



PROJETO DE SOFTWARE COM UML 2.0

Rodrigo Yoshima

Curso UML & Unified Process - www.aspercom.com.br

Aspercom Serviços de Informática Ltda.
CNPJ 02.942.579/00001-37
Autor: Rodrigo Yoshima

Projeto de Software com UML 2.0
Outubro de 2005

Copyright 2005, Aspercom. Todos os direitos reservados.
<http://www.aspercom.com.br>

O conteúdo deste texto é de propriedade da Aspercom Serviços de Informática Ltda.
Qualquer reprodução total ou parcial deste conteúdo é proibida.

Curso UML & Unified Process - www.aspercom.com.br

Índice

Introdução	8
1. Modelando o escopo do sistema com Casos de Uso	9
1.1. Para quê Casos de Uso?	9
1.2. Diagrama de Casos de Uso (ud).....	10
1.3. Atores	10
1.4. Caso de Uso.....	12
1.5. Narrativa do Caso de Uso.....	13
1.6. Relacionamento «include» entre Casos de Uso.....	15
1.7. Fluxos Alternativos.....	17
1.8. O que são cenários?.....	22
1.9. Precondição e Pós-condições.....	23
1.10. Relacionamento «extend» entre Casos de Uso.....	24
1.11. Tópicos não abordados sobre Casos de Uso	27
2. Modelando o fluxo de informações com Diagramas de Atividade.....	29
2.1. Fluxos de informações e modelo de processos	29
2.2. InitialNode, ControlFlow, Activity, Action e ActivityFinalNode.....	30
2.3. DecisionNode e MergeNode	31
2.4. ForkNode e JoinNode: tarefas sendo executadas no mesmo tempo	34
2.5. Passando Objetos como parâmetros	35
2.6. ActivityPartition: organizando tarefas em seus responsáveis	36
2.7. Tópicos não abordados sobre o Diagrama de Atividades	37
3. Conceitos de Programação e Análise Orientada a Objetos	39
3.1. Classes e Objetos.....	39
3.2. Associações	40
3.3. Operações	40
3.4. Encapsulamento	41
3.5. Herança.....	43
3.6. Polimorfismo.....	45
3.7. Interfaces e Classes Abstratas	45
4. Diagrama de Classes, Objetos e Pacotes	48
4.1. Comment.....	49
4.2. Compartimento do Nome e uso de Stereotypes na Classe.....	49

4.3.	Visibilidade	53
4.4.	Multiplicidade	54
4.5.	Compartimento de Atributos	57
4.6.	Compartimento de Operações	59
4.7.	Herança	62
4.8.	Interfaces	63
4.9.	Associações	64
4.10.	Dependencies	71
4.11.	Pacotes (packages)	73
4.12.	Tópicos não abordados sobre Classes	74
5.	Interação entre Objetos	76
5.1.	Realização de Casos de Uso	76
5.2.	Diagrama de Seqüência	77
5.3.	Diagrama de Visão Geral da Interação	84
5.4.	Diagrama de Comunicação	85
5.5.	Tópicos não abordados sobre Interações	86
6.	O Ciclo de Vida do Objeto (Diagrama de Máquina de Estado)	87
6.1.	Uso da Máquina de Estados	87
6.2.	Estado Inicial e Final	88
6.3.	Transições	89
6.4.	Compartimento de Transições Internas	90
6.5.	Estados Compostos	91
6.6.	Pseudo-estados	91
6.7.	Tópicos não abordados sobre a Máquina de Estados	93
7.	Modelando a Arquitetura do Sistema	94
7.1.	Diagrama de Componentes (id)	94
7.2.	Diagrama de Implantação/Deployment (dd)	97
7.3.	Tópicos não abordados sobre Diagramas de Arquitetura	99

Índice de figuras

Figura 1 Elementos de um diagrama de Casos de Uso	10
Figura 2 Papéis x Usuários	11
Figura 3 Generalização de Atores.....	11
Figura 4 Caso de Uso = Objetivo do Ator.....	12
Figura 5 Todos os Casos de Uso = Escopo.....	12
Figura 6 Notação do Relacionamento «include» entre Casos de Uso.....	15
Figura 7 Fluxos de um Caso de Uso.....	21
Figura 8 Relacionamento <<extend>> entre Casos de Uso.....	25
Figura 9 Diagrama de Atividades (ad) do workflow da empresa	29
Figura 10 Activities e Actions	30
Figura 11 DecisionNode	31
Figura 12 MergeNode	32
Figura 13 Decision e Merge unidos.....	32
Figura 14 DecisionNode x Setas.....	33
Figura 15 ForkNode e JoinNode	34
Figura 16 ObjectFlow.....	35
Figura 17 InputPin e OutputPin.....	35
Figura 18 ActivityPartition tarefas e responsabilidades	36
Figura 19 Uma Classe e um Objeto.....	39
Figura 20 Associação	40
Figura 21 Visibilidade public(+) e private(-).....	41
Figura 22 Herança	43
Figura 23 Heranças de produtos de uma loja	43
Figura 24 Extensão do PedidoVenda.....	44
Figura 25 Interface Document e sua implementação	46
Figura 26 Integração com Sistema ERP	46
Figura 27 Interface de integração com o Sistema ERP.....	47
Figura 28 Classe Abstrata	47
Figura 29 Diagrama de Classes.....	48
Figura 30 Diagrama de Objetos	48
Figura 31 Anotação (Comment) da UML.....	49
Figura 32 Compartimentos básicos da Classe.....	49

Figura 33 Classe com stereotype (estereótipo).....	50
Figura 34 Boundary, control e entity	50
Figura 35 Stereotypes com notações de ícones padrão da UML	51
Figura 36 Interface, Proxy e Implementação.....	51
Figura 37 Stereotype definido no projeto	52
Figura 38 Classe Abstract.....	52
Figura 39 Visibilidade.....	53
Figura 40 Multiplicidade atributo e associação.....	54
Figura 41 Multiplicidade {ordered} no atributo.....	56
Figura 42 Multiplicidade {unique} na extremidade da associação	56
Figura 43 Atributos: identificação, descrição e relacionamento.....	57
Figura 44 Atributo static.....	58
Figura 45 Constante = atributo static + readOnly.....	58
Figura 46 Sobrecarga da operação “cancelar()”.....	59
Figura 47 Operação “alterar” com dois parâmetros.....	60
Figura 48 Operação static.....	60
Figura 49 Operação “faturar()” abstrata	61
Figura 50 Notação da Herança	62
Figura 51 Herança no estilo árvore.....	62
Figura 52 Notação comum de Interface	63
Figura 53 Interface exigida e interface fornecida.....	63
Figura 54 Notação circular da interface	64
Figura 55 Notação da associação simples.....	64
Figura 56 Indicação da memberEnd	65
Figura 57 Papéis (roles) na associação	65
Figura 58 Multiplicidade na associação	66
Figura 59 Qualificador de acesso na associação.....	67
Figura 60 Navegabilidade da associação	68
Figura 61 Associação reflexiva	68
Figura 62 Classe de associação.....	69
Figura 63 Promoção da Classe de Associação.....	69
Figura 64 Notação da Agregação	70
Figura 65 Notação da Composição.....	70
Figura 66 Dependência de “Produto” para com “Aliquotaimposto”.....	71
Figura 67 Dependency com stereotype «refine».....	71
Figura 68 Diagrama de Pacotes	73

Figura 69 Merge de packages	74
Figura 70 Diagrama de Seqüência.....	77
Figura 71 Diagrama de Seqüência Macro (Atores e Sistema).....	77
Figura 72 Linhas de Vida (lifelines) do Diagrama de Seqüência	78
Figura 73 Mensagens Síncronas e Assíncronas	80
Figura 74 Expressão de Guarda	80
Figura 75 Quadro (diagrama).....	81
Figura 76 InteractionOccurrence "obterPedido"	81
Figura 77 Fragmento combinado "opt" na interação	82
Figura 78 Operador alt.....	82
Figura 79 Operador loop.....	83
Figura 80 Diagrama de Visão Geral da Interação	84
Figura 81 Diagrama de Comunicação.....	85
Figura 82 Diagrama de Máquina de Estados (sm).....	87
Figura 83 Marcação Estado Inicial e Final	88
Figura 84 Compartimento de Transições Internas	90
Figura 85 Transição sem Evento/Operação.....	90
Figura 86 Estado Composto	91
Figura 87 Pseudo-estado Choice.....	91
Figura 88 Pseudo-estado join	92
Figura 89 Pseudo-estado terminate.....	92
Figura 90 Componente e Artefato.....	94
Figura 91 Componente Estereotipado	95
Figura 92 Dependência de Componentes em Classes	95
Figura 93 Dependência entre Componentes.....	96
Figura 94 Dependência de Artefato e Componente	96
Figura 95 Interfaces do Componente.....	96
Figura 96 Diagrama de Implantação (Deployment).....	97
Figura 97 Instâncias de Nodes de Implantação	98
Figura 98 Node com atributos e operações	98

Introdução

A Unified Modeling Language (UML) é a linguagem mais utilizada atualmente para especificação e projeto de software na abordagem orientada a objetos. A UML é o instrumento que permite a modelagem do software “visualmente”, tornando fácil partir dos requisitos do sistema à implementação de uma forma amigável.

