Capítulo 6

Modelos do Sistema

Neste capítulo vamos fazer uma introdução às abordagens mais comumente utilizadas na análise de requisitos para a elaboração de modelos do sistema. O objetivo desta introdução é desenvolver a habilidade de modelar sistemas utilizando abordagens e ferramentas diversas.

Em outras palavras, neste capítulo veremos como desenvolvedores humanos conseguem conpreender o ambiente à sua volta desenvolvendo modelos (simplificados) deste ambiente e como estes modelos podem ser flexíveis. Para isto vamos:

- Explicar o papel dos **modelos do sistema** no processo de análise de requisitos;
- Mostrar como diferentes tipos de modelos podem apresentar **informações que se complementam** sobre um sistema;
- Descrever diferentes **tipos de modelos do sistema**.

A criação de modelos de um sistema é uma atividade que ajuda o analista a compreender como um sistema funciona. Modelos são baseados em conceitos computacionais como objetos e funções e representam um ponte muito importante entre os processos de análise e projeto do sistema.

A coisa mais importante sobre modelos do sistema é que eles são abstratos, ou seja, eles deliberadamente abstraem (escondem) informações preservando somente aqueles conceitos que caracterizam a estrutura do sistema. Esta abstração permite que o sistema seja visualizado de forma holística sem considerar os detalhes de implementação. Desta forma, os modelos também podem ser utilizados para a comunicação entre clientes e desenvolvedores.

O fundamento das técnicas de análise de requisitos baseadas em métodos é a adequação do sistema á um modelo sistemático. Ou seja, independente do método de analise que se esteja utilizando, a criação de modelos é sempre a base do trabalho de análise de requisitos.

Existem diferentes tipos de modelos do sistema, os tipos mais tradicionais são os modelos de fluxo de dados, entretanto em meados da década de 1980 os modelos de objetos tornaram-se muito mais utilizados. A escola americana de engenharia de software sugere que faça-se a escolha por um único tipo de modelo específico (funcional ou de objetos)

adequado ao sistema. Já a escola européia apregoa que diferentes tipos modelos podem ser complementares e o uso de diferentes modelos elucida diferentes características sobre um mesmo sistema. Alguns dos tipos de modelos de sistema mais genéricos são:

- **Modelo de processamento de dados.** Baseado em diagramas de fluxo de dados que demonstram como os dados são processados em diferentes estágios do sistema.
- **Modelo de composição.** Baseado em diagramas entidade-relacionamento que mostram como as entidades do sistema se relacionam.
- **Modelo de classificação.** Baseado em diagramas de classes de objetos e de herança que demonstram as características em comum entre as entidades.
- **Modelo de estímulo e resposta.** Baseado em diagramas de transição de estado que mostram como o sistema reage a eventos internos e externos ao sistema.
- **Modelo de processos.** Baseado em modelos de processo que descrevem as atividades principais do sistema.

Neste capítulo aprofundaremos nossos estudos sobre os três primeiros tipos de modelos do sistema. O modelo de estímulo e resposta será desenvolvido no capítulo sobre projeto de sistemas de tempo real.

6.1 Modelos de fluxo de dados

Modelos de fluxo de dados compõem um método intuitivo para se mostrar como dados são processados pelo sistema. Este modelo apresenta o processamento do sistema em etapas distintas (processos) e os dados que fluem entre estas etapas. Os dados são transformados em cada etapa antes de serem levados para uma próxima etapa. Estas transformações de dados ou etapas equivalem as funções do sistema que, por sua vez, podem ser atividades desempenhadas tanto por pessoas como por máquinas.

A Figura 6.1 mostra um exemplo de um diagrama de fluxo de dados (DFD) que representa o processamento de um pedido de compra de um determinado equipamento em uma organização. O diagrama mostra como o pedido flui entre os processos (representados por círculos). Também são apresentados depósitos de dados (representados por quadrados) onde os dados são armazenadas. Os fluxos são representados por setas.

O DFD possui uma notação intuitiva que normalmente pode ser explicada aos usuários em potencial do sistema para que estes ajudem na validação do modelo. Ao mesmo tempo que é simples, o DFD é bastante completo pois mostra e documenta todo o processamento de dados realizado por todo sistema. Isto faz com que torne-se possível identificar o que está acontecendo com o dados em cada ponto do sistema.

