai ter também de produzir, e devemos manter uma grande interacção com os actuais e futuros utilizadores, procurando que eles vão dando a sua opinião e apresentem críticas do modelo. Para tal eles têm de o compreender.

3.2.1 Classes de Objectos, Atributos e Métodos

Objectos e Atributos

Um ***objecto*** corresponde a um conceito com individualidade e identidade própria, normalmente identificado com um nome, sem ambiguidade num dado contexto ou situação. Podemos ter objectos concretos ou abstractos, mas um objecto é sempre único. A palavra entidade é por vezes utilizada como sinónimo da palavra objecto, mas pode ter também o significado associado ao conceito de classe, nomeadamente no modelo Entidade-Associação. Normalmente utilizamos a palavra entidade sem preocupação de rigor formal, ao contrário da palavra objecto, que como veremos mais tarde, tem um significado formal preciso.

Numa situação concreta, por exemplo numa reunião com um perito, ou perante a descrição de um problema, podemos começar por identificar os objectos com os nomes próprios presentes nos sintagmas nominais. Nesta acepção, temos como exemplo de objectos o motorista Jorge Sampaio, no contexto do Exemplo 3.3, o poeta Fernando Pessoa ou a Princesa Leia, bem como o serviço OTN3456 ou o teorema da Incompletude de Gödel.

Um objecto é normalmente observado segundo várias perspectivas, seja ele concreto ou abstracto. Em particular, temos a tendência para lhes atribuir nomes, como identificação externa, e para os classificar segundo dimensões estabelecidas.

Em geral chamamos ***atributo*** a uma dimensão de observação que nos permite dar um valor a um objecto, valor esse que pertence a um conjunto bem definido, como veremos adiante. Os valores não têm identidade, ao contrário dos objectos, devido a serem abstracções matemáticas perfeitas. Os atributos estão normalmente associados aos adjectivos, numa descrição do problema.

A Figura 3.2 apresenta alguns exemplos de objectos, seguindo a notação gráfica que propomos, e que está reunida e comentada em anexo. Como podemos ver, um objecto representa-se como um rectângulo dividido em duas partes por um segmento de recta horizontal. Na parte superior aparecem o nome e a classe a que o objecto pertence, sublinhados. Na parte inferior do rectângulo aparecem nomes e valores dos atributos.

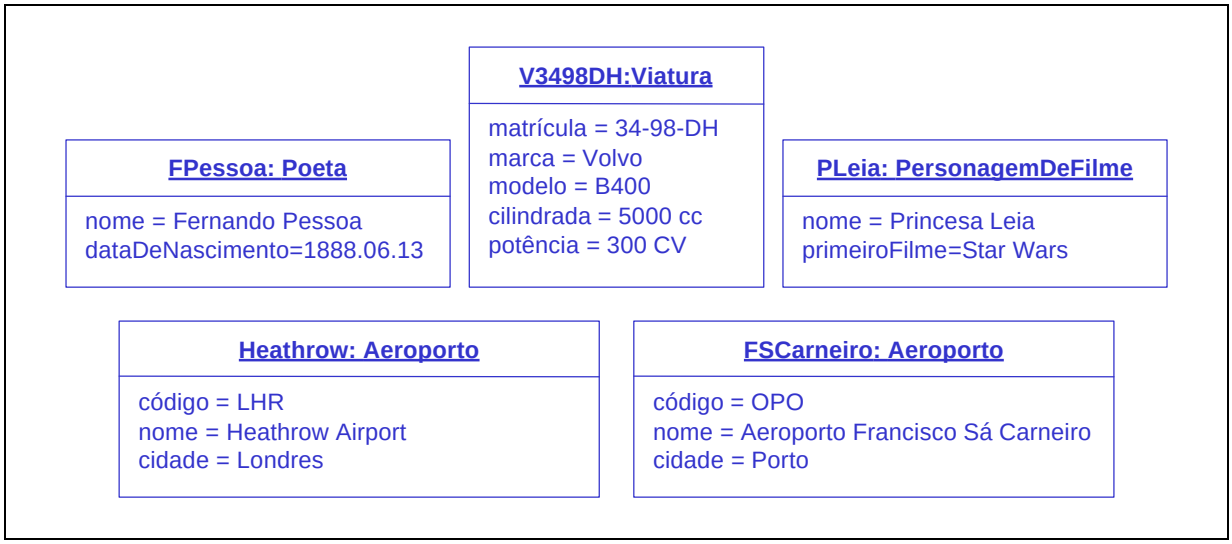


Figura 3.2 Exemplos de objectos, com valores para atributos.

Classes de Objectos

Uma ***classe de objectos*** ou simplesmente ***classe*** corresponde à caracterização formal de um grupo de objectos semelhantes, no que eles têm de comum, em relação com o problema ou situação em estudo. Diz-se então que os objectos são instâncias de uma dada classe. Normalmente utilizamos a palavra grupo sem preocupação de rigor formal, ao contrário da palavra classe, que como veremos mais tarde, tem um significado formal preciso. As classes são os nomes comuns ou sintagmas nominais das descrições.

A notação para classes segue o formato da notação para objectos. As figuras seguintes apresentam vários exemplos de classes. A Figura seguinte representa as classes *Aeroporto*, *CódigoPostal* e *Livro*. Esta última classe deve ser entendida no âmbito de uma biblioteca que pode ter várias cópias de um mesmo livro. As instâncias desta classe, isto é os objectos, correspondem assim aos exemplares reais dos livros existentes na biblioteca. Neste contexto o ISBN não é suficiente para identificar unicamente um dado livro, sendo necessário um outro identificador externo para cada objecto. Neste caso esse identificador pode ser fornecido pelo par *(códigoISBN, nExemplar)* ou apenas por *nExemplar*, dependendo do significado ou Domínio de valores deste último atributo.

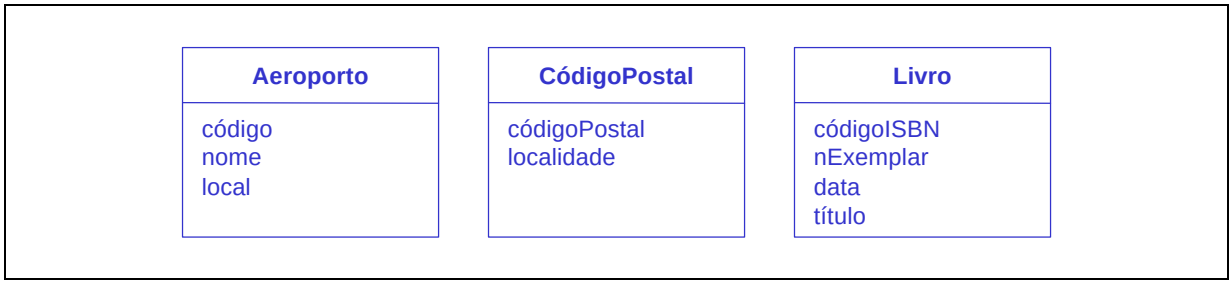


Figura 3.3 Exemplos de classes, com indicação de atributos.

A própria escolha das classes de objectos e respectivos atributos depende também da fase de processo em que nos encontramos. Enquanto numa fase de análise será normalmente suficiente indicar para a classe *Pessoa* os atributos *nome* e *endereço*, numa fase de concepção

de um SGBD é importante separar o nome e o endereço das pessoas nos seus atributos constituintes, por exemplo, para permitir a produção de etiquetas, formatadas adequadamente.

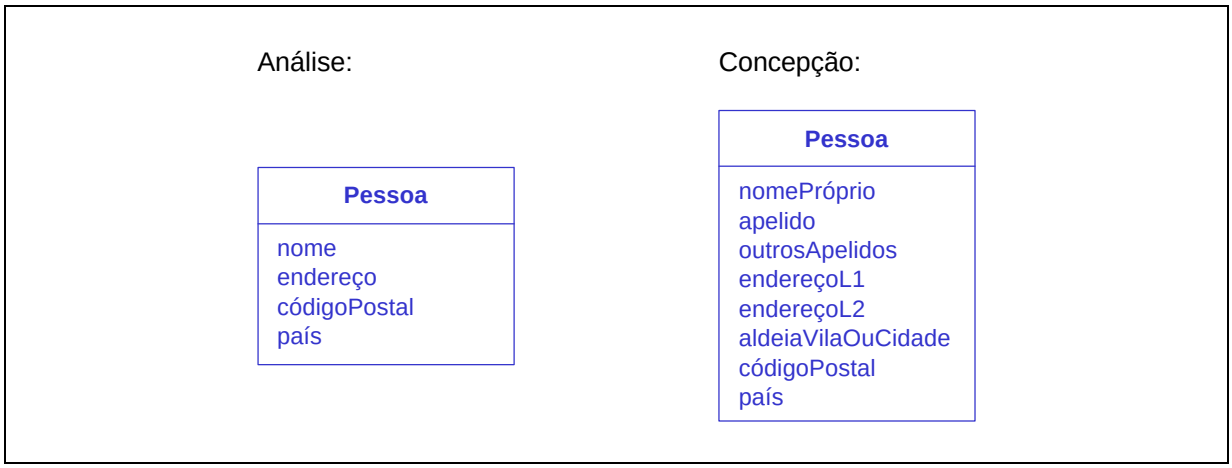


Figura 3.4 Diferentes níveis de detalhe na definição da classe Pessoa.

Na Figura 3.10 são apresentadas várias outras classes de objectos, já num contexto em que os relacionamentos, formalizados na secção seguinte, estão também identificados.

A escolha dos nomes das classes e atributos deve ser feita com extremo cuidado, devendo sempre que possível reflectir a linguagem da situação, tal como é utilizada, e procurando desfazer ambiguidades que existam, sempre em sintonia com os peritos do problema. A participação destes e de futuros utilizadores do sistema nesta fase de análise, e a crítica que possam fazer aos modelos é de fundamental importância para a qualidade global do sistema a desenvolver e sucesso na sua implantação.

Nota 3.1: Devido à necessidade de identificar cada objecto unicamente em todo o sistema, tem de existir um atributo externo, ou conjunto de atributos externos, que o permita fazer. Conforme se pode ver mais adiante, tais atributos são designados chaves candidatas a identificador. Contudo, por várias razões de implementação, deve-se sempre vir a ter para cada classe um único atributo identificador, interno ao sistema que a virá a implementar. Os códigos identificadores externos ou públicos, tais como o número do Bilhete de Identidade ou o código de um motorista na empresa, estão sujeitos a ser alterados com o tempo, por exemplo devido a erros de introdução no sistema pelos operadores, e não são assim adequados a servir de referências internas. Uma vez que nos sistemas informáticos os atributos identificadores são normalmente utilizados para suportar referências aos objectos, não convém que estes sejam atributos externos, pois em caso de alteração do objecto original tornar-se-ia necessário proceder a alterações de todas as suas referências ou ligações. A nível de implementação, a identificação única de objectos é automaticamente suportada nas bases de dados orientadas por objectos, mas não nas bases de dados relacionais, onde esse processo tem de ser garantido por adequada concepção e programação. Durante o processo de tradução de cada classe para uma tabela de uma base de dados relacional, deverá ser garantida a criação de um atributo identificador interno para cada classe. Por tais motivos é suficiente que os atributos apresentados nos modelos de classes sejam apenas os atributos externos, pelo menos ao nível do modelo de análise.

Operações e Métodos de Objectos

As **operações** são funções ou procedimentos definidos no âmbito de um objecto com determinados argumentos e resultado, implementados como **métodos** da respectiva classe, por exemplo. Algumas operações são polimórficas, isto é o tipo dos seus argumentos pode variar (ver 3.2.4). Os nomes das operações representam-se a seguir aos atributos, separados por uma linha horizontal. Na representação dos atributos, podemos também indicar à direita do sinal de igualdade, o seu valor inicial por defeito, se tal for relevante em termos de análise do problema.

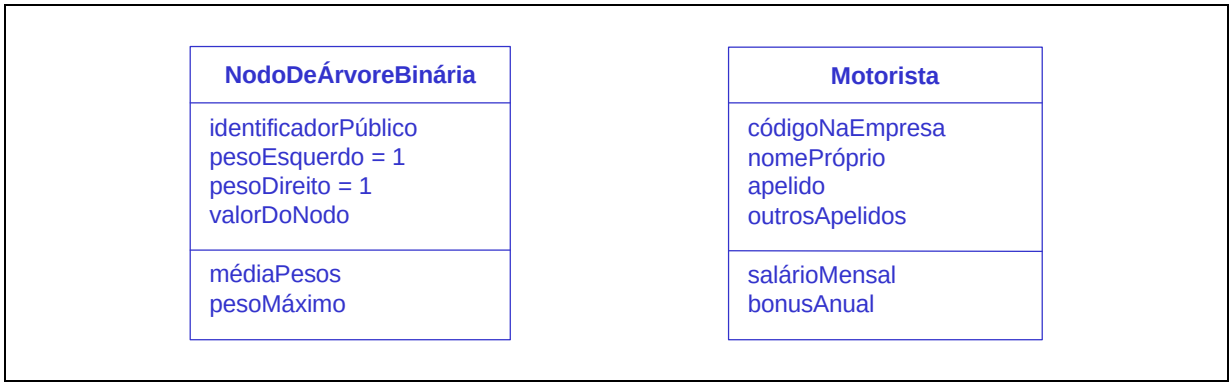


Figura 3.5 Classe para os nodos de uma árvore binária e para motoristas.

