# A notação UML

## 1 Introdução

Este apêndice apresenta os principais conceitos da UML (Unified Modeling Language) que são utilizados neste livro. Não é objetivo deste texto a descrição completa de cada conceito, e muitos são apresentados de forma simplificada; os interessados nas descrições completas devem consultar [Rumbaugh+99] ou as especificações oficiais da UML, que podem ser obtidas no sítio do OMG (Object Management Group).

Por outro lado, utilizam-se aqui algumas convenções de notação que o processo Praxis adota, em situações nas quais a UML permite o uso de alternativas. Por exemplo, as convenções de descrição textual dos casos de uso não são estipuladas pela UML, mas são aqui incluídas por estarem intimamente associadas à maneira de utilização dos casos de uso dentro do Praxis.

## 2 Modelagem funcional

### 2.1 Atores

Os papéis dos usuários de um produto são modelados através dos atores (Figura 1). Cada ator representa uma classe de usuários. Os atores modelam os papéis e não as pessoas dos usuários; por exemplo, o mesmo usuário físico pode agir como "Gerente", "Gestor de Estoque" ou "Gestor de Compras". Pode-se também definir atores não humanos, para modelar outros sistemas que devam interagir com o produto em questão: por exemplo, o "Sistema Financeiro".


*Figura 1 - Exemplos de atores*

Caso exista grande número de atores, deve-se procurar agrupá-los em atores genéricos, que representem características comuns a vários grupos de usuários de comportamento semelhante em relação ao produto. Atores genéricos e específicos são ligados por relacionamentos de herança. Na Figura 2, indica-se que "Gerente de Vendas" e "Gerente de Compras" têm alguns aspectos em comum, que são abstraídos através do ator "Gerente".

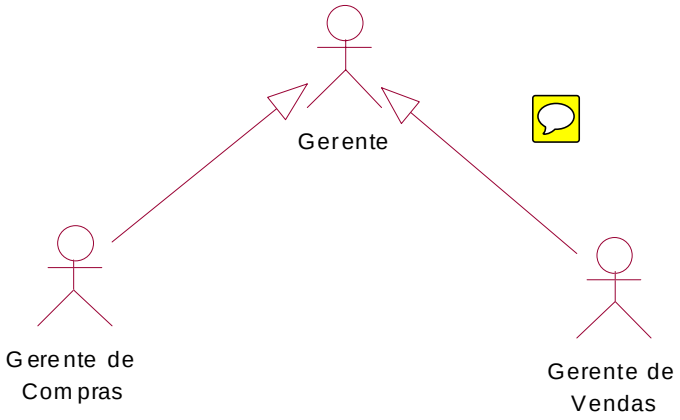


*Figura 2 – Herança entre atores*

## 2.2 Casos de uso

Os **casos de uso** representam funções completas do produto. Um caso de uso realiza um aspecto maior da funcionalidade do produto: deve gerar um ou mais benefícios para o cliente ou os usuários. Na Figura 3 são mostrados os casos de uso que representam a funcionalidade de um produto de informatização de uma mercearia. O conjunto dos casos de uso cobre toda a funcionalidade do produto, e cada caso de uso representa uma fatia independente de funcionalidade.

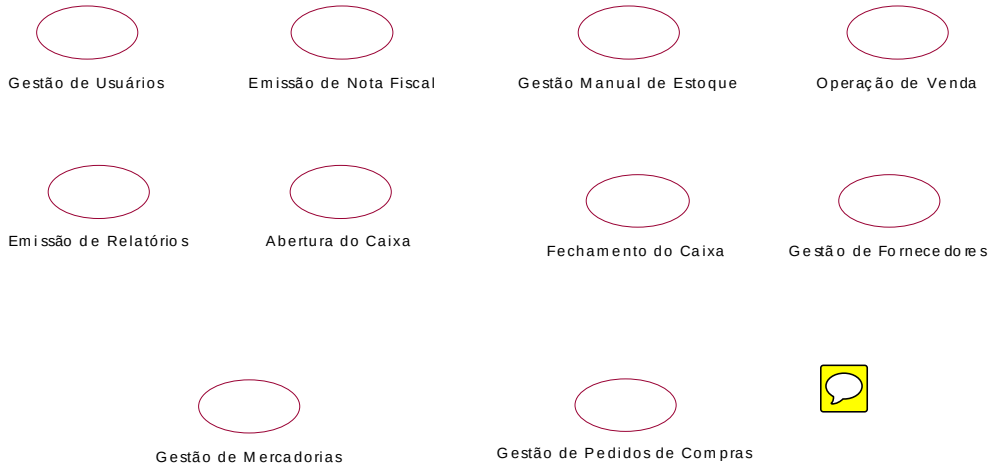


*Figura 3 - Casos de uso*

## 2.3 Diagramas de casos de uso

### *2.3.1 Relacionamentos entre atores e casos de uso*

Diagramas de casos de uso podem especificar os relacionamentos entre casos de uso e atores (Figura 4). Os relacionamentos indicam a existência de comunicação entre atores e casos de uso. Um caso de uso pode estar associado a mais de um ator, quando a sua execução requer a participação de diferentes atores.

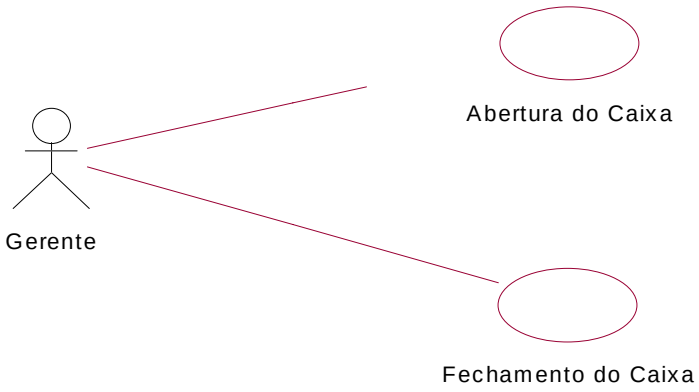


*Figura 4 - Exemplo de casos de uso de um ator*

Normalmente, a comunicação será representada como ligação sem direção; convenciona-se, nesse caso, que a iniciativa de comunicação parte do ator. Quando a iniciativa parte do caso de uso (por exemplo, alarmes, mensagens, dados enviados para outros sistemas etc.), a comunicação deve ser direcionada para o ator (Figura 5). Nesse exemplo, o caso de uso “Gestão Manual de Estoque”, acionado pelo ator “Gestor de Estoque”, envia dados para o “Sistema Financeiro”.

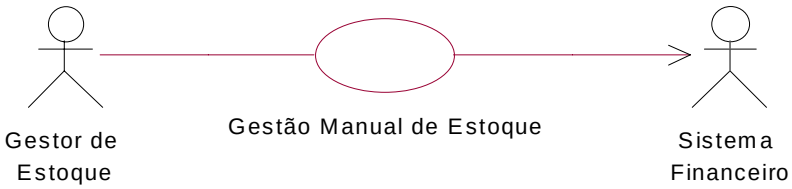


*Figura 5 - Caso de uso com mais de um ator*

Os diagramas de casos de uso podem ser simplificados por meio da herança entre atores. Neste caso, mostra-se um caso de uso comum aos atores específicos, que se comunicam apenas com o ator genérico (Figura 6). Neste exemplo, tanto o "Gerente de Compras" quando o "Gerente de Vendas" podem executar o caso de uso "Emissão de relatórios", porque eles são herdeiros (especializações) de "Gerente".

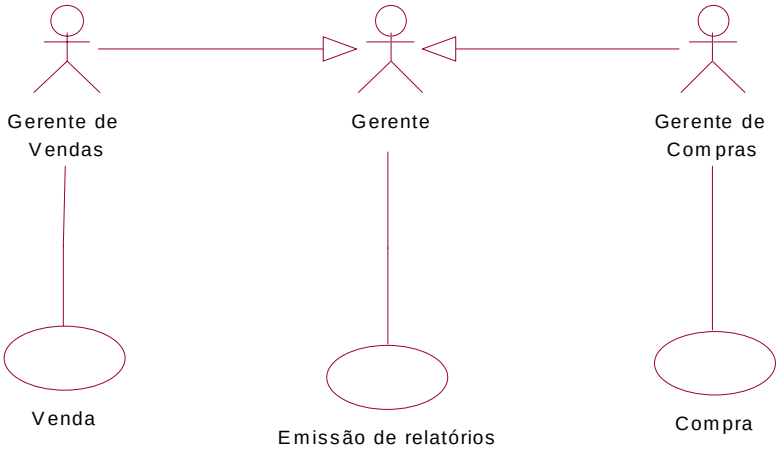


*Figura 6 – Uso de atores genéricos*

O diagrama de contexto (Figura 7) é um diagrama de casos de uso que mostra as interfaces do produto com seu ambiente de aplicação. Os diversos tipos de usuários e outros sistemas com os quais o produto deva interagir são representados por atores situados fora de um retângulo que marca a fronteira do produto.

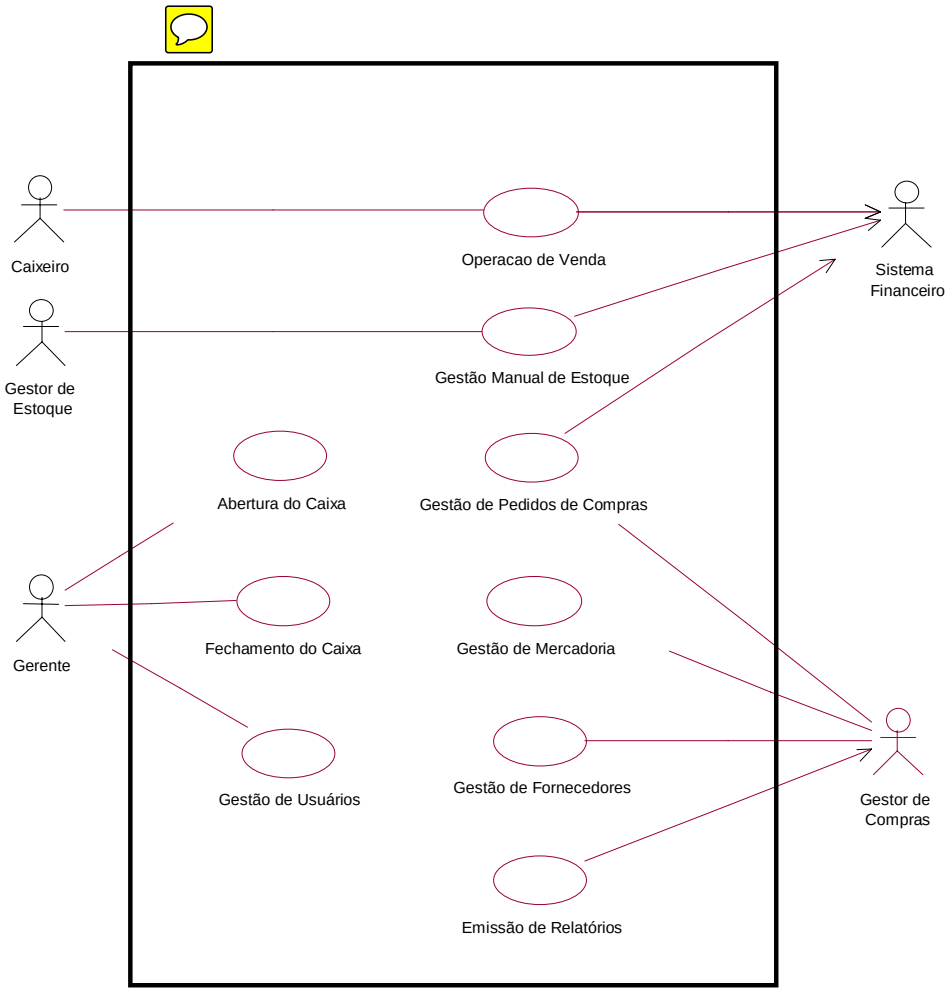


*Figura 7 - Diagrama de contexto*

### *2.3.2 Relacionamentos entre casos de uso*

Notações especiais são utilizadas para facilitar a descrição de funcionalidade mais complexa. Entre estas notações, destacam-se os casos de usos secundários, que simplificam o comportamento dos casos de uso primários através dos mecanismos de **extensão** e **inclusão**. No Praxis, casos de uso primários são aqueles que são invocados por iniciativa direta de um ator; casos de uso secundários são invocados em um passo de outro caso de uso. Os termos “primário” e “secundário”, quando referentes a casos de uso, não fazem parte da UML.