A UML está formalmente em desenvolvimento desde 1994, porém, a UML não é uma idéia que partiu do zero, a UML é uma consolidação de renomadas técnicas retiradas de metodologias já existentes e bastante praticadas até então.

O OMG (Object Management Group, www.omg.org) que também é responsável pelos padrões CORBA, IDL e IIOP, administra o padrão UML. A função do OMG é organizar e divulgar as especificações da linguagem, controlando as solicitações do que deve ou não ser incluído na arquitetura da linguagem. O OMG é uma organização mantida por diversas empresas de renome internacional.

Outra função do OMG é o direcionamento da UML para um plano muito maior e ambicioso chamado MDA (Model Driven Architecture). A MDA é uma arquitetura baseada em modelos que padronizada a forma como um sistema é desenhado, visando que este projeto seja independente de tecnologia ou plataforma. É garantida uma separação entre modelo (negócio) e a implementação (tecnologia), garantindo assim que com a evolução da tecnologia o modelo ainda seja válido. A MDA também é administrada pelo OMG.

A UML abrange todas as etapas da produção de software, mas principalmente é utilizada para traduzir os requerimentos do sistema (em alto nível e mais próximos do usuário) em componentes codificáveis (mais próximos da aplicação). Mesmo estando entre essas duas camadas, a UML pretende ser fácil de entender para todos os envolvidos. A UML é uma **linguagem**, e como tal, é um meio de comunicação. Através de diagramas gráficos é mais fácil discutir e visualizar as idéias e soluções entre a equipe, ou com o usuário. Muito mais simples do que com programas em código.

A quem se destina este livro

Programadores, Analistas, Designers e Arquitetos de Sistemas.

Como este livro está organizado

Este livro dá enfoque ao uso mais geral da UML, isto é, as técnicas e notações que são mais práticas, e que abrangem 95% das necessidades que você terá como analista. A UML 2.0 inteira é muito abrangente e muitas de suas definições estão fora do uso comum.

Os assuntos são apresentados por capítulo, iniciando nos requisitos do sistema até a forma de distribuição. Cada capítulo possui uma parte final chamada “Tópicos não abordados” que cita todos os detalhes que o capítulo não cobriu (os outros 5%).

Não são feitas citações ou comparações a respeito da UML 1.4. Este livro somente cita algumas melhorias da versão 2.0 sobre a 1.4, mas não são explicadas com detalhes as diferenças entre as versões.

Este livro também não detalha aspectos mais baixos da linguagem com relação à **especificação** (superstructure) da UML 2.0 como *metaclasses* ou a *MOF* (meta linguagem que define a UML).

1. Modelando o escopo do sistema com Casos de Uso

1.1. Para quê Casos de Uso?

Quem está iniciando projetos usando objetos e UML pode questionar sobre a importância de Casos de Uso bem modelados e detalhados, talvez por não entender direito por que e para quê definir bonequinhos e elipses com nomes de funcionalidades e o que isso adiciona ao projeto.

Olhando para um diagrama de Casos de Uso, pela sua simplicidade, um analista poderá observar rapidamente as funcionalidades envolvidas no sistema, os usuários envolvidos e integrações com sistemas externos. O propósito maior do Caso de Uso é fornecer uma descrição do comportamento do sistema do ponto de vista do usuário.

O Caso de Uso é uma adição muito importante para a abordagem da Análise Orientada a Objetos e para o Processo Iterativo de Engenharia de Software. A modelagem do diagrama de Casos de Uso é simples e muitos autores ensinam sobre as mais variadas técnicas para descrever em texto um Caso de Uso (destacando o ótimo livro de Alistair Cockburn “Writing Effective Use Cases”). Modelando ou escrevendo, o principal para entender a importância do Caso de Uso são os objetivos dele:

- **Definir Escopo:** Um conjunto de Casos de Uso define o escopo do sistema de uma maneira simples. Se no diagrama aparece um Caso de uso chamado “Plantar Batata”, os usuários não poderão dizer que “plantar couve” está no escopo do sistema.
- **Organizar e dividir o trabalho:** O Caso de Uso é uma importante unidade de organização do trabalho dentro do projeto, geralmente nas equipes de projeto é comum ouvir que “o Zé está trabalhando no Caso de Uso X e o João está com o Caso de Uso Y”. A unidade do Caso de Uso divide o trabalho da equipe entre as pessoas, fora isso, é comum dizer que o Caso de Uso está em Análise, em Programação ou em Teste. Casos de Uso também são entregues separadamente aos usuários em conjuntos divididos em fases ou iterações no projeto. Então, dizemos que a primeira iteração (ou entrega) terá os Casos de Uso X, Y, Z e W e na segunda iteração terá os Casos de Uso T, H, I e J.
- **Estimar o tamanho do projeto:** O Caso de Uso fornece métricas para definir o tempo de desenvolvimento. Uma das métricas que pode ser aplicada sobre Casos de Uso é a UCP (Use Case Point) que consiste em classificar os Casos de Uso em nível de complexidade e somando todos os Casos de Uso do projeto (e mais alguns fatores) você consegue estimar o esforço do projeto em horas. Além do UCP, podem ser aplicadas as técnicas de Pontos de Função (Function Points) que são mais padronizadas e completas.
- **Direcionar os testes:** Os testes do sistema (essencialmente os funcionais que são os mais importantes) são derivados do Caso de Uso. Essa característica é uma das mais importantes e geralmente é desprezada, pois com Casos de Uso, os testes são planejados no **início** do projeto e não no **fim**, diminuindo os riscos. A partir dos Casos de Uso, Casos de Teste são criados para validar o funcionamento do software.

Uma das questões em aberto sobre os Casos de Uso é a confusão que fazem com o diagrama e a narrativa (texto) do Caso de Uso. Isso porque a **UML define somente como deve ser o Diagrama de Casos de Uso, e não a narrativa**. Desse modo, não há um consenso geral sobre como descrever a narrativa, existem muitas técnicas e é difícil julgar que uma técnica é certa e a outra errada, depende muito do projeto, dos seus processos e ferramentas que você tem à disposição.

1.2. Diagrama de Casos de Uso (ud)

Bom, antes de começar a discutir sobre como definir uma narrativa onde não há consenso, vamos nos focar no diagrama que a UML especifica direitinho as regras de como ele deve ser:

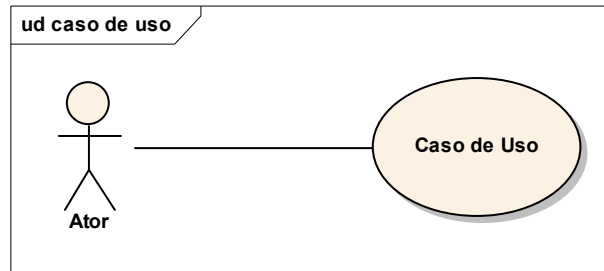


Figura 1 Elementos de um diagrama de Casos de Uso

A notação básica do diagrama são Atores representados pelos bonequinhos. Uma linha conecta atores aos Casos de Uso **informando que o sistema permitirá ao Ator usar o Caso de Uso diretamente**. Os Casos de Uso são representados por elipses (existem notações alternativas para os elementos, mas essas são as mais usuais).

