

Engenharia de Software II

**Aula: Modelagem de
sistemas de software.**



Prof. Anderson Augusto Bosing

Objetivo

Compreender como os modelos gráficos podem ser usados para representar sistemas de software;

Compreender por que diferentes tipos de modelo são necessários e as perspectivas fundamentais de modelagem de sistema de contexto, interação, estrutura e comportamento;

Terá sido apresentado a alguns dos tipos de diagramas da Unified Modeling Language (UML), e como eles podem ser usados na modelagem de sistema.



Introdução a Modelagem de Sistemas

Modelagem de sistema é o processo de desenvolvimento de modelos abstratos de um sistema, em que cada modelo apresenta uma visão ou perspectiva, diferente do sistema.

A modelagem de sistema geralmente representa o sistema com algum tipo de notação gráfica, que, atualmente, quase sempre é baseada em notações de UML (linguagem de modelagem unificada, do inglês Unified Modeling Language).

No entanto, também é possível desenvolver modelos (matemáticos) formais de um sistema, normalmente como uma especificação detalhada do sistema.



O que é modelagem de Sistemas ?

A modelagem de sistemas de software consiste na utilização de notações gráficas e textuais com o objetivo de construir modelos que representam as partes essenciais de um sistema, considerando-se várias perspectivas diferentes e complementares.



Introdução a Modelagem de Sistemas

Os modelos são usados durante o processo de engenharia de requisitos para ajudar a extrair os requisitos do sistema; durante o processo de projeto, são usados para descrever o sistema para os engenheiros que o implementam; e, após isso, são usados para documentar a estrutura e a operação do sistema. Você pode desenvolver modelos do sistema existente e do sistema a ser desenvolvido.

- **Modelos do sistema existente** são usados durante a engenharia de requisitos. Eles ajudam a esclarecer o que o sistema existente faz e podem ser usados como ponto de partida para discutir seus pontos fortes e fracos. Levam, então, os requisitos para o novo sistema.



Introdução a Modelagem de Sistemas

- Modelos do novo sistema são usados durante a engenharia de requisitos para ajudar a explicar os requisitos propostos para outros stakeholders do sistema. Os engenheiros usam esses modelos para discutir propostas de projeto e documentar o sistema para a implementação. Em um processo de engenharia dirigida a modelos, é possível gerar uma implementação completa ou parcial do sistema a partir do modelo de sistema.



Previsão do Comportamento Futuro do Sistema

O comportamento do sistema pode ser discutido mediante uma análise dos seus modelos.

Os modelos servem como um “laboratório”, em que diferentes soluções para um problema relacionado à construção do sistema podem ser experimentadas.



Aspecto Importante

O aspecto mais importante de um modelo de sistema é que ele deixa de fora os detalhes. Um modelo é uma abstração do sistema a ser estudado, e não uma representação alternativa dele. Idealmente, uma representação de um sistema deve manter todas as informações sobre a entidade representada. Uma abstração deliberadamente simplifica e seleciona as características mais salientes.



Diferentes Perspectivas

A partir de perspectivas diferentes, você pode desenvolver diversos modelos para representar o sistema. Por exemplo:

1. Uma perspectiva externa, em que você modela o contexto ou o ambiente do sistema.
2. Uma perspectiva de interação, em que você modela as interações entre um sistema e seu ambiente, ou entre os componentes de um sistema.
3. Uma perspectiva estrutural, em que você modela a organização de um sistema ou a estrutura dos dados processados pelo sistema.
4. Uma perspectiva comportamental, em que você modela o comportamento dinâmico do sistema e como ele reage aos eventos.



E o que essas perspectivas tem em comum ?

Atividade - Visões de Arquitetura.

Quais visões ou perspectivas são úteis durante o projeto e documentação de Arquitetura de um Sistema ?

Quais notações devem ser utilizadas para descrever os modelos de arquitetura ?

É possível representar em um único diagrama todas as informações relevantes a respeito da arquitetura de um sistema ?

Detalhe a visão 4 + 1, de Kruchten, da arquitetura do sistema (KRUCHTEN, 1995).

Leitura Obrigatória :

Capítulo 6.2 Visões de Arquitetura

SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo: Pearson, 2018. E-book.

Disponível em: <https://plataforma.bvirtual.com.br>. Acesso em: 04 ago. 2023.



Atividade - Visões de Arquitetura.

Quais visões ou perspectivas são úteis durante o projeto e documentação de Arquitetura de um Sistema ?

Quais notações devem ser utilizadas para descrever os modelos de arquitetura ?

É possível representar em um único diagrama todas as informações relevantes a respeito da arquitetura de um sistema ?

Detalhe a visão 4 + 1, de Kruchten, da arquitetura do sistema (KRUCHTEN, 1995).

Leitura Obrigatória :

Capítulo 6.2 Visões de Arquitetura

SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo: Pearson, 2018. E-book.

Disponível em: <https://plataforma.bvirtual.com.br>. Acesso em: 04 ago. 2023.



Engenharia de Software II

**Aula: Atividades Típicas
de um Processo de
Desenvolvimento**



Prof. Anderson Augusto Bosing

Levantamento dos Requisitos

Compreensão do problema, visando permitir que usuários e desenvolvedores tenham a mesma visão do problema a ser resolvido.

Análise de requisitos

Estudo detalhado dos requisitos levantados e a construção de modelos para representar o sistema a ser construído.



Levantamento dos Requisitos e Análise de requisitos

- O escopo do software é refinado;
- Nessa fase, a interação entre quem desenvolverá e o cliente é muito grande, contudo não se discute como será feito o software, mas sim o que ele deve fazer.
- Devem ser analisados o domínio do problema e o domínio da solução.
- São usadas as histórias de usuário (User Stories) ou especificação tradicional de requisitos, que são pequenas histórias escritas para auxiliar na definição do requisito entre quem desenvolve e o cliente.
- Todo esse material é compilado e gera um relatório que servirá para os tomadores de decisão definirem, ou não, a continuidade e o desenvolvimento do software.



Projeto

Determina como o sistema funcionará para atender os requisitos, de acordo com os recursos tecnológicos existentes. A modelagem do software pode ser realizada por um conjunto de diagramas, por exemplo, pela UML.

- Utiliza a fase anterior como insumo.
- Essa fase é voltada ao programador do software.
- Definição de como seria a interface que o usuário opera, quais cores seriam utilizadas na interface, como seria o fluxo de funcionamento de cada botão existente na interface, em qual banco de dados ficariam os dados gerados.
- O projeto, diferente da fase anterior, define como as coisas acontecerão.



