

ANALISE E PROJETOS DE SISTEMAS

**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.



ANALISE E PROJETOS DE SISTEMAS

**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).



ANALISE E PROJETOS DE SISTEMAS

**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: 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: Especificação de
Caso de Uso e Regras de
Negócios.**



Prof. Anderson Augusto Bosing

Revisando a Ultima Aula

- **Estudo de Caso de Diagrama de Caso de Uso e Modelo de Caso de Uso.**
- **Sentido das ligações.**
- **Nomes dos Casos de uso em verbos do infinitivo.**



Regras de Negócios

São padrões que condicionam o funcionamento do negócio, sendo comumente aplicadas no contexto da arquitetura de softwares.

No entanto, não ficam restritas a isso. Você pode e deve adotá-las como parâmetro para todas as atividades produtivas da empresa.



Regras de Negócios

Em essência são regras da organização que está construindo o sistema e que refletem na forma como o sistema deve funcionar.



Regras de Negócios

Um negócio funciona por processos que, por sua vez, são formados por atividades relacionadas entre si. As funções das áreas de compras, estoque, logística, finanças, vendas e marketing, por exemplo, compõem um processo de fornecimento de um produto ao cliente.



Regras de Negócios

Dentro destes processos, existem regras que devem ser seguidas durante a execução das atividades, que ajudam a definir **COMO** as operações devem ser realizadas e gerenciadas, por **QUEM**, **QUANDO**, **ONDE** e **POR QUÊ**, de acordo com a definição do BPM CBOK. Podemos dizer que as regras de negócio são limites impostos às operações, de forma que elas sigam corretamente em direção às políticas e aos objetivos da instituição.



Exemplos de Regras de Negócios

Em um controle de qualidade de granja, pode-se dizer que a cada 100 ovos impróprios para consumo, o lote será descartado.

Em um banco, clientes com faturamento mensal de mais de R\$ 25 mil e CPF sem restrições, serão atendidos pelo gerente Premium pessoa física.



Exemplos de Regras de Negócios

Para conclusão de licitações, devem ser feitos três orçamentos e o vencedor será sempre o de menor preço final.

Em um processo de seleção de RH, o candidato só pode ser aprovado se tiver mais de 5 anos de experiência na área, diploma de pós-graduação, espanhol fluente e pretensão salarial abaixo de R\$ 8.000,00.



Especificação de Casos de Uso

Como vimos, caso de uso é definido através da descrição das interações que ocorrem entre as entidades externas e o sistema.

A descrição ou especificação é uma forma de validar se o entendimento de analistas de sistemas e usuários são os mesmos; e serve de base para a criação dos planos de testes do sistema.

As especificações dos casos de uso são feitas normalmente em linguagem natural e existem diversões padrões estruturais, mas não um único padronizado pela UML.



Especificação de Casos de Uso

Cada caso de uso possui um conjunto de ações que precisam ser executadas para que o objetivo da funcionalidade seja alcançado;

No caso de efetuar venda: identificar o vendedor, identificar o produto, a quantidade vendida, etc;

Essas ações constituem os fluxos que são realizados pelos casos de uso para disponibilizar o resultado desejado.



Pré-Condições

Pré-condições:

São condições que devem ser verdadeiras para um caso de uso se iniciar.

Normalmente envolvem um estado de sistema, uma condição temporal ou estão relacionados a permissões de usuário;

Exemplos:

- 1. Solicitação de mudança deve estar no estado “Pendente Aprovação”**
- 2. Deve ser 01:00**
- 3. O usuário logado deve ter perfil “Administrador”**



Fluxos de Eventos

- Provavelmente a parte mais importante de um caso de uso.
- Define a sequencia de ações entre o ator e o sistema.
- É uma sequencia de comandos declarativos que descreve as etapas de execução de um caso de uso.



Fluxos de Eventos

O fluxo de eventos de um caso de uso é composto por:

- **Fluxo Principal:** descreve a funcionalidade principal do caso de uso, quando nenhum desvio é tomado.
- **Fluxos Alternativos:** descrevem desvios pré-definidos ao fluxo principal.
- **Fluxos Excepcionais:** descrevem desvios do fluxo principal que de alguma forma geram exceções na aplicação como erros não previstos ou até mesmo validações.



Exemplo de Fluxo Principal

1. [EV] O Cliente insere o cartão no caixa.
2. [RS] O sistema realiza a leitura do cartão e solicita a senha ao cliente.
3. [EV] O Cliente entra com a senha.
4. [RS] O sistema valida a senha e exibe as opções de menu.
5. [EV] O cliente escolhe a opção “Sacar”.
6. [RS] O sistema solicita ao cliente a quantidade a ser sacada.
7. [EV] O cliente informa a quantia desejada e clica na opção “Confirmar”.



Exemplo de Fluxo Principal

8. [RS] O sistema verifica se o cliente possui saldo suficiente para sacar a quantia informada e entrega a quantia sacada.
9. [EV] Cliente retira a quantidade solicitada.
10. [RS] Sistema solicita a remoção do cartão do cliente e a operação é finalizada.



Exemplo de Fluxo Alternativo

8a.1 [RS] O sistema verifica que o cliente não possui saldo suficiente e exibe um menu para realização de uma saque de adiantamento ao salário do cliente.

8a.2 [EV] O cliente aciona a opção de “Aceitar”.

8a.3 Retorna ao fluxo principal no passo 9.



Exemplo de Fluxo Excepcional

A senha informada é incorreta.

4a.1 [RS] O sistema apresenta a mensagem de erro “A senha informada está incorreta. Verifique!”

4a.2 Retorna ao fluxo principal no passo 3.



Mas o que é [EV] e [RS] ?

WAZLAWICK (2004) sugeriu uma forma de descrição dos casos de uso, nomeado como passos obrigatórios. Estes passos obrigatórios incluem os passos que indicam de alguma forma uma troca de informação entre os atores e o sistema.

Existem dois tipos de passos obrigatórios:

- a) Eventos de Sistema[EV]: passos que mostram alguma informação indo de um ator para o sistema.**
- b) Respostas do Sistema[RS]: passos que mostram alguma informação vindo do sistema para o(s) ator(es)**



Descrição Essencial e Real do Caso de Uso

A descrição essencial é descrita por COCKBURN (2005) como “deixe de fora a interface do usuário; focalize a intenção”.

WAZLAWICK (2004, p.63) apresenta como “o que” acontece entre o usuário e o sistema e não “como” isto ocorre.



Descrição Essencial e Real do Caso de Uso

Na descrição real os casos de uso são descritos em estilo real, ou seja, “como” realmente irá acontecer, sua realização concreta (WAZLAWICK, 2004, p.63-67).

Nesse caso descrevemos o processo em termos de projeto real, comprometido com tecnologias de desenvolvimento, interfaces de entrada e saída. Fluxos de eventos representados diretamente no sistema.



Engenharia de Software II

Aula:
A Linguagem UML.
Visões de um Sistema.
Diagramas UML.



Prof. Anderson Augusto Bosing

A Linguagem UML

A Unified Modeling Language (UML, Linguagem de Modelagem Unificada) é “uma linguagem-padrão para descrever/documentar projeto de software.