Domínios ou Tipos de Dados e Operações Complexas em Classes

Vamos considerar aqui que um domínio corresponde a um tipo de dados primitivo, tal como o tipo **Natural** (abreviado **Nat**), **Inteiro** (abreviado **Int**), **Real** ou conjuntos enumerados. Um domínio é assim semelhante a um conjunto de valores, valores estes que sendo entidades abstractas não têm identidade, ao contrário dos objectos, que correspondem a entidades com identidade na realidade do problema. A noção formal de domínio corresponde à noção tradicional de um conjunto a que se adiciona normalmente um elemento com valor *indefinido*, representado por vezes como $\perp$. Por exemplo, o domínio dos números inteiros inclui também um elemento de valor indefinido. Este elemento é utilizado para modelar os valores nulos em campos de bases de dados ou os resultados indefinidos de operações ou métodos. Este elemento é necessário para estudar o significado matemático das instruções e comandos nas linguagens de programação [Stoy 1983], ou o impacto das indefinições e dos erros na execução das aplicações informáticas [Gordon 1986].

Um domínio corresponde a um tipo de dados definido a partir dos tipos primitivos através de construções simples. Os valores de um domínio não podem ser obtidos por construções complexas, estando à partida excluídos valores de complexidade semelhante aos objectos das próprias classes definidas no modelo (ver questão 3.4, no final do capítulo). Na Figura 3.6 podemos observar que aos atributos e operações da classe *NodoDeÁrvoreBinária* foram acrescentados tipos.

Figura 3.6 Indicação de domínios para os atributos e métodos da classe correspondente aos nodos de uma dada árvore binária. Para os atributos pesoEsquerdo e pesoDireito indica-se também o valor inicial.

No processo de implementação estes tipos têm de ser mapeados em tipos suportados pelos sistemas a utilizar ou pelas suas linguagens de programação. Nos SGBD relacionais a variedade de tipos de dados suportados directamente é normalmente muito limitada, ao contrário por exemplo dos SGBD orientados por objectos. Tipicamente um domínio **Int** seria implementado como um Integer ou LongInteger.

Há também a possibilidade de recorrer a domínios estruturados que se tornam úteis para modelar SGBDOO ou extensões OO aos SGBDR, que suportam tipos de dados mais complexos (ver por exemplo [Blaha & Premerlani 1998; 46] ou [Stonebraker & Moore 1996]).

Atributos Derivados

Um atributo derivado contem informação que pode ser facilmente calculada a partir da informação presente em outros atributos. Por exemplo, dada a hora de partida e a duração de um serviço, é simples calcular o valor da hora de chegada. Em termos de implementação, por exemplo recorrendo a um SGBD, coloca-se a questão de tornar um atributo derivado persistente ou de calcular o seu valor no momento de consulta pelo utilizador. Representa-se um atributo derivado com o símbolo inicial “ ” , como se pode observar na classe Serviço da figura seguinte.

Figura 3.7 Exemplo de atributo derivado correspondente à hora de chegada.

Atributos de Classe e Operações de Classe

Podem por vezes ser necessários os chamados atributos de classe. Estes atributos são por exemplo úteis para manter informação comum a todos os objectos da classe, como parâmetros ou constantes, tal como a Figura 3.8 ilustra. No entanto não é normalmente uma boa solução recorrer a estes atributos, pois existe sempre a possibilidade de definir outra classe, correspondente aos parâmetros ou constantes, e recorrer a associações para manter a caracterização desejada. De igual forma podemos utilizar operações de classe, por exemplo tendo em vista criar um novo objecto [Blaha & Premerlani 1998, 42]. Uma operação de classe tem assim como domínio a própria classe e não objectos ou seus atributos.

Figura 3.8 Exemplos de Atributo de classe, a evitar, e Operação de classe. Atributos e Operações de classe são representados com o nome e valor sublinhado.

Estas noções são sobretudo importantes para evitarmos usar atributos e operações de classe como se fossem normais atributos ou operações de objecto.

Atributos candidatos a identificadores

Cada classe pode ter um atributo externo que permita identificar todas as suas instâncias, isto é nomear cada um dos possíveis objectos da classe, sem ambiguidade. Na realidade pode haver mais do que um atributo que satisfaça esta condição, e assim teremos vários identificadores candidatos. Podemos também ter necessidade de um conjunto de atributos para atingir este objectivo, pelo que então teremos um identificador composto. Neste último caso os atributos têm de ser em número mínimo. Isto é, se retirando do conjunto um ou mais atributos se mantiver a possibilidade de identificação, então será este conjunto menor o identificador. Obviamente nenhum identificador composto pode incluir um atributo identificador simples. Na Figura 3.9 representam-se identificadores, para o caso simples da classe *Aeroporto*, e para o caso composto da classe *Livro*, supondo que o atributo *nExemplar* refere o número do exemplar de um dado livro existente numa biblioteca.

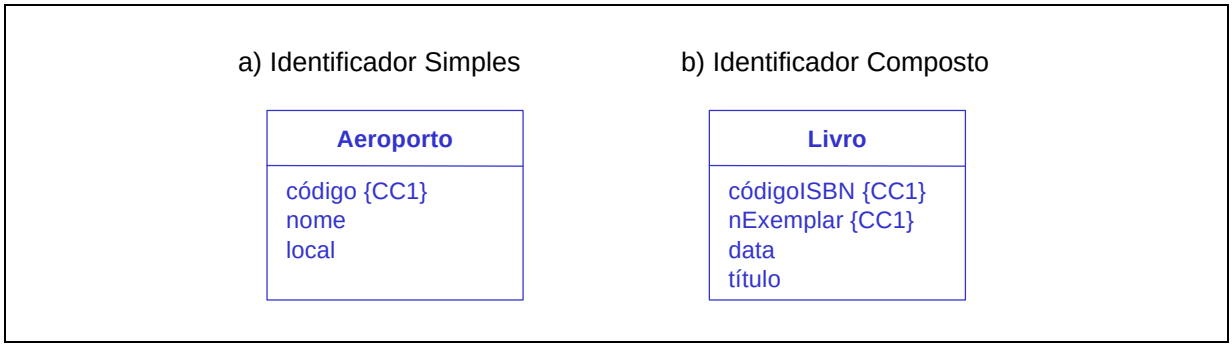


Figura 3.9 Identificadores candidatos simples e compostos para as classes Aeroporto e Livro.

Este conceito estender-se-á mais tarde naturalmente à identificação de associações entre objectos.

De seguida apresentam-se tabelas resumindo os conceitos apresentados.

Tabela 3.1 Tipificação e Exemplo de Atributos

Identificador do objecto	Simples	Composto
Exemplo:	Figura 3.9 a)	Figura 3.9 b)

Contradomínio	Contradomínio sem estrutura	Contradomínio estruturado
Exemplo:	Figura 3.6	-

Nível	Atributo de Objecto	Atributo de Classe
Exemplo:	*(em várias figuras)*	Figura 3.8

Tabela 3.2 Tipificação e Exemplo de Operações

Nível da operação	Operação de Objecto	Operação de Classe
Exemplo:	*(em várias figuras)*	Figura 3.8

Significado implementacional e Significado denotacional

Sem uma fundamentação matemática, ou formal, os conceitos e modelos apresentados não têm a menor aplicabilidade do ponto de vista de uma disciplina e rigor de engenharia, uma vez que lhes falta um padrão semântico e uma consistência objectiva, e assim o seu significado fica dependente de qualquer interpretação particular, pessoal ou empresarial.

Uma semântica formal, referida normalmente aqui por significado denotacional, requer a utilização da noção de Tipo Abstracto de Dados (evolução de conceitos base e implementações respectivas nas linguagem Pascal, Módula, Ada e Eiffel). São também muito relevantes os trabalhos matemáticos aplicados à teoria do significado das linguagens de Scott, Stoy, Hoare, Dijkstra e Meyer, para citar apenas alguns.

Vamos aqui procurar definir dois tipos de significado que podem ser dados aos modelos anteriores: o significado implementacional e o significado denotacional.

- Significado Implementacional: Por exemplo, envolvendo a tradução para o modelo relacional, baseado na álgebra relacional, seguindo-se a utilização de um dado SQL e implementação com auxílio de uma dada linguagem de programação apropriada. Todos estes passos efectuados com base em normas existentes.
- Significado Denotacional: Por exemplo, envolvendo modelos formais recorrendo a uma teoria de conjuntos ou domínios, e à lógica formal para fornecer um significado de referência, abstracto, independente de uma particular sequência de implementação.

Obviamente que estas abordagens têm os seus riscos. Não é garantido que os técnicos das empresas informáticas, que constroem os compiladores e interpretadores, cumpram as especificações das linguagens. Por exemplo, a vontade de melhorar a linguagem, a incapacidade de compreender a sua especificação, ou simplesmente um erro de programação pode levar a que não suceda o que estava previsto e formalmente seria de esperar. De facto passa-se o mesmo com uma empresa de construção civil. O projecto de construção de uma ponte deve estar formalmente correcto, de acordo com modelos físicos experimentalmente validados, por exemplo atendendo às cargas máximas a suportar e ao padrão e frequência de ventos da região. O projecto de construção da ponte deverá igualmente guiar-se pelos modelos padrão existentes e seguir as suas especificações, mas pode haver problemas ou incorrecções no processo de construção que é necessário precaver e controlar.

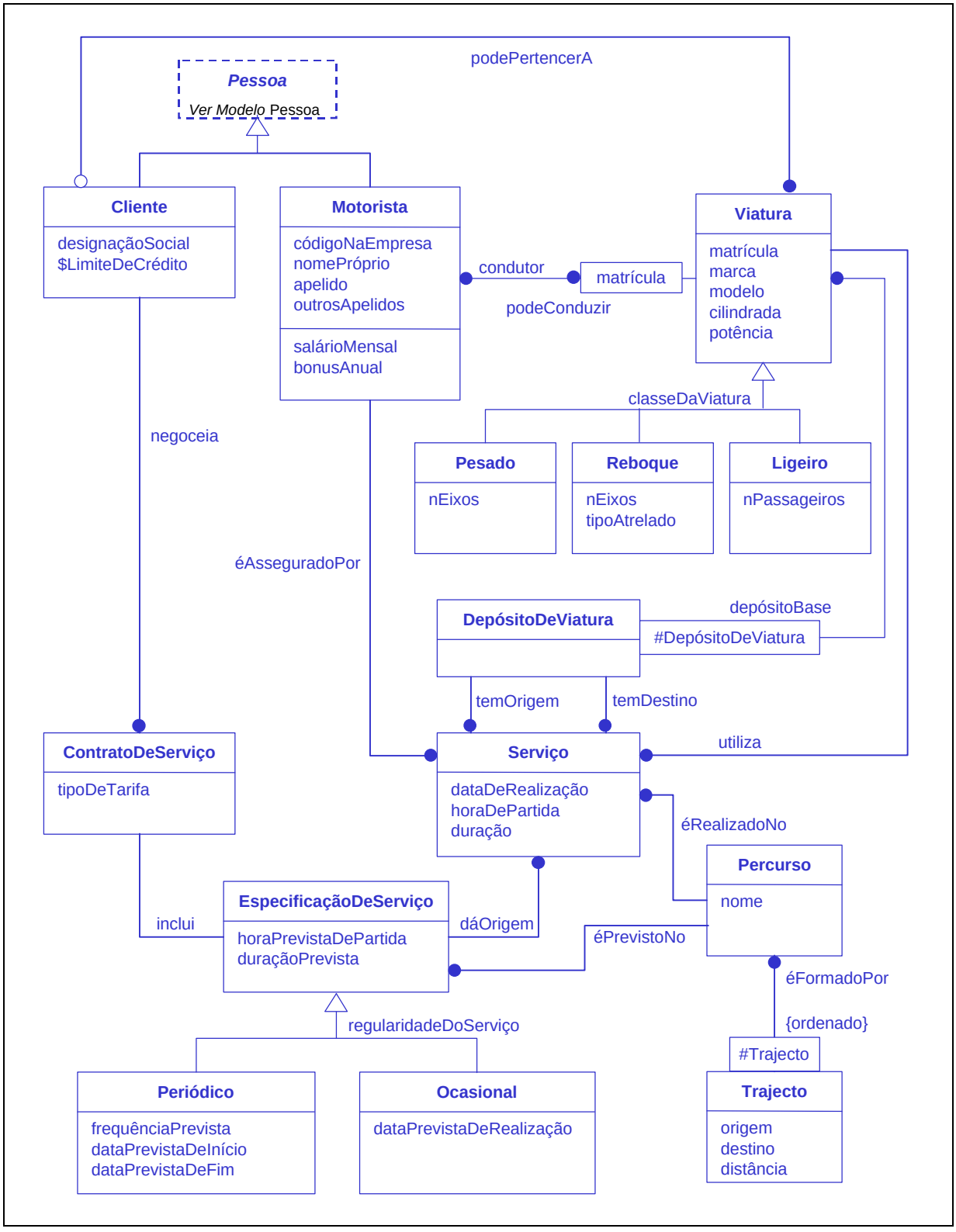


Figura 3.10 Modelo estático do subsistema de gestão de transportes da empresa ONTIME.

3.2.2 Ligações, Associações e Agregações Simples

Uma vez classificados os grupos de entidades, como objectos e classes, precisamos agora de observar como é que estes se relacionam uns com os outros. Intuitivamente, há entidades que

são autónomas entre si, mas que interagem ou comunicam, há outras que se agrupam para formar entidades mais complexas, e há ainda outras que nós próprios criamos, tendo normalmente uma existência abstracta dependente da de outras entidades.

Imaginemos por exemplo o caso de uma autarquia em particular, com os seus clientes, funcionários, departamentos, divisões e serviços. Os clientes são por exemplo os residentes que pagam a contribuição autárquica.

Exemplo 3.4: Descrição particular da Autarquia de Porboa:

> "A Senhora Teresa Ramos tem 3 processos de contribuição autárquica (45677A1989, 356B1995 e 357B1995), geridos pelo Serviço SCAUTA, um dos 7 serviços da DF&P, Divisão de Finanças e Património. O Senhor Alberto Matias, um dos 2998 funcionários actuais da Câmara de Porboa é o Chefe do SCAUTA, onde trabalham mais 25 funcionários".

É possível identificar vários grupos de entidades, relacionando-se entre si das várias formas acima referidas, como o leitor atento facilmente pode concluir.