Implementação

Ocorre a tradução da descrição computacional da fase de projeto em código executável através do uso de linguagens de programação. É nessa fase que os diagramas/modelos criados “ganham vida”.

- Essa fase deve traduzir o projeto em um software, utilizando ferramentas e linguagens adequadas.
- Dadas as inúmeras metodologias de desenvolvimento, cada programador pode desenvolver o código do sistema de uma forma diferente.



Testes

Para verificar a corretude do sistema, levando-se em conta a especificação feita na fase de projeto.

- **Teste de unidade:** cada componente é testado individualmente, sem conexão com outros componentes.
✓ Exemplo: testar apenas o método que faz a soma dos valores das vendas, dados dois números, ele deve retornar a soma dos dois.
- **Teste de módulo:** um módulo é um agrupamento de pequenos componentes, que tem uma função específica.
✓ Exemplo: testar a geração do relatório de vendas de um produto.
- **Teste de subsistemas:** são testados módulos integrados que controlam as interfaces do sistema.
✓ Exemplo: testar as interfaces do sistema de compra e venda.



Testes

- **Teste de sistema:** testa a integração dos subsistemas que formam o sistema principal.
 - ✓ Exemplo: realizar login, operações e geração de relatórios em um sistema.
- **Teste de aceitação:** testes realizados com dados reais fornecidos pelos clientes. Último teste antes de colocar o sistema em operação.
 - ✓ Exemplo: quando há muitos usuários na base e o login não acontece de forma instantânea.



Implantação

O sistema é empacotado, distribuído e instalado no ambiente do usuário. São entregues os manuais do sistema e os usuários são treinados para utilizar o sistema.

➤ O software deve ser instalado em ambiente produção.

➤ Envolve:

- ✓ Treinamento de usuários;
- ✓ Configuração do ambiente de produção;
- ✓ Conversão bases de dados (se necessário).



Engenharia de Software II

**Aula: O componente
humano (participantes do
processo)**



Prof. Anderson Augusto Bosing

O componente humano (participantes do processo)

Gerente de projeto

- Responsável pela gerência e coordenação das atividades necessárias para a construção do sistema, alem de estimar tempo e custo.

Analista

- Possui conhecimento sobre o domínio do negócio para que possa levantar os requisitos.

Projetista

- Avalia as alternativas de solução e gera uma especificação detalhada da solução computacional (Ex: projetista de rede, de banco de dados, etc.).

Programador

- Responsável pela implementação do sistema.

Cliente

- O cliente usuário e especialista no domínio do negócio e interage diretamente com o Analista para levantar os requisitos do sistema.



Engenharia de Software II

**Aula: O Paradigma da
orientação a objetos**



Prof. Anderson Augusto Bosing

O paradigma de Orientação a Objetos

Historicamente antes de 1975 a maioria das empresas de software não usava nenhuma técnica específica, cada indivíduo trabalhava do seu próprio jeito.

Grandes avanços foram feitos aproximadamente entre 1975 e 1985, com o desenvolvimento assim chamado paradigma clássico ou estruturado. Essa abordagem parecia extremamente promissora para a época. Todavia à medida que o tempo foi passando, constatou-se que ela ficou aquém do esperado em dois principais aspectos:

1. Algumas vezes o paradigma e suas técnicas eram incapazes de lidar com o tamanho cada vez maior dos produtos de software, isso é, as técnicas clássicas eram adequadas para elaborar produtos com escala pequena ou média.
2. O paradigma clássico não estava a altura das expectativas iniciais durante a manutenção pós-entrega.

O paradigma de Orientação a Objetos

A principal razão para o sucesso limitado desse paradigma clássico foi que as técnicas são orientadas a operações ou a atributos(dados), mas não a ambos ao mesmo tempo.

Em contraste, o paradigma de orientação a objetos considera igualmente importantes tanto os atributos quanto as operações. Uma maneira simplista de compreender um objeto é vê-lo como um artefato de software unificado, que incorpora tanto os atributos quanto as operações realizadas sobre os atributos



Evolução Histórica

Bezerra (2015) apresenta um breve resumo histórico da evolução das técnicas de desenvolvimento, para explicar como chegamos ao cenário atual.

Décadas de 1950/1960: Os sistemas eram bem simples, e o seu desenvolvimento era direto ao assunto, não tinha um planejamento inicial, como dizem, “ad hoc”. Como os sistemas eram significativamente mais simples, as técnicas de modelagem também: Eram usados fluxogramas e diagramas de módulos.

Década de 1970: Começaram a surgir computadores mais avançados e acessíveis. Houve grande ampliação do mercado computacional. Sistemas mais complexos começavam a surgir. Consequentemente, modelos mais robustos foram propostos. Os autores Larry Constantine e Edward Yourdon são grandes colaboradores nessas técnicas.



Evolução Histórica

Década de 1980: Nessa fase os computadores se tornaram ainda mais avançados e mais baratos. Surge a necessidade por interfaces mais sofisticadas, o que originou a produção de sistemas de softwares mais complexos. A Análise Estruturada surgiu no início desse período, com os trabalhos de Edward Yourdon, Peter Coad, Tom De Marco, James Martin e Chris Gane.

Década de 1990: No início dessa década é o período em que surge um novo paradigma de modelagem, a Análise Orientada a Objetos, como resposta a dificuldades encontradas na aplicação da análise estruturada em algumas aplicações. Grandes colaboradores desse paradigma são Sally Shlaer, Stephen Mellor, Rebecca Wirfs-Brock, James Rumbaugh, Grady Booch e Ivar Jacobson. Já no fim da década, o paradigma da orientação a objetos atinge sua maturidade. Os conceitos de padrões de projeto, frameworks, componentes e qualidade começam a ganhar espaço. Surge a Linguagem de Modelagem Unificada UML.



Evolução Histórica

Década de 2000: O reúso por meio de padrões de projetos e frameworks se solidifica. As denominadas metodologias ágeis começam a ganhar espaço. Técnicas de testes automatizados e refatoração começaram a se difundir entre os desenvolvedores que trabalham com orientação a objeto. Grandes nomes dessa fase são: Rebecca Wirfs-Brock, Martin Fowler e Eric Vans.

Como percebemos, durante a década de 90 surgiram várias propostas e técnicas para modelagem de sistema segundo o paradigma da orientação a objeto. Era comum, durante essa década, duas técnicas possuírem diferentes notações gráficas para modelar a mesma perspectiva de um sistema. Todas tinham pontos fortes e fracos em relação à notação utilizada, mas via-se a necessidade de uma notação que viesse a se tornar um padrão para a modelagem de sistemas orientados a objeto, e que fosse amplamente aceita, nas indústrias e na academia.