Diagramas de fluxo de dados são utilizados em diferentes níveis de abstração, desde o nível mais simbólico e abstrato até o nível mais concreto e detalhado. A construção de um modelo de fluxo de dados pode começar com um DFD descrevendo apenas as funções

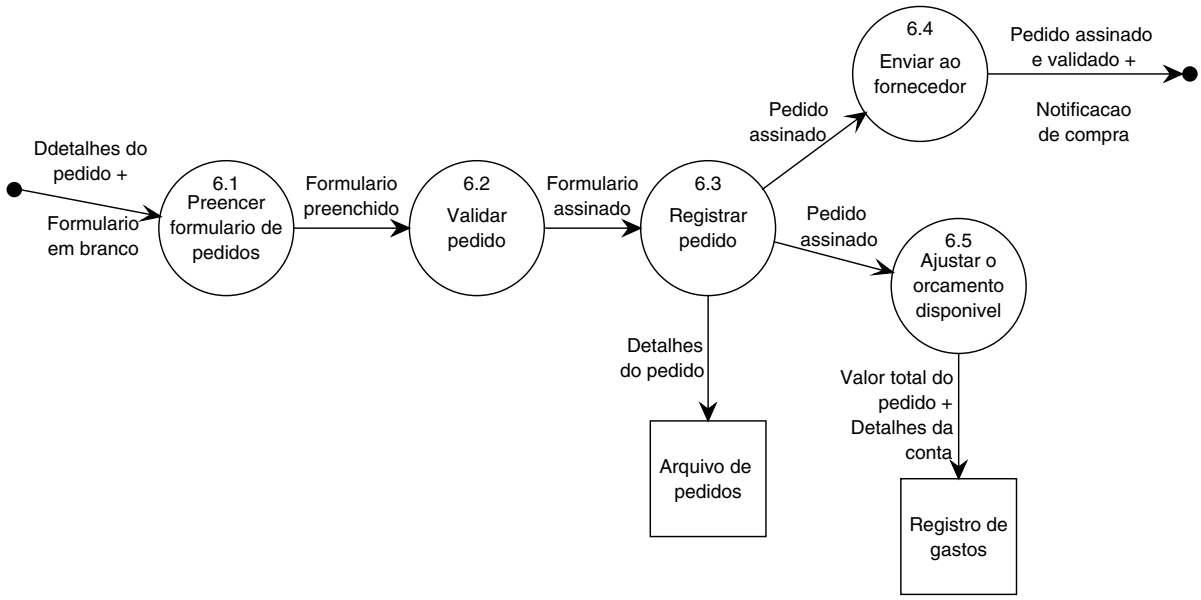


Figura 6.1: Diagrama de fluxo de dados para o processamento de pedidos.

principais do sistema – o DFD de nível zero – seguindo-se então de outros diagramas que descrevem em detalhes cada uma das funções descritas no diagrama inicial. Vai-se então formando uma hierarquia de DFDs que se estende até que todas as funções do sistema sejam descritas. A utilização de numeração junto aos processos é muito útil na identificação da posição de cada diagrama na hierarquia. A Figura 6.2 mostra um modelo de fluxo de dados se estendendo até o nível 2.

Já a Figura 6.3 é uma continuação do exemplo da Figura 6.1. Nesta nova figura vemos um modelo de mais alto nível de um sistema de aquisição de equipamentos de uma organização. Observe que a o DFD da Figura 6.1 é representado na Figura 6.3 como uma única etapa abstrata: o processo seis.

O modelo de fluxo de dados também pode ser utilizado para se fazer a descrição arquitetural mostrando o intercâmbio de dados entre os subsistemas que compõem o sistema. A Figura 6.4 mostra um exemplo de utilização de um DFD para descrever o modelo arquitetural do sistema, no exemplo, a estrutura funcional de uma ferramenta CASE.

Os diagramas de fluxo de dados não são eficiente para descrever subsistemas com interfaces muito complexas. Neste caso, seriam necessários diversos diagramas para descrever cada detalhe da interface e um modelo de objetos poderia ser uma solução mais adequada.

6.2 Modelos semânticos de dados

A maioria dos sistemas de grandes escalas são baseados em um banco de dados que, além de armazenar os dados, descreve a semântica entre as informações que são utilizadas pelo sistema. Assim um modelo que descreve a estrutura lógica dos dados processados pelo