O caso de uso B estende o caso de uso A quando B representa uma situação opcional ou de exceção, que normalmente não ocorre durante a execução de A. Essa notação pode ser usada para representar fluxos complexos opcionais ou anormais. O caso de uso “estendido” é referenciado nas precondições do caso de uso “extensor”. As precondições são a primeira parte dos fluxos dos casos de uso. Na Figura 8, a “Emissão de Nota Fiscal” é uma funcionalidade que pode ser invocada ou não, durante a “Operação de Venda”.

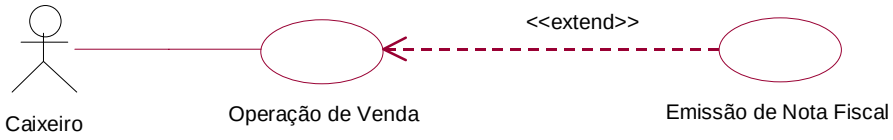


*Figura 8 - Caso de uso de extensão*

O caso de uso A inclui o caso de uso B quando B representa uma atividade complexa, comum a vários casos de uso. Essa notação pode ser usada para representar subfluxos complexos e comuns a vários casos de uso. O caso de uso "incluído" é referenciado no fluxo do caso de uso "incluidor". Na Figura 9, "Baixa no Estoque" representa um comportamento comum a "Gestão Manual de Estoque"e "Operação de Venda".

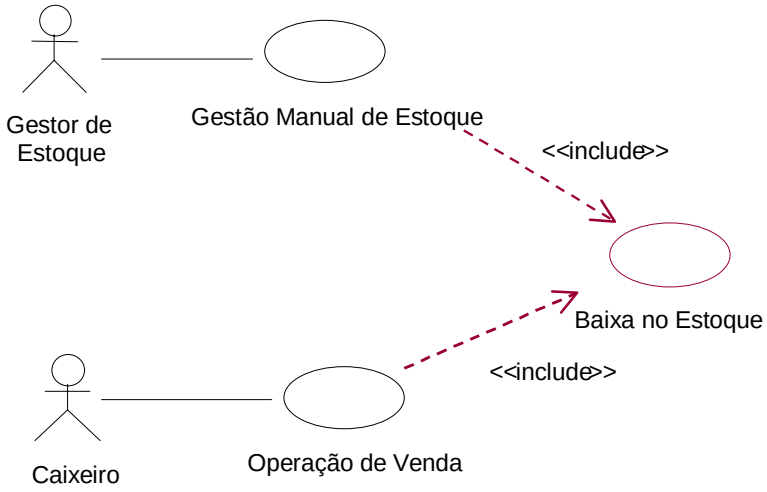


*Figura 9 - Caso de uso de inclusão*

## 2.4 Fluxos dos casos de uso

No Praxis, os fluxos dos casos de uso são detalhados por meio de descrições textuais. A UML não impõe formatos obrigatórios para as descrições dos fluxos, mas a forma de descrição textual aqui apresenta é semelhante às formas usadas pela maioria dos autores que utilizam os casos de uso.

O detalhamento dos fluxos dos casos inclui as suas precondições, ou seja, as condições que supõem estejam satisfeitas, ao iniciar a execução de um caso de uso (Tabela 1); o fluxo principal, que representa a execução mais normal da função; e os subfluxos e fluxos alternativos, que representam variantes que são executadas sob certas condições. A UML permite que diversas notações sejam utilizadas para descrever os detalhes dos casos de uso.

| |
|---|
| Toda mercadoria a ser vendida (item de venda) deve estar previamente cadastrada. |
| O Merci deve estar no Modo de Vendas. |

*Tabela 1 - Precondições de um caso de uso*

Os fluxos são comumente descritos em linguagem natural, na forma de uma seqüência de passos (Tabela 2). Cada passo corresponde a uma ação de um ator ou do produto; estes devem aparecer explicitamente como sujeitos da frase. Outros atores podem aparecer como objetos verbais de uma ação. Condições e iterações podem aparecer, mas os detalhes destas devem ser descritos em subfluxos, de preferência. Isso ajuda a manter a legibilidade do fluxo, que é essencial para garantir o bom entendimento de todas as partes.

| |
|---|
| O Caixeiro faz a abertura da venda. |
| O Merci gera o código da operação de venda. |
| Para cada item de venda, o Merci aciona o subfluxo Registro. |
| O Caixeiro registra a forma de pagamento. |
| O Caixeiro encerra a venda. |
| Para cada item de venda, o Merci aciona o subfluxo Impressão de Linha do Ticket. |
| O Merci notifica o Sistema Financeiro informando: Data, Número da Operação de Venda, “Receita”, Valor Total”, Nome do Cliente (caso tenha sido emitida a nota fiscal). |

*Tabela 2 - Fluxo principal*

Quando uma condição ou iteração for composta por uma seqüência muito curta de passos, elas podem ser representadas por meio de endentação.

| |
|---|
| O Gestor de Compras seleciona a mercadoria.<br>O Merci verifica se existe algum pedido pendente que contenha esta mercadoria.<br>Se não houver pedido pendente contendo a mercadoria a ser excluída:<br>o Merci desvincula a mercadoria dos fornecedores (os fornecedores não mais fornecerão a mercadoria que esta sendo excluída).<br>o Merci faz a remoção da mercadoria.<br>Se houver pedido pendente contendo a mercadoria a ser excluída<br>o Merci emite uma mensagem de erro. |

*Tabela 3 – Fluxo principal com condicionais endentados*

Os subfluxos descrevem geralmente detalhes de iterações, ou de condições executadas com freqüência (Tabela 4).

| |
|---|
| O Caixeiro registra o item de venda, informando a identificação e a quantidade.<br>O Merci totaliza a venda para o cliente da mercearia. |

*Tabela 4 - Exemplo de subfluxo: Registro de item em Operação de Venda*

Os fluxos alternativos descrevem condições pouco habituais ou exceções (Tabela 5).

| |
|---|
| Se o Gestor de Compras solicitar:<br>o Merci imprime o pedido de compra. |

*Tabela 5 - Exemplo de fluxo alternativo: Impressão de Pedido de Compras*

## 2.5 Diagramas de estados

Para casos de uso complexos, pode-se descrever seu comportamento de maneira mais formal, através de um diagrama de estados ou diagrama de atividade da UML. Diagramas de estados são utilizados para descrever fluxos de lógica complexa ou com muitos detalhes, como costuma acontecer nos casos de uso de desenho (Figura 10). As principais convenções são mostradas na Tabela 6.

| Símbolo | Descrição |
|---|---|
| Retângulo com cantos arredondados | Estado. |
| Seta cheia | Transição entre estados diferentes ou de um estado para si mesmo (transições reflexivas). |
| Pequeno círculo cheio | Estado inicial. |
| Círculo cheio dentro de círculo vazio | Estado final. |

*Tabela 6 - Símbolos dos diagramas de estados*

Os eventos que provocam as transições aparecem como rótulos destas. Se a transição depender de uma condição, esta pode ser representada por uma guarda (texto entre colchetes). Anotações são utilizadas para complementar o modelo; por exemplo, indica-se, na Figura 10, que se pode invocar a tela de nota fiscal no estado "Atualizada".

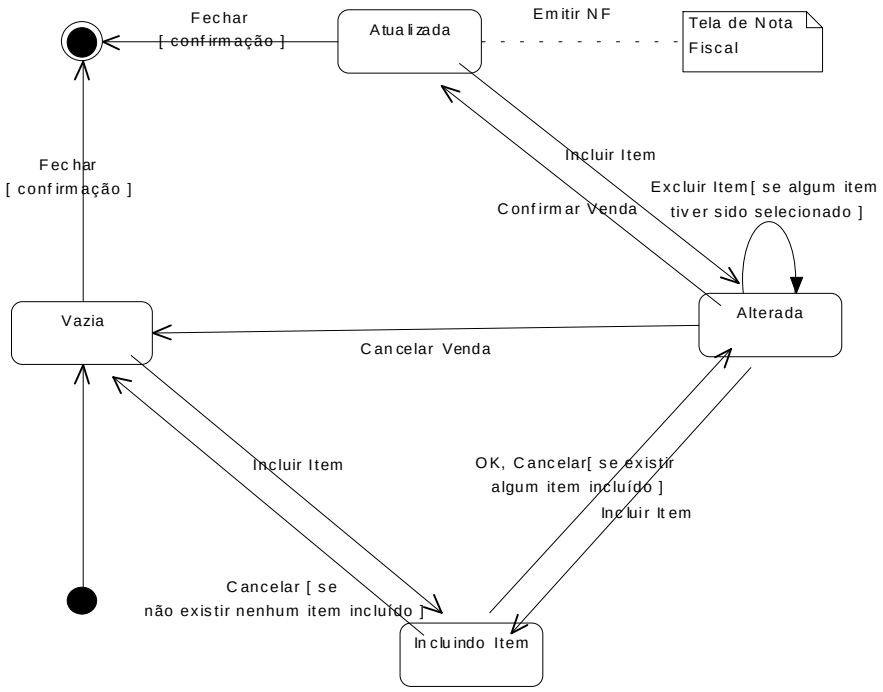


*Figura 10 – Diagrama de estado de caso de uso*

Diagramas de estado podem ser usados para mostrar lógica bastante complexa. Na Figura 11, os estados complexos X e Y tem subestados, que são usados para exprimir a seguinte lógica:

- O estado inicial conduz a X, dentro qual existe um subestado inicial que conduz a X1. Ao entrar em X1, a ação A0 é executada. O evento E1 leva ao subestado X2.

- No subestado X2, se o evento E2 ocorrer com a condição C1, é executada a ação A1 e continua-se no subestado X2. Se E2 ocorrer com a condição C1 falsa, executa-se a ação A2 e passa-se para o subestado Y1 do estado Y.

- Durante a permanência no subestado Y1 é executada a ação A4. Se no subestado Y1 ocorrer o evento E3, executa-se a ação A3 e passa-se ao subestado Y2.

- Se no subestado Y2 ocorrer o evento E4, volta-se ao estado X; na saída de Y2 envia-se o evento E5, que pode provocar alguma transição não mostrada nesse diagrama.

- Se no estado Y (em qualquer subestado) ocorrer o evento E6, vai-se para um estado final.

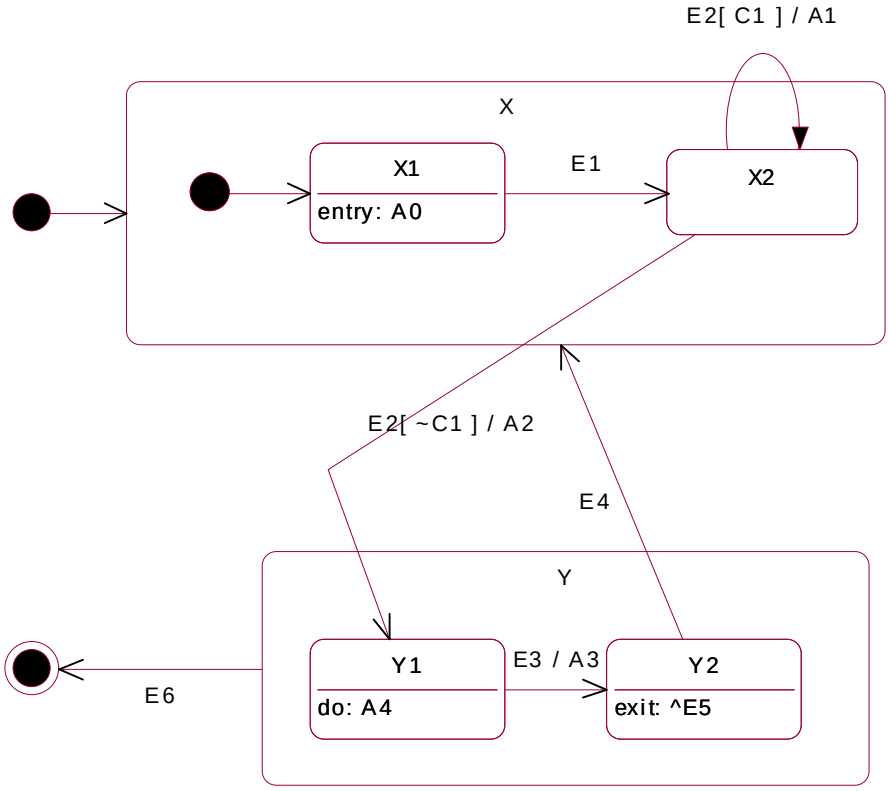


*Figura 11 – Diagrama de estados complexo*

## 2.6 Diagramas de atividade

Estes diagramas são uma variante dos fluxogramas, sendo geralmente usados para descrever processos de negócio, ou outros fluxos onde o paralelismo de atividades seja importante. A Figura 12 apresenta um exemplo de diagrama de atividades. A Tabela 7 explica os símbolos usados.

| Símbolo | Descrição |
|---|---|
| Retângulo ovalado | Atividade (passo do fluxo). |
| Retângulo | Objeto (artefato do fluxo). |
| Seta cheia | Relação de precedência entre atividades. |
| Seta pontilhada | Consumo ou produção de objeto por atividade. |
| Linha horizontal cheia | Ponto de sincronização (onde subfluxos paralelos se juntam). |
| Pequeno círculo cheio | Estado inicial. |
| Círculo cheio dentro de círculo vazio | Estado final. |

*Tabela 7 - Símbolos dos diagramas de atividades*

Nesse exemplo, mostra-se que as atividades “Detalhamento dos requisitos de interface” e “Detalhamento dos casos de uso” são realizadas em paralelo, mas a atividade “Detalhamento dos requisitos não funcionais” requer que ambas estejam completas. Os objetos mostrados à direita representam os resultados das atividades.