A UML pode ser usada para visualizar, especificar, construir e documentar os artefatos de um sistema de software.



A Linguagem UML

Motivação

Em outras palavras, assim como os arquitetos criam plantas e projetos para serem usados por uma empresa de construção, os arquitetos de software criam diagramas UML para ajudar os desenvolvedores de software a construir o software. Se você entender o vocabulário da UML (os elementos visuais do diagrama e seus significados), poderá facilmente entender e especificar um sistema e explicar o projeto desse sistema para outros interessados.



A Linguagem UML

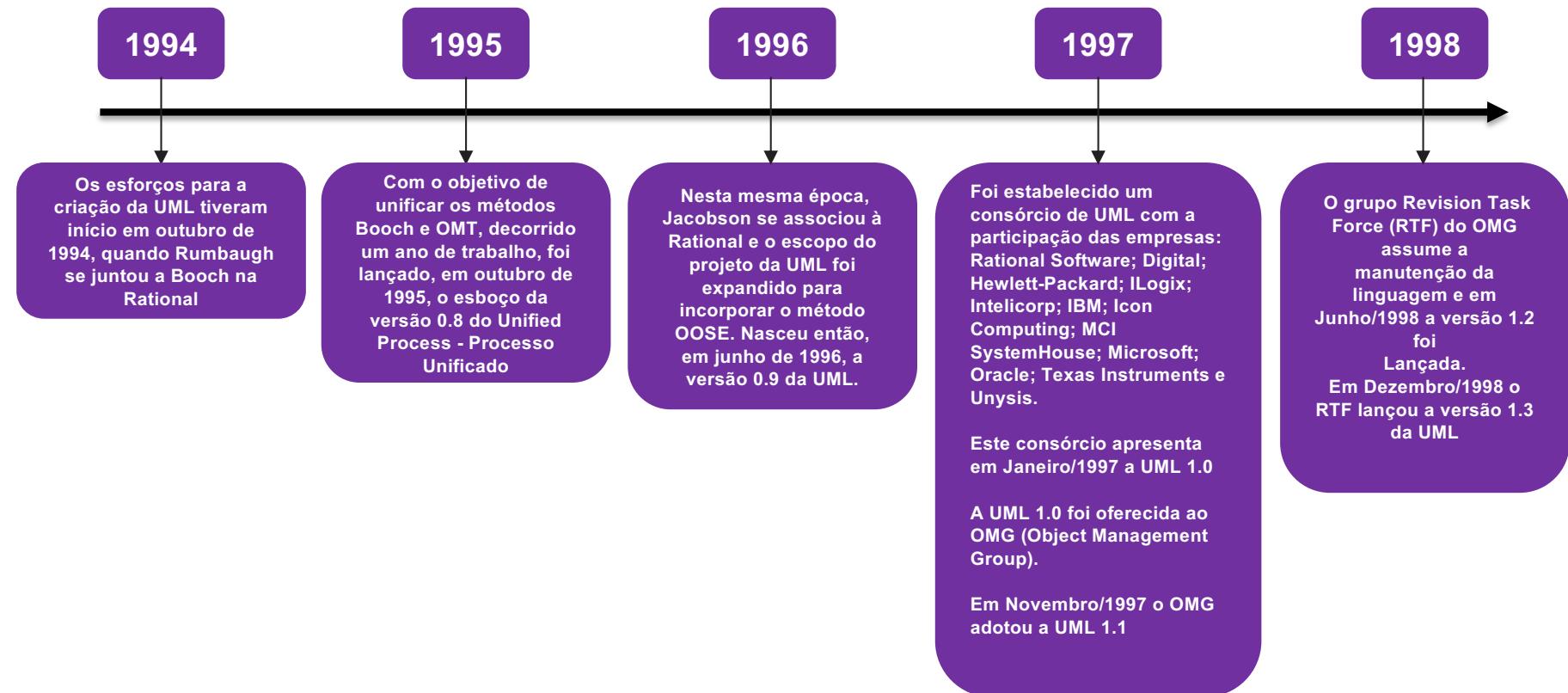
A construção da UML teve muitos contribuintes, mas os principais atores no processo foram Grady Booch, James Rumbaugh e Ivar Jacobson.

Em 1997 a UML foi aprovada como padrão pelo OMG(Object Management Group).

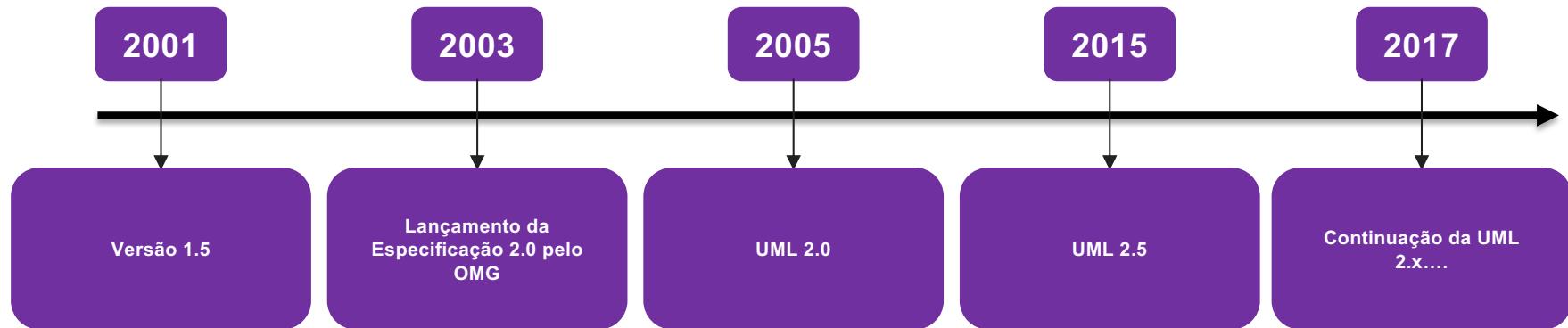
Desde então, a UML tem tido grande aceitação pela comunidade de desenvolvedores de sistemas.



A Linguagem UML Timeline



A Linguagem UML Timeline

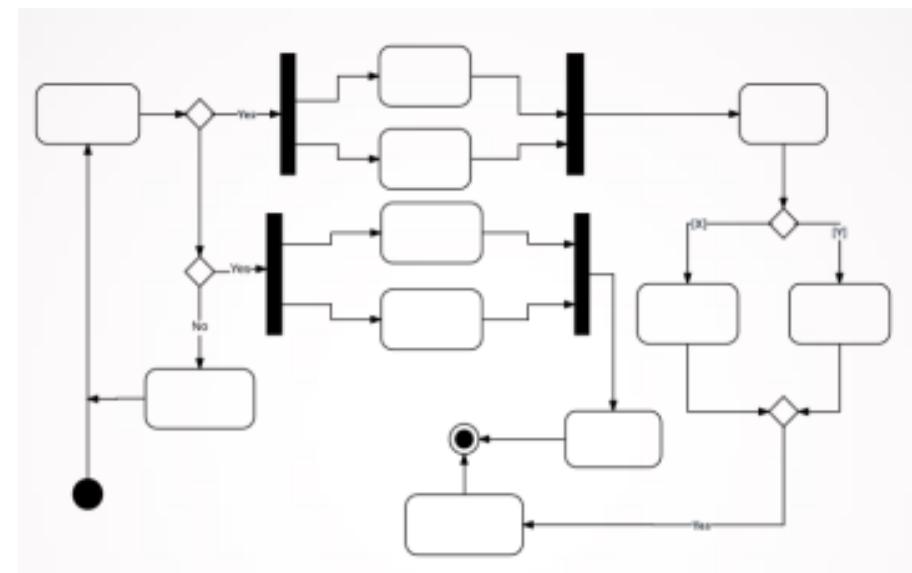


Visão Geral da UML



A UML é uma linguagem visual para modelar sistemas orientados a objetos.