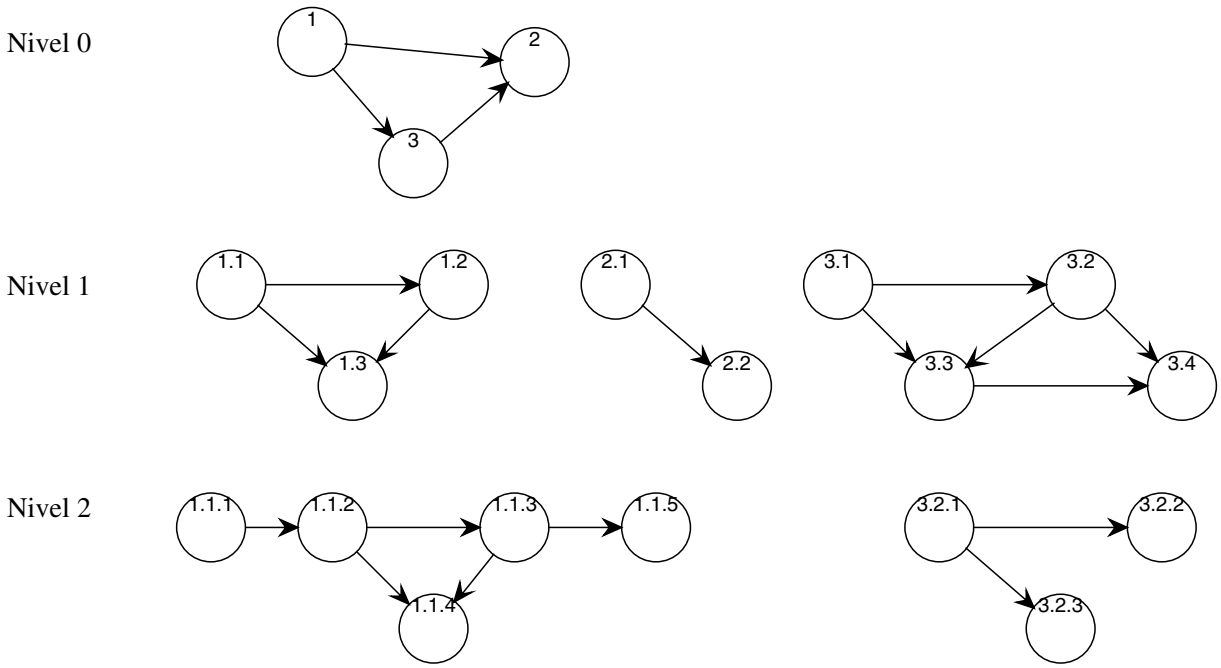


Figura 6.2: Modelo de fluxo de dados multinível.

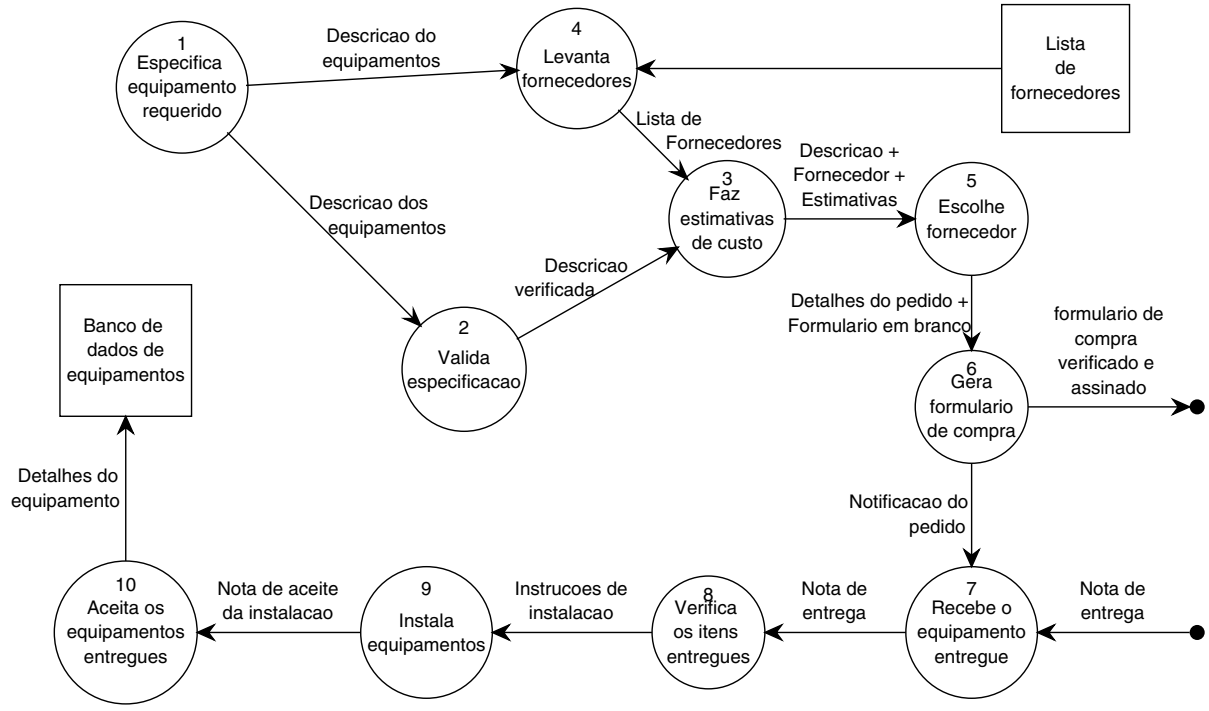


Figura 6.3: DFD do processo de obtenção de equipamentos.