Alguns esforços surgiram em 1996 para essa padronização, o que resultou na definição da UML (Unified Modeling Language), que detalharemos a seguir em um tópico específico, mas antes falaremos na Modelagem Estruturada e da Modelagem Orientada a Objetos.



Evolução Histórica

Pode-se construir uma casa para um cachorro apenas juntando algumas tábuas, alguns pregos e algumas ferramentas básicas. Com um planejamento mínimo, em poucas horas e com um pouco de habilidade, nosso cachorro terá uma casa (BOOCH et al., 2000).



Engenharia de Software II

**Aula: Utilização da UML no
processo iterativo e
incremental**



Prof. Anderson Augusto Bosing

Utilização da UML no processo iterativo e incremental

A UML é independente de processo de desenvolvimento.

- Vários processos podem utilizar a UML para modelagem de sistemas OO.

Os artefatos de software construídos através da UML evoluem à medida que as iterações são realizadas.

- A cada iteração, novos detalhes são adicionados a esses artefatos.
- Além disso, a construção de um artefato fornece informações para adicionar detalhes a outros.



Modelo Incremental

- Segundo Pressman (2011) “modelos evolucionários são iterativos. Apresentam características que possibilitam desenvolver versões cada vez mais completas do software”.
- Dessa forma, o primeiro modelo evolucionário que surgiu é o modelo incremental de desenvolvimento de software.
- Para Sommerville (2014) o sistema incremental tem o objetivo de reduzir o retrabalho custoso do modelo cascata, possibilitando ao cliente postergar decisões e requisitos conforme necessidade, mas mantendo o poder de gerenciamento que o modelo oferece.



Prof. Anderson Augusto Bosing

Modelo Incremental

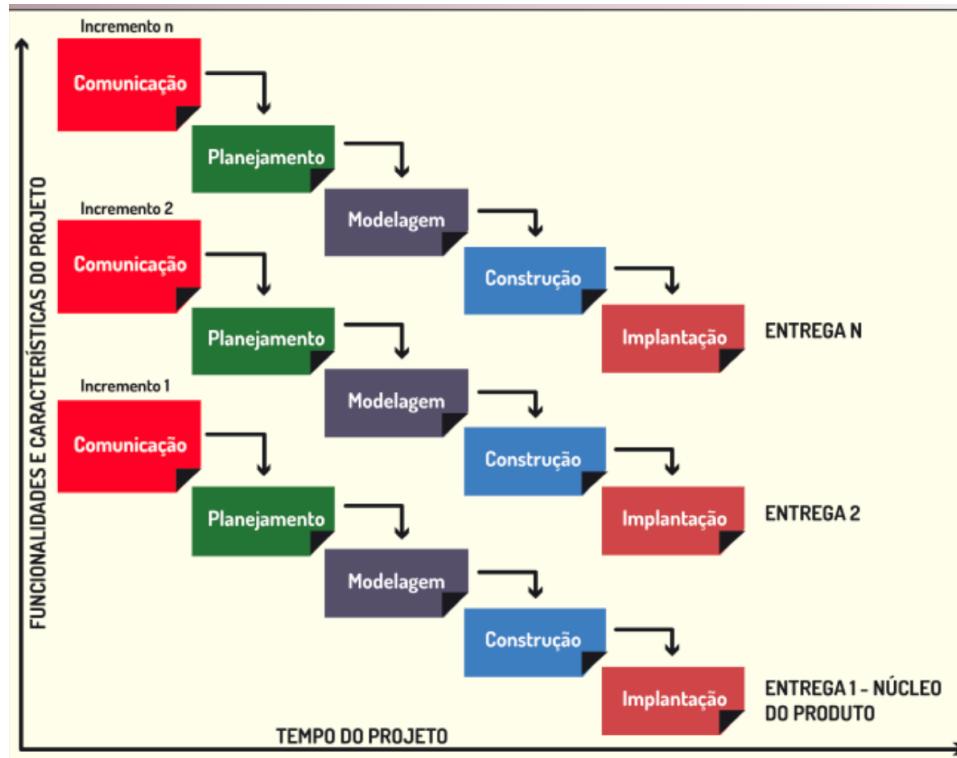
- Essa metodologia divide a fase de desenvolvimento do software em ciclos completos, passando por todas as fases existentes no modelo clássico, levantamento e análise de requisitos, projeto, implementação e testes.

- O modelo trabalha com 5 (cinco) fases distintas: comunicação, planejamento, modelagem, construção e implantação.



Prof. Anderson Augusto Bosing

Modelo Incremental



1. **Comunicação:** trata das informações do negócio, como são geradas, processadas e quem utilizará. Define os requisitos.
2. **Planejamento:** define os objetos, suas características e como se relacionarão entre si.
3. **Modelagem:** desenha o processo, visando atender à função do negócio, e define os procedimentos necessários para a manipulação dos objetos de dados definidos na etapa anterior.
4. **Construção:** ocorre a geração da aplicação, normalmente por meio de reutilização de componentes que serão integrados posteriormente.
5. **Implantação:** acontece alguns pequenos testes e a entrega de parte do produto.

PROTOTIPAGEM

Um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções. O desenvolvimento rápido e iterativo do protótipo é essencial para que os custos sejam controlados e os stakeholders do sistema possam experimentá-lo no início do processo de software.

Um protótipo de software pode ser usado em um processo de desenvolvimento de software para ajudar a antecipar as mudanças que podem ser requisitadas:

1. No processo de engenharia de requisitos, um protótipo pode ajudar na elicitação e validação de requisitos de sistema.
2. No processo de projeto de sistema, um protótipo pode ser usado para estudar soluções específicas do software e para apoiar o projeto de interface de usuário.



PROTOTIPAGEM

A prototipagem é uma técnica aplicada quando:

- Há dificuldades no entendimento dos requisitos do sistema.
- Há requisitos que precisam ser mais bem definidos.

A construção de protótipos utiliza ambientes com facilidades para a construção da interface gráfica.

Procedimento geral da prototipagem:

- Após o levantamento de requisitos um protótipo é construído para ser usado na validação.
- Usuários fazem críticas sobre e o protótipo é corrigido ou refinado.
- O processo de revisão e refinamento continua até que o protótipo seja aceito.
- Após a aceitação, o protótipo é descartado ou utilizado como uma versão inicial do sistema.



PROTOTIPAGEM

A prototipagem é uma técnica aplicada quando:

- Há dificuldades no entendimento dos requisitos do sistema.
- Há requisitos que precisam ser mais bem definidos.

A construção de protótipos utiliza ambientes com facilidades para a construção da interface gráfica.