Por meio dos elementos gráficos definidos nesta linguagem pode-se construir diagramas que representam diversas perspectivas de um sistema.



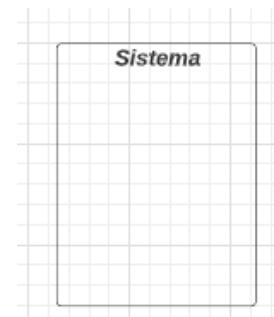
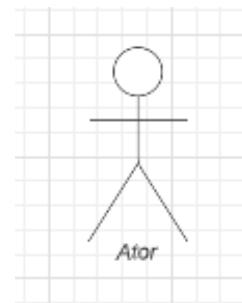


Visão Geral da UML

Cada **elemento gráfico** da UML possui uma sintaxe e uma semântica.

A **sintaxe** de um elemento corresponde à forma predeterminada de desenhar o elemento.

A **semântica** define o que significa o elemento e com que objetivo ele deve ser utilizado.



Visão Geral da UML

A UML é independente tanto de linguagens de programação quanto de processos de desenvolvimento.

Esse é um fator importante para a utilização da UML, pois diferentes sistemas de software requerem abordagens diversas de desenvolvimento.



Visão Geral da UML

O desenvolvimento de um software complexo demanda que seus desenvolvedores tenham a possibilidade de examinar e estudar esse sistema a partir de perspectivas diversas.

Os autores da UML sugerem que um sistema pode ser descrito por cinco visões:

- Visão de Casos de Uso.
- Visão de Projeto.
- Visão de implementação.
- Visão de Implantação.
- Visão de Processo.



Visão Geral da UML

Visão de Caso de Uso

Descreve o sistema de um ponto de vista externo como um conjunto de interações entre o sistema e os agentes externos ao sistema.

Esta visão é criada em um estágio inicial e direciona o desenvolvimento das outras visões do sistema.



Visão Geral da UML

Visão de Projeto(Lógica)

Enfatiza as características do sistema que dão suporte, tanto estrutural quanto comportamental, às funcionalidades externamente visíveis do sistema.

Ligada ao problema do negócio.



Visão Geral da UML

Visão de Implementação

Abrange o gerenciamento de versões do sistema, construídas pelo agrupamento de módulos (componentes) e subsistemas.



Visão Geral da UML

Visão de Implantação

Corresponde à distribuição física do sistema em seus subsistemas e à conexão entre essas partes.



Visão Geral da UML

Visão de Processo

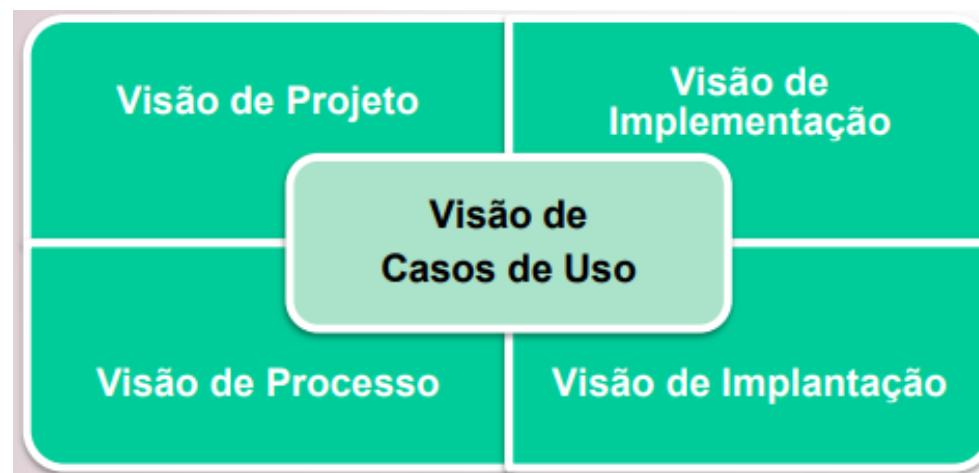
Esta visão enfatiza as características de concorrência (parallelismo), sincronização e desempenho do sistema.



Visão Geral da UML

Visão da UML

Dependendo das características do sistema, nem todas as visões precisam ser construídas.



Visão Geral da UML

Visão da UML

Um processo de desenvolvimento que utilize a UML envolve a criação de diversos documentos.

Esses documentos podem ser textuais ou gráficos.