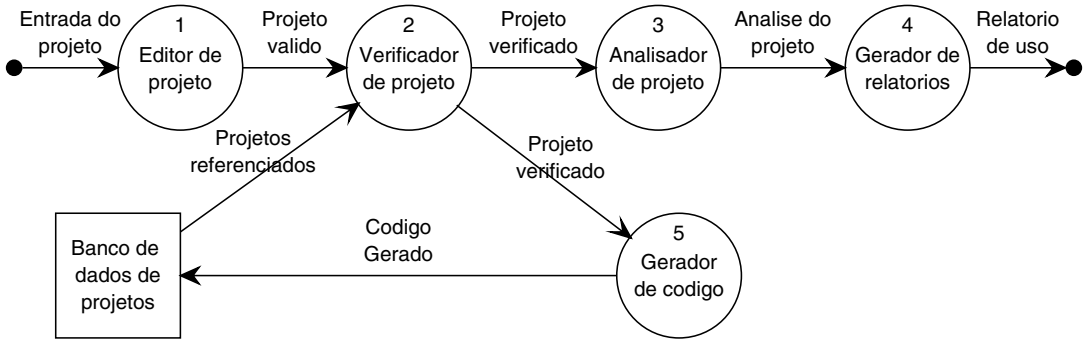


Figura 6.4: DFD descrevendo a arquitetura de uma ferramenta CASE.

sistema se torna importante. Os modelos semânticos se baseiam no modelo entidade-relacionamento que apresenta as entidades do sistema, os relacionamentos entre as entidades e seus atributos.

A Figura 6.5.a apresenta um diagrama entidade-relacionamento típico descrevendo a relação existente entre clientes de um banco e sua conta corrente. Existem diversas notações para este tipo de diagrama. A notação apresentada na Figura 6.5.a segue um padrão mais comumente utilizado por projetistas de bancos de dados, no qual:

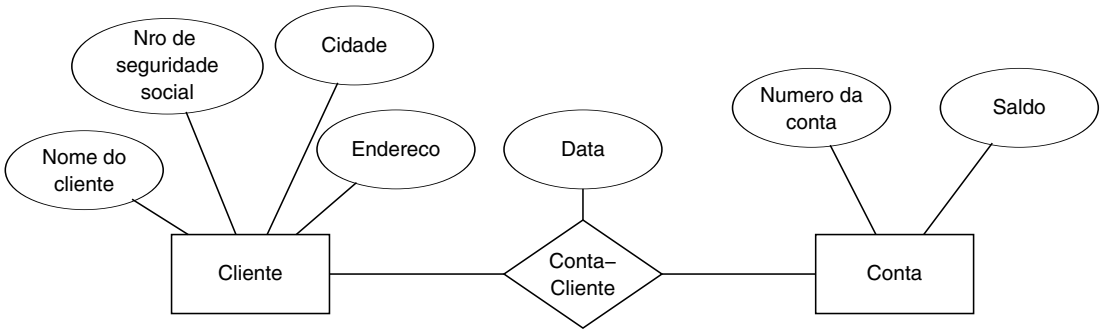
- Uma **entidade** é representada por um retângulo;
- Um **relacionamento** é representado por um losango.
- Um **atributo** de uma entidade ou de um relacionamento é representado por uma elipse.

A Figura 6.5.b demonstra o uso de cardinalidade no diagrama entidade-relacionamento. A cardinalidade diz respeito ao número de instâncias de uma determinada entidade permitidas na relação desta entidade com outra. No exemplo da figura temos um exemplo de cardinalidade do tipo um-para-muitos, isto equivale a dizer que *uma turma possui muitos alunos*. Na Figura 6.5.c temos uma notação mais tradicional para a cardinalidade, que também é bastante recorrente.

Na Figura 6.5.d temos três exemplos de cardinalidade:

- um-para-muitos: *uma turma possui muitos alunos.*
- um-para-um: *um professor ministra um curso.*
- muitos-para-muitos: *muitos professores atendem muitos alunos.*

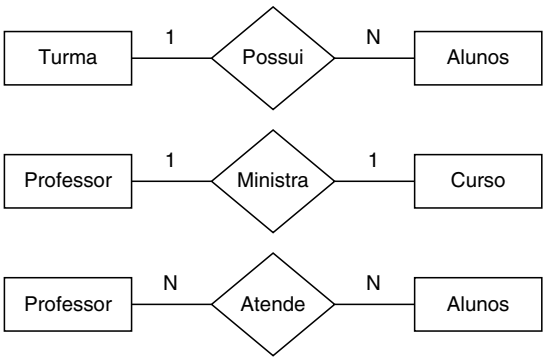
Na Figura 6.5.e temos ainda o conceito de generalização que é uma forma de representar entidades que possuem atributos em comum e podem ser agrupadas em generalizações.