Uma entidade pode naturalmente desempenhar diversos papéis, consoante os relacionamentos em que está envolvida, exactamente como um actor pode tomar o lugar de várias caracteres em filmes diferentes, ou inclusivamente no mesmo.

Estas noções vão de seguida ser formalizadas, recorrendo aos conceitos de *ligação*, *associação* e *agregação*, sendo este último conceito um caso particular de associação, envolvendo um grau de composição física.

Ligações entre objectos

Estabelece-se uma **ligação** entre objectos quando é importante manter a informação que os relaciona de alguma forma relevante para a situação em estudo. A Figura 3.11 representa por exemplo as duas ligações possíveis entre um dado motorista e as viaturas que ele pode conduzir. A maior parte das ligações estabelecem-se entre dois objectos mas há a possibilidade de haver ligações entre três ou mais objectos. Como veremos, este facto é uma consequência de se utilizarem preferencialmente associações binárias para modelar a realidade, devido à sua simplicidade conceptual. No caso do exemplo da Figura 3.20, haveria necessidade de utilizar instâncias com ligações entre três objectos.

Pode também suceder que a existência de um dos objectos na ligação dependa da do outro. Considere-se por exemplo uma ligação entre um objecto do tipo *Cliente* e outro do tipo *Empréstimo*. A existência deste último objecto depende da existência do primeiro, não fazendo sentido sem a presença do outro, ou isoladamente dele. Diz-se deste último que é um objecto dependente. O modelo E-A designa como *entidade fraca* este tipo de objecto.

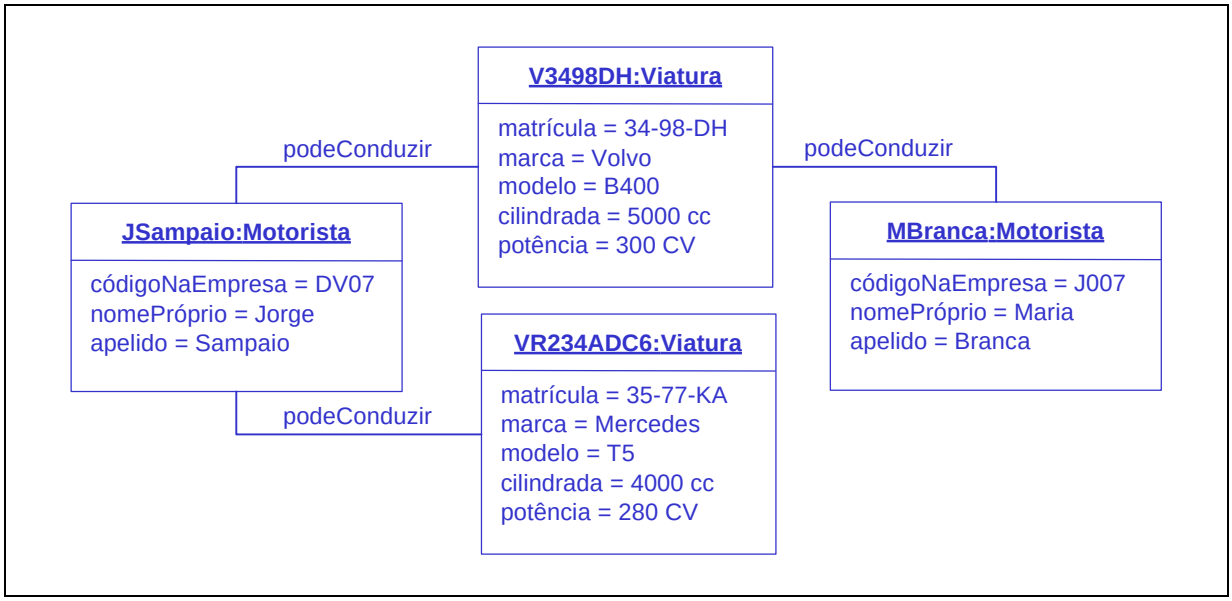


Figura 3.11 Ligações podeConduzir entre objectos das classes Motorista e Viatura.

Associações entre Classes

Uma **associação** entre duas ou mais classes corresponde à caracterização e tipificação de ligações semelhantes entre objectos dessas classes. Assim como um objecto é uma instância de uma classe, uma ligação é uma instância de uma associação. As Figuras Figura 3.12 e Figura 3.13 apresentam associações entre classes. As ligações e associações são representadas por linhas unindo os objectos ou classes. No caso das associações representa-se também a multiplicidade através da notação da Figura 3.14. Não havendo ambiguidade ou dúvidas sobre o nome da associação pode-se omitir, embora seja boa prática referi-lo. É importante apresentar os nomes de todas as associações de multiplicidade [N]x[N] visto que mais tarde no processo de concepção e implementação essa associação poderá vir a ser transformada numa classe ou relação independente. Tal acontece por exemplo se se utilizar na concepção do sistema o CASE da ORACLE, ou na sua implementação um SGBDr.

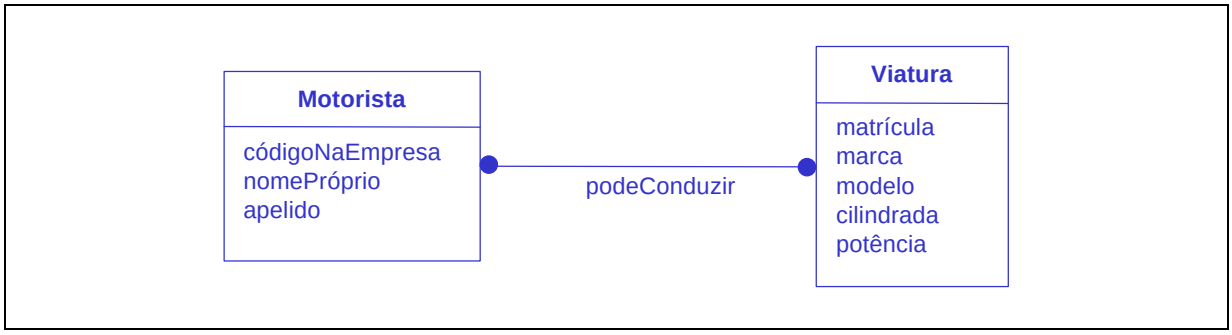


Figura 3.12 Associação podeConduzir correspondente às ligações da Figura 3.11.

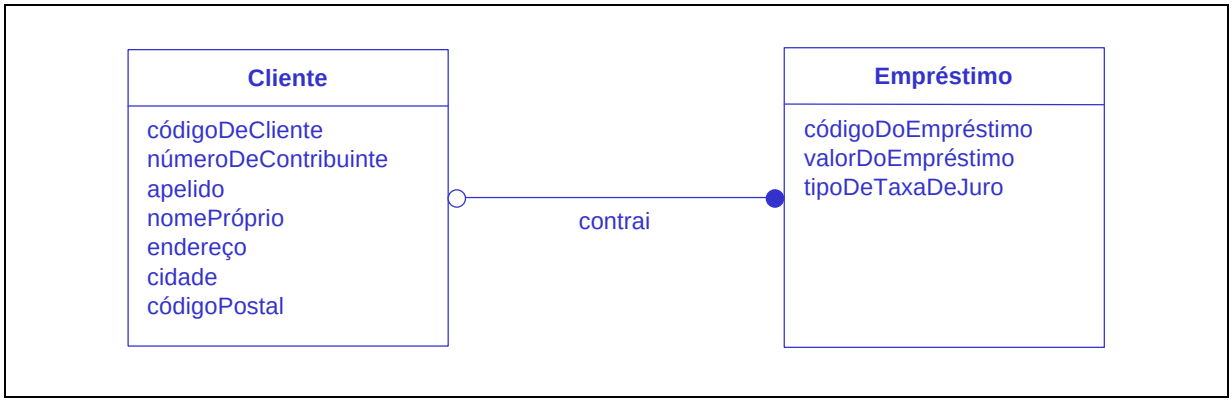


Figura 3.13 Associação Contrai: um Cliente contrai um Empréstimo numa instituição de crédito.

Note-se que, numa associação binária aB, de multiplicidade [1] x [N], entre classes A e B, cada objecto da classe B aceita apenas uma ligação aB, mas cada objecto da classe A aceita múltiplas ligações desse tipo.

Multiplicidade de associações

Uma associação pode ter diversas multiplicidades, conforme os exemplos mostram. Há cinco tipos de multiplicidades representáveis na notação gráfica, conforme indicado na Figura 3.14, com abreviaturas correspondentes e respectiva explicação. Em alguns casos pode ser interessante indicar que a classe do lado da multiplicidade superior a [1] vai ter os objectos ordenados segundo algum critério, devido à existência da associação. Para tal pode-se utilizar a notação referida.

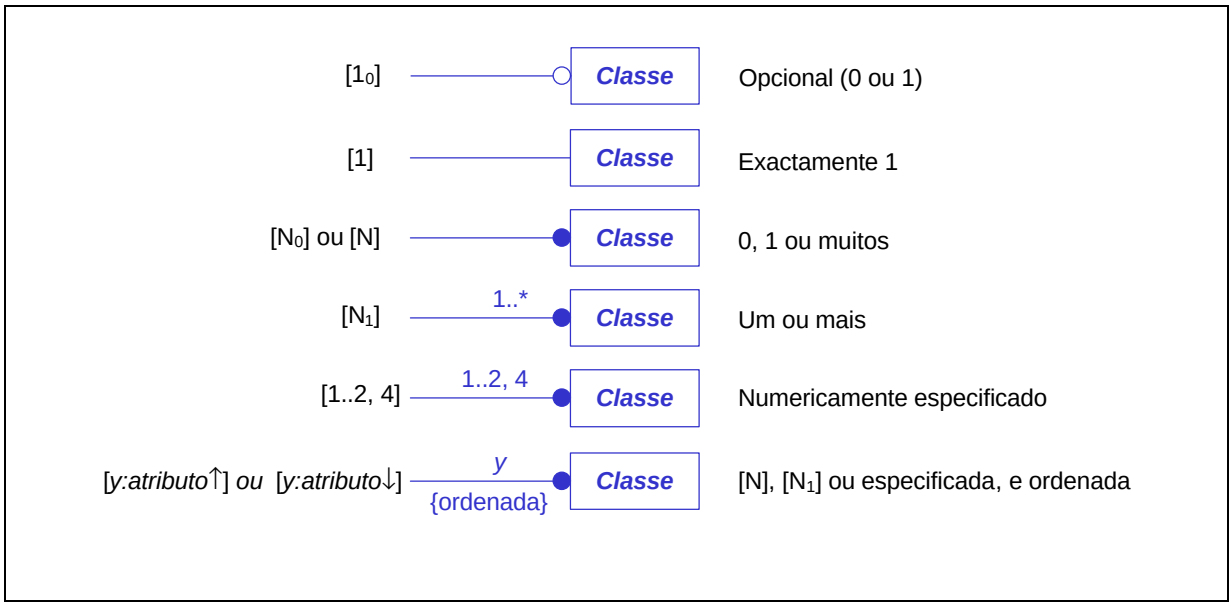


Figura 3.14 Exemplos de multiplicidades de uma associação.

É também normal referir a multiplicidade de associações entre classes utilizando a preposição para, conforme exemplo da Figura 3.15.

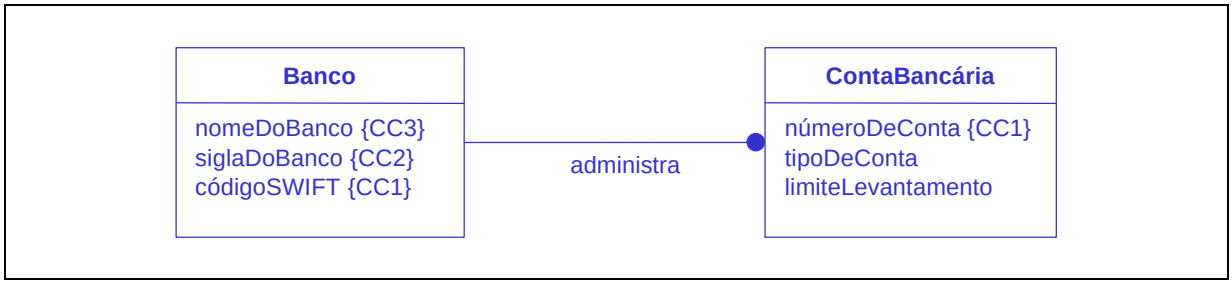


Figura 3.15 Exemplo de associação um-para-muitos, [1] x [N_0] ou simplesmente [1] x [N].

Exemplo de notações EA e UML para associações

As figuras seguintes ilustram a notação E-A [Teorey *et al* 1986] e UML [Erikson & Penker 1998] para o exemplo anterior. Em modelos complexos a notação com origem no OMT [Rumbaugh et al 1991] para as multiplicidades superiores a [1] parece-nos superior à adoptada pelo UML, concordando inteiramente com a posição de [Blaha & Premerlani 1998]. Visualmente o símbolo "∗" é menos perceptível do que o símbolo "●", tornando por esse motivo menos óbvia uma propriedade importante das associações.