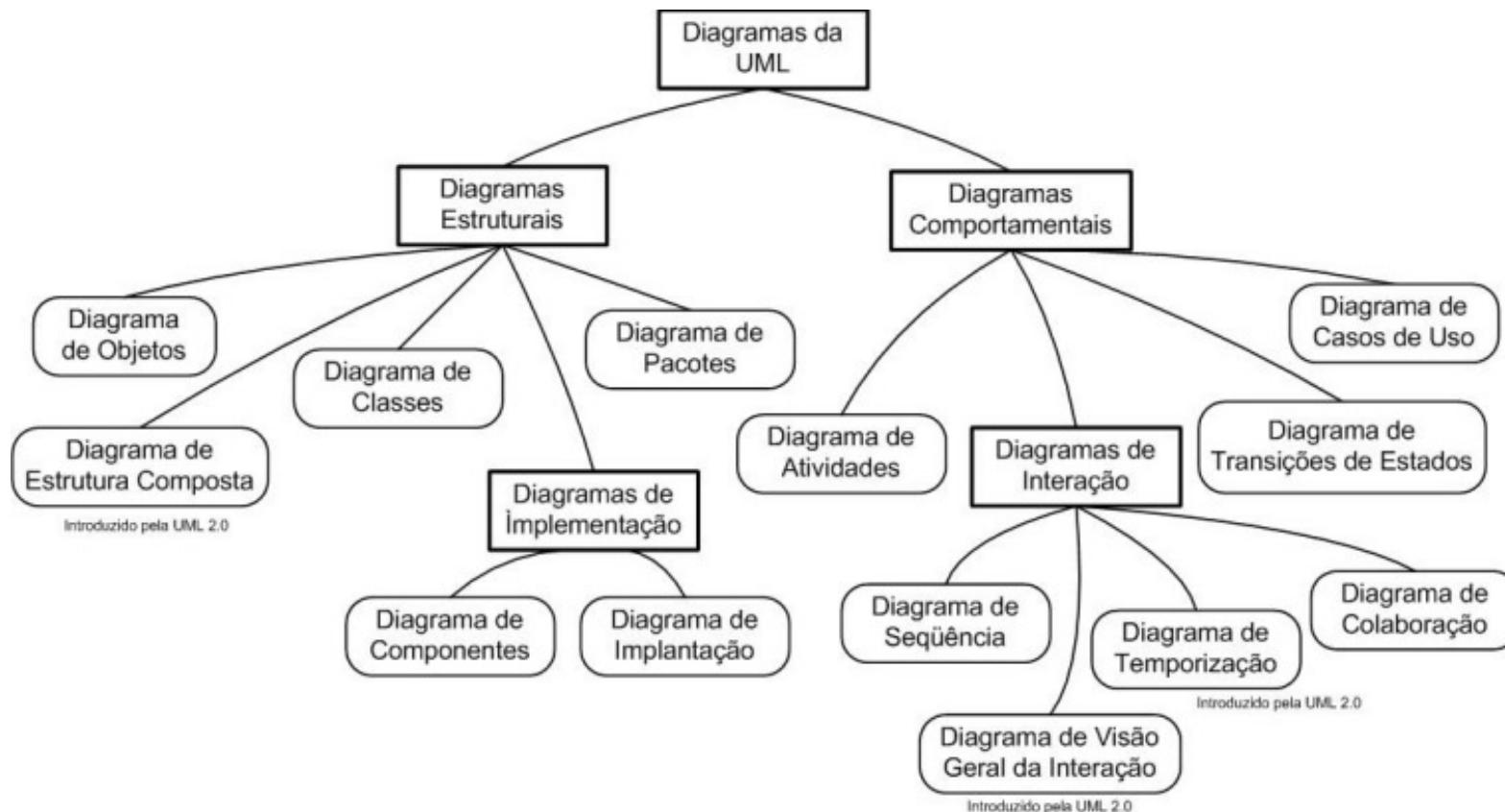
Eles são denominados artefatos de software, ou simplesmente artefatos.

São os artefatos que compõem as visões do sistema.