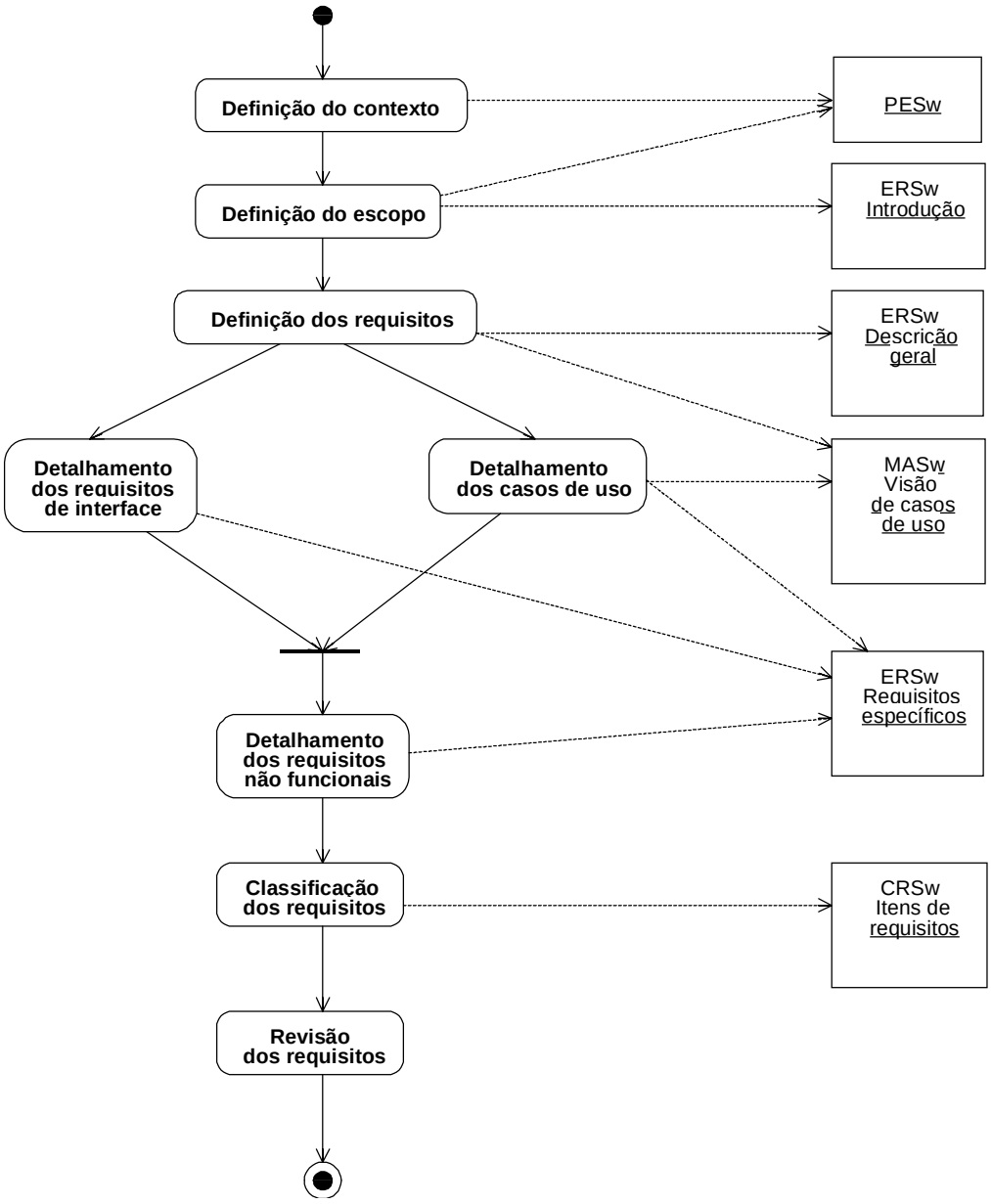


*Figura 12 – Exemplo de diagrama de atividades*

Um diagrama de atividades mais complexo é mostrado na Figura 13. Ele é dividido em **raias** (*swimlanes*), que representam os diversos papéis de pessoas ou grupos envolvidos no processo (no exemplo, cliente da mercearia, caixeiro e gerente). Os objetos situados sobre as fronteiras das raias representam objetos partilhados entre os papéis. Em um processo de negócio, eles podem representar tanto objetos físicos quanto informacionais. Um losango de decisão tem como saídas subfluxos alternativos (no exemplo, a emissão opcional de nota fiscal).

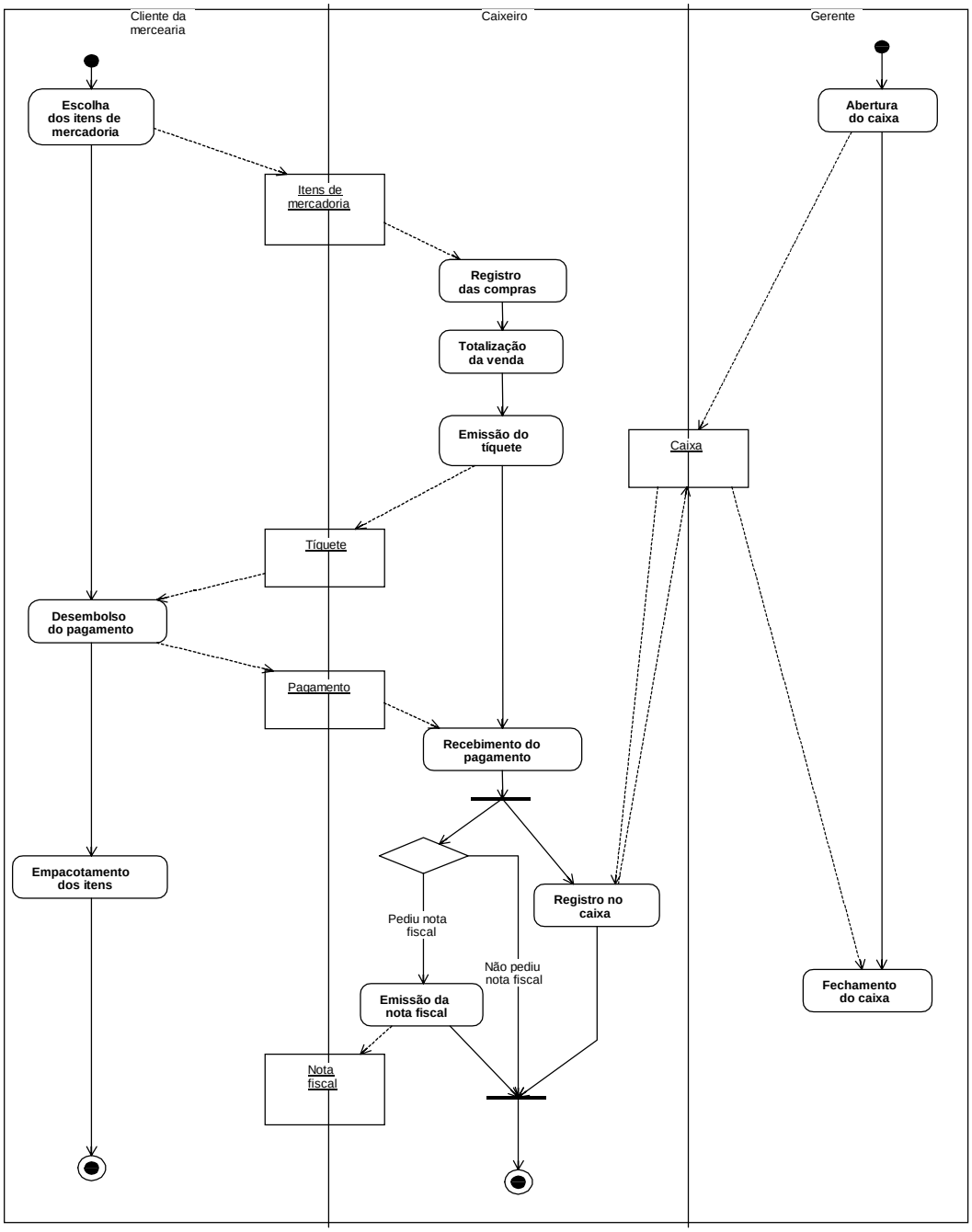


*Figura 13 – Exemplo de modelo de processo de negócio*

# 3 Modelagem lógica

## 3.1 Objetos e classes

Nas metodologias de modelagem orientadas a objetos, os objetos representam entidades discretas, com fronteira bem definida e identidade própria, que encapsulam estado e comportamento. O estado é representado pelos atributos do objeto, e o comportamento pelas respectivas operações. Os objetos interagem entre si trocando mensagens, que são invocações das operações. Objetos similares são agrupados em classes.

Na UML, um objeto é representado por um retângulo, onde o nome do objeto é sublinhado. Quando um objeto aparece em um diagrama UML, podem acontecer as seguintes situações:

- Um objeto pode ter classe indeterminada (Figura 14). Esta é uma situação transitória que pode acontecer durante a análise, mas deve ser resolvida.

- Um objeto pode ter denominação própria e pertencer a uma determinada classe (Figura 15). O nome da classe é separado do nome do objeto por um sinal de dois pontos.

- Um objeto pode ser anônimo, representando uma instância genérica de determinada classe (Figura 16). O campo à esquerda dos dois pontos fica vazio.

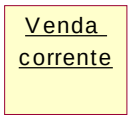


*Figura 14 - Objeto sem classe determinada*

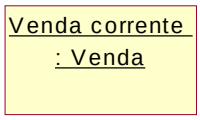


*Figura 15 - Objeto com indicação da respectiva classe*

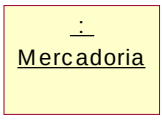


*Figura 16 - Objeto anônimo*

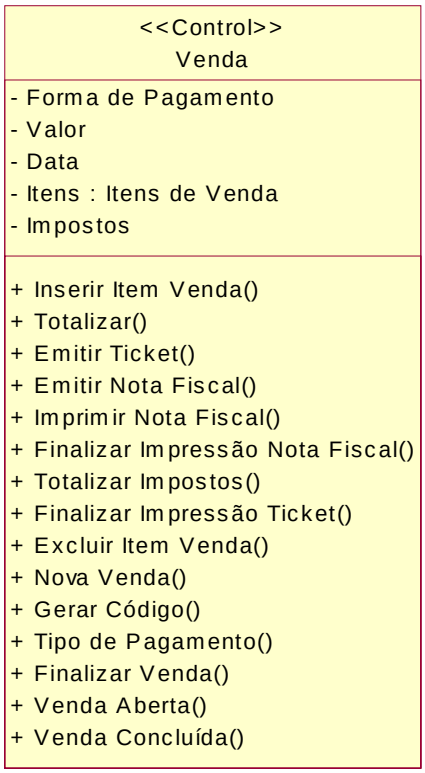


*Figura 17 - Representação de classe*

Na UML, a classe é representada por um retângulo dividido em três compartimentos, que contêm respectivamente o nome da classe, os atributos e as operações. Para maior clareza nos diagramas, pode-se suprimir cada um dos compartimentos de atributos e operações, ou deixar de mostrar determinados atributos ou operações. São opcionais também: a indicação da visibilidade por caracteres ou ícones; a assinatura (lista de argumentos e tipo de retorno) das operações; e o tipo e valor padrão dos atributos.