1.3. Atores

O Ator, em um diagrama de Casos de Uso (ud), também pode ser confuso para os novatos na abordagem, mas para todo e qualquer efeito é bom sempre ter em mente que um Ator é um **PAPEL DESEMPENHADO POR ALGUMA COISA EXTERNA** ao sistema (não necessariamente uma pessoa). Até mesmo o tempo pode ser um Ator para tarefas que ocorrem com frequência temporal.

Essa definição ainda aparenta não ser tão simples, vou demonstrar os erros mais comuns a respeito dos Atores:

- **Componentes internos do sistema não são Atores:** É comum o analista imaginar que porque sistemas externos podem ser atores, o banco de dados da aplicação que está sendo desenvolvida deve ser um ator também, mas isso é errado, lembre que Ator é um papel de responsabilidade externa ao sistema. O que deve funcionar dentro da responsabilidade do sistema não pode ser extraído como um Ator.

- **Hardware internos do sistema não são Atores:** Você pode imaginar que um servidor é externo ao sistema, ou uma impressora em um Caso de Uso que precise imprimir alguma coisa é externa ao sistema também, porém, esses itens externos ao sistema são componentes que o software pressupõe que existem e cumprirão seu papel. Então, são como os componentes do funcionamento interno do software (como o banco de dados da aplicação) e assim sendo, não são Atores. **Alguns** tipos de hardware podem ser Atores, mas somente quando for importante para a análise do sistema destacar a presença desse hardware. Por exemplo, softwares para controles industriais podem ter alguns sensores que indicam algum tipo de problema na linha de produção e disparam algum comportamento (Caso de Uso) no sistema. Este sim pode ser um hardware que é um Ator no sistema, pois é importante destacar isso para o funcionamento do Caso de Uso. Outro hardware que pode ser um Ator seria a catraca do controle de ponto (entrada e saída de funcionários) para um sistema de Administração de Pessoal. Ela desempenha um papel no sistema, seria um Ator.

• **A definição de Atores (e Casos de Uso) não é o controle de acessos da aplicação:** Esse erro também é comum: confundir a definição de Atores com o que os usuários podem ou não fazer dentro do sistema. Não é essa a idéia! A representação do Ator somente destaca o papel que um usuário assume ao usar o Caso de Uso.

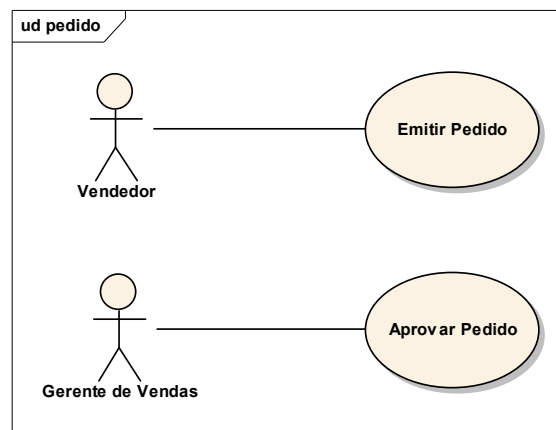


Figura 2 Papéis x Usuários

Neste exemplo, o diagrama de Casos de Uso somente diz que para “Emitir Pedido” o usuário deverá assumir o papel de “Vendedor”, isso não implica que um usuário que possui o cargo de vendedor na organização não possa “Aprovar Pedido”.

O que consideramos mais importante na definição de Atores é dar nome aos papéis que pessoas assumirão ao usar o sistema e informar integrações com sistemas externos. Enfim, em 90% dos casos um Ator será uma pessoa (usuário do sistema) ou um sistema externo. Isso serve para dar clareza a quem lê o diagrama deixando as coisas o mais simples possível.

Também é possível definir uma herança entre Atores para estabelecer generalizações de Atores. Veja o conceito geral de generalização no Capítulo 4 - Diagrama de Classes.

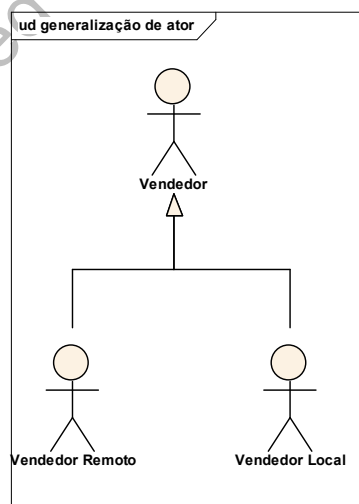


Figura 3 Generalização de Atores

A generalização de Atores serve para passar a idéia de algo em comum entre papéis no sistema. Assim, se um Caso de Uso requer um “Vendedor”, vendedores remotos e locais se encaixam no papel pela herança. Essa notação também não é muito comum, pois para pessoal não técnico, o conceito de generalização não é fácil de compreender.

1.4. Caso de Uso

A representação do Caso de Uso no Diagrama é simples: a elipse representa uma forma que o sistema se comporta do ponto de vista do Ator. O nome do Caso de Uso é uma forma de elucidar esse comportamento do sistema, assim sendo, **o nome do caso de uso define o OBJETIVO do Ator**, isto é, o que ele quer fazer no sistema.

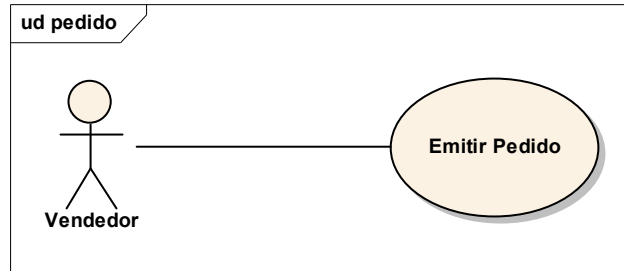


Figura 4 Caso de Uso = Objetivo do Ator

Todo o conjunto de Casos de Uso e Atores do sistema organiza o escopo do sistema a respeito dos objetivos que os usuários atingirão quando o sistema estiver pronto.

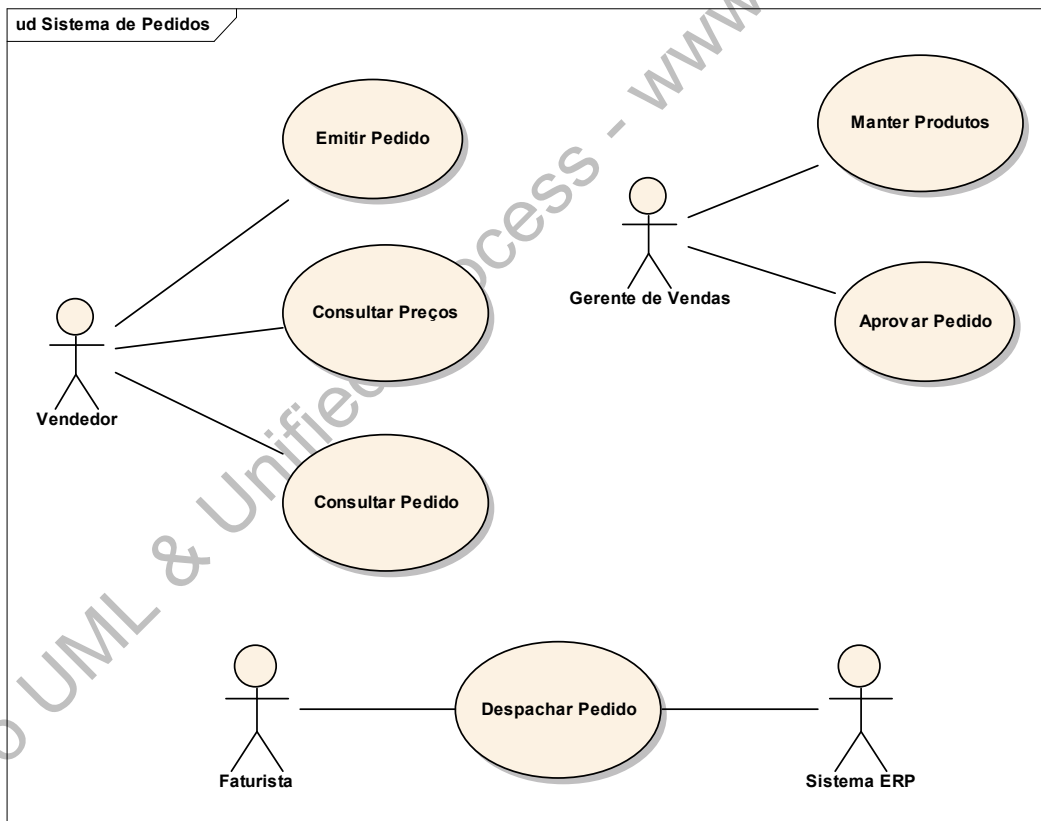


Figura 5 Todos os Casos de Uso = Escopo

Dar nomes aos Casos de Uso com o **objetivo do Ator** é a maneira de tornar o diagrama claro para o pessoal menos técnico saber exatamente o que será possível fazer no sistema. A narrativa (texto) do Caso de Uso deve ser consistente com esse objetivo definido pelo seu nome.