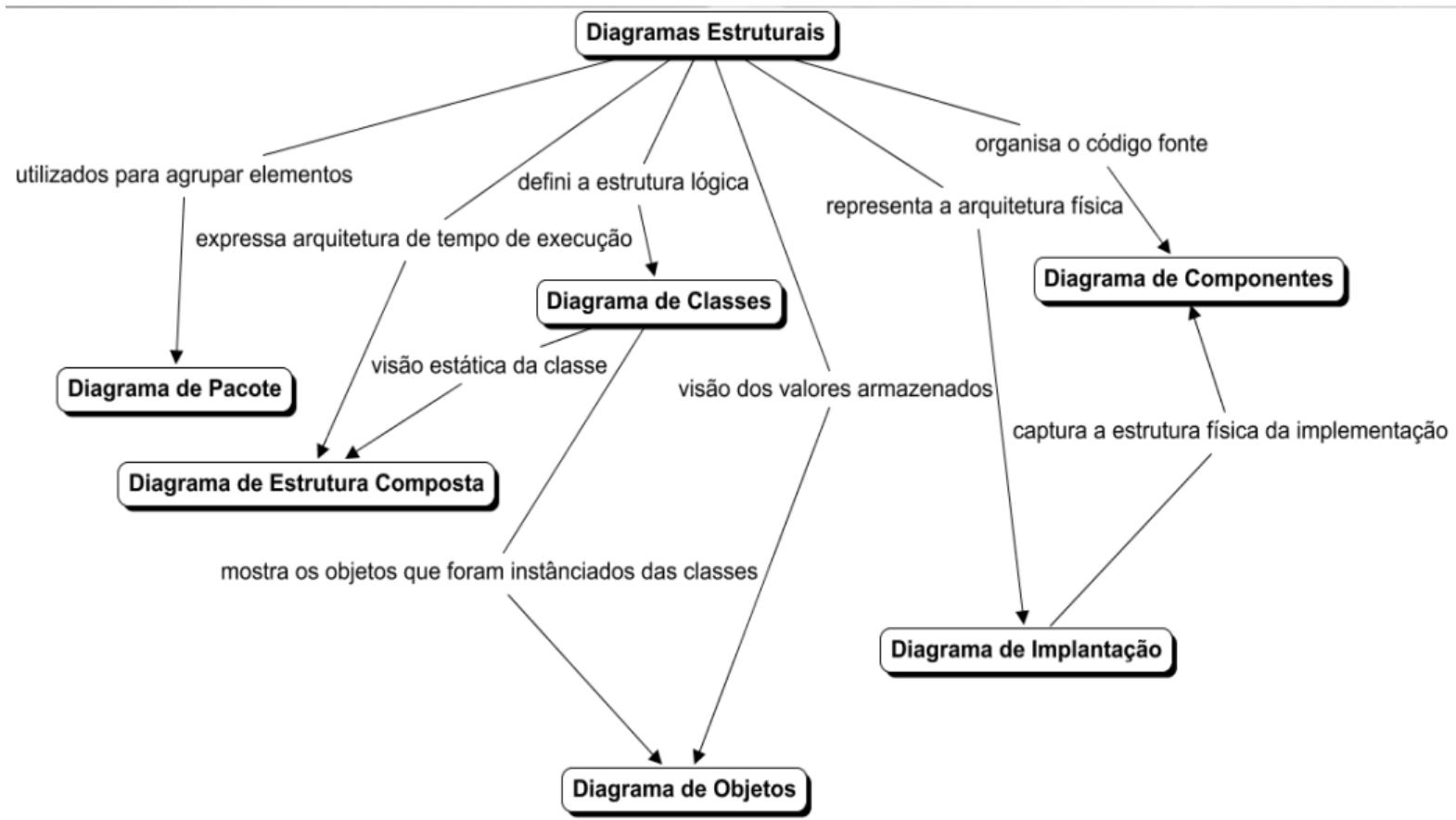
Visão Geral da UML

Artefatos da UML



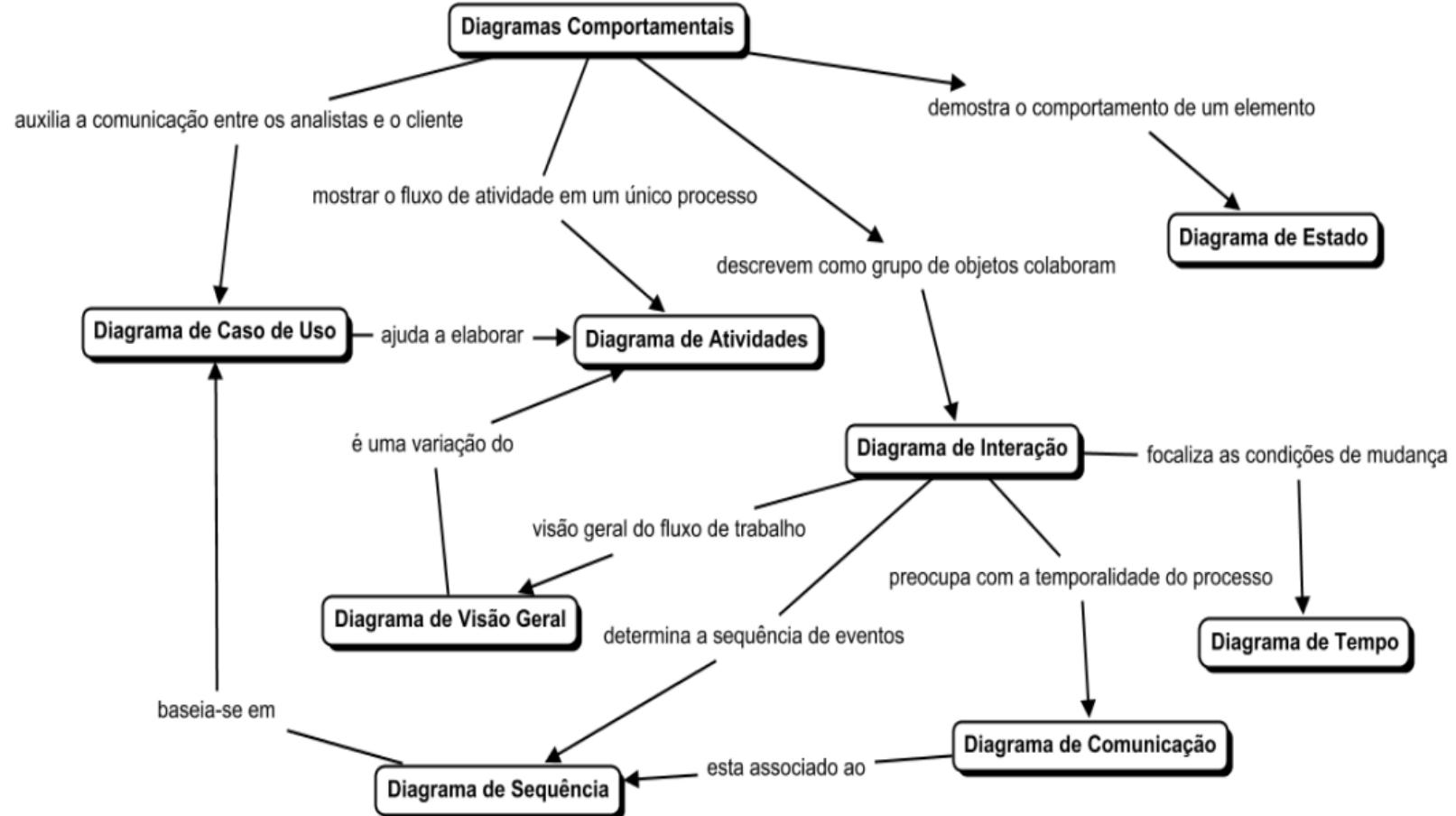
Visão Geral da UML

Mapa Conceitual - Artefatos da UML



Visão Geral da UML

Mapa Conceitual - Artefatos da UML



A Linguagem UML Site, Certificações e o futuro

<https://www.uml.org/>



Engenharia de Software II

Aula: Diagrama de Classes.



Prof. Anderson Augusto Bosing

Diagrama de Classes

O diagrama de classes é um dos mais importantes e utilizados da UML.

Seu principal enfoque está em permitir a visualização das **classes que compõem o sistema com seus respectivos atributos e métodos**, bem como em demonstrar como as classes do diagrama **se relacionam, complementam e transmitem informações entre si**.

Esse diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em **como definir a estrutura lógica delas**.



Diagrama de Classes Atributos e Métodos

Classes costumam ter **atributos** como já explicado anteriormente armazenam os dados dos objetos da classe.

Classes também costumam possuir **métodos**, também chamados operações, que são as funções que uma instância da classe pode executar.

Embora os métodos sejam declarados no diagrama de classes, identificando os possíveis parâmetros que são por eles recebidos e os possíveis valores por eles retornados, **o diagrama de classes não se preocupa em definir as etapas que tais métodos deverão percorrer quando forem chamados.**



Diagrama de Classes

Classes

ContaComum
numeroConta: long
aberturaConta: Date
fechamentoConta: Date
situacaoConta: int
senhaConta: int
saldoConta: double
+ abrirConta(): int
+ consultarConta(): int
+ validarSenha(): int
+ emitirSaldo(): double
+ emitirExtrato(): String
+ sacarValor(): int
+ depositarValor(): int
+ encerrarConta(): int

Uma classe, na linguagem UML, é representada como um retângulo com até três divisões.

A primeira contém a descrição ou nome da classe, que, nesse exemplo, é ContaComum.



Diagrama de Classes

Classes

ContaComum
numeroConta: long
aberturaConta: Date
fechamentoConta: Date
situacaoConta: int
senhaConta: int
saldoConta: double
+ abrirConta(): int
+ consultarConta(): int
+ validarSenha(): int
+ emitirSaldo(): double
+ emitirExtrato(): String
+ sacarValor(): int
+ depositarValor(): int
+ encerrarConta(): int

A segunda armazena os atributos e seus tipos de dados. A classe **ContaComum** contém os atributos **numeroConta**, do tipo **long**; **aberturaConta** e **fechamentoConta**, do tipo **Date**; **situacaoConta** e **senhaConta**, do tipo **int**; e **saldoConta**, do tipo **double**.

Diagrama de Classes

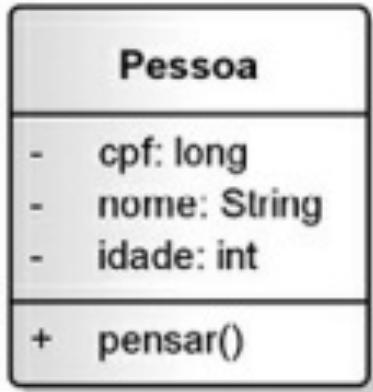
Classes

ContaComum
numeroConta: long
aberturaConta: Date
fechamentoConta: Date
situacaoConta: int
senhaConta: int
saldoConta: double
+ abrirConta(int): long
+ consultarConta(long): int
+ validarSenha(int): int
+ emitirSaldo(): double
+ emitirExtrato(Date, Date): String
+ sacarValor(double): int
+ depositarValor(long, double): int
+ encerrarConta(long): int

Finalmente, a terceira divisão lista os métodos da classe. Nesse exemplo, a classe ContaComum contém os métodos **abrirConta**, **consultarConta**, **validarSenha**, **emitirSaldo**, **emitirExtrato**, **sacarValor**, **depositarValor** e **encerrarConta**. Os métodos são mapeados seguindo o padrão: **nomeMetodo(ParamEntrada):TipoRetor** no

Diagrama de Classes

Tipos de Visibilidade

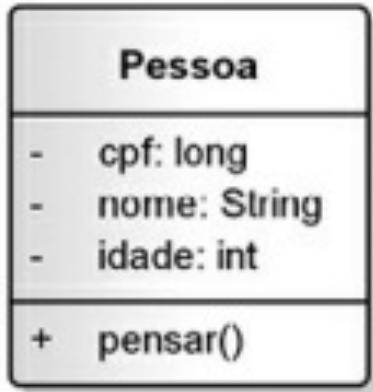


A visibilidade é utilizada para indicar o nível de acessibilidade de um determinado atributo ou método, sendo representada à esquerda destes.