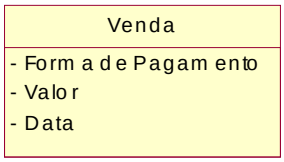


*Figura 18 - Representação de classe com supressão das operações*

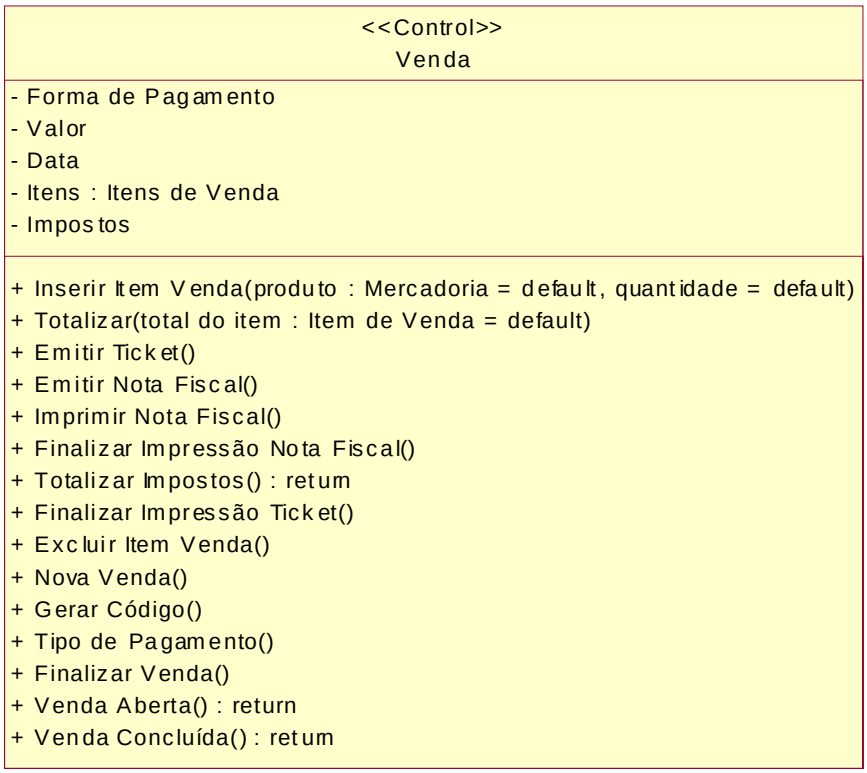


*Figura 19 - Representação de classe com detalhamento das assinaturas das operações*

## 3.2 Organização das classes

### *3.2.1 Pacotes lógicos*

Os **pacotes lógicos** são agrupamentos de elementos de um modelo. No modelo de análise, eles podem ser utilizados para formar grupos de classes com um tema comum. Na Figura 20, os pacotes lógicos Compras, Vendas e Administração são usados para agrupar classes especializadas em relação a esses aspectos.

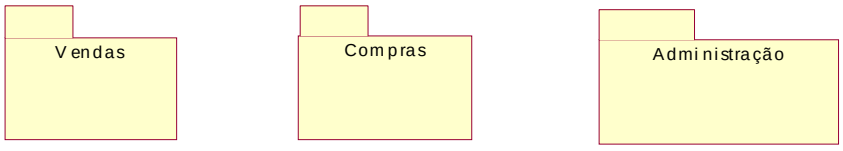


*Figura 20 - Pacotes lógicos*

Pacotes lógicos podem ter relações de dependência e continência entre si. Na Figura 21, os pacotes lógicos “Classes de fronteira” e “Classes de controle” dependem (importam recursos) de “Coleções de entidades”. “Coleções de entidades” contém os pacotes “Coleções” e “Elementos”. “Classes VB” é um pacote global: seus recursos estão disponíveis para os elementos de todos os outros pacotes.

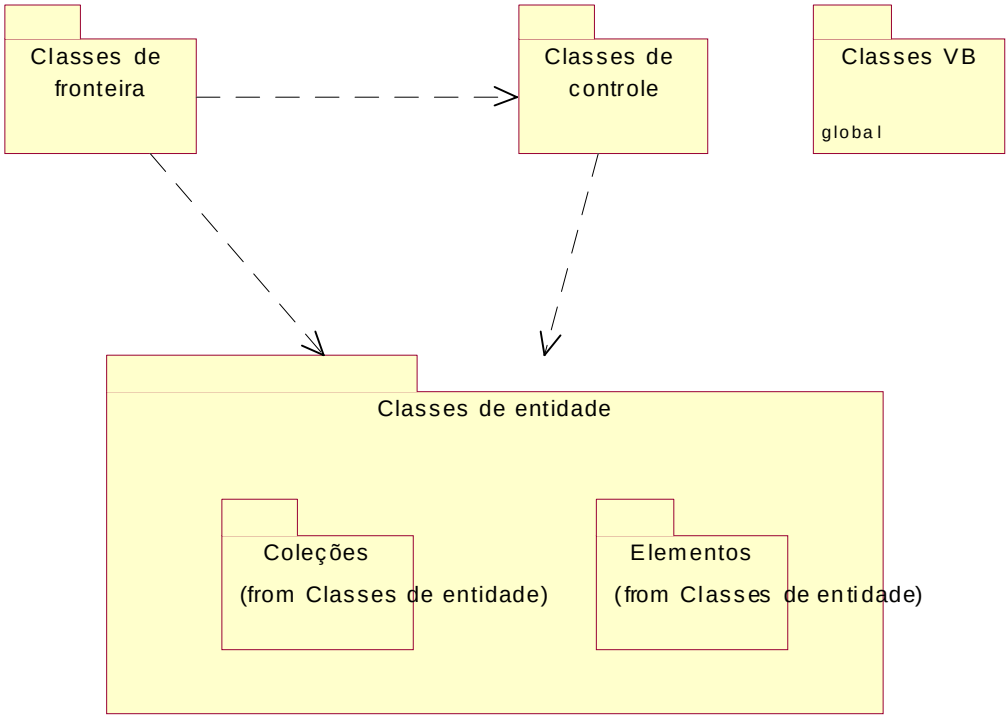


*Figura 21 – Relacionamentos entre pacotes lógicos*

### *3.2.2 Estereótipos*

Os **estereótipos** são extensões de elementos do modelo. Podem ser usados para denotar especializações significativas de classes. Os atores, por exemplo, podem ser tratados pelas ferramentas de modelagem como classes estereotipadas. Como mostra a Figura 22, estereótipos podem ser indicados através de ícones próprios, ou incluindo-se o nome do estereótipo em aspas francesas (os caracteres « », representados nos desenhos por << >>).

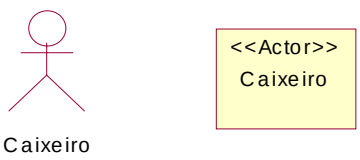


*Figura 22 - Representações alternativas do estereótipo "Ator"*

Estereótipos são usados para criar especializações da UML em relação a determinadas áreas de modelagem. Jacobson ([Jacobson94], [Jacobson+99]) propõe a divisão das classes do Modelo de Análise de acordo com os estereótipos descritos a seguir, que foram incorporados ao padrão oficial da UML. Na Figura 23 eles são representados de forma textual, e na Figura 24 são representados na forma icônica.

- **Entidades** ("entity") – modelam informação persistente, sendo tipicamente independentes da aplicação. Geralmente são necessárias para cumprir alguma responsabilidade do produto, e freqüentemente correspondem a tabelas de bancos de dados.

- **Fronteiras** ("boundary")– tratam da comunicação com o ambiente do produto. Modelam as interfaces do produto com usuários e outros sistemas, e surgem tipicamente de cada par ator – caso de uso.

- **Controles** ("control") – coordenam o fluxo de um caso de uso complexo, encapsulando lógica que não se enquadra naturalmente nas responsabilidades das entidades. São tipicamente dependentes de aplicação.

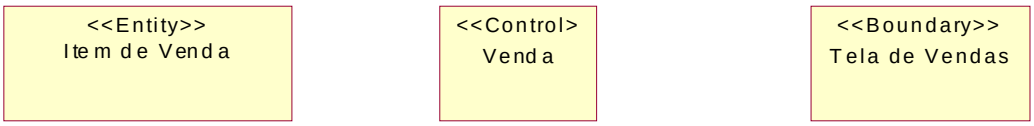


*Figura 23 - Exemplo de classes de análise com estereótipos textuais*

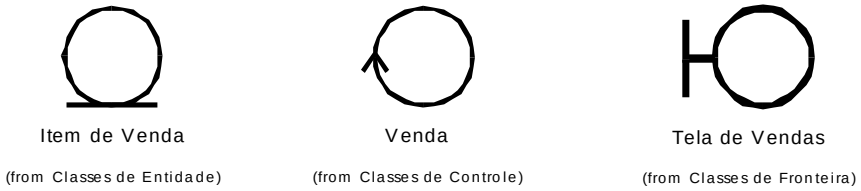


*Figura 24 - Exemplo de classes de análise com estereótipos icônicos*

## 3.3 Relacionamentos

### *3.3.1 Relacionamentos de associação*

As associações expressam relações bidirecionais de dependência semântica entre duas classes. Elas indicam a possibilidade de comunicação direta entre os respectivos objetos. Isso significa que faz parte das responsabilidades de um objeto de uma das classes determinar os objetos correspondentes da outra classe. Normalmente, existirão em cada classe operações para cumprir essa responsabilidade.


*Figura 25 - Relacionamento de associação*

### *3.3.2 Multiplicidades e papéis*

A especificação das associações inclui o seu nome, descrição e possíveis restrições. Em certos casos, é útil batizar também os papéis, assim como especificar vários detalhes destes. Os nomes das

associações devem ser simples e significativos. Recomenda-se usar um substantivo que descreva bem a semântica do relacionamento. Pode-se também usar um verbo, desde que esteja claro qual classe é sujeito e qual classe é objeto desse verbo (Figura 26). Uma convenção habitual é batizar o relacionamento de modo que ele seja lido corretamente de cima para baixo ou da esquerda para a direita. Na última versão da UML, um pequeno triângulo pode ser usado para indicar a direção de leitura, caso necessário.

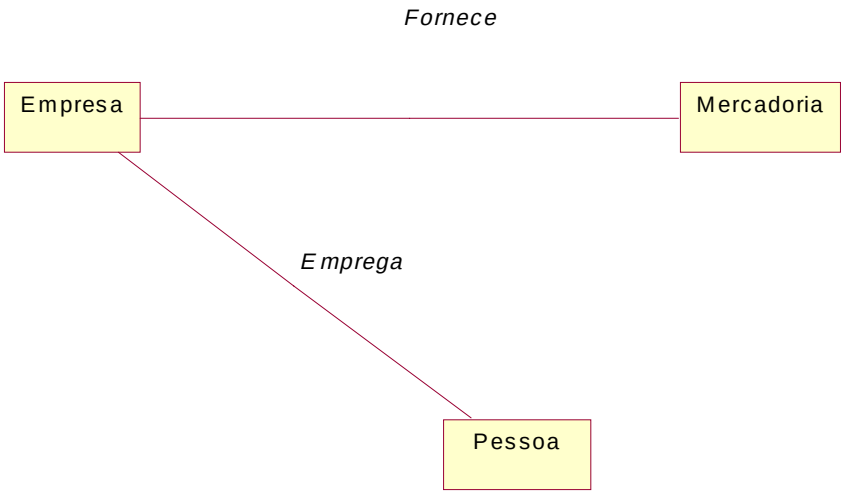


*Figura 26 - Relacionamento com nome*

A multiplicidade de um participante de um relacionamento indica quantos objetos de uma classe se relacionam com cada objeto da outra classe. Relacionamentos obrigatórios têm multiplicidade mínima 1. A multiplicidade máxima indica o número máximo de instâncias da classe alvo que podem existir simultaneamente. Na Figura 27 modela-se o fato de que uma “Pessoa” pode estar ligada a nenhuma ou uma “Empresa”, enquanto uma “Empresa” pode estar relacionada com uma ou mais instância de “Pessoa” (por exemplo, empregar). Uma “Empresa” pode estar relacionada com zero ou mais instâncias de “Mercadoria” (por exemplo, fornecer).

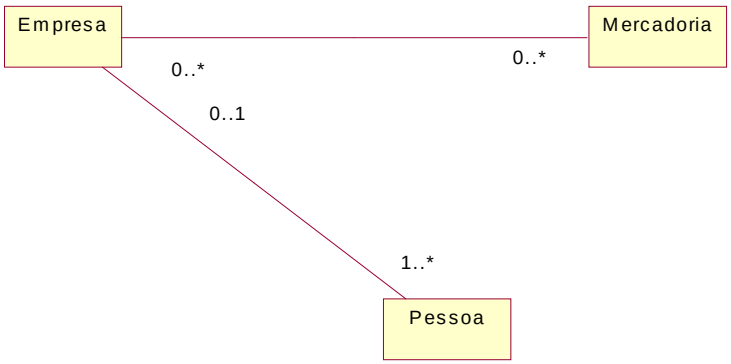


*Figura 27 - Relacionamentos com multiplicidades*

Os papéis são denominações que exprimem em que qualidade um objeto de uma das classes do relacionamento se relaciona com um objeto da outra classe.Os papéis dos participantes devem ser batizados explicitamente quando participarem de um relacionamento em uma qualidade que não é implícita no respectivo nome da classe. Na Figura 28 indica-se que um objeto da classe "Empresa" participa no papel de "fornecedor" do relacionamento "Fornece", com zero ou mais objetos da classe

"Mercadoria", e um objeto dessa classe se relaciona com zero ou mais objetos da classe "Empresa" na qualidade de "produto". Uma "Empresa" pode participar de outros relacionamentos em outras qualidades; por exemplo, no papel de "empregador", em um relacionamento "Emprega" com um objeto da classe "Pessoa".