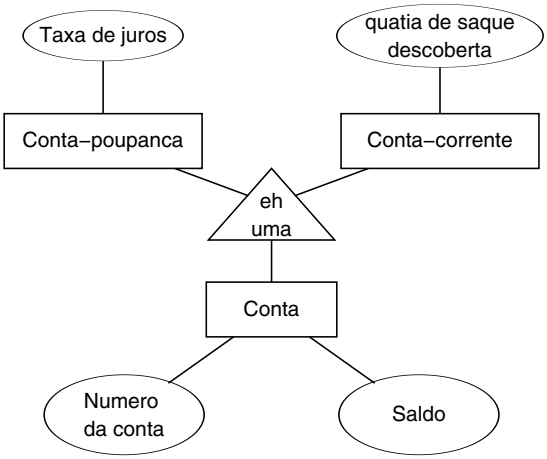
(a) Um diagrama entidade-relacionamento genérico.

(b) Cardinalidade.

(c) Cardinalidade (repr. tradicional).



(d) Tipos de cardinalidade.



(e) Generalização.

Figura 6.5: Diagramas entidade-relacionamento (DER).

No exemplo, a entidade *Conta* é uma generalização das entidades *Conta-corrente* e *Conta-poupança*, Isto significa que as entidades *Conta-corrente* e *Conta-poupança* possuem, além de seus próprio atributos individuais, os atributos da entidade *Conta*.

O modelo entidade-relacionamento é utilizado no projeto de bancos de dados pois este modelo pode ser diretamente implementado utilizando-se um sistema de bancos de dados relacional.

6.3 Modelos de objetos

O desenvolvimento de sistemas utilizando linguagens de programação que oferecem o recurso da orientação a objetos – como Smalltalk, C++ e Java entre outras – precisa ser baseado em uma abordagem de análise e projeto também orientada a objetos.

Modelos orientados a objetos podem ser utilizados para representar tanto os dados como o processamento realizado pelo sistema, combinando atributos dos modelos anteriores. Mas, mais do que isto, os modelos de objetos apresentam o sistema como entidades do sistema podem ser classificadas e compostas por outras entidades, além de trocar serviços entre si.

Modelos de objetos descrevem o sistema em termos de classes de objetos. Uma classe de objetos é uma abstração sobre um conjunto de objetos com atributos comuns e serviços (operações) oferecidas por cada objeto. Vários tipos de objetos podem ser produzidos demonstrando com as classes de objetos se relacionam entre si, como objetos se agregam para formar outros objetos, como objetos utilizam serviços oferecidos por outros objetos e assim por diante. Todas estas possibilidades trabalham em favor do aumento da compreensão sobre o domínio do sistema.

Modelos de objetos representam uma forma natural de refletir o mundo real em entidades manipuladas pelo sistema. Isto fica particularmente claro quando o sistema processa dados oriundos de entidades concretas. Entidades abstratas são mais difíceis de se modelar como objetos. A própria identificação de classes de objeto em si é reconhecidamente uma atividade difícil que requer uma compreensão profunda do domínio da aplicação.

Uma das características interessantes sobre o modelo de objeto é que classes de objetos refletindo entidades do domínio podem ser reutilizadas em outros sistemas. A reusabilidade destas classes pode ser medida pela qualidade da generalização: quanto mais genérica a classe, mais reutilizável ela é.

Uma classe é notada por um retângulo particionado em três subretângulos. No primeiro subretângulos temos o nome da classe, no segundo uma lista de atributos e no terceiro uma lista de serviços que a classe oferece. Nas figuras que seguem você perceberá esta notação.

6.3.1 Modelos de herança

A modelagem orientada a objetos baseia-se na identificação de classes de objetos que são importantes para o domínio que se está estudando. Estas classes são então ordenadas

segundo uma taxonomia, ou seja, segundo um esquema de classificação que mostra como os objetos se relacionam.

A organização das classes de objeto do domínio segundo um modelo de heranças ou em uma hierarquia é uma das taxonomias utilizadas. Segundo este modelo, classes no topo da hierarquia refletem características comuns a todas as classes abaixo. Classes de objetos herdam os atributos e serviços da classe imediatamente acima, chamada sua superclasse.

Os objetos são derivados – ou *instanciados à partir* – do último nível de classes e podem ser tão especializados como necessário.

A Figura 6.6 mostra parte de uma hierarquia simplificada de classes para um sistema de bibliotecas. O diagrama nos mostra que além de livros a biblioteca também disponibiliza outros tipos de materiais como revistas, filmes, e programas de computador. Observamos que o ítem mais genérico encontra-se no topo do diagrama em árvore contendo atributos e serviços que são comuns a todos os ítens da biblioteca. Estes atributos e serviços são herdados pelas classes *Ítem publicado* e *Ítem armazenado*, que também possuem seus próprio atributos à serem passados ao nível inferior.