Existem basicamente 3 modos de visibilidade: público, protegido, privado.

Diagrama de Classes

Tipos de Visibilidade



A visibilidade privada é representada por um símbolo de menos (-) e significa que somente os objetos da classe detentora do atributo ou método poderão enxergá-lo.

- A visibilidade protegida é representada pelo símbolo de sustenido (#) e determina que, além dos objetos da classe detentora do atributo ou método, também os objetos de suas subclasses poderão ter acesso a este.
- A visibilidade pública é representada por um símbolo de mais (+) e determina que o atributo ou método pode ser utilizado por qualquer objeto.

Diagrama de Classes

Características Extras dos Atributos

ContaComum
numeroConta: long
/aberturaConta: Date
/fechamentoConta: Date [0..1]
situacaoConta: int = 1
senhaConta: int
/saldoConta: double = 0
+ abrirConta(int): long
+ consultarConta(long): int
+ validarSenha(int): int
+ emitirSaldo(): double
+ emitirExtrato(Date, Date): String
+ sacarValor(double): int
+ depositarValor(long, double): int
+ encerrarConta(long): int

Os atributos aberturaConta, encerramentoConta e saldoConta têm uma barra (/) antes de seus nomes, significando que os valores desses atributos sofrem algum tipo de cálculo.

No caso das datas, quando for realizada a operação de abertura de conta, o valor da data de abertura será tomado da data do sistema, o mesmo ocorrendo com o valor da data de encerramento quando do encerramento da conta.



Diagrama de Classes

Características Extras dos Atributos

ContaComum
numeroConta: long
/aberturaConta: Date
/fechamentoConta: Date [0..1]
situacaoConta: int = 1
senhaConta: int
/saldoConta: double = 0
+ abrirConta(int): long
+ consultarConta(long): int
+ validarSenha(int): int
+ emitirSaldo(): double
+ emitirExtrato(Date, Date): String
+ sacarValor(double): int
+ depositarValor(long, double): int
+ encerrarConta(long): int

Pode-se perceber também que o atributo fechamentoConta, após a definição de seu tipo (Date), contém os valores [0..1]. Isso é chamado multiplicidade e, nesse contexto, significa que existirão, no mínimo, nenhuma (0) e, no máximo, uma (1) data de encerramento

Diagrama de Classes

Características Extras dos Atributos

ContaComum
numeroConta: long
/aberturaConta: Date
/fechamentoConta: Date [0..1]
situacaoConta: int = 1
senhaConta: int
/saldoConta: double = 0
+ abrirConta(int): long
+ consultarConta(long): int
+ validarSenha(int): int
+ emitirSaldo(): double
+ emitirExtrato(Date, Date): String
+ sacarValor(double): int
+ depositarValor(long, double): int
+ encerrarConta(long): int

Também podemos verificar que o valor inicial dos atributos situacaoConta e saldoConta será, respectivamente, 1 e 0 quando da instanciação de um objeto dessa classe durante a abertura de uma nova conta. Dessa forma, sempre que uma nova conta for aberta, sua situação inicial terá o valor 1 (está ativa) e o seu saldo permanecerá com valor 0 até que um depósito seja realizado.



Vamos Praticar?

```
public class Calculadora {  
  
    private double resultado = 0;  
  
    public double somar(double a, double b) {  
        resultado = a + b;  
        return resultado;  
    }  
  
    public double diminuir(double a, double b) {  
        resultado = a - b;  
        return resultado;  
    }  
  
    public double dividir(double a, double b) {  
        resultado = a / b;  
        return resultado;  
    }  
  
    public double multiplicar(double a, double b) {  
        resultado = a * b;  
        return resultado;  
    }  
}
```

Representar a Classe Calculadora no diagrama de classes.

<https://online.visual-paradigm.com/>



Vamos Praticar?

Representar a Classe Calculadora no
diagrama de classes.

<https://online.visual-paradigm.com/>



Diagrama de Classes Interfaces

Notações para representar interfaces na UML:

A primeira notação é a mesma para classes. São exibidas as operações que a interface especifica. Deve ser usado o estereótipo <<interface>>.

A segunda notação usa um segmento de reta com um pequeno círculo em um dos extremos e ligado ao classificador.

Classes clientes são conectadas à interface através de um relacionamento de notação similar à do relacionamento de dependência.

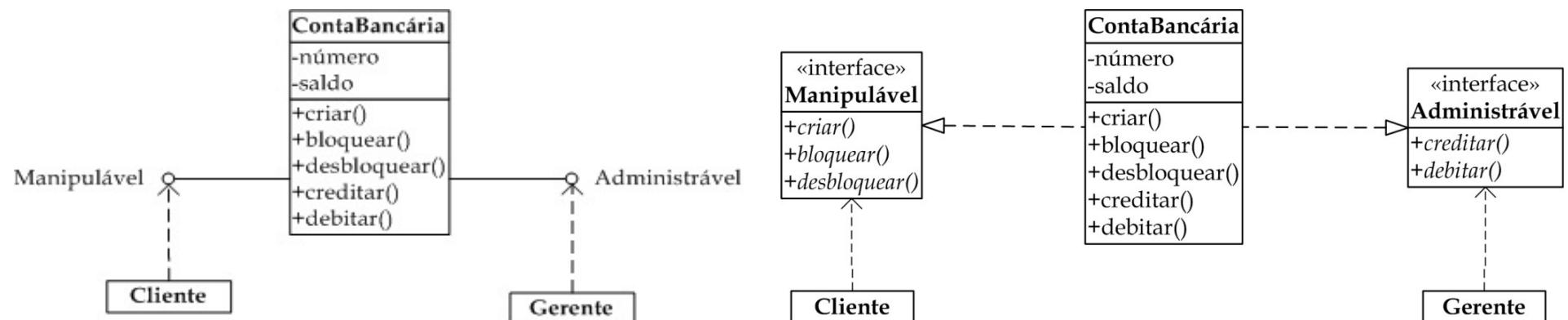
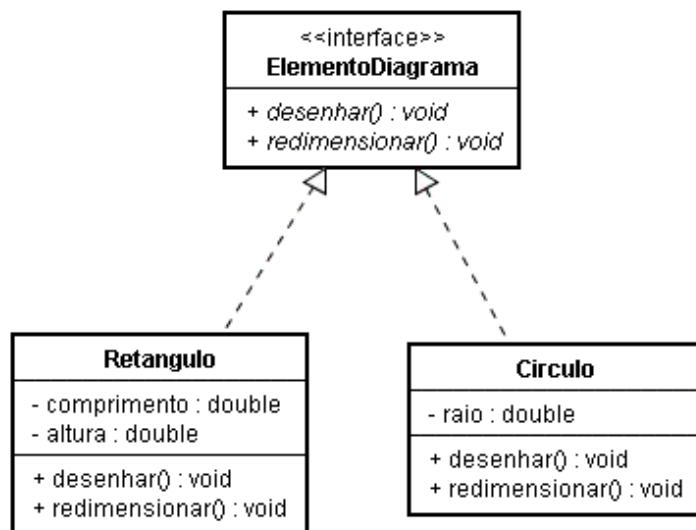


Diagrama de Classes Interfaces



```
public interface ElementoDiagrama {
    double PI = 3.1425926; //static and final
    constant.
    void desenhar();
    void redimensionar();
}

public class Circulo implements ElementoDiagrama {
    ...
    public void desenhar() { /* draw a circle */ }
    public void redimensionar() { /* draw a circle */ }
}

public class Retangulo implements
ElementoDiagrama {
    ...
    public void desenhar() { /* draw a rectangle */ }
    public void redimensionar() { /* draw a
rectangle */ }
}
```

Chegou a vez de vocês?