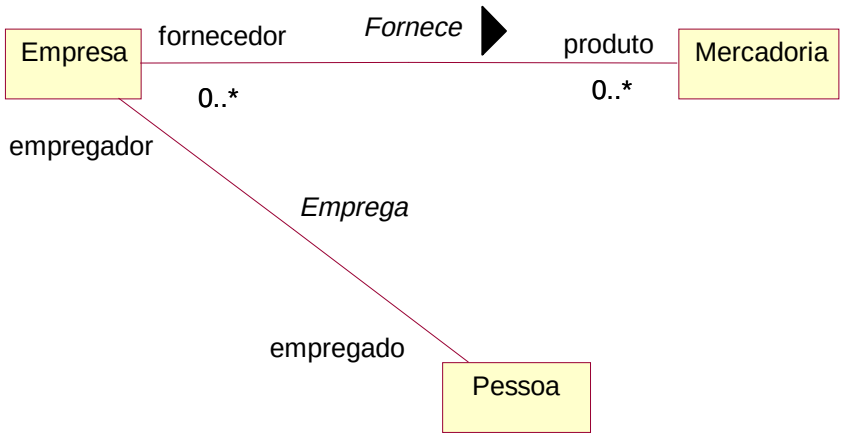


*Figura 28 - Relacionamentos com denominação dos participantes*

### *3.3.3 Navegabilidade*

Os relacionamentos podem ter direção de navegação. Um relacionamento é navegável da classe A para a classe B se, dado um objeto da classe A, consegue-se obter de forma direta (por exemplo, através de uma operação da classe A) os objetos relacionados da classe B. Por exemplo, a Figura 29 indica que, dada uma "Mercadoria", é possível localizar diretamente o respectivo "Fornecedor", mas a recíproca não é verdadeira.

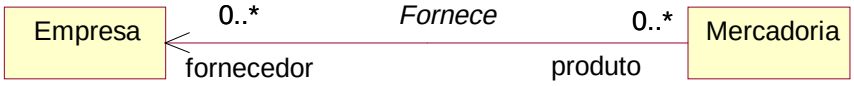


*Figura 29 - Relacionamento com restrição de navegabilidade*

### *3.3.4 Relacionamentos de herança*

O relacionamento de herança existe entre classes de natureza mais geral (chamadas de superclasses ou classes bases) e suas especializações (chamadas de subclasses ou classes derivadas). As superclasses contêm atributos ou operações comuns a um grupo de subclasses. Na Figura 30 modela-se o fato de que um “Item de Compra” é um tipo (uma especialização) de “Item de Mercadoria”, assim com um “Item de Venda”.

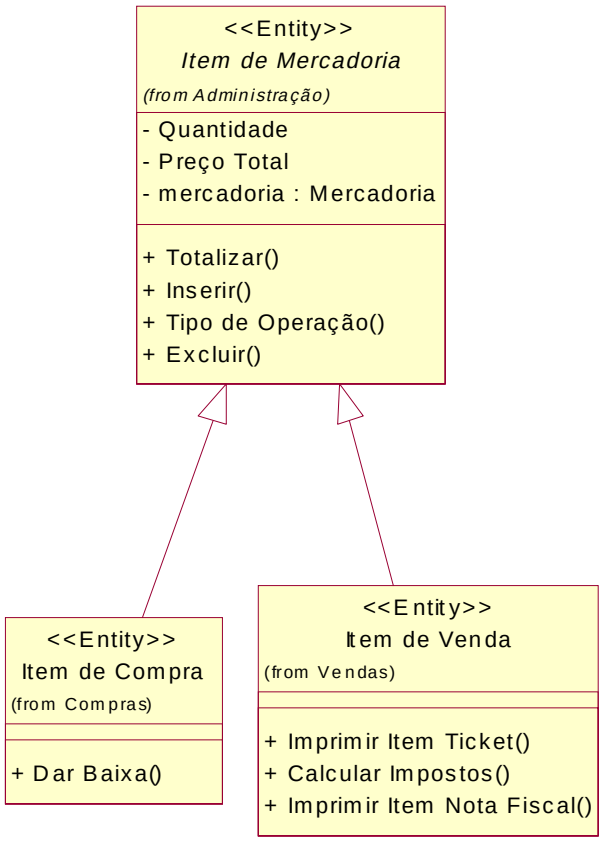


*Figura 30 - Exemplo de relacionamentos de herança*

### *3.3.5 Relacionamentos avançados*

Um **relacionamento de agregação** é uma associação que reflete a construção física ou a posse lógica. Relacionamentos de agregação são casos particulares dos relacionamentos de associação, e só é necessário distingui-los quando for conveniente enfatizar o caráter "todo-parte" do relacionamento. Geralmente um relacionamento de agregação é caracterizado pela presença da expressão "parte de" na descrição do relacionamento, e pela assimetria da navegação.

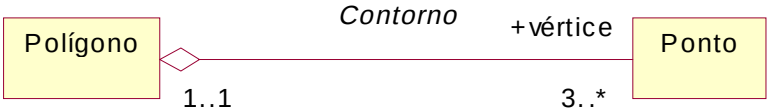


*Figura 31 - Relacionamento de agregação*

Um tipo mais forte de relacionamento todo-parte é o **relacionamento de composição**. Nesse caso, os objetos da classe parte não têm existência independente da classe todo. A Figura 32 indica que o "centro" de um "Círculo" não tem existência independente deste, enquanto que a Figura 31 indica que cada "ponto" existe independentemente do "Polígono" ao qual serve de "vértice".


*Figura 32 - Relacionamento de composição*

Uma associação pode ser reflexiva. Uma auto-associação indica um relacionamento entre objetos de mesma classe que desempenham diferentes participações. Na Figura 33 indica-se uma “Pessoa” na qualidade de “chefe” relaciona-se com uma ou mais instâncias de “Pessoa” que exercem o papel de “subordinado”.

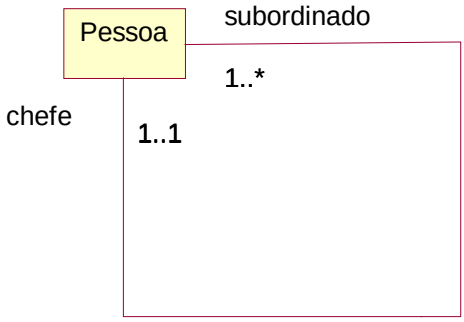


*Figura 33 - Associação reflexiva*

Os relacionamentos podem ser de natureza complexa. Em certos casos, um relacionamento é mais bem explicado através de uma classe de associação, que exprime atributos e até operações que são propriedades do relacionamento como um todo e não de cada participante isoladamente. Na Figura 34 a classe “Emprego” inclui atributos e operações que não estão ligados a uma “Empresa” ou uma “Pessoa” isolada, mas ao relacionamento entre elas; por exemplo, o atributo “data de início” ou a operação “imprimir atestado de emprego”.

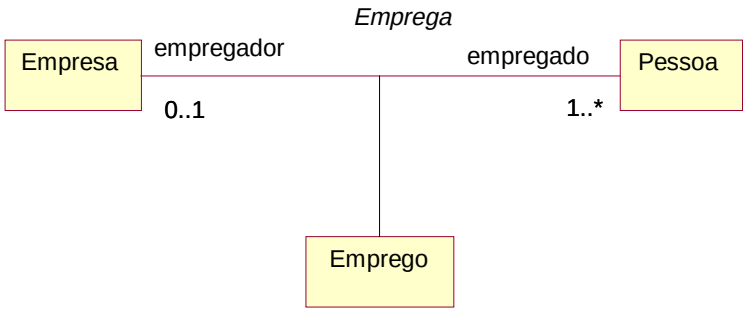


*Figura 34 - Classe de associação*

Um qualificador é um atributo que restringe um relacionamento através de uma seleção. Na maioria dos casos, cada valor do qualificador está associado a uma única instância da classe alvo. Na Figura 35, o qualificador "número de conta" restringe a participação de objetos da classe "Pessoa" no relacionamento. Um "número de conta" pode estar associado a zero ou um cliente (contas conjuntas não são modeladas).


*Figura 35 - Relacionamento com qualificador*

## 3.4 Detalhes das classes

### *3.4.1 Mensagens e operações*

Em modelos orientados a objetos, as mensagens representam os mecanismos de interação entre estes. Um objeto só pode receber mensagens que correspondam à invocação de uma operação da respectiva

classe. Durante a execução da modelagem, os diagramas podem conter, em caráter provisório, mensagens não associadas a operações de alguma classe. Entretanto, ao término da análise todas as mensagens têm de ser resolvidas em termos de operações de alguma classe.

Uma mensagem tem as seguintes partes:

- receptor - o objeto que recebe a mensagem;
- operação - a função requisitada do receptor;
- parâmetros - os dados para a operação.

Na Figura 36 modela-se uma interação simples entre dois objetos, através de uma mensagem sem parâmetros.

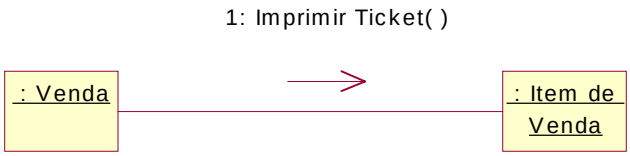


*Figura 36 - Exemplo de mensagem sem parâmetro*

Na Figura 37 modela-se uma colaboração complexa que envolve um ator (a rigor, uma instância de ator) e vários objetos. Os parâmetros das mensagens são incluídos no modelo.

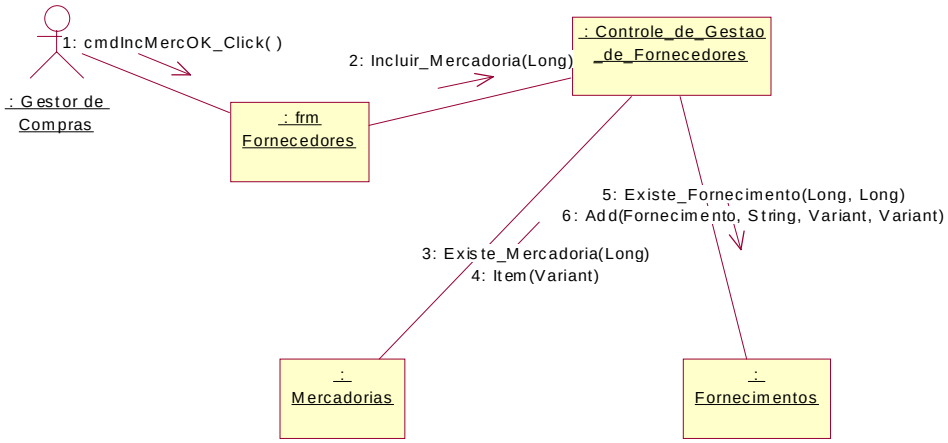


*Figura 37 – Exemplo de mensagens com parâmetros*

### *3.4.2 Atributos*

Atributos são propriedades que descrevem as classes. Atributos equivalem a relacionamentos de composição em que:

- a classe parte é o tipo do atributo;
- o papel é o nome do atributo.

Atributos e relacionamentos podem ser alternativas para se expressar os mesmos conceitos (Figura 38). A escolha entre atributo e relacionamento deve visar à clareza do modelo. Geralmente atributos são de tipos equivalentes a classes pequenas e reutilizáveis, que representam abstrações de nível superior ao do domínio do problema.