A Figura 6.7 mostra um outro tipo de entidade que pode compor uma hierarquia de objetos: os usuários do sistema. Este diagrama não descreve partes do sistema à serem automatizadas – pois os usuários serão sempre humanos – mas também neste caso o modelo de objetos nos dá uma visão de como estes usuários se organizam.

A Figura 6.8 mostra um esquema de herança múltipla. Um sistema que suporta herança múltipla permite que uma classe de objetos herde atributos e serviços de mais de uma superclasse. Este recurso pode causar conflitos semânticos quando existem atributos ou serviços de mesmo nome oriundos de superclasses diferentes e com semânticas diferentes. O uso de heranças múltiplas torna o processo de reconhecimento de classes mais difícil e deve ser evitado em situações onde o domínio do sistema ainda não está bastante claro. No exemplo vemos que um *Livro falado* é um misto de um *Livro* e uma *Gravação de audio*, podendo herdar atributos e serviços de ambas as superclasses.

O projeto de herança de classes é um processo difícil pois exige uma compreensão muito clara do domínio do sistema em jogo, entretanto a própria elaboração do modelo de herança ajuda a clarear este domínio. Contudo é sempre difícil determinar as melhores generalizações a fim de evitar duplicações de atributos e serviços em diferentes encaminhamentos da hierarquia de classes.

6.3.2 Agregação de objetos

Além de adquirir atributos e serviços de superclasses, objetos também podem ser formados à partir de outros objetos. O modelo de agregação de objetos, portanto, mostra coleções de classes, ou seja, classes que podem ser compostas de outras classes. É similar a agregação no modelo semântico de dados.

A Figura 6.9 mostra um agregado de classes que representam um *Curso*. Este curso é composto por atributos próprios como *Título*, *Ano* e *Instrutor*, além de outros objetos como uma lista de *Atividades*, um conjunto de *Slides*, *Anotações de aula* e alguns *Videotapes*. A