Representar as Classes do projeto do repositório em um diagrama de classes.

<https://github.com/profandersonbosing/unipar-diagram-class-1-main>



<https://online.visual-paradigm.com/>



Perguntas?



Análise e Projetos de Sistemas

Aula: Diagrama de Classes(Relacionamentos ou Associações).



Prof. Anderson Augusto Bosing

Relacionamentos ou Associações

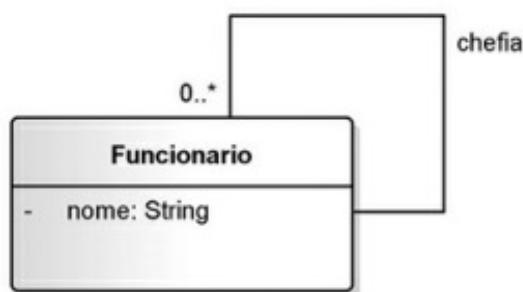
As classes costumam ter relacionamentos entre si, chamados associações, que permitem que elas compartilhem informações entre si e colaborem para a execução dos processos executados pelo sistema. Uma associação descreve um vínculo que ocorre normalmente entre os objetos de uma ou mais classes. As associações são representadas por linhas ligando as classes envolvidas.

Tais linhas podem ter nomes ou títulos para auxiliar a compreensão do tipo de vínculo estabelecido entre os objetos das classes envolvidas nas associações.



Associação Unária ou Reflexiva

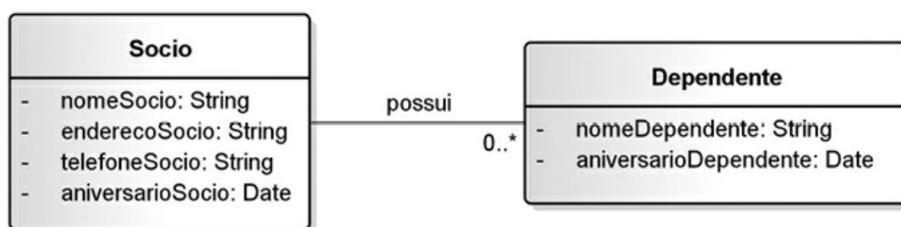
Este tipo de associação ocorre quando existe um relacionamento de um objeto de uma classe com objetos da mesma classe.



O título ou nome da associação não é realmente obrigatório, mas seu uso é recomendado para auxiliar a compreender o que a associação representa.
Normalmente é escolhido um verbo ou uma expressão verbal.

Associação Binária

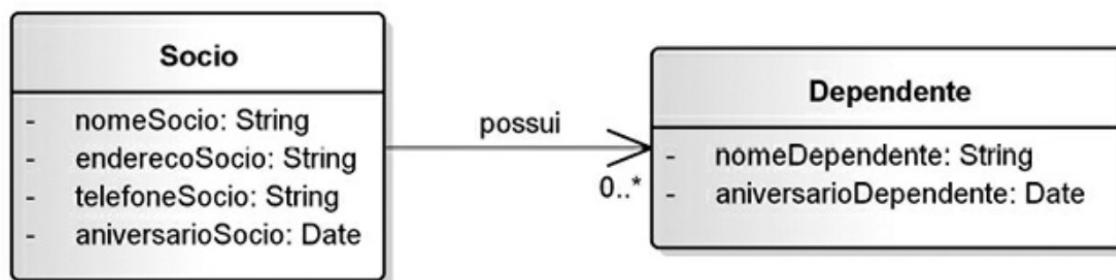
Associações binárias ocorrem quando são identificados relacionamentos entre objetos de duas classes distintas. Em geral, esse tipo de associação é o mais comumente encontrado.



Um objeto da classe **Socio** pode relacionar-se ou não com instâncias da classe **Dependente**

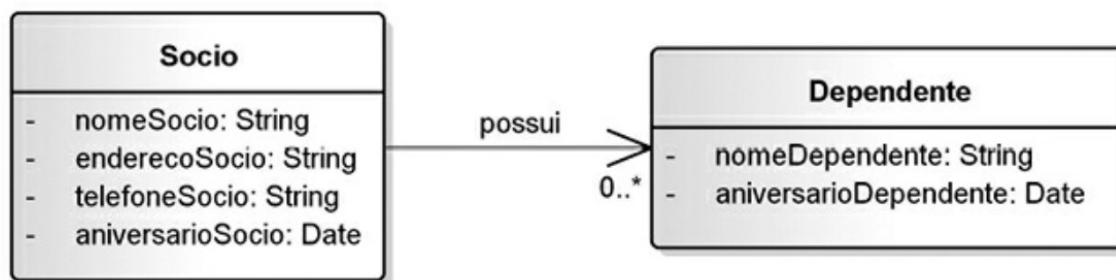
Associação Binária

A navegabilidade é representada mais comumente por uma seta em um dos fins da associação, embora seja possível representá-la nos dois sentidos, se isso for considerado necessário. Quando há navegabilidade unilateral, ela determina que os objetos da classe para onde a seta aponta não têm conhecimento dos objetos aos quais estão associados na outra extremidade da associação.



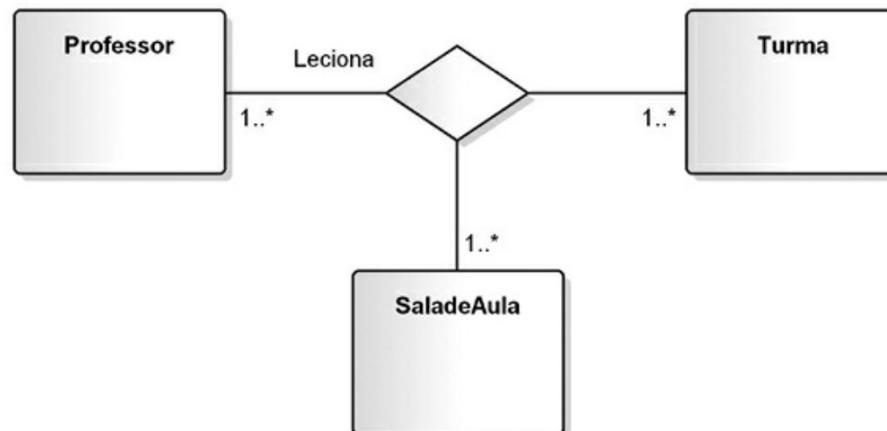
Associação Binária

No exemplo da figura abaixo, o uso da direção de leitura nos transmitiria a informação de que a leitura da associação deveria ser feita da seguinte forma: “Uma instância da classe Socio possui, no mínimo, nenhuma instância e, no máximo, muitas instâncias da classe Dependente e uma instância da classe Dependente é possuída por uma e somente uma instância da classe Socio”.