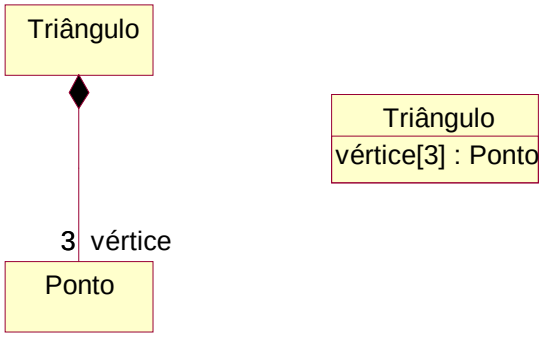


*Figura 38 – Equivalência entre atributos e composições*

### *3.4.3 Controle de acesso*

Os tipos de acesso a operações ou atributos de cada classe incluem:

- **público**: qualquer outra classe pode usar (símbolo na UML +);
- **protegido**: só subclasses dessa classe podem usar (símbolo na UML #);
- **privado**: nenhuma outra classe pode usar diretamente (símbolo na UML -);
- **implementação**: declarado no corpo de uma operação.

Exceções ao controle normal de acesso, para determinadas classes clientes, podem ser feitas declarando-se essas classes como **amigas** da classe fornecedora.

Na Figura 39 são mostrados os controles de acesso dos atributos e operações de uma classe implementada em Visual Basic. Note-se que todos os atributos são privados. São públicas as operações provenientes do modelo de análise e as operações de acesso aos atributos; são privadas as operações de início e término da vida dos objetos.

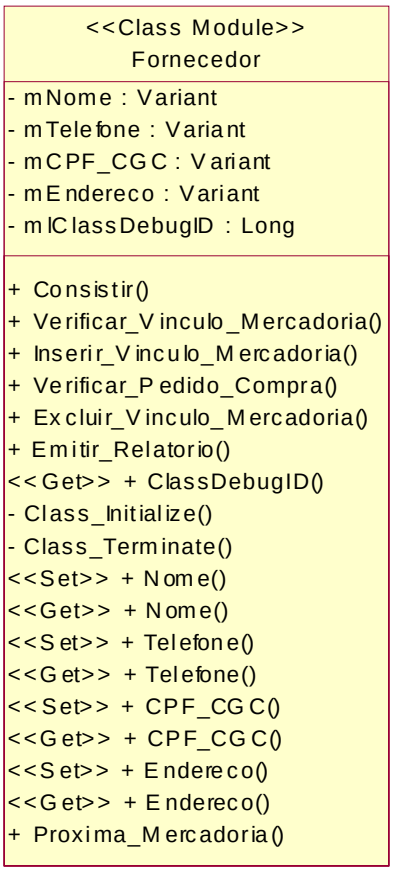


*Figura 39 - Exemplo de controles de acesso*

Note-se também, na Figura 39, o uso de estereótipos para indicar que construções da linguagem alvo serão utilizadas para implementar a classe (`<< Class Module >>`) e as operações de acesso aos atributos (`<< Get >>`, `<< Set >>`).

## 3.5 Diagramas de interação

### *3.5.1 Diagramas de seqüência*

Os diagramas de seqüência enfatizam o ordenamento temporal das ações. Eles são construídos de acordo com as seguintes convenções:

- linhas verticais representam os objetos;
- setas horizontais representam as mensagens passadas entre os objetos;
- rótulos das setas são os nomes das operações;
- a posição na vertical mostra o ordenamento relativo das mensagens;
- o diagrama pode ser complementado e esclarecido por anotações.

A UML prevê que os retângulos situados nas linhas verticais devem indicar o tempo de vida dos objetos. Esta convenção não foi seguida neste texto, por limitação da ferramenta de modelagem utilizada.

| |
|---|
| O Gestor de Compras seleciona a mercadoria. |
| O Merci verifica se existe algum pedido pendente que contenha esta mercadoria. |
| Se não houver pedido pendente contendo a mercadoria a ser excluída, |
| o Merci desvincula a mercadoria dos fornecedores (os fornecedores não mais fornecerão a mercadoria que esta sendo excluída); |
| o Merci faz a remoção da mercadoria. |
| Se houver pedido pendente contendo a mercadoria a ser excluída, |
| o Merci emite uma mensagem de erro. |

*Tabela 8 - Exemplo de fluxo de caso de uso*

Os diagramas de seqüência são orientados para exprimir, de preferência, o desenrolar temporal de seqüências de ações. É mais difícil representar lógicas de seleção e repetição sem prejudicar a inteligibilidade do diagrama. Os roteiros representam desdobramentos da lógica do caso de uso. É preferível usar diagramas separados para representar roteiros resultantes de diferentes caminhos lógicos. Por exemplo, a lógica contida no fluxo da Figura 40 pode ser descrita através dos diferentes roteiros (Figura 41 e Figura 42). Para simplificar o exemplo, os diagramas não usam classes de fronteira.

| |
|---|
| O Caixeiro registra itens de mercadoria. |
| O Merci totaliza venda. |
| O Caixeiro registra modo de venda. |
| Se venda a prazo, |
| o Caixeiro insere venda em contas a receber. |
| Senão, |
| o Caixeiro registra pagamento. |

*Figura 40 – Exemplo de fluxo com lógica de seleção: Operação de Venda*

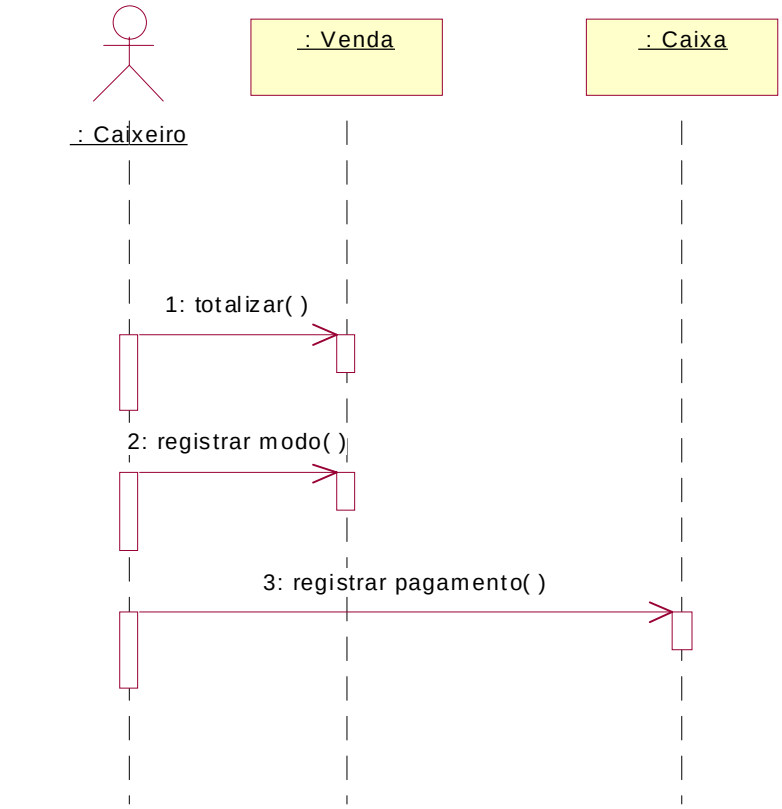


*Figura 41 – Exemplo de roteiro alternativo: Operação de Venda à Vista*

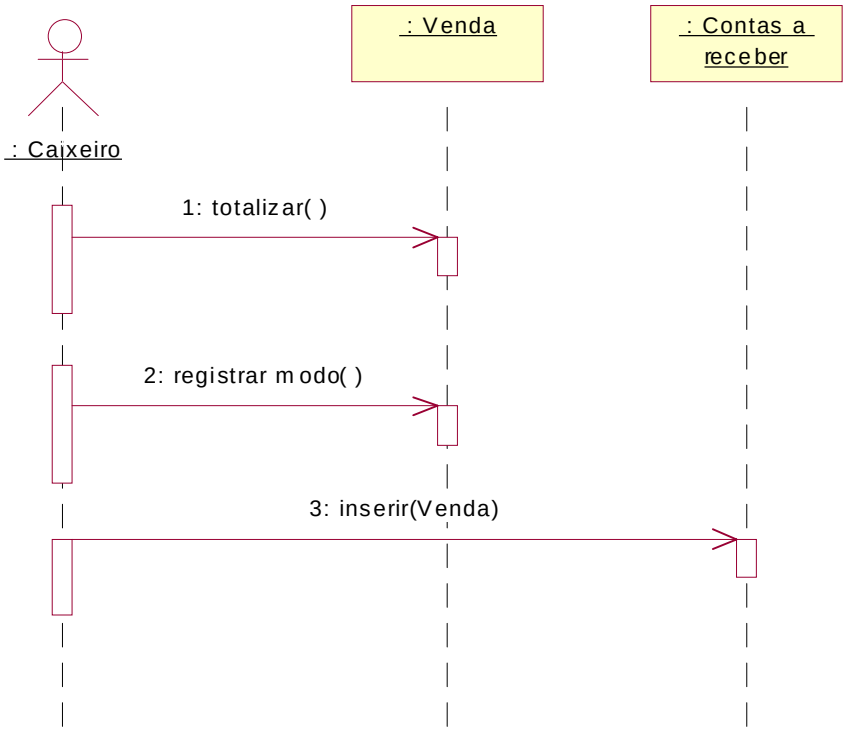


*Figura 42 – Exemplo de roteiro alternativo: Operação de Venda a Prazo*

Subfluxos curtos podem ser inseridos em um roteiro, indicando-se a lógica de ativação deles por meio de **expressões de recorrência**:

- [condição] - lógica de seleção;
- *[condição] - lógica de iteração.

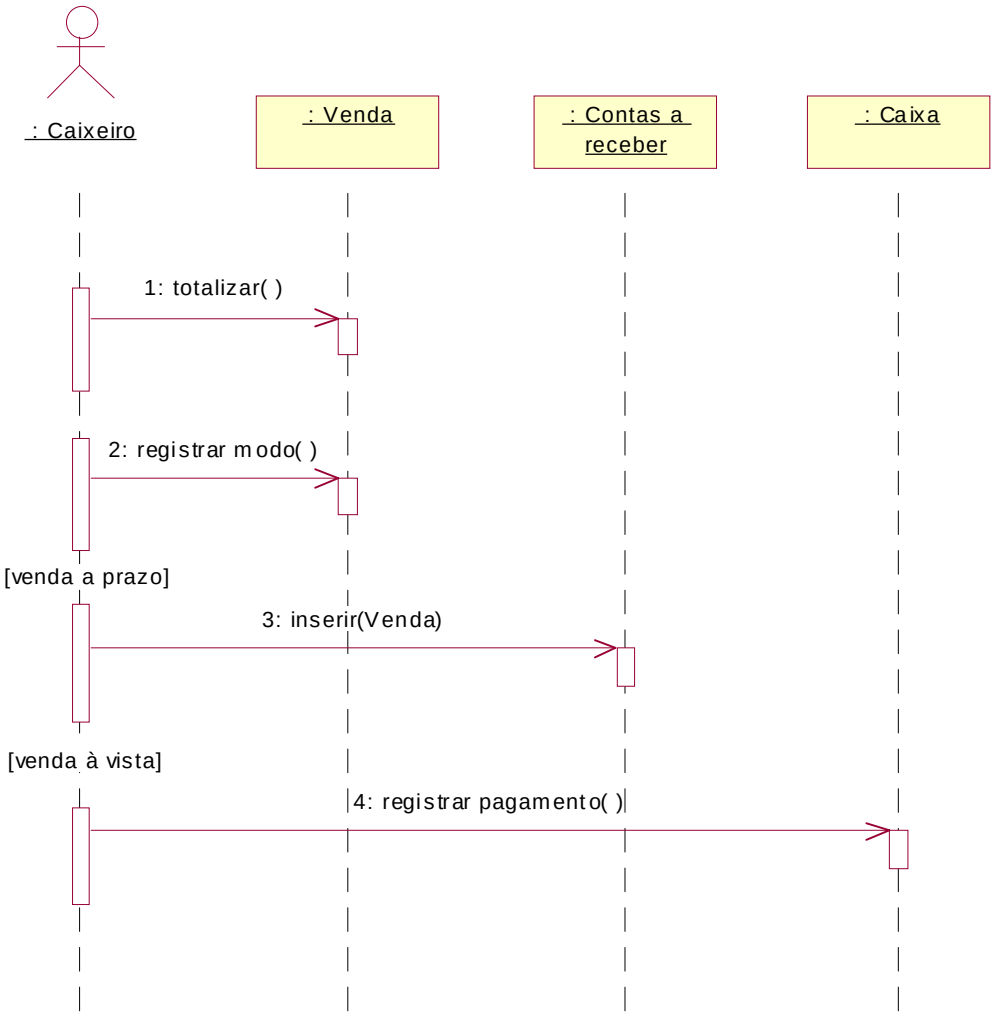


*Figura 43 – Representação de lógica por meio de restrições*

Diagramas de seqüência podem ser complementados por meio de anotações. Os detalhes de processamento que não correspondem a interações entre classes foram lançados como anotações no diagrama da Figura 44. Estas anotações podem servir como especificações das operações das classes envolvidas.