Procedimento geral da prototipagem:

- Após o levantamento de requisitos um protótipo é construído para ser usado na validação.
- Usuários fazem críticas sobre e o protótipo é corrigido ou refinado.
- O processo de revisão e refinamento continua até que o protótipo seja aceito.
- Após a aceitação, o protótipo é descartado ou utilizado como uma versão inicial do sistema.



PROTOTIPAGEM

A prototipagem NÃO é um substituto à construção de modelos do sistema.

- A prototipagem é uma técnica complementar à construção de modelos do sistema.
- Mesmo com o uso de protótipos, ou modelos do sistema devem ser construídos.
- Os erros detectados na validação do protótipo devem ser utilizados para modificar e refinar os modelos de sistema.



EXERCICIO FERRAMENTAS CASE

- Definição de Conceito.
- Qual sua aplicação?
- Como elas podem ser classificadas ?
- Elas podem ser Classificadas de acordo com os serviços que oferecem ?
- Identifique exemplos de ferramentas case e para que são aplicadas.
- Escolha duas delas que possuem aplicação para o mesmo propósito e realize um benchmark entre elas identificando os pontos positivos e negativos de cada uma delas.



FERRAMENTAS CASE

A sigla CASE significa “Computer Aided Software Engineering”, em português: “Engenharia de Software Auxiliada por Computador”.

Ferramentas CASE são ferramentas utilizadas como suporte para desenvolver um software. Essas ferramentas oferecem um conjunto de serviços, fortemente relacionados, para apoiar uma ou mais atividades do processo de desenvolvimento de software e podem minimizar o tempo de desenvolvimento do programa, mantendo o alto nível de qualidade.

As ferramentas CASE podem auxiliar no desenvolvimento de software desde a análise de requisitos e modelagem até programação e testes. Podem ser consideradas como ferramentas automatizadas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de desenvolvimento de software.



FERRAMENTAS CASE

As ferramentas CASE podem ser classificadas em:

- Horizontais: oferecem serviços utilizados durante todo o processo de software;
- Verticais: utilizadas em fases específicas do processo de software.

Elas também podem ser classificadas de acordo com os serviços que oferecem, dentre as quais, cita-se.

- Documentação;
- Planejamento e gerenciamento de projetos;
- Especificações formais;
- Comunicação;
- Análise e projeto de software;
- Projeto e desenvolvimento de interfaces;
- Suporte a Programação;
- Gerenciamento de Configuração;
- Controle de Qualidade;
- Testes de software;
- Depuração;
- Reengenharia



Engenharia de Software II

**Aula: Mecanismos gerais
(estereótipos, notas
explicativas, etiquetas
valoradas, restrições,
pacotes e OCL)**



Prof. Anderson Augusto Bosing

Estereótipos

Um estereótipo é um dos mecanismos de uso geral da UML, que é utilizado para estender o significado de determinado elemento em um diagrama.

A UML predefine diversos estereótipos. A UML também permite que o usuário defina os estereótipos a serem utilizados em determinada situação de modelagem. Ou seja, a própria equipe de desenvolvimento pode definir os estereótipos que serão utilizados em situações específicas.

Dessa forma, podemos ter estereótipos de dois tipos: predefinidos ou definidos pela equipe de desenvolvimento.

Os estereótipos definidos pela própria equipe de desenvolvimento devem ser documentados de tal forma que a sua semântica seja entendida sem ambiguidades por toda a equipe. Além disso, um estereótipo definido pelo usuário deve ser utilizado de forma consistente na modelagem de todo o sistema. Ou seja, não é correto utilizar um mesmo estereótipo para denotar diferentes significados.



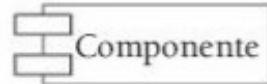
Estereótipos

Outra classificação que pode ser aplicada aos estereótipos (tanto os predefinidos quanto os definidos pela equipe de desenvolvimento) é quanto à sua forma: estereótipos gráficos (ou ícones) e textuais.

Um estereótipo gráfico é representado por um ícone que lembre o significado do conceito ao qual ele está associado.



Ator



Componente



Servidor
HTTP



Portal de
segurança
(firewall)

Estereótipos

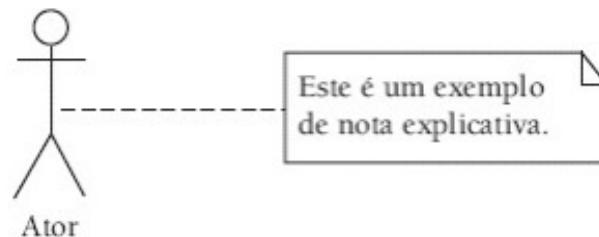
Um estereótipo de rótulo é representado por um nome delimitado pelos símbolos << e >> (esses símbolos são chamados de aspas francesas), e posicionado próximo ao símbolo. Os estereótipos <<document>>, <<interface>>, <<control>> e <<entity>> são exemplos de estereótipos textuais predefinidos da UML. Nesse sentido, os estereótipos permitem estender a UML e adaptar o seu uso em diversos casos.



Notas Explicativas

Notas explicativas são utilizadas para definir uma informação que comenta ou esclarece alguma parte de um diagrama. Podem ser descritas em texto livre; também podem corresponder a uma expressão formal utilizando a linguagem de restrição de objetos da UML, a OCL(Vamos falar sobre ela no futuro).

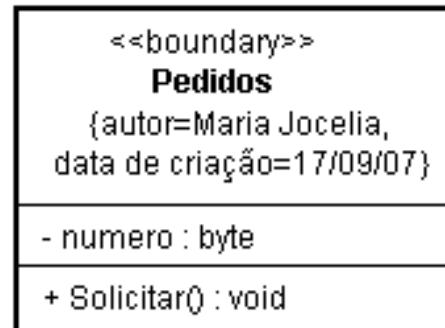
Graficamente, as notas são representadas por um retângulo com uma “orelha”. O conteúdo da nota é inserido no interior do retângulo e este é ligado ao elemento que se quer esclarecer ou comentar por meio de uma linha tracejada.



Etiquetas Valoradas

Os elementos gráficos de um diagrama da UML possuem propriedades predefinidas. Uma classe tem três propriedades predefinidas: um nome, uma lista de atributos e uma lista de operações.