1.5. Narrativa do Caso de Uso

É importante citar que quando dizemos “Caso de Uso”, estamos mais preocupados com a narrativa do Caso de Uso do que com o desenho da elipse no diagrama, ou o próprio diagrama. Isso porque nos projetos, associamos o termo “Caso de Uso” ao texto, à narrativa, e não ao desenho. É a narrativa do Caso de Uso que é importante, é ela que permite os benefícios já citados, e não o desenho. O desenho serve mais para organizar os Casos de Uso e representar graficamente (que é mais amigável do que o texto) as funcionalidades do sistema.

CASO DE USO = DIAGRAMA + NARRATIVA

Como já disse, existem inúmeros formatos ou templates de preenchimento de uma narrativa de Caso de Uso. Meu objetivo aqui é apresentar o conceito que se aplica a maioria dos formatos com exemplos claros. Gostaria reforçar mais uma vez que a UML **não define** o modo como uma narrativa deve ser descrita. A UML se limita apenas à forma de diagramar o modelo de Casos de Uso (notação).

A narrativa do Caso de Uso é um texto passo a passo sobre as ações que o Ator pode tomar e como o Sistema responderá a esta ação. A narrativa vai então evoluindo, entre ações do Ator e as respostas do Sistema, para o objetivo do Ator, até este ser alcançado.

Um dos erros mais comuns que vemos nas narrativas de Casos de Uso é o analista imaginar o Caso de Uso como um programa e tratar a sua narrativa como um passo a passo sobre as tarefas internas do sistema. O analista também pode errar se quebrar o objetivo do Ator em diversos Casos de Uso menores já imaginando como o sistema será implementado (como a decomposição funcional). Lembre-se: nesse momento o seu foco deve ser o **objetivo do Ator**, e não como o sistema resolverá esse objetivo.

Como exemplo: o Caso de Uso “Emitir Pedido” envolve várias tarefas menores como selecionar produtos, escolher forma de pagamento, calcular descontos, escolher forma de entrega, porém, tudo isso são partes do objetivo maior que é emitir o pedido.

O Caso de Uso é um objetivo do Ator e não uma tarefa do sistema. Uma das formas de evitar essa proliferação de Casos de Uso no sistema é perguntar a si mesmo ao criar um Caso de Uso:

- *Se eu entregar esse Caso de Uso sozinho para os usuários do sistema, resolveria algum problema deles? Agregaria algum valor para os usuários? Com esse Caso de Uso o usuário conseguiria resolver algum problema que o sistema deve atender?*

Acho que essas perguntas são suficientes. Como exemplo, no caso acima, se o analista criar um Caso de Uso chamado “Escolher Forma de Pagamento” e entregar ele sozinho para os usuários teria algum valor? Claro que não. Pode ter certeza que os usuários diriam que não serve pra nada fora da funcionalidade “Emitir Pedido”.

IMPORTANTE: Na narrativa do Caso de Uso a **resposta do sistema** deve se limitar somente ao que o **Ator consegue ver**. Não é necessário se preocupar em como o sistema obteve ou calculou os dados. Limite-se a escrever o **que** o sistema responde e não **como** ele obtém a resposta.

Para exemplificar uma narrativa, vamos descrever o Caso de Uso “Emitir Pedido” em texto:

Caso de Uso: Emitir Pedido

Ator: Vendedor

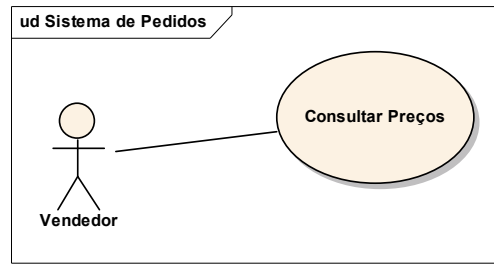
1. O Ator inicia o caso de uso selecionando “Emitir Pedido”;
2. O Sistema oferece a interface para emissão de pedidos;
3. O Ator seleciona um cliente para o pedido;
4. O Sistema exibe as informações do cliente;
5. O Ator seleciona um grupo de produtos;
6. O Sistema lista os subgrupos do grupo selecionado;
7. O Ator seleciona um subgrupo de produtos;
8. O Sistema apresenta os produtos do subgrupo selecionado;
9. O Ator seleciona os produtos desejados pelo cliente;
10. O Sistema calcula os preços e impostos dos produtos;
11. O Ator informa que deseja finalizar o pedido;
12. O Sistema questiona sobre a forma de pagamento e entrega;
13. O Ator seleciona a forma de pagamento e entrega;
14. O Sistema informa o adicional de juros, o frete e solicita uma confirmação de todos os dados do pedido;
15. O Ator confirma o pedido;
16. O Sistema informa que o pedido foi emitido com sucesso;

Note que o Caso de Uso não revela sobre como o sistema deverá resolver algumas questões difíceis como calcular preços e impostos. Pode ter certeza esses pontos envolvem regras complexas e cálculos com várias informações de diversas tabelas do sistema, porém, os detalhes de como serão resolvidos internamente **o Ator não consegue ver**. Nesse ponto do projeto nosso trabalho se limita ao que o sistema deve fazer (escopo), e não como ele irá fazer (implementação).

Essa regra de **escrever somente o que o usuário vê** é importante para agilizar a fase de análise do sistema e principalmente é esta regra que permite derivar os Casos de Teste a partir da narrativa, pois sabendo **o que** o sistema deve fazer é possível **planejar** como testar a funcionalidade independente de como ficará a implementação.

1.6. Relacionamento «include» entre Casos de Uso

Vamos descrever também o Caso de Uso “Consultar Preço”:



Caso de Uso: Consultar Preço

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Consultar Preço”;
2. O Sistema oferece a interface para consulta de preços;
3. O Ator seleciona um grupo de produtos;
4. O Sistema lista os subgrupos do grupo selecionado;
5. O Ator seleciona um subgrupo de produtos;
6. O Sistema apresenta os produtos do subgrupo selecionado;
7. O Ator seleciona os produtos;
8. O Sistema calcula os preços;

Observando o Caso de Uso “Emitir Pedido” descrito anteriormente, os passos para a seleção de produtos possuem um comportamento exatamente igual. Nesse caso, é possível extrair os passos iguais e criar um novo Caso de Uso separado com um relacionamento «include» entre eles:

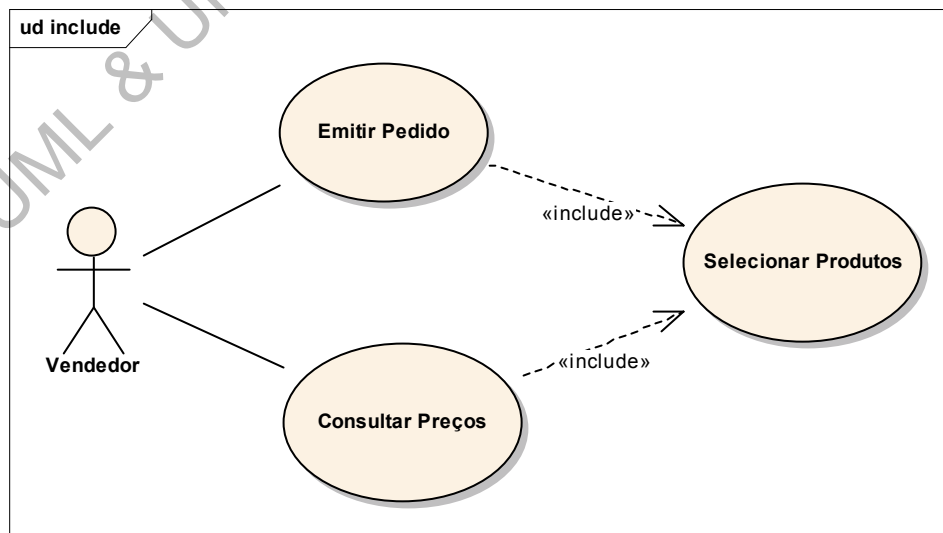


Figura 6 Notação do Relacionamento «include» entre Casos de Uso

Este novo Caso de Uso “Selecionar Produtos” seria assim:

Caso de Uso: Selecionar Produtos

1. O Ator seleciona um grupo de produtos;
2. O Sistema lista os subgrupos do grupo selecionado;
3. O Ator seleciona um subgrupo de produtos;
4. O Sistema apresenta os produtos do subgrupo selecionado;
5. O Ator seleciona os produtos;

É provável que você esteja questionando: “Isso não é quebrar o objetivo em Casos de Uso menores ou decomposição funcional?” Eu respondo que **não**. A técnica de extrair um Caso de Uso de dentro de outro só deve ser aplicada quando **visar reutilização do comportamento** que está sendo extraído. Como foi descrito no exemplo, o Caso de Uso “Consultar Preço” possuía um trecho de comportamento que já existia em “Emitir Pedido”, por esse motivo o Caso de Uso “Selecionar Produtos” foi extraído. Se não existisse “Consultar Preço”, o comportamento de selecionar produtos ainda ficaria dentro de “Emitir Pedido”.

Ou então você poderia perguntar: “Agregaria algum valor para o usuário entregar o Caso de Uso Selecionar Produtos isoladamente?” Observe melhor o diagrama da figura 6, o Caso de Uso “Selecionar Produtos” não está relacionado com nenhum Ator diretamente, assim sendo, ele não pode ser utilizado fora de “Emitir Pedido” ou “Consultar Preços”. Deste modo, ele não pode ser entregue sozinho para o usuário, o escopo não define que um Ator pode utilizá-lo diretamente.

Do mesmo modo, outro conceito que deve ser levado em consideração são as **dependências no modelo UML**. Exploraremos melhor este conceito no Capítulo 4 - Diagrama de Classes, adiantando, a linha tracejada que liga os Casos de Uso com o relacionamento «include» significa que o elemento que está lançando a seta depende do elemento que está sendo atingido pela seta. Assim, o Caso de Uso “Emitir Pedido” depende de “Selecionar Produtos”. “Emitir Pedido” só pode ser testado e entregue aos usuários **após ou juntamente com** “Selecionar Produtos”. Não é possível atingir o objetivo do Ator em “Emitir Pedido” sem “Selecionar Produtos”, o mesmo acontece com “Consultar Preços”.

Após a criação do relacionamento, o Caso de Uso “Consultar Preços” ficaria assim:

Caso de Uso: Consultar Preço

Versão: 1.1

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Consultar Preço”;
2. O Sistema oferece a interface para consulta de preços;
3. O Ator seleciona produtos; **Usa “Selecionar Produtos”**;
4. O Sistema calcula os preços;

A mesma alteração se aplica ao “Emitir Pedido”:

Caso de Uso: Emitir Pedido

Versão: 1.1

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Emitir Pedido”;
2. O Sistema oferece a interface para emissão de pedidos;
3. O Ator seleciona um cliente para o pedido;
4. O Sistema exibe as informações do cliente;
5. O Ator seleciona produtos; **Usa “Selecionar Produtos”**;
6. O Sistema calcula os preços e impostos dos produtos;
7. O Ator informa que deseja finalizar o pedido;
8. O Sistema questiona sobre a forma de pagamento e entrega;
9. O Ator seleciona a forma de pagamento e entrega;
10. O Sistema informa o adicional de juros, o frete e solicita uma confirmação de todos os dados do pedido;
11. O Ator confirma o pedido;
12. O Sistema informa que o pedido foi emitido com sucesso;

As alterações acima nos demonstram uma regra importante sobre o relacionamento «include»: o Caso de Uso que atira a seta passa o controle para o outro Caso de Uso mencionando isso da narrativa (destacado em negrito). Por outro lado, o Caso de Uso incluído “Selecionar Produto” não precisa se preocupar sobre quem o iniciou. O Caso de Uso incluído é independente de quem o está usando.

Como já foi citado, o Caso de Uso é um importante meio de dividir o sistema em fases entregáveis (iterações) visando disponibilizar o sistema em pedaços incrementais para os usuários, e não tudo no final do projeto. Este é um modo importante de reduzir o risco. Observar as dependências de Casos de Uso é um instrumento importante para definir o que entra em uma iteração.

1.7. Fluxos Alternativos

Até agora descrevemos narrativas somente com o que chamamos “fluxo básico”. O fluxo básico do Caso de Uso é a maneira mais comum que o Ator usará para atingir o seu objetivo, mas como a vida não é um mar de rosas na produção de software, pode ser que existam outros caminhos para atingir o mesmo objetivo, ou por alguma razão o objetivo não pode ser alcançado.

O sistema necessitará atender a outros fluxos que o Ator deseja para chegar lá. Tendo isso em vista, a técnica de Casos de Uso nos fornece a ferramenta do “fluxo alternativo”.

Vamos continuar nossa análise com o próximo Caso de Uso “Consultar Pedido”:

Caso de Uso: Consultar Pedido

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Consultar Pedido”;
2. O Sistema oferece a interface de consulta para pedidos;
3. O Ator informa o número do pedido desejado;
4. O Sistema exibe os dados do pedido;

Julgando que o Caso de Uso estaria finalizado, o analista foi checar se a sua narrativa estaria atendendo a todos os requisitos que o Caso de Uso deveria atender. Ele deixou escapar o seguinte parágrafo dos requisitos:

“O sistema deverá permitir consulta do pedido por número ou através de uma lista de pedidos por cliente, nesse último caso, a lista deverá ter todos os pedidos não faturados do cliente em ordem cronológica decrescente para seleção”.

A narrativa que ele descreveu não atende a este requisito. O usuário pode escolher entre o Caso de Uso do jeito que está e através de uma lista de pedidos do cliente. Usaremos um Fluxo Alternativo para atender essa necessidade:

Caso de Uso: Consultar Pedido

Versão: 1.1

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Consultar Pedido”;
2. O Sistema oferece a interface de consulta para pedidos;
3. O Ator informa o número do pedido desejado [A1];
4. O Sistema exibe os dados do pedido;

Fluxo Alternativo A1 – Consultar por Cliente

3. O Ator informa um cliente;
 - 3.1. O Sistema exibe uma lista de pedidos do cliente selecionado em ordem cronológica decrescente;
 - 3.2. O Ator seleciona um pedido do cliente; volta ao fluxo básico;

Nesse exemplo, o fluxo alternativo foi provocado por uma escolha do Ator no passo 3. O fluxo do Caso de Uso foi desviado para A1 e após esse desvio, voltou para o Fluxo Básico (veja o passo A1 - 3.2). Assim, vemos que um Fluxo Básico pode ser desviado para um Fluxo Alternativo devido a uma escolha ou ação do Ator.

Fora o desvio causado pela escolha do Ator, existe somente mais um modo do fluxo do Caso de Uso ser desviado: **uma ocorrência de sistema ou o estado do sistema**. Imagine que o nosso analista novamente achou mais um requisito do sistema que o Caso de Uso dele deveria atender:

“Pedidos cancelados não podem ser consultados.”

Vamos criar mais um fluxo alternativo:

Caso de Uso: Consultar Pedido

Versão: 1.2

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Consultar Pedido”;
2. O Sistema oferece a interface de consulta para pedidos;
3. O Ator informa o número do pedido desejado [A1];
4. O Sistema exibe os dados do pedido [A2];

Fluxo Alternativo A1 – Consultar por Cliente

3. O Ator informa um cliente;
- 3.1. O Sistema exibe uma lista de pedidos do cliente selecionado em ordem cronológica decrescente;
- 3.2. O Ator seleciona um pedido do cliente; volta ao fluxo básico;

Fluxo Alternativo A2 – Pedidos Cancelados não podem ser consultados

4. O Sistema informa que o pedido está cancelado e volta ao passo 2 do fluxo básico;

Nesse caso, não foi opção do Ator o desvio no passo 4 para A2, e sim o estado do pedido dentro do sistema que provocou esse desvio.