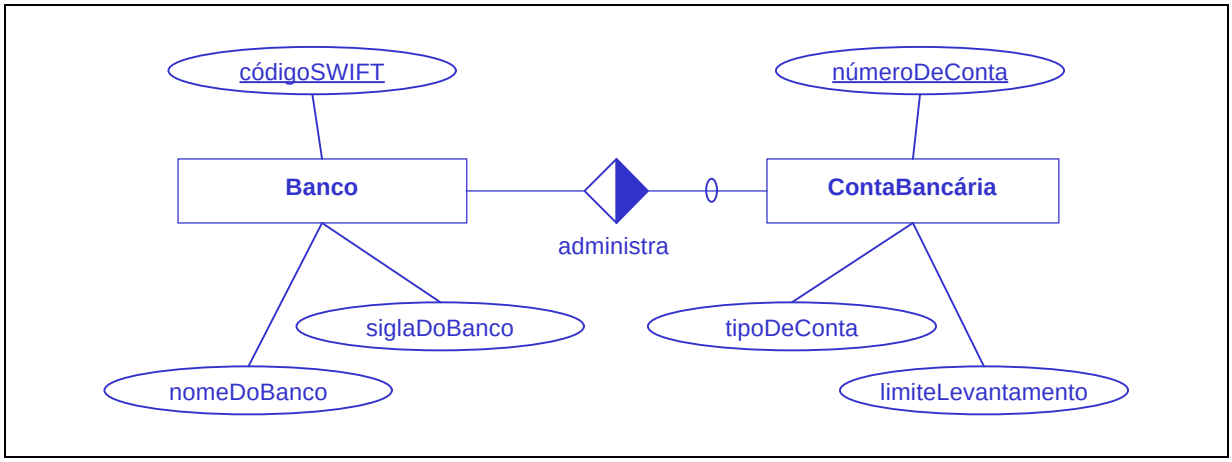


Figura 3.16 Notação E-A relativa ao exemplo anterior.

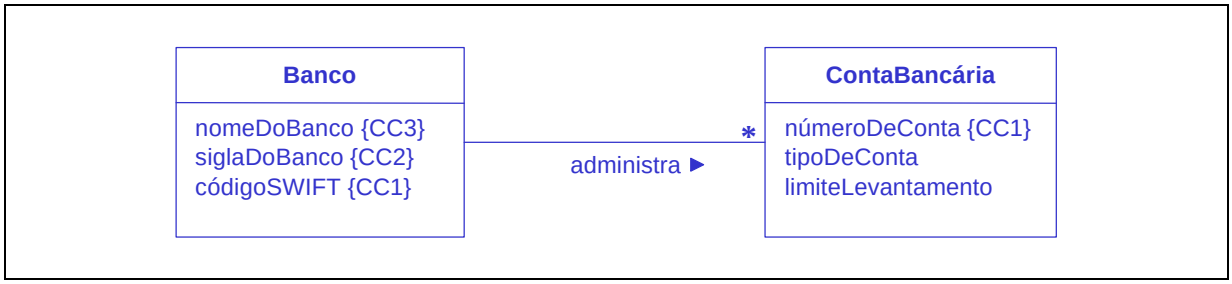


Figura 3.17 Notação UML relativa ao exemplo anterior. O símbolo "▶" localizado junto ao nome da associação administra, indica o sentido de leitura para tradução para português, apontando para a classe correspondente ao complemento directo do verbo que lhe deu origem.

Associações de aridade Unária e Ternária

A aridade de uma associação pode ser unária (ou reflexiva), binária e ternária. As associações apresentadas anteriormente eram todas binárias, sendo essas as mais frequentes. Há situações muito raras em que se pode encontrar ou justificar a modelação com uma associação de aridade superior à ternária. No entanto, a maior parte das ferramentas CASE de apoio à modelação conceptual, exige a utilização de associações simples, isto é unárias ou binárias, e de multiplicidade igual ou inferior a um-para-muitos. Desta forma existe uma correspondência directa entre todas as classes de modelação aí definidas e as tabelas de uma base de dados relacional. Estão neste caso por exemplo as ferramentas de modelação da Oracle e os diagramas de relacionamentos do Access.

As duas figuras seguintes apresentam-se exemplos de associações unárias.

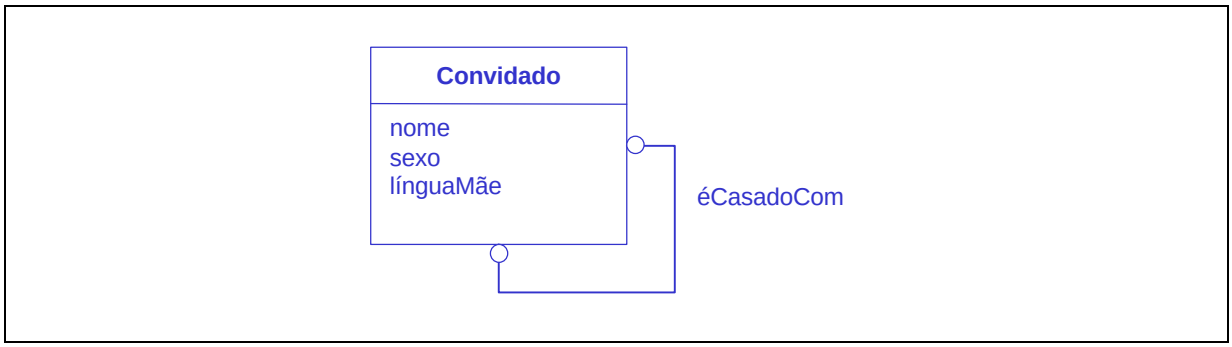


Figura 3.18 Associação unária éCasadoCom; utilizada por um sistema de uma empresa de gestão de recepções para atribuição de lugares nas mesas aos convidados.

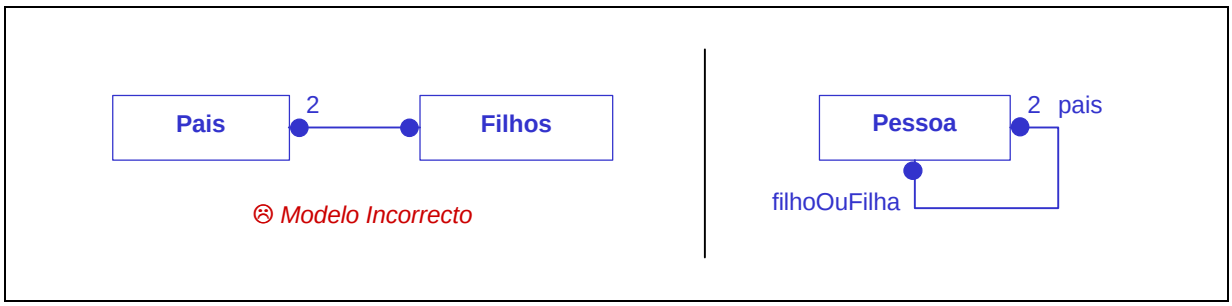


Figura 3.19 A associação unária com os dois papéis indicados é preferível à outra solução representada.

A figura seguinte apresenta um exemplo de uma associação ternária, em que a multiplicidade é representada do diagrama de duas formas alternativas: graficamente através de um triângulo com triângulos menores preenchidos ou não, ou através da indicação explícita da chave candidata identificadora.

Uma associação ternária pode ter sete chaves candidatas diferentes: cada um dos três atributos identificadores das classes associadas, cada um dos três pares diferentes desses atributos e finalmente, como se mostra na figura, a tripla de atributos. Com base nestas sete possibilidades básicas de chaves candidatas, há ainda que escolher para cada caso particular em estudo, quais são todas as chaves candidatas. Assim, dada uma associação entre três classes A, B e C, podemos ter uma situação com duas chaves candidatas, por exemplo os

pares (#a, #b) e (#b, #c). Obviamente que nem todas as combinações de chaves candidatas são admissíveis, pela própria definição de chave candidata. Por exemplo, não é possível ter como chaves candidatas (#b, #c) e (#a, #b, #c).

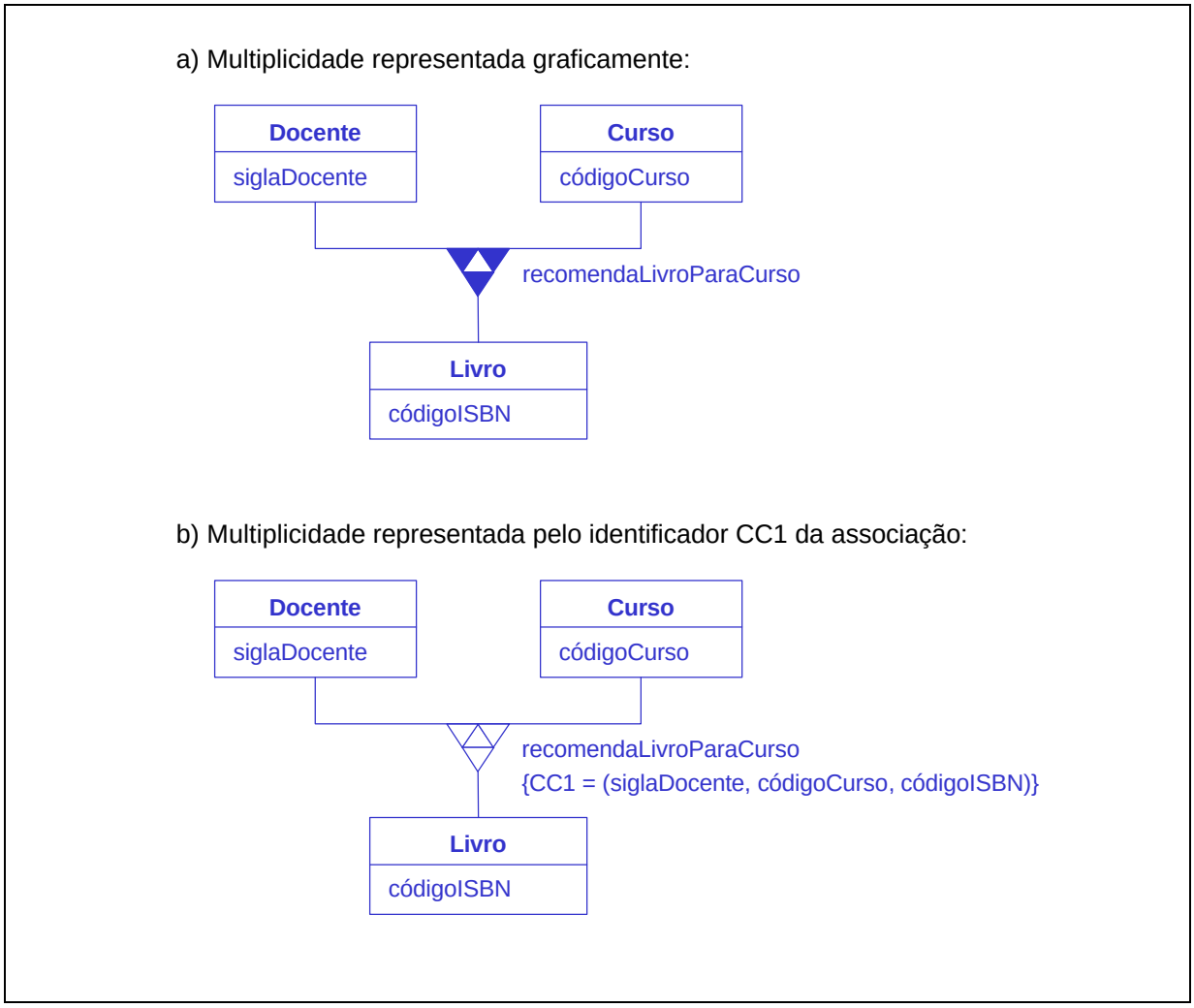


Figura 3.20 Associação ternária recomendaLivroParaCurso; utilizada por um sistema para inquérito aos docentes de uma Universidade para recomendarem livros para todos os cursos de Licenciatura e Mestrado.

Na prática é sempre possível uma alternativa de modelação que diminua a aridade das associações inicialmente identificadas, através da promoção de associações a novas classes, e da ligação destas classes às classes anteriores. Em termos de modelo relacional esta alternativa corresponde à introdução de um novo código interno que codifica o anterior grupo de atributos que formavam a chave primária. A tabela anterior é substituida por uma nova com mais uma coluna com o novo atributo que passa a ser a chave primária. Cada um dos atributos da chave primária inicial mantém agora apenas o estatuto de chave alheia. A estratégia de promoção de associações pode aplicar-se a associações binárias [N] x [N] e deve aplicar-se a associações ternárias, em particular se estas últimas não forem de multiplicidade [N] x [N] x [N] (por exemplo no caso [1] x [N] x [N], pois é normalmente complicado nestes casos identificar o par de chaves candidatas).

Na Tabela 3.3 resumem-se os vários tipos de associações apresentadas anteriormente.

Tabela 3.3 Tipificação e Exemplos de Associações

Multiplicidade	exactamente um	zero ou um	zero, um ou mais	um ou mais	indicada exactamente
Exemplo:	*Figura 3.15*	*Figura 3.18*	-	*Figura 3.15*	*Figura 3.19*

Aridade	unária	binária	ternária
Exemplo:	*Figura 3.19*	*(em várias figuras)*	*Figura 3.20*

Atributos ou Classes em Ligações

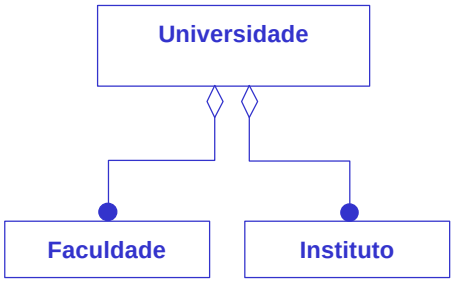
É possível definir atributos de ligações, em particular no caso de associações muitos-para-muitos, onde o atributo é claramente uma propriedade das ligações (associação). Numa associação um-para-um ou um-para-muitos é sempre possível considerar esse atributo no âmbito de um dos objectos (classe) envolvidos na ligação.

De igual forma, é possível definir classes de ligações, em particular no caso de associações muitos-para-muitos. Esta possibilidade é particularmente importante se a associação puder participar em novas associações. Fica a questão de saber qual é o significado formal destas construções.