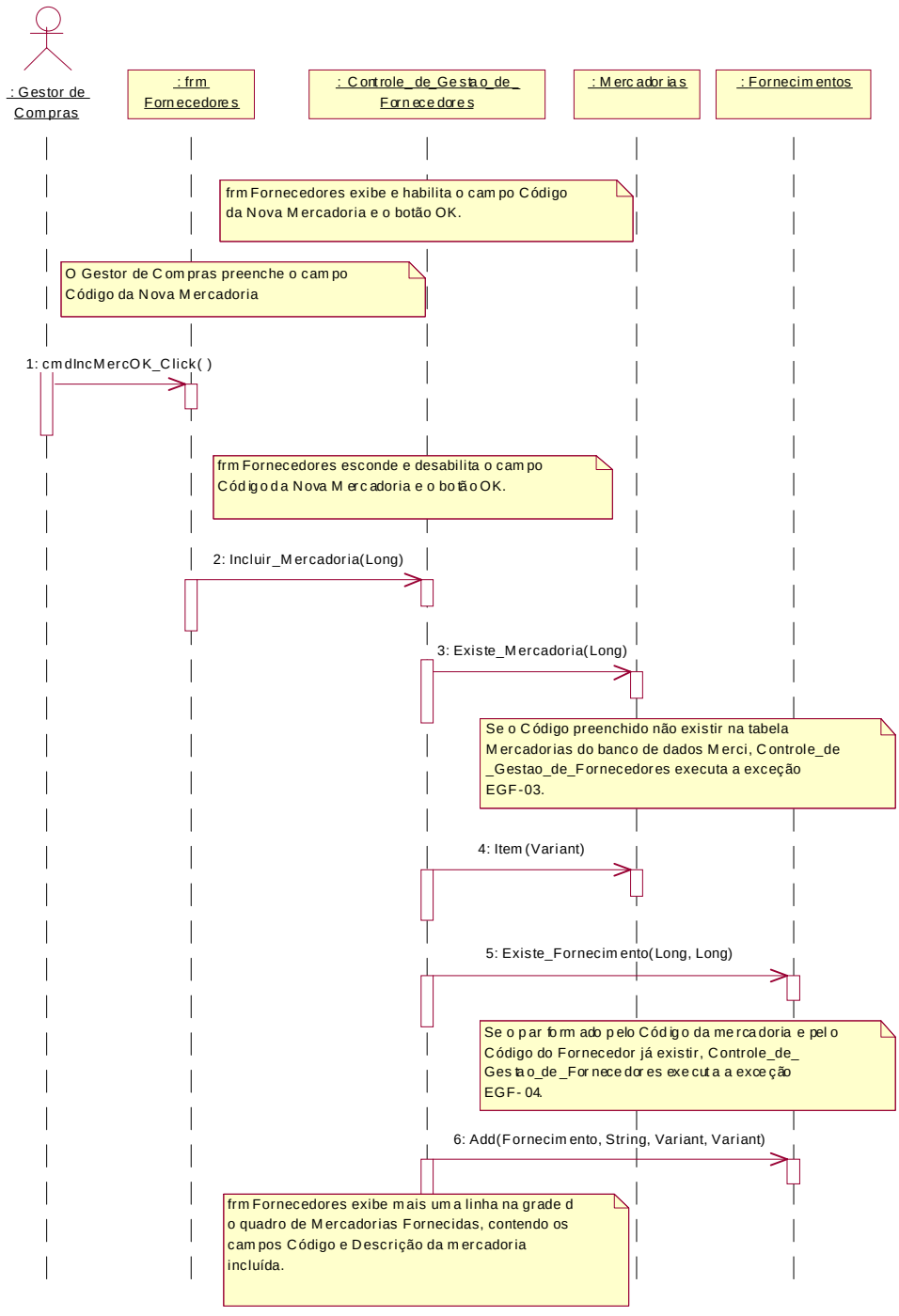


*Figura 44 – Exemplo de diagrama de seqüência para realização de um caso de uso*

### *3.5.2 Diagramas de colaboração*

Os diagramas de colaboração enfatizam os relacionamentos entre os objetos participantes. Eles são construídos de acordo com as seguintes convenções:

- nodos representam os objetos;

- os arcos representam as mensagens passadas entre os objetos;
- os rótulos dos arcos são os nomes das operações;
- os números de seqüência mostram o ordenamento relativo das mensagens;
- as anotações podem complementar o diagrama.

Nos diagramas de colaboração a ordenação das mensagens é mostrada apenas pela numeração delas. Na Figura 46 mostra-se o diagrama de colaboração que corresponde ao diagrama de seqüência da Figura 45.

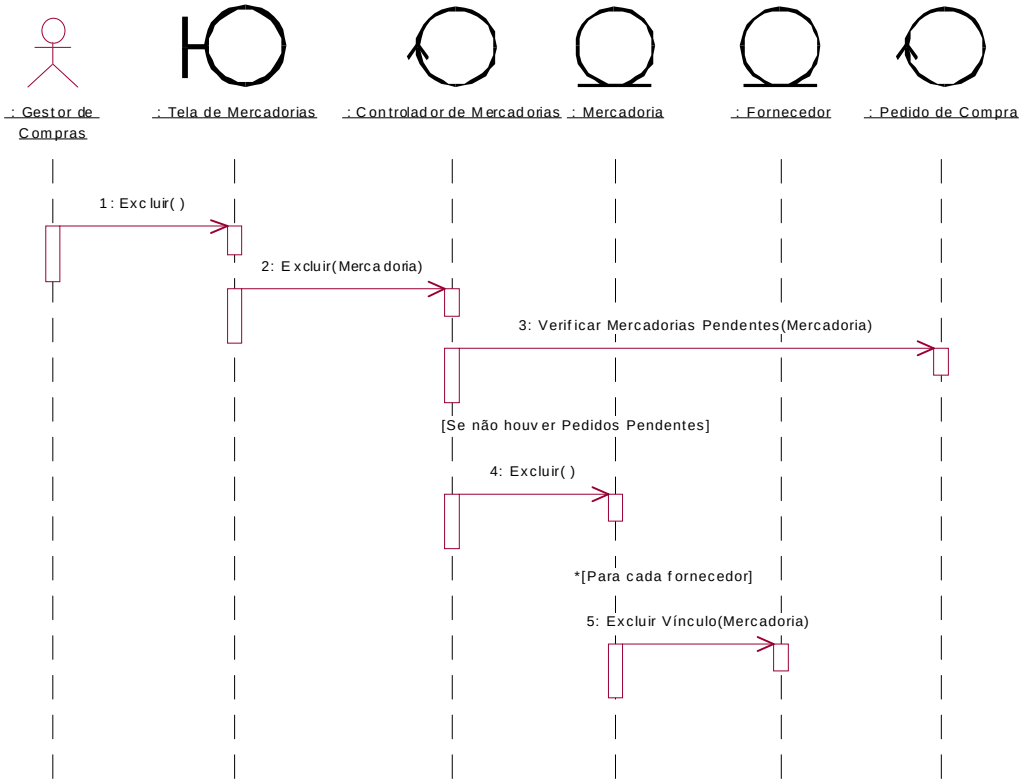


*Figura 45 - Realização de fluxo através de diagrama de seqüência*

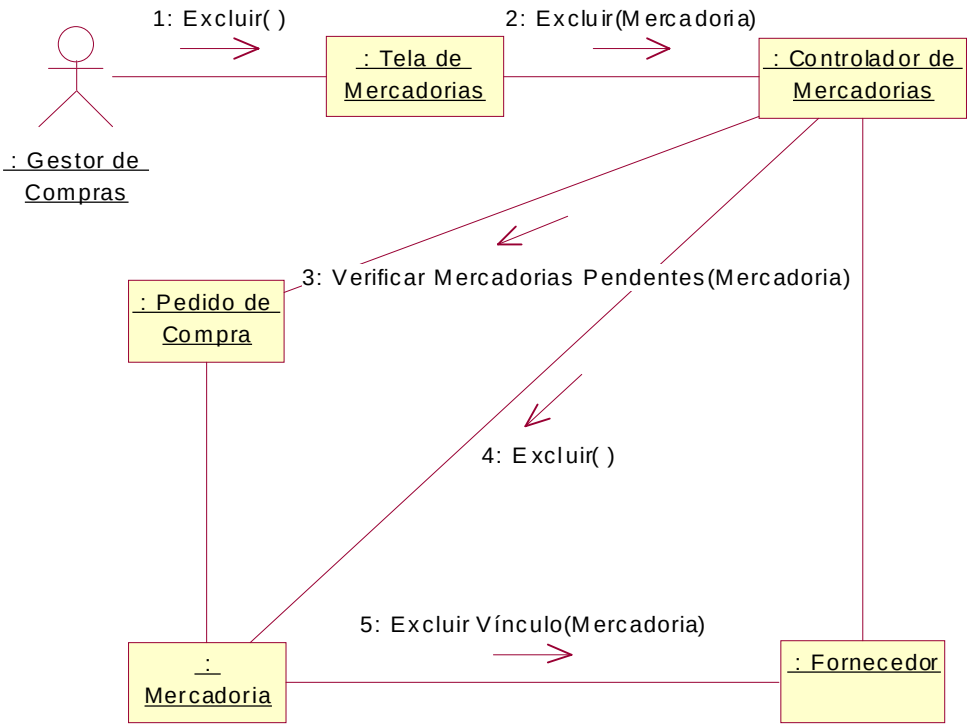


*Figura 46 - Realização de fluxo através de diagrama de colaboração*

### *3.5.3 Diagramas de objetos*

Diagramas de colaboração mostram interações entre objetos, que geralmente têm o objetivo de cooperar para realizar roteiro de um caso de uso. Diagramas de objetos podem ser utilizados também para outras finalidades. Na Figura 47, um diagrama de objetos ilustra a integração *bottom-up*. No exemplo, as unidades U21 e U22 são testadas utilizando-se os controladores C1 e C2. No passo seguinte da integração, os controladores são substituídos pela unidade U1. "Substituído por" é uma ligação entre dois objetos.

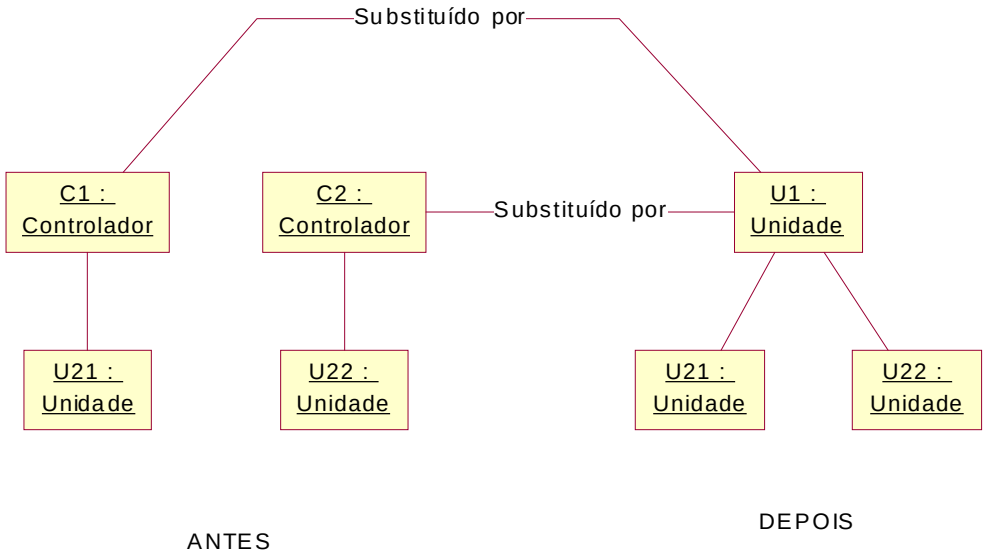


*Figura 47 – Exemplo diagrama de objetos utilizado para ilustrar testes*

Os diagramas de objetos podem ser usados para mostrar a passagem de dados e de controle. Na Figura 48, por exemplo, mostram-se como objetos uma unidade sob teste e os componentes utilizados para testa-la. As setas simples representam a direção de fluxo de controle: do objeto que chama (cliente do serviço ou emissor da mensagem) para o objeto chamado (fornecedor do serviço ou receptor da mensagem). As setas com bolinhas indicam o fluxo de dados (direção de passagem de parâmetros).