Outra diferença que deve ser notada é que A2 solicitou que fosse retornado para o passo 2 no fluxo básico. Já o fluxo alternativo A1 solicitou que o fluxo básico continuasse. Uma outra situação que pode ocorrer é um fluxo alternativo solicitar que o Caso de Uso seja encerrado. Vamos demonstrar:

Caso de Uso: Consultar Pedido

Versão: 1.3

Ator: Vendedor

1. O Ator inicia o caso de uso selecionando “Consultar Pedido”;
2. O Sistema oferece a interface de consulta para pedidos [A3];
3. O Ator informa o número do pedido desejado [A1];
4. O Sistema exibe os dados do pedido [A2];

Fluxo Alternativo A1 – Consultar por Cliente

3. O Ator informa um cliente;

3.1. O Sistema exibe uma lista de pedidos do cliente selecionado em ordem cronológica decrescente;

3.2. O Ator seleciona um pedido do cliente; volta ao fluxo básico;

Fluxo Alternativo A2 – Pedidos Cancelados não podem ser consultados

4. O Sistema informa que o pedido está cancelado e volta ao passo 2 do fluxo básico;

Fluxo Alternativo A3 – Não existem pedidos para consulta

2. O Sistema informa que não existem pedidos a serem consultados; o caso de uso é encerrado;

O Fluxo A3, assim como o A2, ocorreu pelo estado do sistema (o sistema não possui pedidos para consulta), e não por opção do Ator. O Fluxo A3 encerra o Caso de Uso.

IMPORTANTE: Note que em toda narrativa do Caso de Uso estão descritas várias regras de condições para que fluxos alternativos ocorram, mas você não consegue encontrar em todo o texto uma só ocorrência da palavra “se” como: “se o Ator fizer isso, então ocorrerá aquilo”. Todas as alternativas estão em forma de afirmação.

Casos de Uso bem descritos não possuem a palavra “se” para definir o rumo dentro dos fluxos possíveis. Toda ocorrência da palavra “se” pode ser substituída por um Fluxo Alternativo para permitir a decomposição de todos os caminhos possíveis do Caso de Uso (cenários). Cenários são importantes para a criação dos Casos de Teste que validarão a funcionalidade implementada. Os cenários extraídos da narrativa para teste são **TODAS** as combinações válidas de caminhos possíveis. Todas as maneiras possíveis de se navegar em um Caso de Uso são testadas.

SOBRE O DIAGRAMA: Fluxos Alternativos da narrativa de um Caso de Uso não alteram em nada a notação do diagrama UML.

Assim, em linhas gerais, resumimos as causas e efeitos dos Fluxos Alternativos:

O que causa um Fluxo Alternativo:	O que um Fluxo Alternativo pode fazer:
<ul style="list-style-type: none">• uma escolha do Ator;• o estado do Sistema.	<ul style="list-style-type: none">• retroceder para um passo anterior;• avançar para um passo posterior;• finalizar o Caso de Uso;

Uma ferramenta importante para o entendimento dos Cenários do Caso de Uso é o Diagrama de Atividades que será explicado no Capítulo 2. Somente para que o quadro acima seja mais bem entendido, verifique o diagrama a seguir:

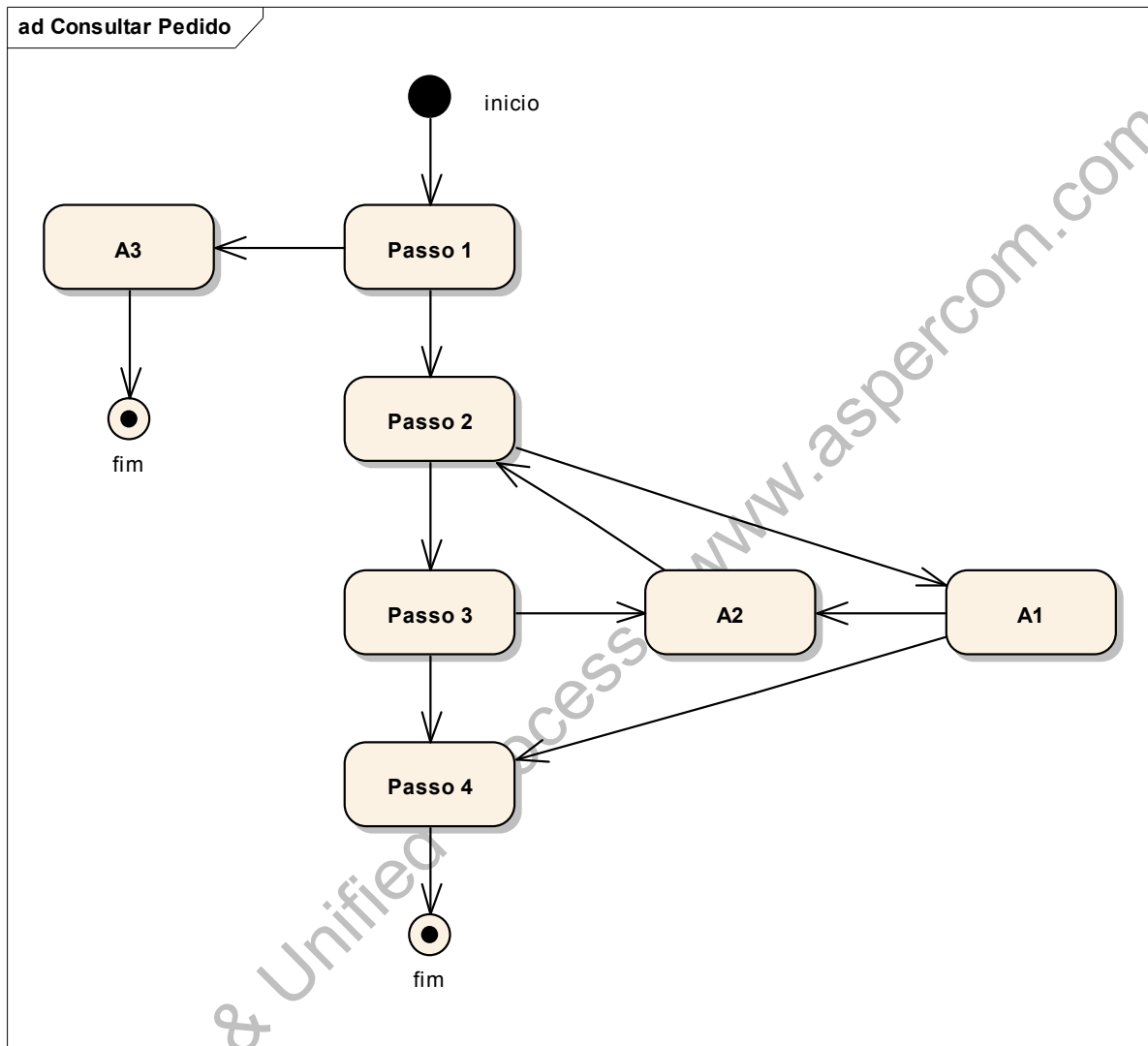
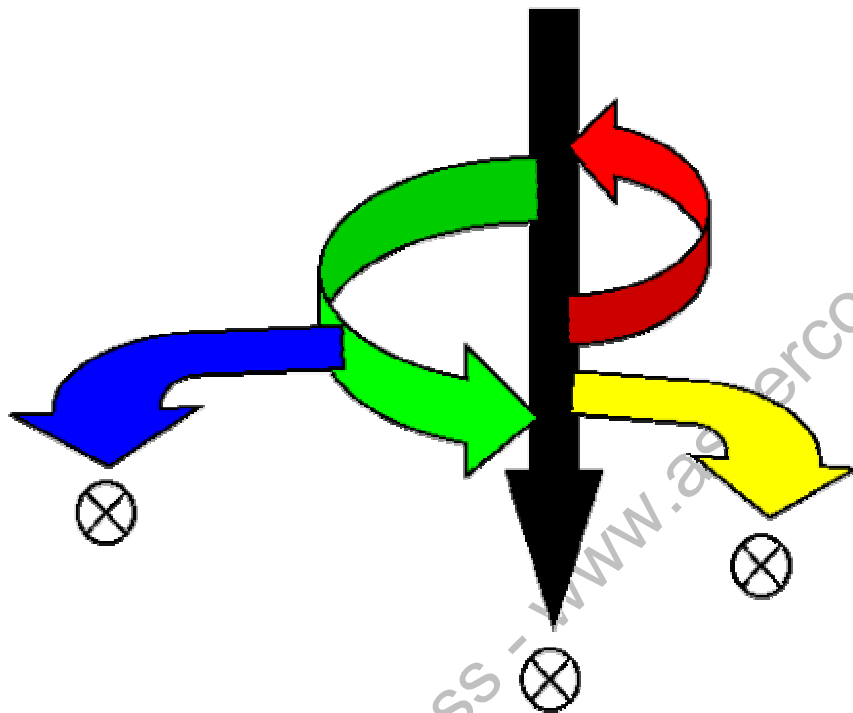