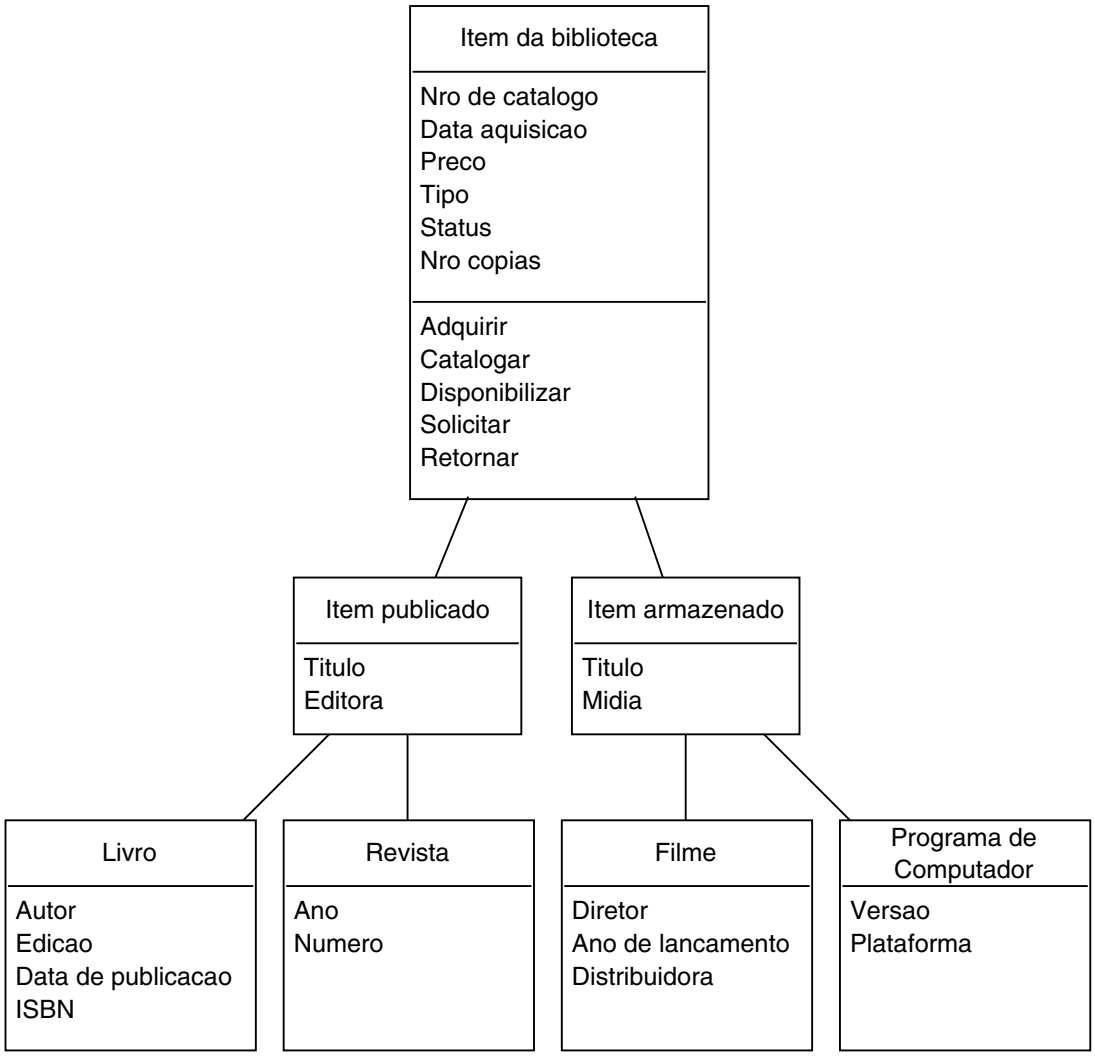


Figura 6.6: Hierarquia de classes para um sistema de biblioteca.

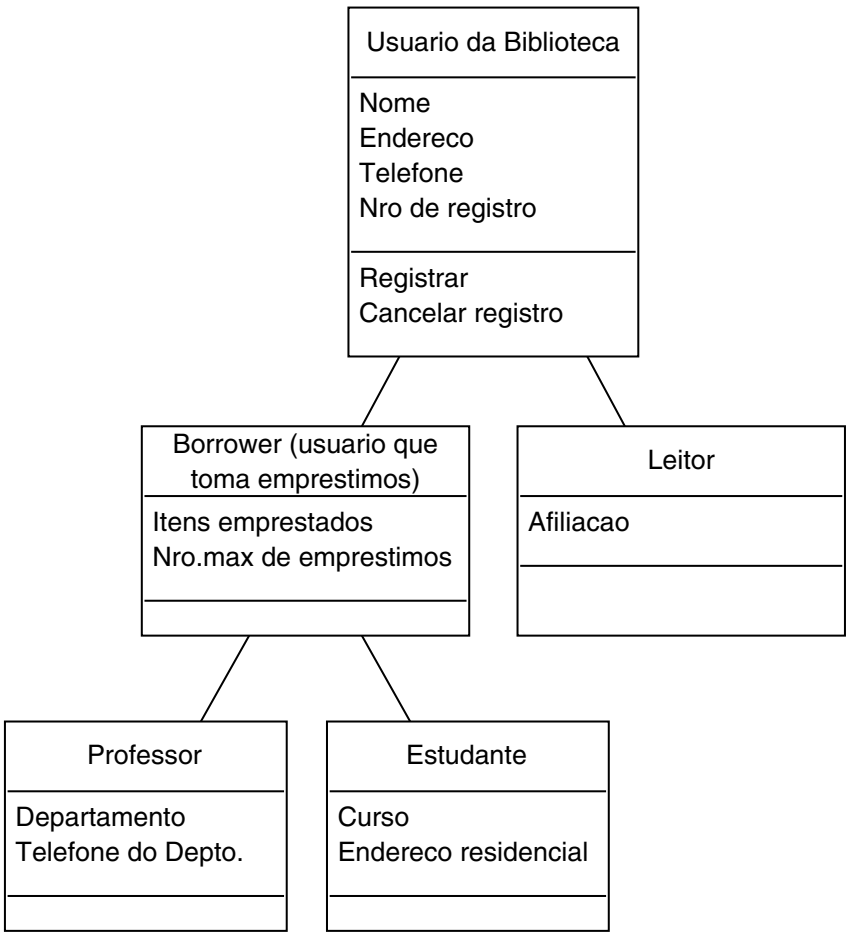


Figura 6.7: Hierarquia de usuários para um sistema de biblioteca.

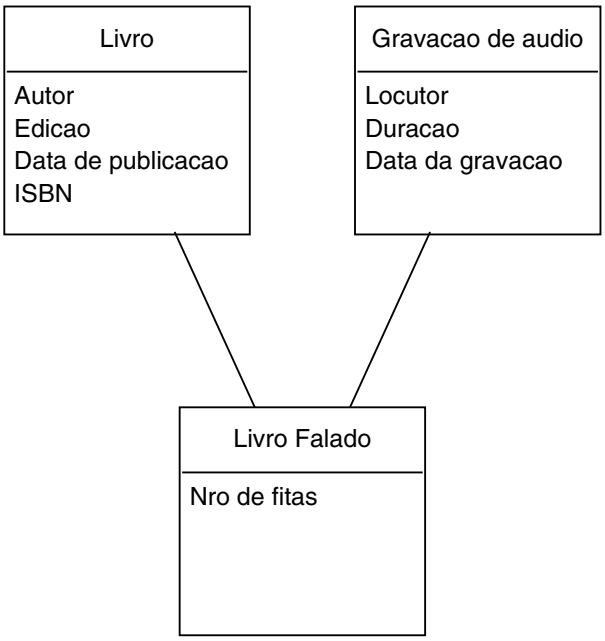


Figura 6.8: Herança múltipla.

notação utilizada no exemplo é a de uma flecha que significa que a relação entre os objetos é uma relação do tipo *parte de* ao invés de uma relação de herança.