Agregação entre classes

A agregação é um tipo especial de associação, utilizada por exemplo para indicar a composição física de um tipo de produto, ou a estrutura de organização de uma entidade, conforme os exemplos seguintes. O conjunto de ligações entre os objectos que a agregação descreve, visto como uma relação matemática, caracteriza-se pelas propriedades de transitividade e anti-simetria.

a) Agregação aberta, indicando algumas classes constituintes:



b) Agregação fechada, com indicação de todas as classes constituintes:

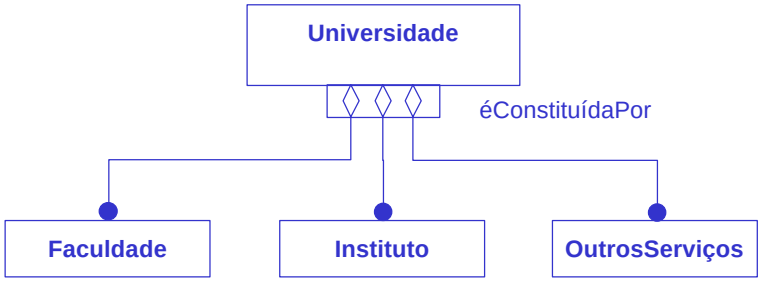


Figura 3.21 Exemplos de agregação aberta e fechada. A notação para a agregação é semelhante à da associação, com a adição de um losango junto à classe composta.

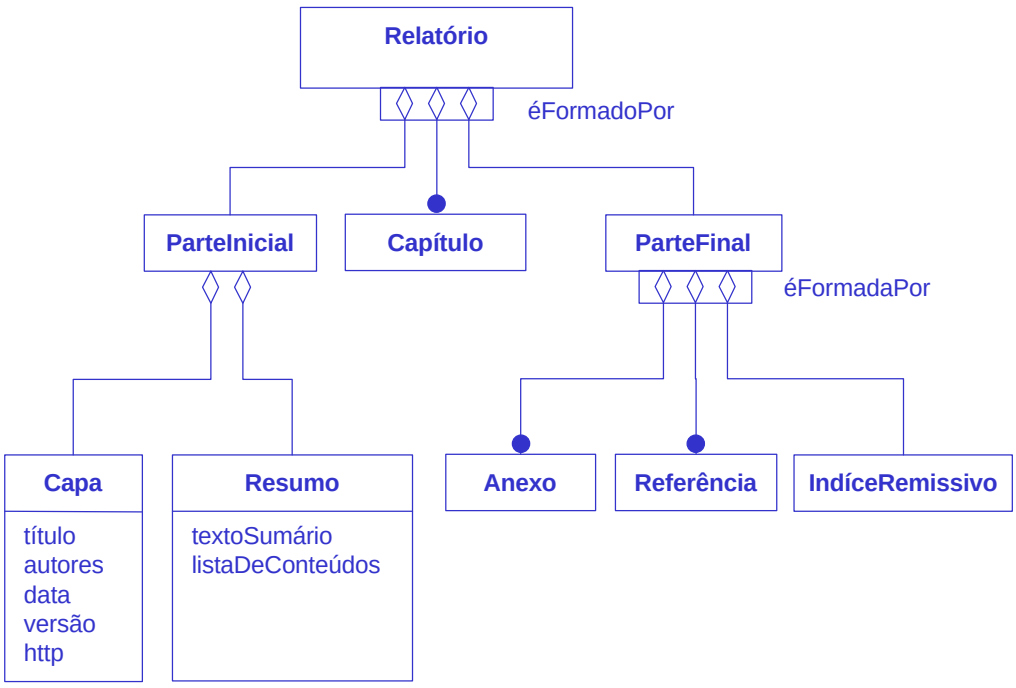


Figura 3.22 Exemplo de agregação a vários níveis de composição.

Frequentemente utiliza-se agregação para especificar a associação entre os produtos e os seus componentes em ambientes de produção envolvendo listas ou árvores de materiais. Esta composição pode ser especificada e mantida ao nível do catálogo dos produtos e componentes, ou pode ser ainda mantida ao nível dos produtos realmente fabricados, conforme a Figura 3.23 ilustra.

Um produto pode ser formado por componentes simples, ou por produtos ainda por sua vez compostos, numa relação transitiva e anti-simétrica. Esta relação pode ser vista como um grafo orientado sem ciclos. O fecho transitivo de um dado produto, visto como nodo do grafo, corresponde à lista de componentes necessários por exemplo para a sua fabricação.

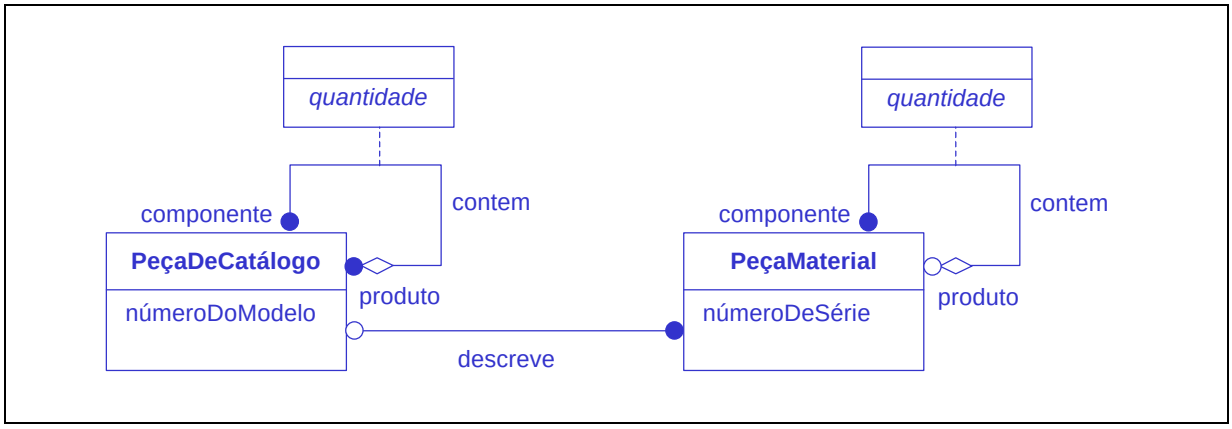


Figura 3.23 Exemplo de agregação de catálogo e de produto: uma peça em catálogo pode pertencer a muitos produtos (exemplo adaptado de [Blaha & Premerlani 1998; 53]).

3.2.3 Outras Associações

Apresentam-se de seguida extensões do conceito de associação, nomeadamente: associações qualificadas, associações com atributos e associações com classes. Estas últimas originam os chamados atributos de uma associação e as classes de uma associação.

Associação Qualificada

Normalmente uma associação qualificada tem origem numa associação de multiplicidade [1]x[N] (ou [N]x[N]) entre uma classe origem A e uma classe alvo B. Numa situação destas, dado um objecto de A não é possível habitualmente identificar um único objecto de B que lhe esteja associado, a menos que seja apresentada uma característica particular de B que nesse contexto o permita identificar.

Obtemos uma associação qualificada se existir um atributo na classe alvo que permita identificar inequivocamente os objectos dessa classe, a partir de um dado objecto da classe origem. O exemplo da Figura 3.24 mostra o caso em que a combinação de uma especificação de um serviço periódico planeado com uma data de realização, permite identificar um serviço particular realizado, ou eventualmente que não foi realizado nenhum serviço. O modelo não qualificado também está correcto, mas é mais geral pois não captura ou especifica esta característica. Em termos de notação, o rectângulo qualificador está junto à classe alvo, apresentando os símbolos de multiplicidade da associação antes e após a qualificação, respectivamente do lado da classe origem e da classe alvo. A Figura 3.26 ilustra outra associação qualificada, mas em que o atributo é agora um candidato a chave identificadora.

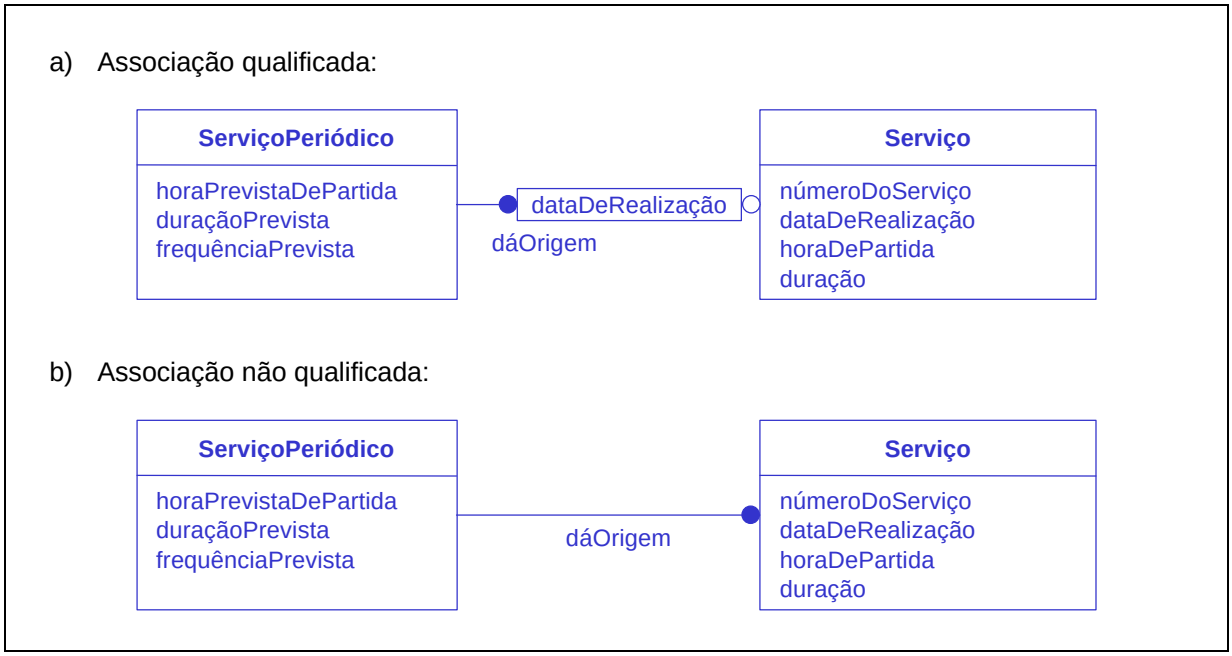


Figura 3.24 Uma associação qualificada aumenta a informação capturada pelo modelo.

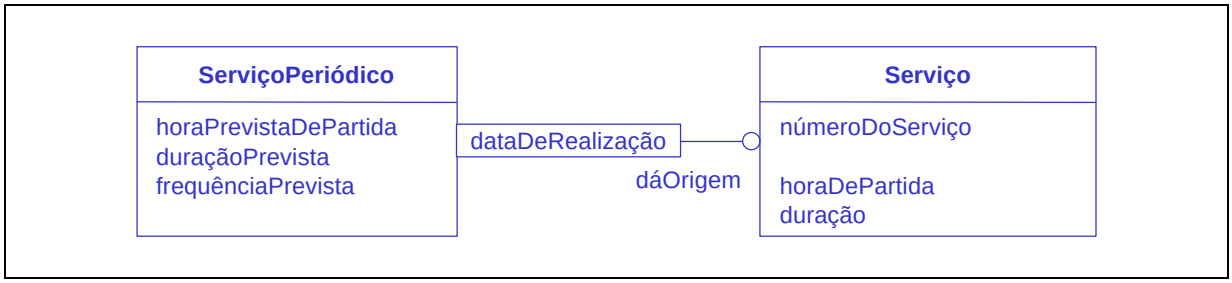


Figura 3.25 Associação qualificada da Figura 3.24 segundo a notação OMT de [Blaha & Premerlani 1998].

Ambas as figuras ilustram a notação aqui proposta, que difere da notação OMT e da notação UML, visto que nestas o atributo não permanece na classe, o qualificador é localizado graficamente junto da classe origem, e mantém-se apenas a indicação da multiplicidade após qualificação. Julgamos que as características da nova notação são importantes, em particular se os gráficos forem utilizados mais tarde para apoiar o processo de conversão para o modelo relacional, particularmente em modelos complexos.

Julgamos que o facto de o atributo qualificador dataDeRealização vir a definir a relação Serviço, correspondente à tradução para o modelo relacional da classe Serviço, não aconselha a que graficamente este atributo seja colocado exactamente junto da outra classe da associação. Por outro lado a manutenção do símbolo de multiplicidade [N] recorda o facto de continuar a ser preferível acrescentar à relação Serviço um identificador da classe ServiçoPeriódico, como chave alheia, devido à multiplicidade [1]x[N] da associação sem qualificação. Na notação aqui proposta o nome do qualificador da associação repete-se também como nome do atributo da classe alvo da qualificação. Podia-se ter evitado esta repetição, mas julgamos que facilita o processo de conversão para o modelo relacional e permite alguma forma de verificação da possibilidade de qualificação.