Além das propriedades predefinidas, podem-se também definir outras propriedades para determinados elementos de um diagrama através do mecanismo de etiquetas valoradas. A partir da UML 2.0, uma etiqueta valorada somente pode ser utilizada como um atributo definido sobre um estereótipo (que pode ser predefinido ou definido pelo usuário). Dessa forma, um determinado elemento de um modelo deve primeiramente ser estendido por um estereótipo e só depois por uma etiqueta valorada

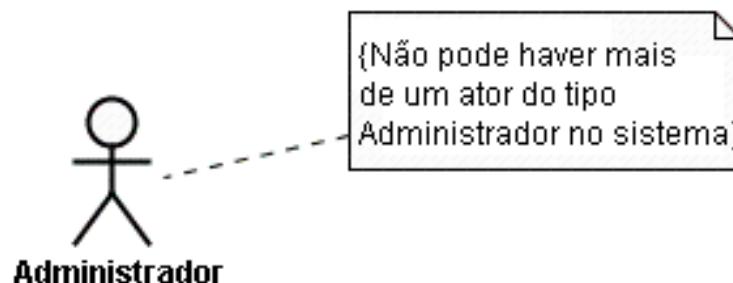


Restrições

A todo elemento da UML está associada alguma semântica. Isso quer dizer que cada elemento gráfico dessa linguagem possui um significado bem definido que, uma vez entendido, fica implícito na utilização do elemento em algum diagrama.

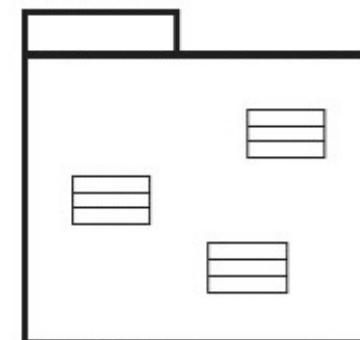
As restrições permitem estender ou alterar a semântica natural de um elemento gráfico. Esse mecanismo geral especifica restrições sobre um ou mais valores de um ou mais elementos de um modelo. Restrições podem ser especificadas tanto formal quanto informalmente. A especificação formal de restrições se dá pela OCL. Além de poderem ser especificadas com o uso da OCL, restrições também podem ser definidas informalmente pelo texto livre (linguagem natural). Assim como para as etiquetas, uma restrição, seja ela formal ou informal, também deve ser delimitada por chaves. Essas restrições devem aparecer dentro de notas explicativas

.

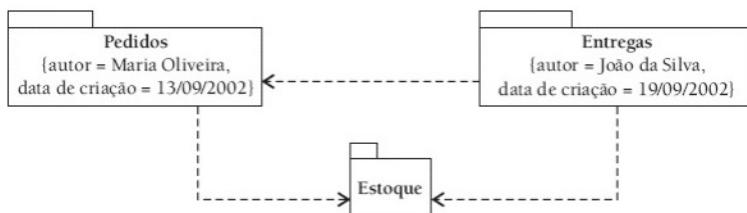


Pacotes

Um pacote é um mecanismo de agrupamento definido pela UML. Esse mecanismo pode ser utilizado para agrupar elementos semanticamente relacionados. A notação para um pacote é a de uma pasta com uma aba.



As ligações entre pacotes são relacionamentos de dependência. Um pacote P1 depende de outro P2 se algum elemento contido em P1 depende de algum elemento contido em P2. O significado específico dessa dependência pode ser definido pela própria equipe de desenvolvimento com o uso de estereótipos.



OCL

A UML define uma linguagem formal que pode ser utilizada para especificar restrições sobre diversos elementos de um modelo. Essa linguagem se chama OCL, a Linguagem de Restrição de Objetos. A OCL pode ser utilizada para definir expressões de navegação entre objetos, expressões lógicas, precondições, pós-condições etc..

A maioria das declarações em OCL consiste nos seguintes elementos estruturais: contexto, propriedade e operação. Um contexto define o domínio no qual a declaração em OCL se aplica. Por exemplo, uma classe ou uma instância de uma classe. Em uma expressão OCL, a propriedade corresponde a algum componente do contexto. Pode ser, por exemplo, o nome de um atributo em uma classe, ou uma associação entre dois objetos. Finalmente a operação define o que deve ser aplicado sobre a propriedade. Uma operação pode envolver operadores aritméticos, operadores de conjunto e operadores de tipo. Outros operadores que podem ser utilizados em uma expressão OCL são: and, or, implies, if, then, else, not, in.

Véiculo
- proprietario : Pessoa
- cor : String
- marca : String

Context Véiculo

inv: self.proprietário.idade >= 18

Engenharia de Software II

**Aula: Modelagem de
Caso de Uso**



Prof. Anderson Augusto Bosing

Modelagem de Caso de Uso

O modelo de casos de uso (MCU) é uma representação das funcionalidades externamente observáveis do sistema e dos elementos externos ao sistema que interagem com ele. O MCU é um modelo de análise que representa um refinamento dos requisitos funcionais do sistema em desenvolvimento.

A ferramenta da UML utilizada na modelagem de casos de uso é o diagrama de casos de uso.



Modelo de Casos de Uso(MCU)

O MCU representa os possíveis usos de um sistema da maneira como são percebidos por um observador externo a este sistema.

Cada um desses usos está associado a um ou mais requisitos funcionais identificados para o sistema.

É um modelo de análise que representa um refinamento dos requisitos funcionais.

Possui diversos componentes: casos de uso, atores e relacionamentos entre eles.



O que é Modelagem de Caso de Uso?

- Faz a ligação entre as necessidades dos envolvidos para os requisitos de software.
- Define limites claro para um sistema.
- Captura e comunica o comportamento desejado do sistema.
- Identifica quem ou o que interage com o sistema.
- Valida/Verifica os requisitos.
- Um instrumento de planejamento.



O que é um Caso de Uso ?

Um caso de uso define uma sequência de ações a serem executadas pelo sistema para produzir um resultado de valor observável para um ator.(RUP)

Usamos casos de uso para captar os comportamentos pretendidos de um sistema, sem especificar como esse comportamento é implementado.



Casos de Uso são comumente identificados por nomes ou identificadores.

Todo caso de uso possui um nome que o identifica e diferencia dos demais casos de uso do sistema.

O nome é uma sequência de caracteres de texto e deve ser único no pacote que o contém.

No geral, os nomes são expressões verbais ativas, que nomeiam um comportamento específico do sistema.

Exemplos: Registrar Venda, Fazer Pedido, Manter Usuários, Manter Produtos.



Casos de Uso são comumente identificados por nomes ou identificadores.

Os identificadores representam através de um código o caso de uso a que se referenciam.

Exemplo:

UC.001 - Registrar Venda

UC.002 - Fazer Pedido

UC.003 - Manter Usuários

UC.004 - Manter Produtos



O que é um Caso de Uso ?

Um caso de uso é a especificação de uma sequência completa de interações entre um sistema e um ou mais agentes externos a esse sistema.