Figura 7 Fluxos de um Caso de Uso

O Diagrama de Atividades é muito utilizado para demonstrar Casos de Uso complexos (com muitos fluxos alternativos). Muitas vezes, um Diagrama de Atividades que é mais visual simplifica a maneira do leitor visualizar as situações possíveis do fluxo do Caso de Uso. Sempre use um Diagrama de Atividades quando a leitura da sua narrativa ficar confusa ou complexa. Vamos detalhar os cenários no tópico seguinte.

1.8. O que são cenários?

Uma demonstração importante para o entendimento dos Fluxos Alternativos do Caso de Uso são os Cenários. Veja a figura a seguir:



Essa ilustração nos mostra o Fluxo Básico como a seta preta. É a maneira mais usual do ator atingir o seu objetivo no Caso de Uso. As outras setas são os Fluxos Alternativos. Essas setas avançam ou retornam no Fluxo Básico ou encerram o Caso de Uso (ver o indicador X). Os cenários são todos os caminhos possíveis que o Caso de Uso pode ter desde o Fluxo Básico até todos os Fluxos Alternativos combinados entre si.

Vamos verificar quais seriam alguns cenários do “Consultar Pedido”. Analisando todas as possibilidades dos caminhos que o Caso de Uso pode tomar, nesse exemplo simples, os cenários existentes são:

- Cenário 1 : Passo 1, Passo 2, Passo 3, Passo 4 (Fluxo Básico);
- Cenário 2 : Passo 1, Passo 2, A1 , Passo 4;
- Cenário 3 : Passo 1, Passo 2, Passo 3, A2 , Passo 2;
- Cenário 4 : Passo 1, Passo 2, A1 , A2 , Passo 2;
- Cenário 5 : Passo 1, A3.

O cenário 1 é o mais simples, pois é o próprio Fluxo Básico. O cenário 2 seria o seguinte:

Caso de Uso: Consultar Pedido	Versão: 1.3
Cenário 2	
1. O Ator inicia o caso de uso selecionando "Consultar Pedido";	
2. O Sistema oferece a interface de consulta para pedidos;	
3. O Ator informa um cliente;	
3.1. O Sistema exibe uma lista de pedidos do cliente selecionado em ordem cronológica decrescente;	
3.2. O Ator seleciona um pedido do cliente;	
4. O Sistema exibe os dados do pedido;	

1.9. Precondição e Pós-condições

Explicado o conceito de cenário, podemos introduzir mais um elemento presente nas narrativas de Casos de Uso: a precondição e as pós-condições. Esses dois elementos são importantes, pois demonstram restrições para um Caso de Uso iniciar e garantias mínimas alcançadas quando este terminar.

Verifique a figura das setas da página anterior para podemos entender melhor este mecanismo. A precondição seria a condição do Sistema que permitiria a seta preta iniciar. Já todos os símbolos (X), seriam as pós-condições: os resultados observáveis do Caso de Uso (sejam de sucesso ou de fracasso com relação ao objetivo do Ator).

Com este desenho chegamos a uma importante definição: **O Caso de Uso possui uma precondição e geralmente várias pós-condições.**

A precondição é uma restrição sobre quando um Caso de Uso pode ser iniciado. **PRESTE ATENÇÃO!** É a CONDIÇÃO que o Sistema deve se encontrar para permitir que o Caso de Uso inicie. Não confunda com o evento que inicia o Caso de Uso. A precondição mais comum nos sistemas é "O usuário deve estar logado". Vamos imaginar que a arquitetura da nossa aplicação de pedidos seja integrada com a autenticação do domínio da rede, nosso Caso de Uso "Consultar Pedido" ficaria assim:

Caso de Uso: Consultar Pedido	Versão: 1.3
Ator: Vendedor	
Precondição: O usuário deve estar logado.	
1. O Ator inicia o caso de uso selecionando "Consultar Pedido";	
2. O Sistema oferece a interface de consulta para pedidos [A3];	
3. O Ator informa o número do pedido desejado [A1];	
4. O Sistema exibe os dados do pedido [A2];	
[os fluxos alternativos foram suprimidos]	

É importante ressaltar que a precondição é um elemento opcional da narrativa do Caso de Uso. O que descrevemos na precondição deve ser importante e agregar um valor observável à análise. Fazemos essa observação, pois é muito comum achar alguns escritos "óbvios" nas precondições, veja alguns exemplos:

- O usuário deve ter acesso a um terminal
- O usuário deve saber escrever
- O sistema deve estar funcionando
- A base de dados deve estar configurada

Geralmente a precondição de um Caso de Uso é uma pós-condição de um Caso de Uso anterior. Se a nossa aplicação de pedidos não fosse integrada com o login da rede possivelmente teríamos um Caso de Uso "Login" que teria como pós-condição "O usuário deve estar logado".

Vamos ver quais seriam as pós-condições de "Consultar Pedidos":

Caso de Uso: Consultar Pedido

Versão: 1.3

Ator: Vendedor

Precondição: O usuário deve estar logado.

5. O Ator inicia o caso de uso selecionando "Consultar Pedido";
6. O Sistema oferece a interface de consulta para pedidos [A3];
7. O Ator informa o número do pedido desejado [A1];
8. O Sistema exibe os dados do pedido [A2];

[os fluxos alternativos foram suprimidos]

Pós-condições: Um pedido selecionado; Não existem pedidos para consulta.

As pós-condições descrevem os resultados observáveis de sucesso ou de falha do Caso de Uso. As pós-condições são importantes, pois mostram as garantias mínimas que um Caso de Uso deve oferecer.

1.10. **Relacionamento «extend» entre Casos de Uso**

A melhor maneira de explicar um conceito é entender quais foram os problemas e necessidades que foram enfrentadas para definir a solução. Quando a técnica de Caso de Uso se desenvolveu os analistas tinham um problema para acrescentar comportamentos em Casos de Uso que já estavam definidos ou até implantados. Os analistas imaginavam que seria muito bom se o Caso de Uso definido abrisse uma porta para que os novos comportamentos da evolução do software fossem incorporados. Essa foi a motivação do relacionamento «extend».

Um Caso de Uso disponibiliza um ponto de extensão (extension point) que outros Casos de Uso podem **observar** e de acordo com **uma condição**, este Caso de Uso que está observando pode assumir o controle e embutir os seus comportamentos.