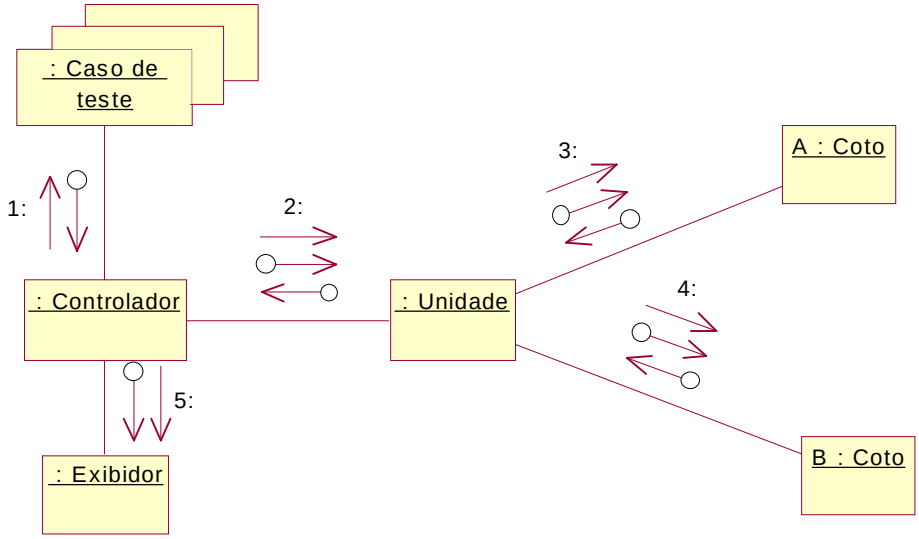


*Figura 48 - Componentes de teste*

# 4 Modelagem física

## 4.1 Visão de componentes

A visão de componentes descreve os aspectos estáticos do modelo físico. Ela mostra os arquivos de código fonte e objeto utilizados na implementação, assim como componentes reutilizados do ambiente e de outros projetos. Mostra também as dependências existentes entre eles, que indicam o impacto das alterações realizadas em cada componente. Na Figura 49, mostra-se que o componente executável "Merci_10" depende de vários componentes de código fonte correspondentes a formulários do Visual Basic e de um componente de código binário da tecnologia COM (um controle ActiveX de grade de entrada).

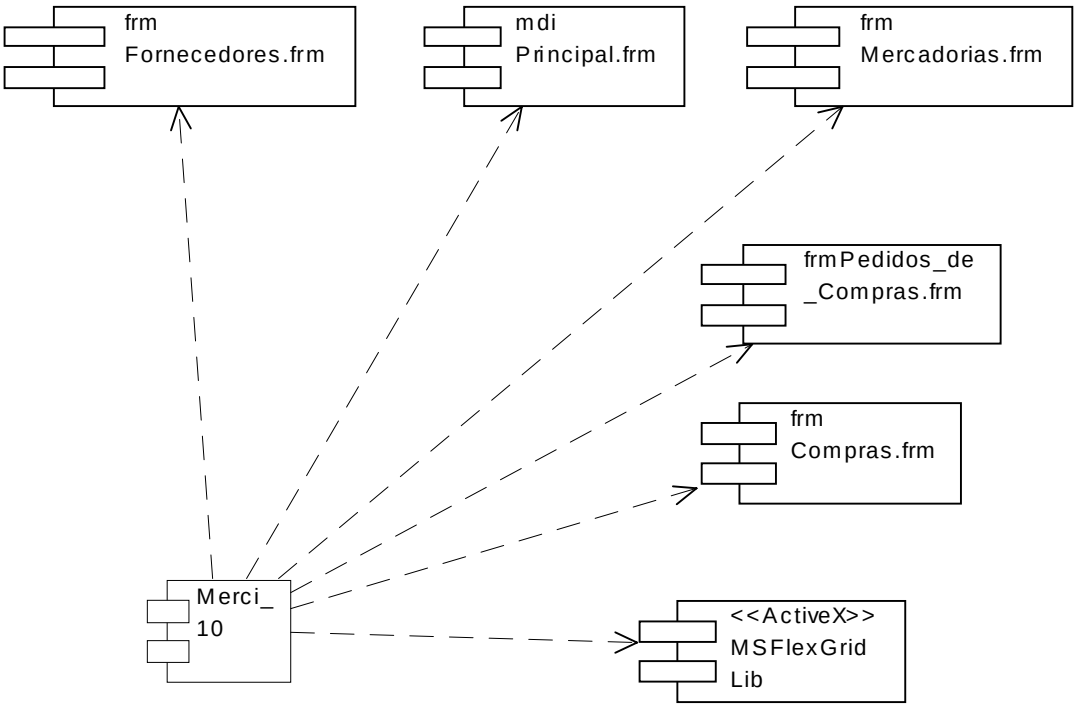


*Figura 49 – Exemplo de diagrama de componentes (visão estática)*

Os componentes podem ser implementados por meio de muitos tipos diferentes de arquivos. Nos diagramas de componentes, esses tipos de arquivos podem ser destacados por meio de estereótipos, representados por meio de identificadores ou de ícones próprios. A Figura 50 mostra alguns tipos de arquivos fontes, com os respectivos estereótipos icônicos. A Figura 51 mostra alguns exemplos de componentes de código objeto comuns em ambientes Windows, mostrados com estereótipos textuais.

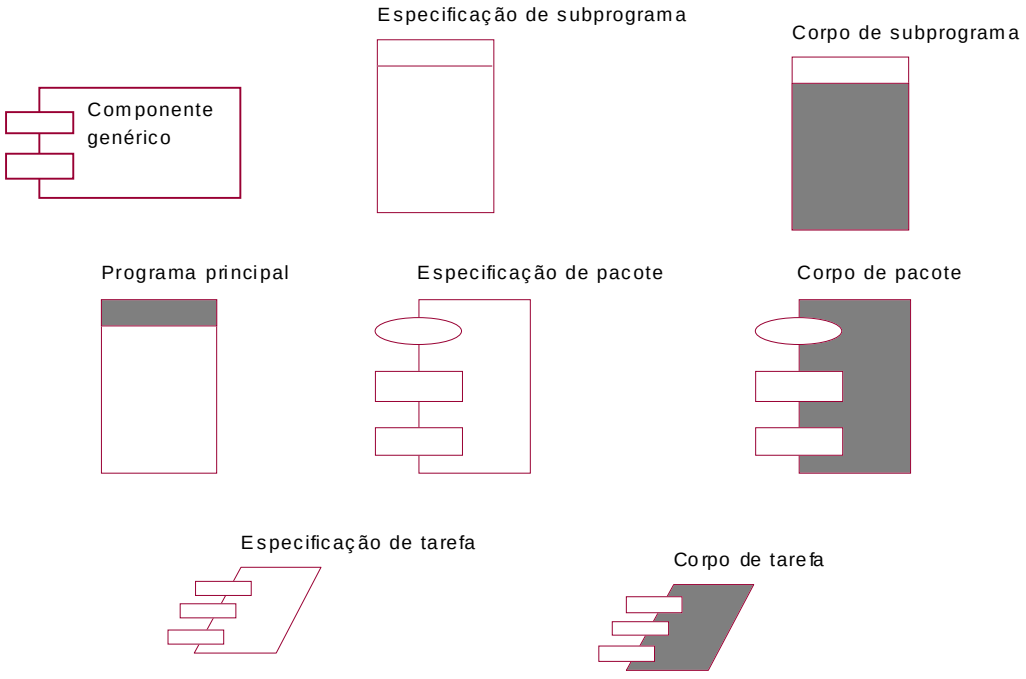


*Figura 50 – Exemplo de componentes de código fonte*

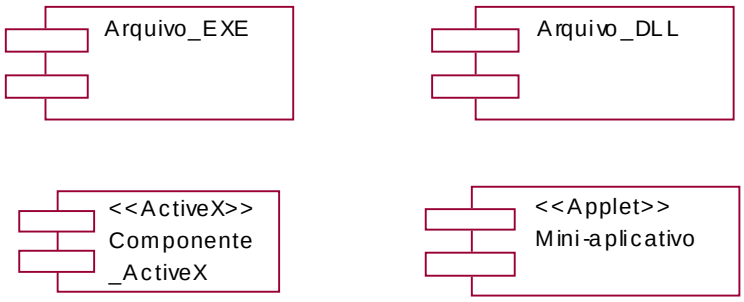


*Figura 51 – Exemplo de componentes de código objeto*

As interfaces representam um subconjunto dos recursos implementados por um componente, que têm um tema comum. Por exemplo, na Figura 52, um componente ActiveX oferece duas interfaces distintas. Aplicativos que não tratem de vendas a prazo, por exemplo, possivelmente não utilizarão a interface `Gestao_de_Clientes`, utilizando apenas os recursos oferecidos através da interface `Operacao_de_Venda`.

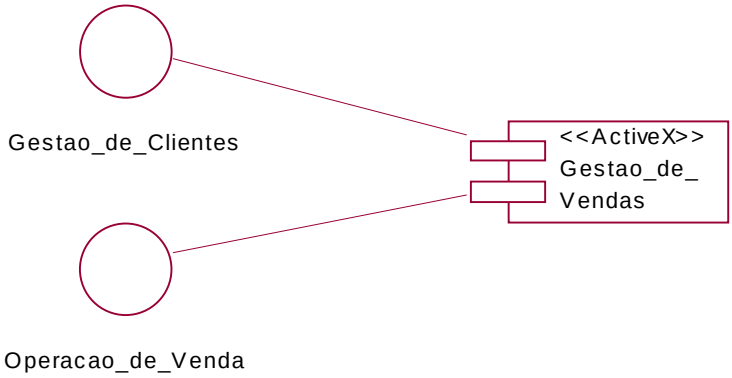


*Figura 52 – Exemplo de interfaces de componente*

## 4.2 Visão de desdobramento

O modelo físico dinâmico é descrito pelo diagrama de desdobramento, que mostra os processadores, os dispositivos e as conexões utilizados na implementação de um sistema. Em cada processador, pode-se mostrar os processos que nele executarão. Esse modelo é particularmente importante no caso de sistemas distribuídos. É comum a utilização de estereótipos para indicar diferentes tipos de processadores e dispositivos.

Na Figura 53 mostra-se que os processadores “Cliente de WWW 1”, “Cliente de WWW 2” e “Servidor de bancos de dados” são ligados ao processador “Servidor de WWW” através de conexões “Internet”. O “Servidor de WWW” executa os processos “Personal Web Server” e “Aplicativo CGI”. O “Cliente de WWW 1” está ligado a um dispositivo “Impressora”.

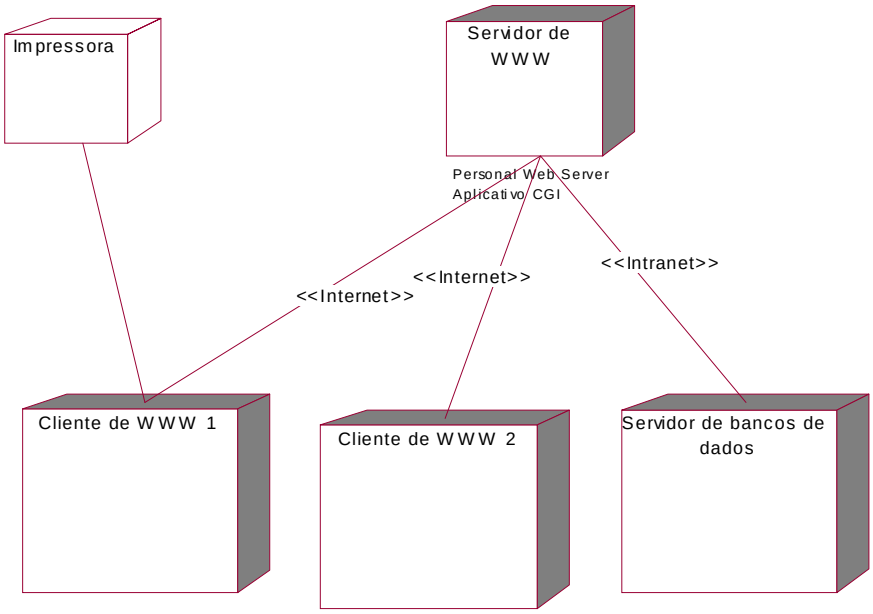


*Figura 53 - Exemplo de diagrama de desdobramento*

# 5 Referências


[Jacobson+99] Ivar Jacobson, James Rumbaugh e Grady Booch. *Unified Software Development Process.* Addison-Wesley, Reading - MA, 1999.

[Jacobson94] Ivar Jacobson. *Object-Oriented Software Engineering.* Addison-Wesley, Reading - MA, 1994.

[Rumbaugh+99] James Rumbaugh, Ivar Jacobson e Grady Booch. *Unified Modeling Language Reference Manual.* Addison-Wesley, Reading - MA, 1999.