Representa um relato de uso de certa funcionalidade do sistema em questão, sem revelar a estrutura e o comportamento internos desse sistema.



O que é um Caso de Uso ?

Cada caso de uso de um sistema se define pela descrição narrativa das interações que ocorrem entre o elemento externo e o sistema.

A UML não define uma estrutura textual a ser utilizada na descrição de um caso de uso.

Há vários estilos de descrição propostos para definir casos de uso.



Podemos dizer que há três dimensões em que o estilo de descrição de um caso de uso pode variar:

- ✓ **Formato**
- ✓ **Grau de detalhamento**
- ✓ **Grau de abstração**



Formato

O formato de uma descrição de caso de uso diz respeito à estrutura utilizada para organizar a sua narrativa textual.

Os formatos comumente utilizados são:

- ✓ Contínuo**
- ✓ Numerado**
- ✓ Tabular**



Formato Contínuo – Exemplo Realizar Saque

Este caso de uso inicia quando o cliente chega ao caixa eletrônico e insere seu cartão. O sistema requisita a senha do cliente. Após o cliente fornecer sua senha e esta ser validada, o sistema exibe as opções de operações possíveis. O cliente opta por realizar um saque. Então o sistema requisita o total a ser sacado. O cliente fornece o valor da quantidade que deseja sacar. O sistema fornece a quantia desejada e imprime o recibo para o cliente. O cliente retira a quantia e o recibo, e o caso de uso termina.



Formato Numerado - Exemplo Realizar Saque

- 1) Cliente insere seu cartão no caixa eletrônico.**
- 2) Sistema apresenta solicitação de senha.**
- 3) Cliente digita senha.**
- 4) Sistema valida a senha e exibe menu de operações disponíveis.**
- 5) Cliente indica que deseja realizar um saque.**
- 6) Sistema requisita o valor da quantia a ser sacada.**
- 7) Cliente fornece o valor da quantia que deseja sacar.**
- 8) Sistema fornece a quantia desejada e imprime o recibo para o cliente.**
- 9) Cliente retira a quantia e o recibo, e o caso de uso termina.**



Formato Tabular - Exemplo Realizar Saque

Cliente	Sistema
Insere seu cartão no caixa eletrônico.	Apresenta solicitação de senha.
Digita senha.	Valida senha e exibe menu de operações disponíveis.
Solicita realização de saque.	Requisita a quantia a ser sacada.
Fornece o valor da quantia que deseja sacar.	Fornece a quantia desejada e imprime o recibo para o cliente
Retira a quantia e o recibo.	



Grau de Detalhamento

O grau de detalhamento a ser utilizado na descrição de um caso de uso pode variar desde o mais sucinto até a descrição com vários detalhes (expandido).

Um caso de uso sucinto descreve as interações entre ator e sistema sem muitos detalhes.

Um caso de uso expandido descreve as interações em detalhes.



Grau de Abstração

O grau de abstração de um caso de uso diz respeito à existência ou não de menção a aspectos relativos à tecnologia durante a descrição desse caso de uso.

Essencial: é completamente desprovido de características tecnológicas.

Real: a descrição das interações cita detalhes da tecnologia a ser utilizada na interação entre o ator e o sistema.



Descrição Essencial – Exemplo (numerada)

- 1) Cliente fornece sua identificação.**
- 2) Sistema identifica o usuário.**
- 3) Sistema fornece opções disponíveis para movimentação da conta.**
- 4) Cliente solicita o saque de determinada quantia.**
- 5) Sistema requisita o valor da quantia a ser sacada.**
- 6) Cliente fornece o valor da quantia que deseja sacar.**
- 7) Sistema fornece a quantia desejada.**
- 8) Cliente retira dinheiro e recibo, e o caso de uso termina.**



Descrição Real – Exemplo já utilizado (numerada)

- 1) Cliente insere seu cartão no caixa eletrônico.**
- 2) Sistema apresenta solicitação de senha.**
- 3) Cliente digita senha.**
- 4) Sistema valida a senha e exibe menu de operações disponíveis.**
- 5) Cliente indica que deseja realizar um saque.**
- 6) Sistema requisita o valor da quantia a ser sacada.**
- 7) Cliente fornece o valor da quantia que deseja sacar.**
- 8) Sistema fornece a quantia desejada e imprime o recibo para o cliente.**
- 9) Cliente retira a quantia e o recibo, e o caso de uso termina.**



Casos de Uso contém Requisitos de Software

Cada caso de uso:

- Descreve ações que o sistema faz para entregar algo de valor para um agente.
- Mostra a funcionalidade do sistema que o agente usa.
- Modela um diálogo entre o sistema e os agentes.
- É um completo e significativo fluxo de eventos da perspectiva de um agente em particular.



Benefícios dos Casos de Uso

- Dá um contexto para os requisitos

Coloca os requisitos do sistema em sequências lógicas.

Ilustra o porque o sistema é necessário.

Ajuda a verificar se todos os requisitos foram capturados.

- São fáceis de entender.

Usam terminologia que clientes e usuários entendem.

Descreve a história concreta de uso do sistema.

Verifica o entendimento dos envolvidos.

- Facilita o acordo com os clientes.

- Facilita o reuso: teste, documentação e design.



Engenharia de Software II

**Aula: Elementos do Caso de
Uso e Notação UML**



Prof. Anderson Augusto Bosing

“Um caso de uso é a especificação de uma sequência completa de interações entre um sistema e um ou mais agentes externos a esse sistema.”

O que são os agentes externos? Onde vivem? Do que se alimentam ?



Atores

Qualquer elemento externo ao sistema que interage com o mesmo é denominado ator.

“Externo” nessa definição indica que atores não fazem parte do sistema.

“Interage” significa que um ator troca informações com o sistema (envia ou recebe informações).



Atores

Podem ser Categorizados em:

Um **ator humano** é uma pessoa física, que no diagrama deve possuir como nome o papel que a pessoa executa no contexto empresarial onde o sistema será utilizado. Por exemplo: Cliente, Fornecedor, Atendente.

Um **ator sistêmico** é um sistema, ou módulo de um sistema, ou componente de um sistema, que realizará a execução da funcionalidade que está especificada pelo caso de uso. No diagrama deve possuir seu nome de fato (se o ator é o sistema “Disparador de rotinas batch” por exemplo, este deve ser o nome do ator sistêmico).



Atores

Exemplos de Atores:

- **Cargos (Ex. Empregado, Cliente, Gerente, Vendedor).**
- **Organizações ou divisões de uma organização (Ex. Fornecedor, Administradora de Cartões, Almoxarifado).**
- **Outros sistemas de software (Ex. Sistema de Cobrança, Sistema de Estoque de Produtos).**
- **Equipamentos com os quais o sistema deve se comunicar (Ex. Leitora de Código de Barras, Sensor).**



Atores

Um ator corresponde a um papel representado em relação ao sistema.