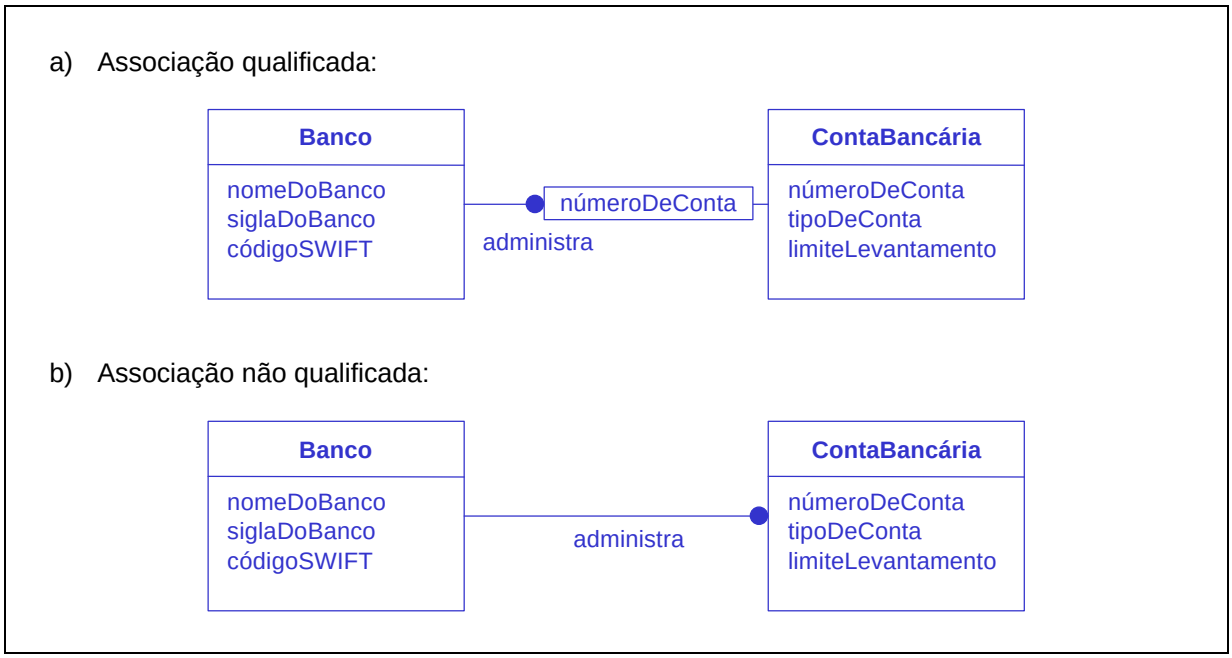


Figura 3.26 Exemplo de associação qualificada e da mesma associação não qualificada.

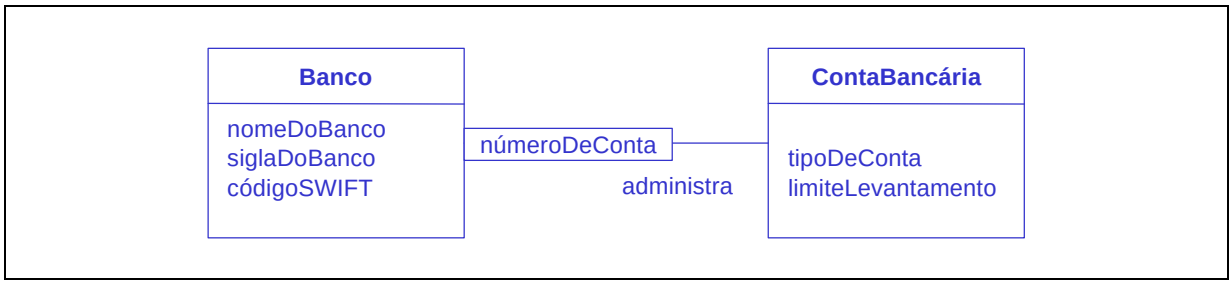


Figura 3.27 Associação qualificada da Figura 3.26 segundo a notação OMT de [Blaha & Premerlani 1998].

Associação com atributo ou com classe

Atributo de associação (associação com um ou mais atributos) e classe de associação (associação com uma classe).

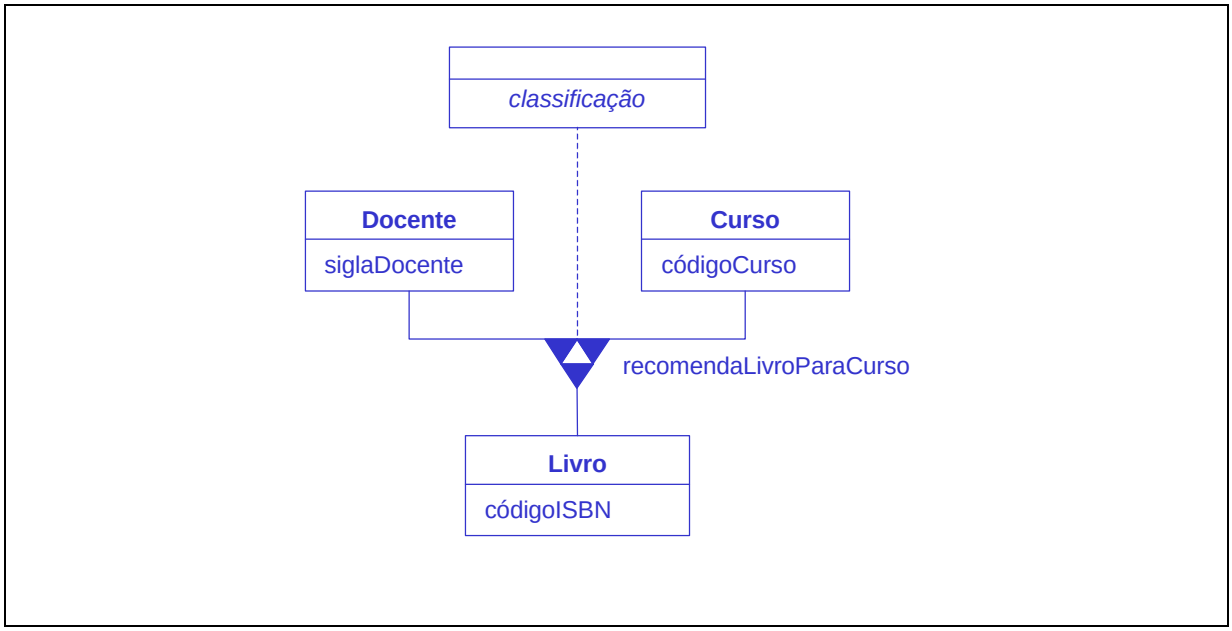


Figura 3.28 Atributo na associação ternária recomendaLivroParaCurso da Figura 3.20; o atributo permite aos docentes classificarem os livros de acordo com a importância que lhes atribuem para cada curso.

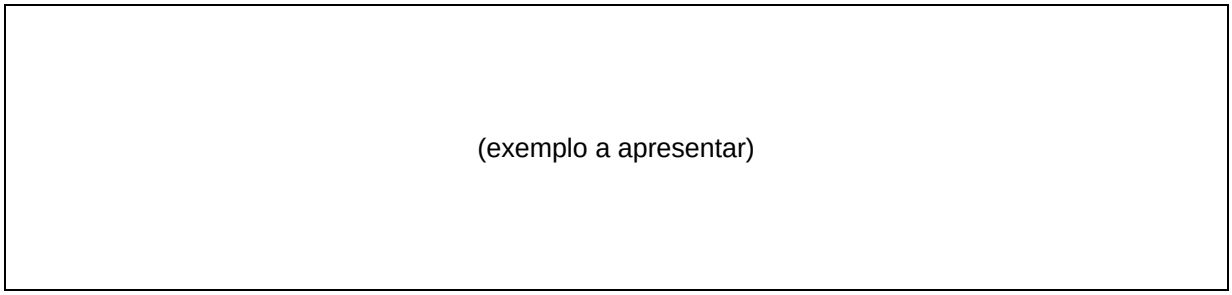


Figura 3.29 Classe de Associação

Na tabela seguinte resumem-se as extensões ao conceito de associação apresentadas anteriormente.

Tabela 3.4 Extensões ao conceito de Associação

Extensões	associação qualificada	atributo de associação	classe de associação
Exemplo:	*Figura 3.24 e Figura 3.26*	*Figura 3.28*	*Figura 3.29*

3.2.4 Generalização e Herança

As noções de associação e generalização capturam as duas formas mais tradicionais de análise e especificação de sistemas, sejam eles informáticos ou não: a associação indica-nos a organização do sistema através da sua decomposição em partes, a generalização permite-nos identificar semelhanças entre sistemas ou seus componentes através da uma **decomposição em tipos**. Por exemplo, podemos analisar as viaturas utilizadas por uma empresa nos seus diversos tipos, por exemplo pesados e ligeiros, dado que um ligeiro é uma viatura. Podemos

também alargar esta análise a níveis mais gerais: uma viatura é um bem móvel, e um bem móvel é um bem patrimonial. Temos assim uma hierarquia, do tipo das hierarquias muito utilizadas para classificar espécies em Botânica, Zoologia ou Geologia. Como é óbvio a noção de particularização e de generalização correspondem a sentidos opostos de navegação nestas hierarquias. A utilidade de qualquer hierarquia esgota-se na sua capacidade de resumir ou explicar conhecimento. Sendo assim aplicam-se as regras normais, pelo que não é natural permitir por exemplo menos do que 3 ou mais do que 7 particularizações distintas para cada classe de objectos, num dado nível.

Uma parte significativa do esforço científico é colocado na identificação de padrões comuns aos objectos em análise, com vista a compreender e explicar as diferenças, desde a Física à Psicologia. Esta tarefa permite economias de conhecimento, uma vez que regras simples e compactas, embora ricas em conhecimento, passam a explicar as particularidades. De igual forma, na análise ou síntese de sistemas de informáticos, a identificação de classes gerais permite economias a todos os níveis, desde o espaço em memória ou as linhas de código, até à manutenção e documentação do próprio sistema. Considere-se por exemplo o sistema de informação de uma dada instituição. Havendo necessidade de manter endereços de clientes e fornecedores, ou de outros contactos, e do próprio pessoal da empresa, é mais económico a todos os níveis identificar uma classe genérica que permite gerir toda essa informação. A implementação com base num SGBD, ou o recurso à interface com um produto comercial de gestão de contactos, facilita o funcionamento de todo o sub-sistema de produção de cartas ou de outros tipos de correspondência, uma vez que há apenas um tipo de dados a considerar (uma tabela e funcionalidades associadas, no caso de um SGBDr ou uma classe de interface no caso de um outro produto comercial aberto). Obviamente que o esforço de análise e concepção é superior, e terá que ser justificado se o benefício futuro o justifica, no contexto particular.

A identificação de classes genéricas deve ainda ser orientada por forma a facilitar futuramente todo o processo de reutilização, desde a própria especificação, aos programas ou outras estruturas entretanto construídas. A orientação para a reutilização deve também condicionar o grau de detalhe de cada classe e os níveis de particularização a considerar.

Generalização ou Particularização Exclusiva

A particularização é uma relação estrutural que permite funcionar o mecanismo de herança (uma subclasse herda os atributos, métodos e associações da sua super-classe). Há dois tipos de herança normalmente utilizada: a herança simples e herança múltipla. No primeiro caso, exemplificado na figura seguinte, um objecto está directamente ligado apenas a um outro objecto mais geral através de uma relação de generalização. No segundo caso, um objecto pose estar directamente ligado a dois ou mais objectos gerais. Este último tipo de modelo pode criar alguns problemas, e não é normalmente aconselhado.

A particularização apresentada na figura seguinte diz-se **exclusiva**, pois não se admite a possibilidade de por exemplo uma viatura pesada ser também uma viatura ligeira.

Há alguns problemas de implementação, dependentes do sistema a utilizar. Por exemplo na conversão para o modelo relacional, prévia à utilização de uma SGBDr, há que ter em consideração a quantidade de atributos e operações particulares de cada classe antes de decidir se se deve manter uma relação e correspondente tabela particular, ou se pelo contrário a

relação ou tabela geral deve ser alargada para incluir os atributos particulares. A decisão depende obviamente dos volumes de informação a gerir, do tipo de operações a solicitar ao sistema e dos tempos de resposta que os utilizadores podem esperar.

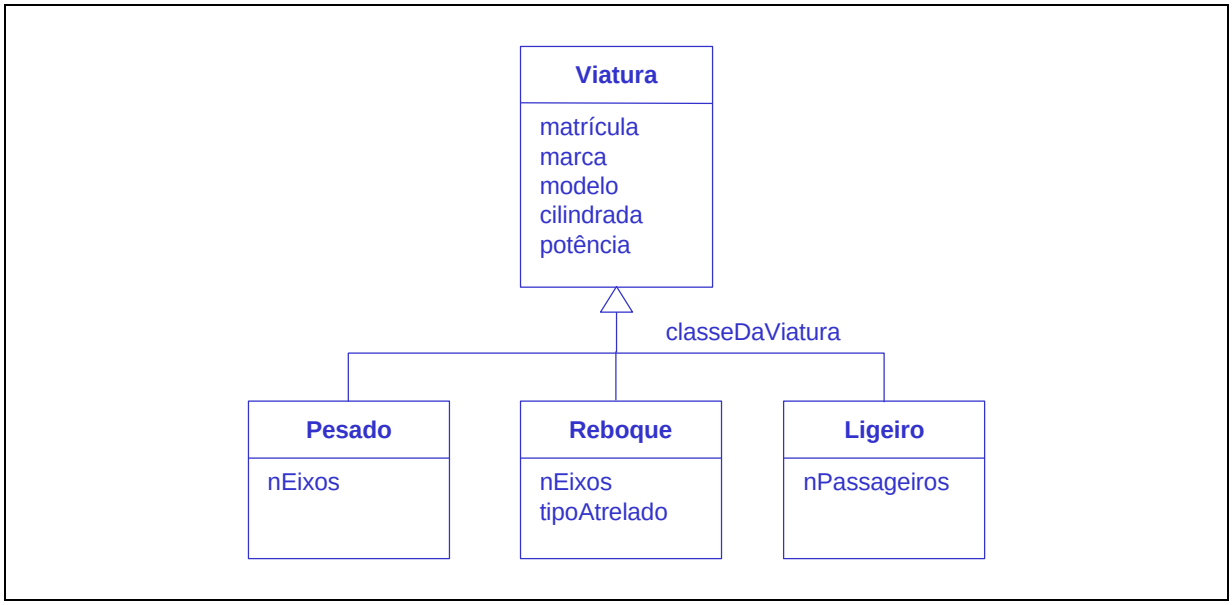


Figura 3.30 Exemplo de Generalização na classe Viatura da informação comum às várias classes de viaturas utilizadas pela empresa OnTime: Pesados, Reboques e Ligeiros.