6.3.3 Modelos de uso de serviço

Além dos modelos de objetos de hierarquia e de agregação descritos acima, modelos de uso de serviço que mostram como os serviços oferecidos por um objeto são utilizados por outros objetos também podem ser importantes. A Figura 6.10 mostra um exemplo com algumas classes do modelo hierárquico do sistema de biblioteca. O exemplo demonstra que os *Usuários da biblioteca* fazem uso dos serviços *Solicitar* e *Retornar* associados ao *Ítem da Biblioteca*. Já os *Funcionários da biblioteca* utilizam os serviços *Adquirir*, *Catalogar* e *Disponibilizar*, e assim por diante.

Conclusões

Modelos de objetos desenvolvidos durante a análise de requisitos simplificam a transição para o projeto e a programação orientadas a objetos. Entretanto usuários finais costumam considerar modelos de fluxo de dados mais fáceis de serem compreendidos. Na dúvida faça os dois!!!

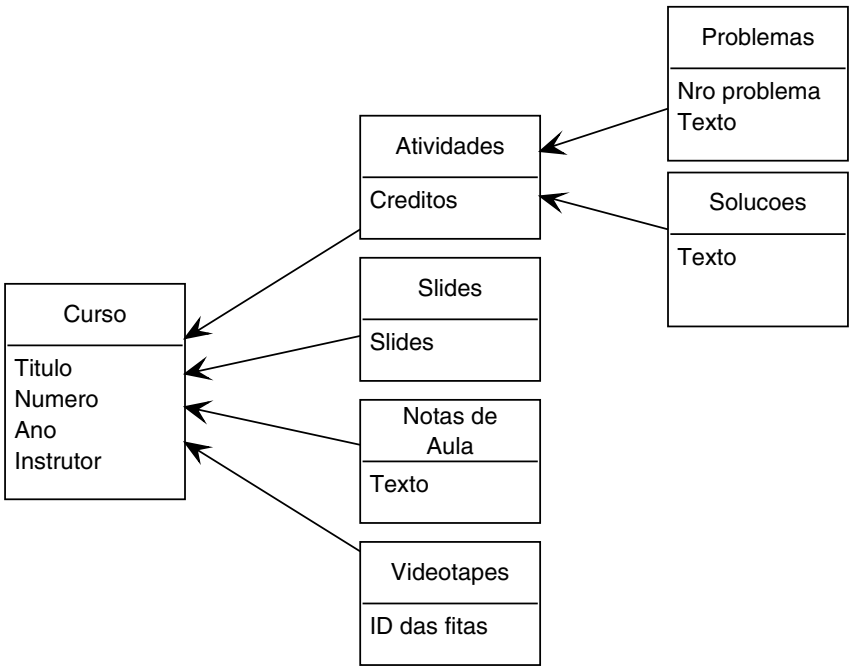


Figura 6.9: Um agregado de classes representando uma disciplina.

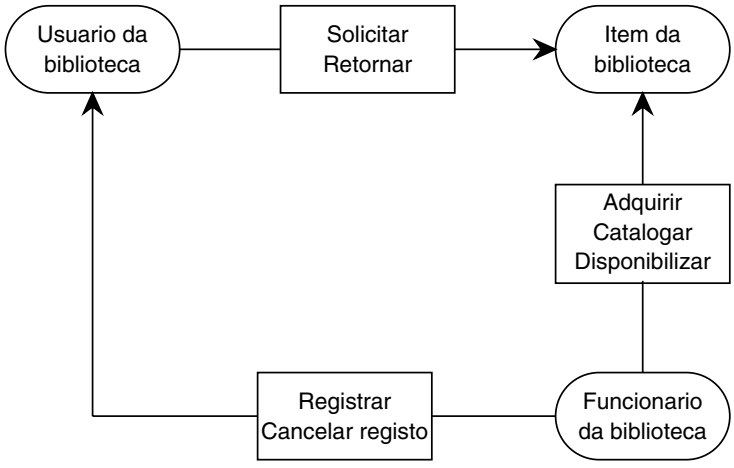


Figura 6.10: Uso de serviço.