Exemplos:

O mesmo indivíduo pode ser o Cliente que compra mercadorias e o Vendedor que processa vendas.

Uma pessoa pode representar o papel de Funcionário de um banco que realiza a manutenção de um caixa eletrônico, mas também pode ser o Cliente do banco que realiza o saque.



Atores

É uma boa prática de modelagem fazer com que o nome dado a esse ator lembre o seu papel, em vez de lembrar quem o representa.

Exemplos de bons nomes para atores:

✓ Cliente, Estudante, Fornecedor etc.

Exemplos de maus nomes para atores:

✓ João, ANVISA etc.



Atores

Um ator pode participar de muitos casos de uso (situação muito comum na prática).

Do mesmo modo, um caso de uso pode envolver a participação de vários atores, nesses casos, os atores podem ter duas classificações:

Primários ou Secundários.



Atores

Um ator primário é aquele que inicia uma sequência de interações de um caso de uso.

São eles os agentes externos para os quais o caso de uso traz benefício direto.

As funcionalidades principais do sistema são definidas tendo em mente os objetivos dos atores primários.



Atores

Já um ator secundário supervisiona, opera, mantém ou auxilia na utilização do sistema pelo atores primários.

Atores secundários existem apenas para que os atores primários possam utilizar o sistema.



Atores

Exemplo:

Em um navegador de internet.

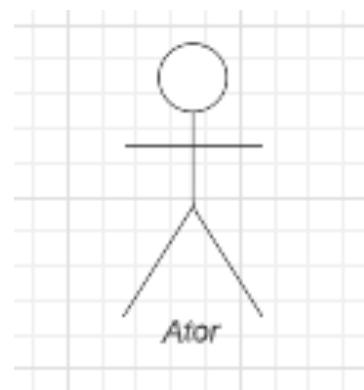
Para que o Usuário (ator primário) requisite uma página ao programa, outro ator (secundário) está envolvido: o Servidor Web.

O ator primário Usuário é auxiliado pelo secundário, Servidor Web, uma vez que é através deste último que o primeiro consegue alcançar seu objetivo.



Atores

Notação de um Ator:



Caso de Uso

Um caso de uso define uma sequência de ações a serem executadas pelo sistema para produzir um resultado de valor observável para um ator.(RUP)

Usamos casos de uso para captar os comportamentos pretendidos de um sistema, sem especificar como esse comportamento é implementado.



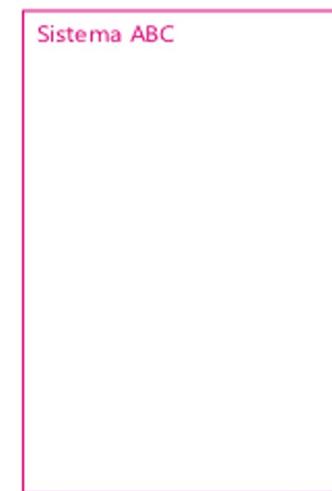
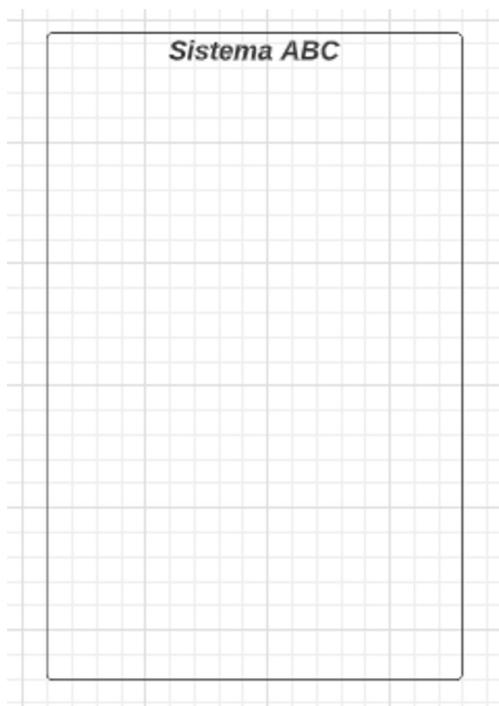
Caso de Uso

Notação de um Caso de Uso:



Sistema

Representa a fronteira do sistema ou componentes de sistemas e subsistemas.



Como os Atores e Casos de Uso se relacionam ?

Um ator deve estar relacionado a um ou mais casos de uso do sistema.

Além disso, pode haver relacionamentos entre os casos de uso ou entre os atores de um sistema.

A UML define os seguintes relacionamentos para o modelo de casos de uso: comunicação, inclusão, extensão e generalização.



Associação de Comunicação

Um relacionamento de comunicação informa a que caso e uso o ator está associado.

Isso significa que esse ator interage (troca informações) com o sistema com ajuda daquele caso de uso.

O relacionamento de comunicação é, de longe, o mais comumente utilizado de todos.



Associações

Notação de um Caso de Uso:



Associação
bidirecional

Associação
unidirecional



Relacionamento de inclusão.

Na modelagem UML, um relacionamento de inclusão é aquele no qual um caso de uso (o caso de uso base) inclui a funcionalidade de outro caso de uso (o caso de uso de inclusão). O relacionamento de inclusão suporta a reutilização da funcionalidade em um modelo de caso de uso.

O relacionamento de inclusão existe somente entre casos de uso. Quando dois ou mais casos de uso incluem uma sequência comum de interações, essa sequência comum pode ser descrita em outro caso de uso.

A partir daí, vários casos de uso do sistema podem incluir o comportamento desse caso de uso comum.



Relacionamento de inclusão.