A escolha de um modelo de generalização com classes particulares tem sempre que ser feita no contexto de um problema. A escolha de classes particulares pode não ser apropriada, sendo possível ser suficiente a utilização de um atributo na classe geral para distinguir os vários tipos de objectos. No caso da figura anterior, só se justificam as classes particulares devido à vontade ou necessidade de as caracterizar com mais pormenores, no exemplo, o número de eixos para pesados e reboques, e o número de passageiros para ligeiros. A existência de operações particulares seria outro motivo para modelar as classes particulares, de forma distinta da geral.

Classes Concretas e Abstractas

A distinção entre classes concretas e abstractas tem a ver com a existência de ou não de objectos, instâncias da classe. Uma classe abstracta não tem instâncias. Uma hierarquia de particularização tem necessariamente de terminar em classes concretas, podendo as classes gerais ser abstractas ou concretas. Pode ser necessário indicar o nome e tipo de uma operação na classe abstracta, mas diferir para as classes particulares a sua implementação concreta. Dizemos então que estamos perante uma operação abstracta com implementação diferida.

Particularização Inclusiva

A particularização apresentada na figura seguinte diz-se **inclusiva**, pois por exemplo admite-se a possibilidade de um estudante ser simultaneamente estudante trabalhador, estudante dirigente associativo, estudante dirigente sindical ou estudante militar. A opção de utilizar um modelo de generalização só deve ser tomado no caso de as classes particulares terem existência por motivos próprios, por exemplo com atributos ou operações particulares

distintas das de estudante. De outra forma seria talvez mais adequado utilizar atributos booleanos na classe estudante, ou uma associação com uma outra classe.

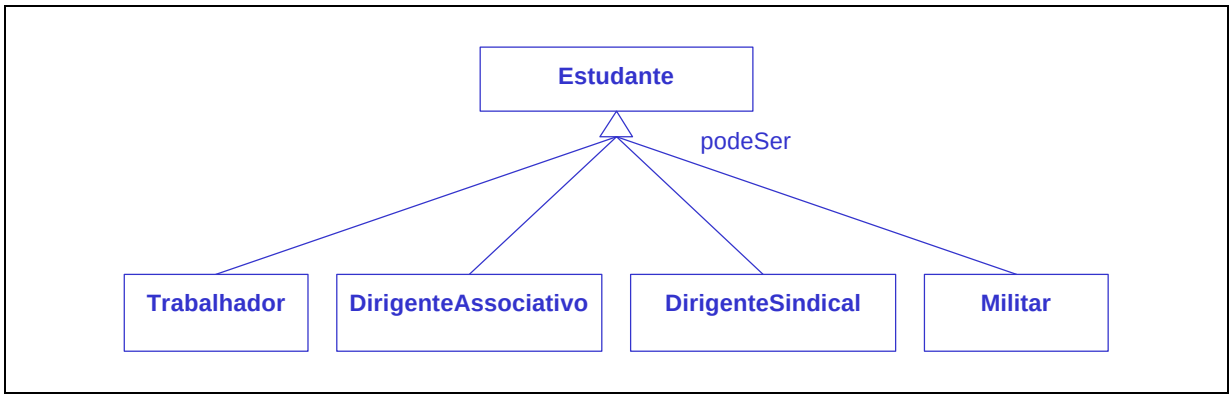


Figura 3.31 Exemplo de Generalização Inclusiva na classe estudante das classes trabalhador, dirigente associativo, dirigente sindical ou militar.

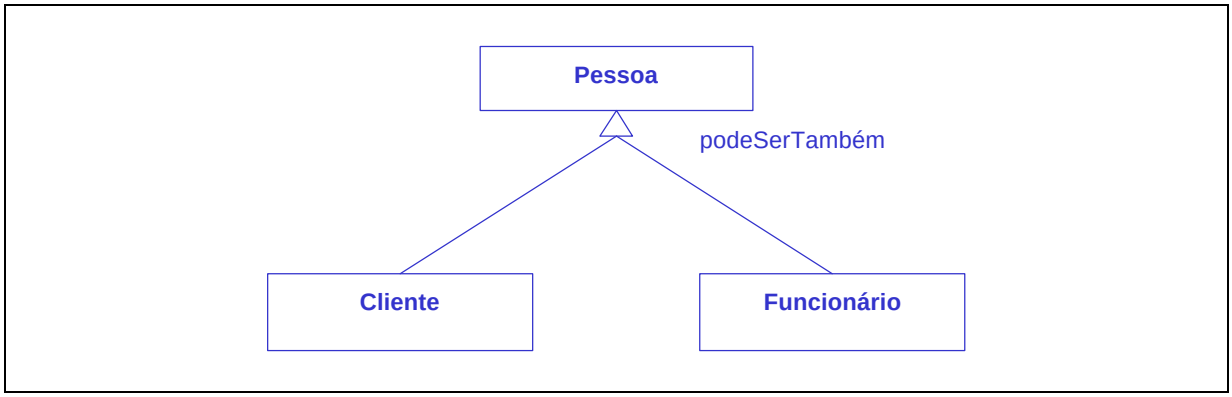


Figura 3.32 Exemplo de Generalização Inclusiva na classe pessoa das classes cliente e funcionário. Enquanto pessoas, os funcionários de uma empresa podem também ser seus clientes.

3.2.5 Exemplo de Modelos, sugestões de visualização e outros aspectos

Horizonte da classe; organização de grandes diagramas; impacto visual; padrões de concepção (design patterns) (experiência)

Como podemos ver nos exemplos anteriores, repetem-se em muitos casos classes semelhantes em termos de atributos e operações. Por exemplo, as classes Pessoa, Motorista e Cliente necessitam normalmente os mesmos atributos. Seria desejável existir uma classe genérica que ...

3.3 Regras para a Construção do Modelo de Classes

Como foi referido brevemente, … Com base em todas as fontes de conhecimento de [Blaha & Premerlani 1998; 127]).:

Abordagem do problema: levantar questões, estudar relatórios interfaces gráficas, brainstorming … como vai ser o futuro, que recursos tenho disponíveis ….

Identificar Classes (nomes próprios e comuns)

1. Identificar Associações (verbos transitivos e preposições)
2. Acrescentar Atributos detalhando as Classes e Associações (possessivos)
3. Utilizar Generalizações para caracterizar semelhanças e diferenças entre objectos
4. Testar caminhos de acesso e pesquisa
5. Iterar e refinar o modelo, acrescentando, eliminando ou alterando o detalhe ou nível de abstracção
6. Organizar graficamente o modelo final

Finalmente: traduzir para o modelo relacional

3.4 Conversão do Modelo de Classes para o Modelo Relacional

Embora não seja do âmbito deste capítulo a apresentação detalhada do modelo relacional de bases de dados, apresentamos de seguida uma breve introdução às bases de dados relacionais. Esta apresentação servirá para definir as principais regras de derivação da estrutura de tabelas ou de relações a partir de um modelo de classes.

Numa base de dados relacional toda a informação está logicamente organizada num conjunto finito de tabelas bidimensionais. Um Sistema de Gestão de Bases de Dados relacional, ou SGDBr, administra toda essa informação, bem como todas as estruturas que são necessárias para garantir funcionalidades e desempenho adequado. Cada tabela, com um conjunto definido de colunas e um número arbitrário de linhas, pode guardar um único valor em cada campo, ou seja em cada intersecção de uma coluna com uma linha. Cada coluna representa um atributo e cada linha representa os valores dos atributos simples de um objecto.

Qualquer SGBDr tem SQL como linguagem de interface. O SQL tem funções para definição da estrutura das tabelas, tem funções para pesquisa, actualização e eliminação de dados, bem como outras funções de gestão e optimização (como veremos em outro capítulo). Os comandos SQL de manipulação de dados aplicam-se a tabelas e têm como resultado também tabelas.

Um SGBDr de boa qualidade deve também suportar mecanismos para definir e garantir regras de integridade ou restrições simples associadas a tabelas. Em particular deve suportar integridade referencial, isto é, deve garantir automaticamente que os objectos referidos em tabelas através dos seus identificadores, estão definidos na tabela apropriada. Um SGBDr deve assegurar independência lógica e independência física dos dados, por forma a que as aplicações hospedeiras apenas dependam da informação de que necessitam e não da forma lógica ou física como ela está guardada. A independência lógica deve garantir por exemplo que essas aplicações não terão de ser modificadas por alterações da estrutura das tabelas, a menos que essa alteração afecte directamente o tipo de dados utilizado.

O processo de derivação do modelo relacional é por si só um processo complexo, exigindo um bom conhecimento de todos os aspectos do problema e de todos os casos de utilização. Para sistemas que necessitem de mais do que cinco tabelas, essa compreensão é facilitada pela construção de um bom modelo de classes, que identifique todos os pedidos e solicitações dos utilizadores. Para além dos aspectos funcionais de utilização, o sistema deve ainda atingir um desempenho adequado em todas as situações. Para tal contribui um modelo relacional que permita a definição de um conjunto apropriado de índices de acesso a dados. A afinação

destes índices e dos comandos SQL é de fundamental importância para se conseguirem atingir bons níveis de desempenho do sistema, sobretudo em condições reais complexas com grandes volumes de dados.

As regras de derivação do modelo relacional a partir do modelo de classes que se apresentam de seguida visam essencialmente obter um bom esquema relacional do ponto de vista funcional. Considerando a tecnologia actual, ao nível dos equipamentos e do suporte automático que o SGBDr dá à independência de dados, pode ser necessário alterar este esquema para permitir que níveis de desempenho aceitáveis. De qualquer forma, os requisitos de desempenho (exemplo de requisitos não funcionais) devem quase sempre ser considerados após os requisitos capturados num modelo de classes (requisitos funcionais).

3.4.1 Conversão de Classes

Numa implementação através de um SGBDr, cada classe dá origem a uma tabela, conforme exemplo da figura seguinte. Recomenda-se uma conversão baseada no princípio da *identidade de existência* de objectos. Para tal, cada objecto deverá ter um identificador único na base de dados, identificador este que não deve ter qualquer significado externo, podendo e devendo ser invisível aos utilizadores normais. Assim, a tabela relacional deverá ser definida com uma coluna correspondente a este atributo identificador e ainda com tantas colunas quantos os atributos da classe. Por convenção, o atributo identificador deverá ter o nome da classe precedido por "#" (ler *código*). Este atributo será a chave primária da tabela, identificada graficamente por sublinhado duplo. A chave primária de uma tabela é uma combinação mínima de atributos que identifica unicamente cada linha de valores de uma tabela. No caso de tabelas representando classes de objectos, a chave primária é constituída por um único atributo, correspondendo a uma coluna onde não pode haver valores repetidos. De facto não tem sentido existirem na base de dados dois ou mais objectos diferentes com a mesma identificação interna.

Uma conversão alternativa assenta princípio da *identidade de valores* de objectos. Esta conversão implica que é necessário seleccionar na classe uma das chaves candidatas (conjuntos de atributos da classe que podem identificar unicamente os seus objectos). Esta chave candidata seria então a chave primária da tabela correspondente à conversão, não se utilizando então códigos internos. Para sistemas experimentais e com poucas classes, é indiferente basear a conversão no princípio da identidade de valor ou de existência. No entanto, devido à utilização de referências baseadas em valores de chaves primárias para garantir a implementação das associações, e à frequência de erros que aparecem na introdução de dados nos sistemas reais, não é apropriado normalmente recorrer a conversão com chaves candidatas.

a) Classe (sem operações):

b) Tabela:

Motorista			
#motorista	*códigoNaEmpresa*	*nomePróprio*	*apelido*
1	DV07	Jorge	Sampaio
3	J007	Maria	Branca
4	P235	José	Lami
5	Y666	Ana	Chaves

Figura 3.33 Exemplo de implementação relacional de uma Classe

Não é possível converter directamente as operações para o modelo relacional, tendo a sua implementação de ser efectuada utilizando macros ou programação numa linguagem apropriada. No caso de utililização de modelos de SGBD orientados por objectos há possibilidade de um melhor suporte à implementação dessas operações de uma forma intergrada com os dados.

3.4.2 Conversão de Associações e Agregações

A implementação de uma associação entre duas ou mais classes num SGBDr pode ou não necessitar de uma tabela independente das tabelas correspondentes a essas classes. De seguida apresentam-se regras de conversão para os principais tipos de associações utilizadas, havendo normalmente várias opções de implementação, umas mais recomendadas do que outras. As regras de conversão para agregações são idênticas às das associações com a mesma multiplicidade, pelo que não se apresentam exemplos.

Conversão de associações um-para-um

Atendendo à regra de implementação de uma classe A (conseguida através de uma tabela com uma chave primária constituída por um único atributo #a, sem valores repetidos), é possível numa linha dessa tabela representar a ligação de um objecto dessa classe A com um objecto de uma outra classe B, através de um valor da chave primária desse outro objecto (cf. Figura 3.34).

a) Classes e associação binária um-para-um:

Classe A Classe B

b) Tabelas correspondentes (uma das duas opções recomendadas):

ContratoDeServiço

#contratoDeServiço	*tipoDeTarifa*	*#especificaçãoDeServiçoIncluída*
1	Alfa	1
2	Beta	5
4	Beta	4
8	Gama	3
…		

EspecificaçãoDeServiço

#especificaçãoDeServiço	*horaPrevistaDePartida*	*duraçãoPrevista*
1	10:00	2:44
3	15:00	3:23
4	10:00	5:44
5	23:23	3:30
…		

c) Opção de conversão não recomendada:

ContratoDeServiço

#contratoDeServiço	*tipoDeTarifa*	*#especificaçãoDeServiço*	*horaPrevistaDePartida*	*duraçãoPrevista*
1	Alfa	1	10:00	2:44
2	Beta	5	23:23	3:30
4	Beta	4	10:00	5:44
8	Gama	3	15:00	3:23
…				

☹ *Não Recomendado*

Figura 3.34 Associação [1] x [1]: exemplos de implementação relacional.