Essa técnica permitiu a solução do problema que eles tinham e abriu também outras possibilidades importantes para os Casos de Uso. Mesmo para Casos de Uso ainda em construção, é possível acrescentar vários comportamentos simplesmente abrindo uma extension point. Como um exemplo vale mais que palavras, vamos diagramar:

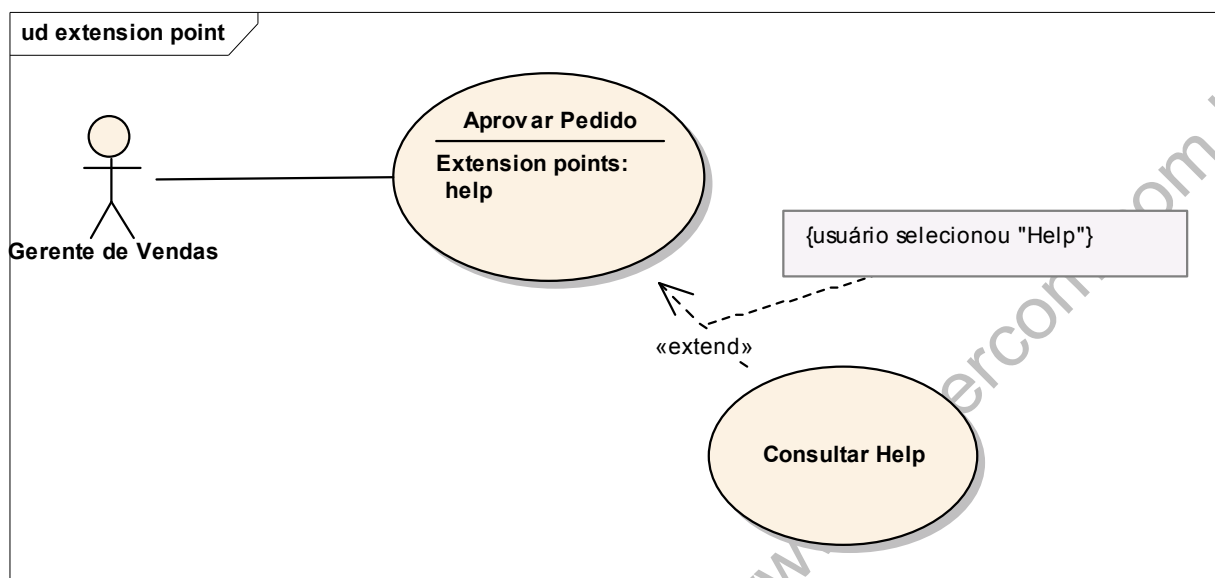


Figura 8 Relacionamento «extend» entre Casos de Uso

Os usuários do sistema solicitaram que fosse criado um “Help” para o Caso de Uso “Aprovar Pedido”. A qualquer momento em “Aprovar Pedido” o usuário poderá selecionar o “Help”. Assim, “Consultar Help” observa “Aprovar Pedido”, se a condição destacada {usuário selecionou “Help”} for satisfeita o Caso de Uso “Aprovar Pedido” irá parar (simplesmente parar, e não encerrar) e o “Consultar Help” iniciará. Quando “Consultar Help” for encerrado “Aprovar Pedido” continuará de onde parou.

Para direcionar melhor o uso do relacionamento «extend», podemos afirmar que você usará esta técnica quando necessitar que **a qualquer momento** dada **uma condição**, o Caso de Uso base deverá ser interrompido e outro Caso de Uso deverá assumir o controle.

Geralmente um «extend» é utilizado quando o Caso de Uso que está estendendo (que atira a seta) é um serviço “assíncrono” que pode ser utilizado se a condição descrita ocorrer no Caso de Uso base (que foi atingido pela seta).

Para exemplificar a narrativa, vamos descrever o Caso de Uso base:

Caso de Uso: Aprovar Pedido

Ator: Vendedor

Extension Points: help

1. O Ator inicia o caso de uso selecionando “Aprovar Pedido”;
2. O Sistema oferece a interface exibindo uma lista de pedidos para aprovação;
3. O Ator seleciona o pedido;
4. O Sistema aprova o pedido;

Note como não existe qualquer menção ao Caso de Uso que está estendendo “Consultar Help”, vamos descrevê-lo:

Caso de Uso: Consultar Help

1. O Ator inicia o caso de uso selecionando a opção Help em “Aprovar Pedido”;
2. O Sistema oferece a interface exibindo a ajuda do sistema;

O ponto de extensão (extension point) não precisa necessariamente ser definido, e também, ele pode ser definido em um ponto específico da narrativa, e não como “em qualquer momento” como o exemplo ditou. Uma narrativa deste modo poderia ser assim:

Caso de Uso: Aprovar Pedido (exemplo 2)

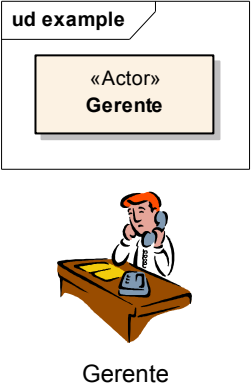
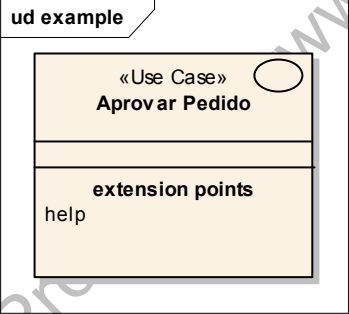
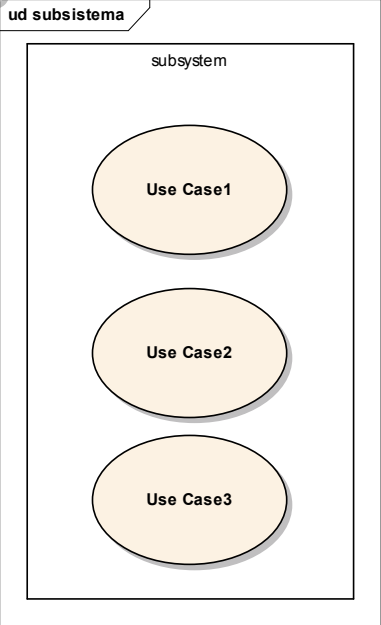
Ator: Vendedor

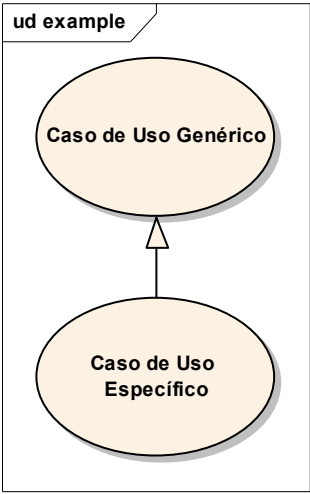
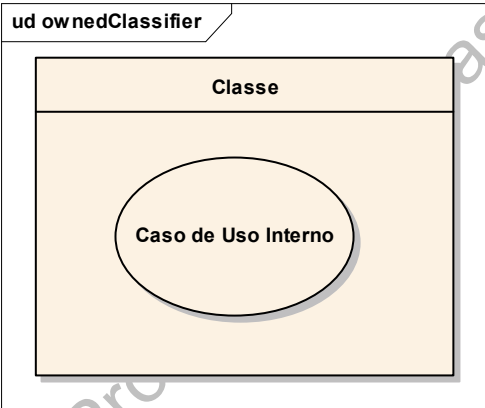
Extension Points: help

1. O Ator inicia o caso de uso selecionando “Aprovar Pedido”;
2. O Sistema oferece a interface exibindo uma lista de pedidos para aprovação;
3. [extension point: help]
4. O Ator seleciona o pedido;
5. O Sistema aprova o pedido;

Neste exemplo 2 do Caso de Uso, a condição {usuário selecionou “Help”} só seria avaliada no passo 3.

1.11. Tópicos não abordados sobre Casos de Uso

Conceito	Notação	Comentários
<p>Notações alternativas para Ator</p>		<p>Um ator também pode ser demonstrado através de um retângulo com «Actor» ou também através de um ícone.</p>
<p>Notação alternativa para Casos de Uso</p>		<p>Um caso de uso também pode ser retratado como retângulo.</p>
<p>Boundary</p>		<p>Você também pode agrupar Casos de Uso em Boundaries separando-os em Subsystems.</p>

<p>Generalização de Caso de Uso</p>		<p>A UML também permite que Casos de Uso sejam generalizados, porém, esta técnica complica o diagrama e o escopo para pessoal não técnico que desconhece o conceito de herança. As formas que existem de abordar essa generalização na narrativa também é complicada e difícil de manter.</p>
<p>Caso de Uso dentro de um Classifier</p>		<p>A UML também permite definir o Caso de Uso dentro de um Classifier de modo que o Classifier possua o Caso de Uso como um comportamento. É uma técnica interessante para definir o comportamento de componentes, porém, também não é usual.</p>

Curso UML & Unified Pro