Exemplo: Sistema bancário

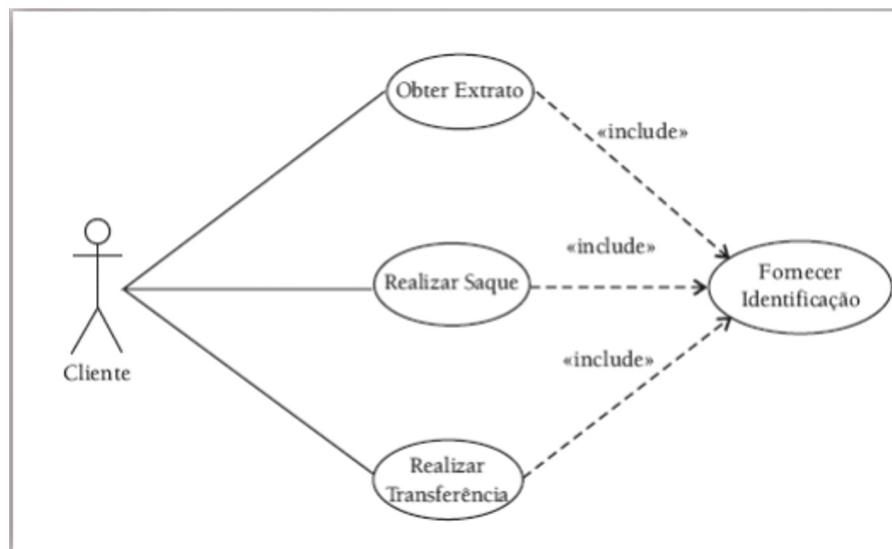
Esses casos de uso têm uma sequência de interações em comum: a que valida a senha do cliente do banco.

Essa sequência de interações em comum pode ser descrita em um caso de uso Fornecer Identificação.

Alguns casos de uso desse sistema são Obter Extrato, Realizar Saque e Realizar Transferência.



Relacionamento de inclusão.



Relacionamento de extensão.

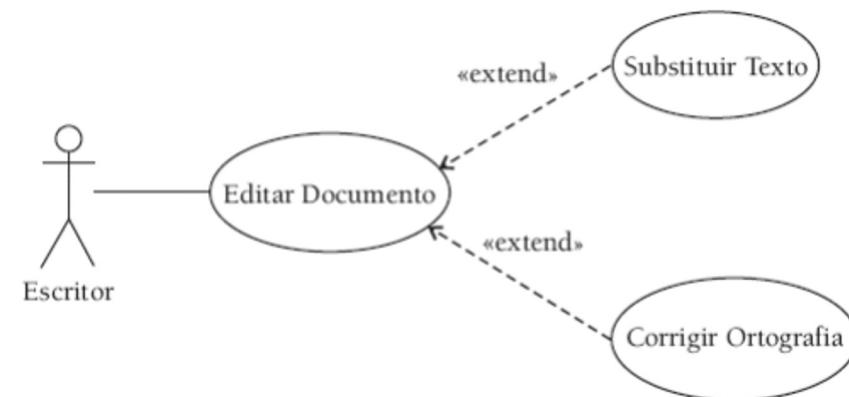
Na modelagem UML, é possível utilizar um relacionamento de extensão para especificar que um caso de uso (extensão) estende o comportamento de outro caso de uso (base). Esse tipo de relacionamento revela detalhes sobre um sistema ou aplicativo que normalmente estão ocultos em um caso de uso.

Cada uma dessas diferentes sequências representa um comportamento eventual, ou seja, um comportamento que só ocorre sob certas condições, ou cuja realização depende da escolha do ator.



Relacionamento de extensão.

Mostra que os casos de uso Corrigir Ortografia e Substituir Texto têm sequências de interações que são eventualmente utilizadas quando o ator Escritor estiver usando o caso de uso Editar Documento

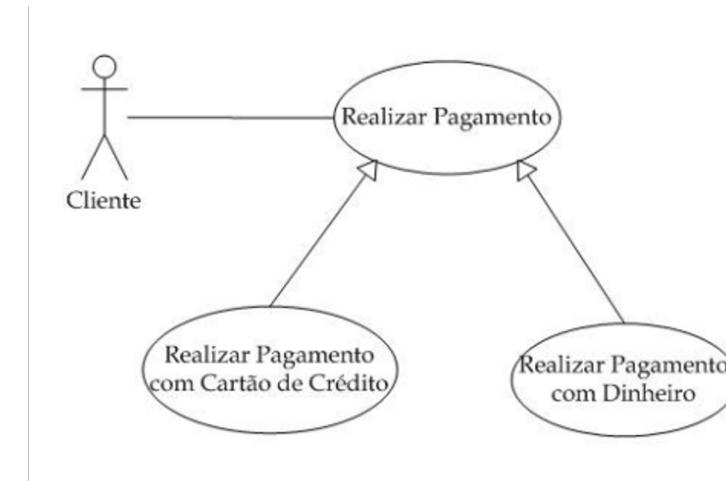


Fonte: Bezerra (2007)

Relacionamento de generalização.

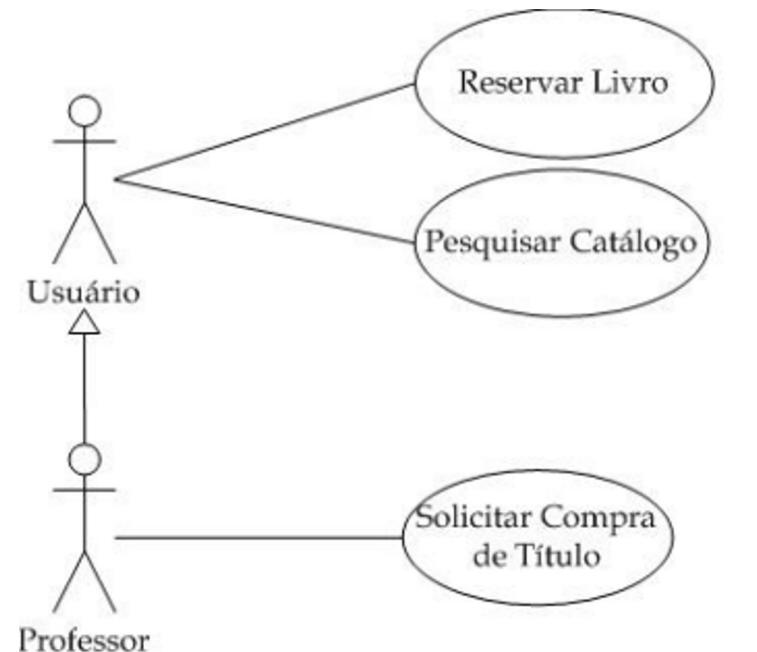
Na modelagem UML, um relacionamento de generalização é aquele no qual um elemento de modelo (o filho) tem como base outro elemento de modelo (o pai). Os relacionamentos de generalização são utilizados em diagramas de classe, componente, implementação e caso de uso para indicar que o filho recebe todos os atributos, operações e relacionamentos definidos no pai.

Realizar Pagamento com Cartão de Crédito e Realizar Pagamento com Dinheiro herdam características em relação ao caso de uso Realizar Pagamento, especializando o seu comportamento.



Relacionamento de generalização.

A generalização entre os atores Usuário e Professor indica que este último pode interagir com qualquer caso de uso que um usuário comum interage.



Resumo das Notações