O atributo #especificaçãoDeServiço, na tabela ContratoDeServiço, designa-se como **chave alheia**, uma vez que toma valores de uma chave primária de uma outra tabela na base de dados. Normalmente é desejável que a base de dados seja integra referencialemente, pelo que esta deve ter mecanismos para não aceitar que sejam introduzidos valores em chaves alheias que não existam nas chaves primárias da tabela correspondente à definição da classe de objectos referidos. Os conceitos de chave alheia e de chave primária serão explicados em maior detalhe no capítulo sobre modelação e projecto de bases de dados relacionais.

Conversão de associações um-para-muitos

Na tabela correspondente a uma dada classe B, é também possível representar as ligações de vários objectos distintos dessa classe a um mesmo objecto (ou a objectos distintos) de uma outra classe A. Sendo assim, no caso de associações binárias [1] x [N], entre classes A e B, podemos efectuar a conversão recorrendo apenas à introdução na tabela B de uma coluna com um atributo do mesmo tipo de #a, identificador dos objectos da classe A (cf. Figura 3.35). Este atributo também é uma chave alheia.

Um processo de conversão idêntico se pode aplicar ao caso de associações unárias de multiplicidade [1] x [N]. As regras de tal processo são deixadas como exercício para o leitor.

a) Classes e associação binária um-para-muitos:

b) Tabelas:

Tabela A

Banco			
#banco	*nomeDoBanco*	*siglaDoBanco*	*códigoSWIFT*
22	Banco Português do Atlântico	Atlântico	334455
33	Banco Comercial Português	BCP	778907
34	Caixa Geral de Depósitos	CGD	333333

Tabela B

ContaBancária				
#contaBancária	*númeroDeConta*	*tipoDeConta*	*limiteLevantamento*	*#bancoQueAdministra*
910988876	382-98989-210	O	200	34
910988878	12120009-32	O	100	34
910988880	AA23347800	O	50	22
910988881	334-00899-230	H		34

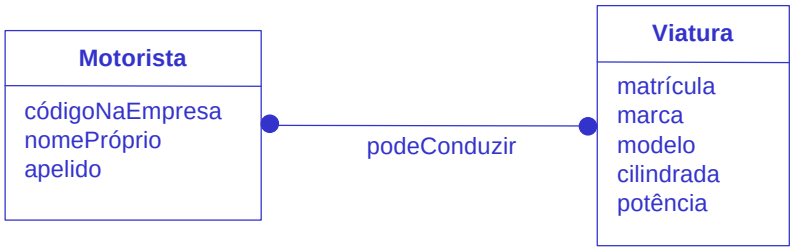
Figura 3.35 Associação [1] x [N_0]: exemplo de implementação relacional.

Conversão de associações muitos-para-muitos

No caso de associações de multiplicidade do tipo [N] x [N], entre classes A e B, teremos em geral de criar uma tabela independente das tabelas correspondentes a cada uma dessas classes, (cf. exemplo da Figura 3.36). Esta nova tabela deverá ter como chave primária o par de atributos (#a,#b). Cada um destes atributos será também uma chave alheia, uma vez que cada um deles está relacionado com a chave primária de outras tabelas (correspondentes respectivamente às classes de obejctos A e B).

Da forma idêntica para associações ternárias [N] x [N] x [N].

a) Classes e associação binária muitos-para-muitos:



b) Tabelas:

Motorista			
#motorista	*códigoNaEmpresa*	*nomePróprio*	*apelido*
…			

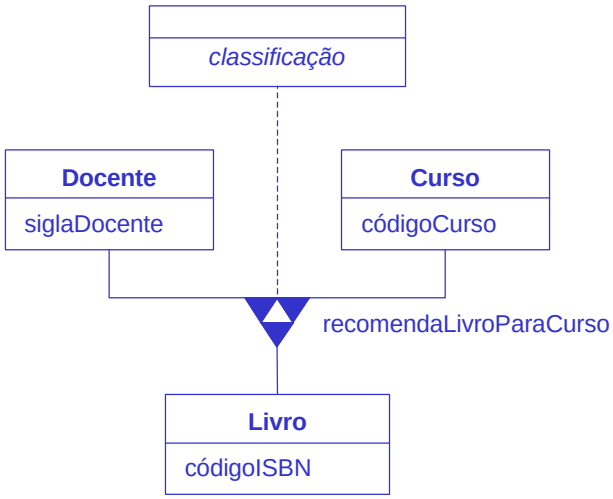
Motorista_podeConduzir_Viatura	
#motorista	#viatura
…	

Viatura					
#viatura	*matrícula*	*marca*	*modelo*	*cilindrada*	*potência*
…					

Figura 3.36 Associação [N_0] x [N_0]: exemplo de implementação relacional.

Conversão de associações ternárias

a) Classes, associação ternária e atributo de ligação:



b) Tabelas:

Docente

#docente	*siglaDocente*
…	

Curso

#curso	*códigoCurso*
…	

Livro

#livro	*códigoISBN*
…	

RecomendaLivroParaCurso

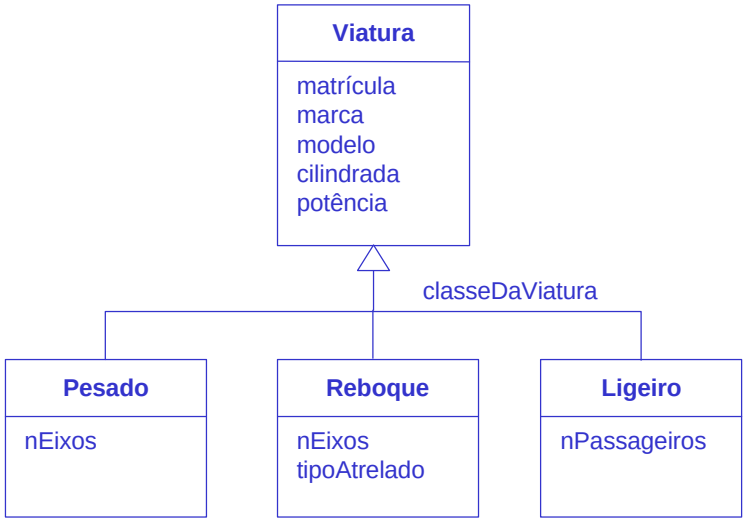
#docente	*#curso*	*#livro*	*classificação*
7	33	1	18
7	33	2	15
7	33	3	7
7	22	1	7
7	22	3	20
9	33	1	15
9	33	3	12
…			

Figura 3.37 Associação [N]x[N]x[N]: exemplo de implementação relacional.

3.4.3 Conversão de Generalizações

Conversão de generalizações exclusivas

a) Classes particulares, generalização exclusiva e classe geral:



b) Tabelas (opção recomendada):

Viatura

#viatura	*matrícula*	*marca*	*modelo*	*cilindrada*	*potência*	*classeDaViatura*
12	12-12-TR	Volvo	Z707	2002	250	Ligeiro
13	13-44-TR	Volvo	Z704	2002	250	Ligeiro
77	22-98-ZR	Volvo	P4	8000	400	Pesado
78	22-99-ZR	Volvo	X700	2002	200	Ligeiro
79	22-12-ZZ	Volvo	P4	8000	500	Pesado
…						

Pesado

#viaturaPesado	*neixos*
77	8
79	8
116	12
…	

Reboque

#viaturaReboque	*neixos*	*tipoAtrelado*
101	8	Articulado
102	8	Articulado
117	12	Longo
…		

Ligeiro

#viaturaLigeiro	*nPassageiros*
12	5
13	4
78	2
…	

c) Tabelas (opção menos recomendada):

Viatura

#viatura	*matrícula*	*marca*	*modelo*	*cilindrada*	*potência*	*classeDaViatura*	*neixos*	*tipoAtrelado*	*nPassageiros*
12	12-12-TR	Volvo	Z707	2002	250	Ligeiro			4
13	13-44-TR	Volvo	Z704	2002	250	Ligeiro			5
77	22-98-ZR	Volvo	P4	8000	400	Pesado	8		
78	22-99-ZR	Volvo	X700	2002	200	Ligeiro			2
79	22-12-ZZ	Volvo	P4	8000	500	Pesado	8		
…									

Figura 3.38 Generalização exclusiva: exemplo de implementação relacional.

Conversão de generalização inclusiva

a) Classes e associação binária muitos-para-muitos:

b) Tabelas:

Estudante	
#estudante	*podeSer*
1	Trabalhador
1	Militar
3	Dirigente Associativo
3	Militar
5	Militar
…	

Trabalhador	
#estudanteTrabalhador	
1	
7	
8	
…	

DirigenteAssociativo	
#estudanteDirigenteAssociativo	
3	
15	
19	
…	

DirigenteSindical	
#estudanteDirigenteSindical	
10	
11	
45	
…	

Militar	
#estudanteMilitar	
1	
3	
5	
…	

Figura 3.39 Generalização inclusiva: exemplo de implementação relacional.

3.5 Conclusões

O processo de concepção inicia-se com a recolha e organização de todo o conhecimento disponível sobre o problema, em particular dando atenção às necessidades dos vários tipos de

utilizadores. É normalmente extremamente importante partir de uma descrição breve e sumária do sistema que se deseja conceber. A partir desta descrição passa-se para a caracterização exaustiva de casos de utilização do sistema. Estes casos podem eventualmente considerar interdependências temporais e interfaces de utilização com outros sistemas.

Deste processo de análise e organização resulta um documento UTIL com a especificação de requisitos dos utilizadores, dirigido precisamente aos utilizadores. Resulta também uma primeira versão do documento SLIE contendo a especificação dos suportes lógicos e das interfaces externas do sistema, documento dirigido aos técnicos que virão a conceber e construir o sistema. Deste documento consta por exemplo a organização da árvore de interacção porposta, com exemplos de maquetes de baixa resolução das interfaces com o utilizador (em particular da janela ou página inicial da aplicação), definições de interfaces com outros sistemas, quer ao nível de dados quer de protocolos, e um dicionário definindo o conjunto de termos específicos da situação em estudo.

Com base nesta informação, e mantendo um contacto regular com os principais interlocutores, inicia-se um processo iterativo de concepção do modelo de classes. Este modelo vai oferecer uma visão estática das estruturas de informação e das principais operações, procedimentos ou funções a realizar sobre esses dados. O modelo de classes inclui assim todos os requisitos funcionais do sistema. O modelo conceptual inclui um conjunto de classes de objectos, estruturadas em hierarquias de associação (análise da composição de objectos, *nas suas partes*) e hierarquias de generalização (análise de tipificação de objectos, *nos seus tipos*).

Finalmente um modelo conceptual terá de dar origem a um sistema funcional. Como geralmente se trata de problemas de gestão de dados, informação e conhecimento de grande dimensão, que se deseja manter persistente, recorre-se a um Sistema de Gestão de Bases de Dados. Para problemas de complexidade pequena ou média, menos de 100 classes, mas com um volume de dados muito elevado, mais de 10^5 objectos, e com vários tipos de utilizadores, deve-se recorrer a um SGBDr. Para problemas com maior complexidade, ainda raros actualmente, pode-se recorrer a sistemas relacionais com extensões para gestão persistente orientados por objectos. Normalmente é necessário transformar o modelo de classes num modelo relacional que seja adequado aos requisitos não funcionais, tais como desempenho desejado e recursos informáticos disponíveis. As regras de transformação indicadas permitem obter uma primeira versão do esquema relacional, que terá de ser posteriormente optimizado, por exemplo, através de uma definição e gestão cuidadosa de indíces ou chaves de indexação. As necessidades de desempenho podem obrigar em alguns casos a uma revisão do modelo conceptual, dado que este servirá posteriormente para documentar o sistema construído e orientar alterações ditadas pela alteração de requisitos informacionais ou funcionais.

3.6 Exercícios

3.1 Considere o conceito Classe de uma Ligação. Qual é o significado implementacional desta construção? E o seu significado denotacional?

3.2 Considere o exemplo da Figura 3.31. Em vez de utilizar uma generalização inclusiva, proponha uma classe que permita caracterizar os tipos de estudantes e defina um modelo alternativo recorrendo à associação da classe estudante com essa outra classe.

3.3 Considere as duas classes funcionário e departamento. Entre estas classes definem-se duas associações que permitem identificar quem trabalha num dado departamento e quem chefia cada departamento. Suponha que qualquer chefe de um dado departamento trabalha também nesse departamento e que um funcionário trabalha apenas num dado departamento. Indique as multiplicidades das associações necessárias. Que restrições existem entre estas duas associações de forma a garantir a integridade da informação? O modelo de classes de objectos permite exprimir esta restrição de integridade? Como exprimia esta restrição em EA, OMT e UML?

3.4 Referiu-se que os valores de um domínio não podem ser obtidos por construções complexas, estando à partida excluídos valores de complexidade semelhante aos objectos das próprias classes definidas no modelo. Se se permitisse que os atributos de uma classe pudessem representar qualquer tipo de dados, que modelos poderiamos representar? Apresente um exemplo de uma situação desta natureza.

3.5 Considere o modelo conceptual de classes para o sistema de informação proposto para a empresa OnTime.

3.5.1 Verifique se é possível calcular o número de quilómetros percorridos por cada viatura entre duas datas.

3.5.2 Verifique se é possível calcular o número de quilómetros que se prevê que cada viatura venha a fazer desde a data actual e uma data futura.

3.5.3 Indique como se pode obter a seguinte listagem no final de cada semana:

Relatório semanal com o total de quilómetros percorridos por cada viatura e com os quilómetros previstos para a próxima semana - 7 dias.

2001-04-06 Segunda-feira

Matrícula	*Marca*	*Quilometragem Actual*	*Quilometragem prevista*
23-24-OF	Volvo	83456	3509
56-33-PF	Ford	156777	8509
56-34-PF	Ford	122562	7512
56-37-PF	Ford	142533	9072

…

3.5.4 Que alteração seria necessário introduzir no modelo do sistema para permitir programar automáticamente a manutenção preventiva de viaturas? Suponha que cada viatura tem uma periodicidade fixa medida em quilómetros percorridos para ser submetida a manutenção períodica.