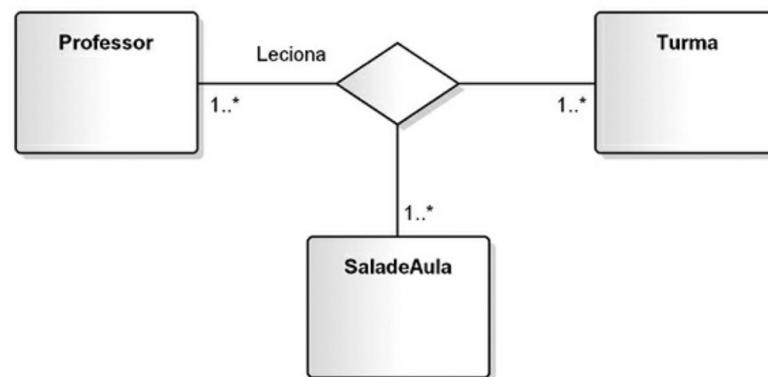
Associação Ternária ou N-ária

Associações ternárias ou n-árias conectam objetos de mais de duas classes. São representadas por um losango para onde convergem todas as ligações da associação. A figura abaixo apresenta um exemplo de associação ternária.



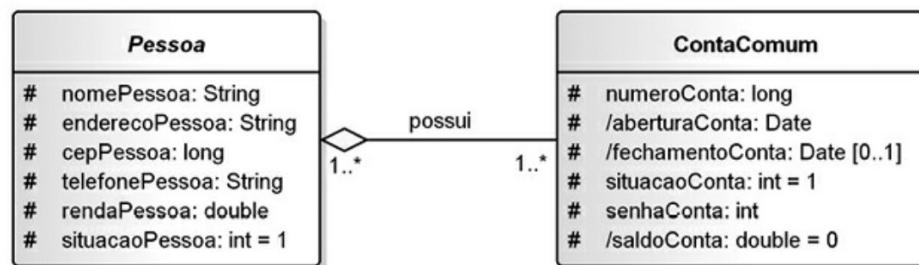
Associação Ternária ou N-ária

Assim, podemos ler a associação apresentada na figura da seguinte forma: “Um professor leciona para, no mínimo, uma turma e, no máximo, para muitas, uma turma tem, no mínimo, um professor e, no máximo, muitos lecionando para ela e um professor, ao lecionar para uma determinada turma, utiliza, no mínimo, uma sala de aula e, no máximo, muitas”. As associações ternárias são úteis para demonstrar associações complexas. No entanto, convém evitar utilizá-las, pois sua leitura é, por vezes, difícil de ser interpretada, todavia seu uso pode ser inevitável em algumas situações.



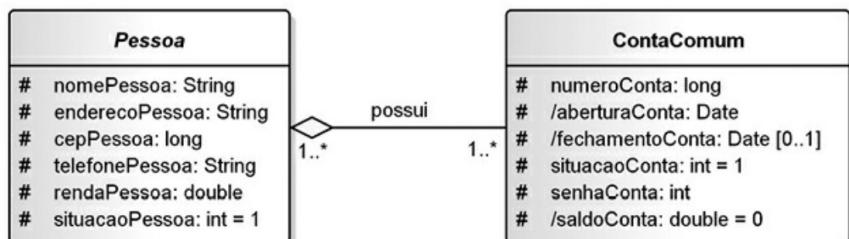
Agregação

Agregação é um tipo especial de associação em que se tenta demonstrar que as informações de um objeto (objeto-todo) são complementadas pelas informações contidas em um ou mais objetos no outro fim da associação (chamados objetos-parte). Esse tipo de associação tenta demonstrar uma relação todo/parte entre os objetos associados. O símbolo de agregação difere do de associação por conter um losango no fim da associação que contém os objetos-todo.



Agregação

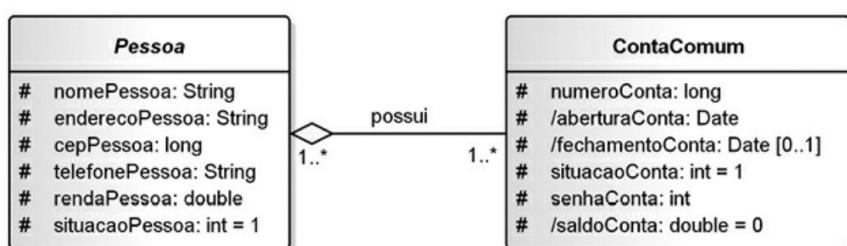
Esse exemplo demonstra uma associação de agregação existente entre uma classe Pessoa e uma classe ContaComum, o que determina que os objetos da classe Pessoa são objetos-todo que precisam ter suas informações complementadas pelos objetos da classe ContaComum, que, nessa associação, são objetos-parte. Dessa maneira, sempre que uma pessoa for consultada, além das informações pessoais, serão apresentadas todas as contas que ela possui.



A decisão de utilizar a AGREGAÇÃO é uma questão de julgamento. Nem sempre é evidente que uma associação deve ser modelada como uma AGREGAÇÃO.

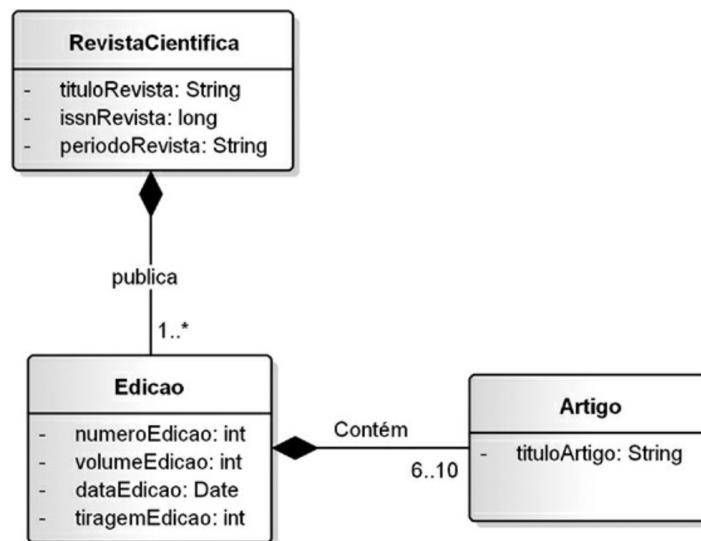
Agregação

A associação de agregação pode, em alguns casos, ser substituída por uma associação binária simples, dependendo da visão de quem faz a modelagem. A função principal de uma associação do tipo agregação é identificar a obrigatoriedade de uma complementação das informações de um objeto-todo por seus objetos-parte, quando este for consultado. Em uma associação binária, todavia, essa obrigatoriedade não está explícita.



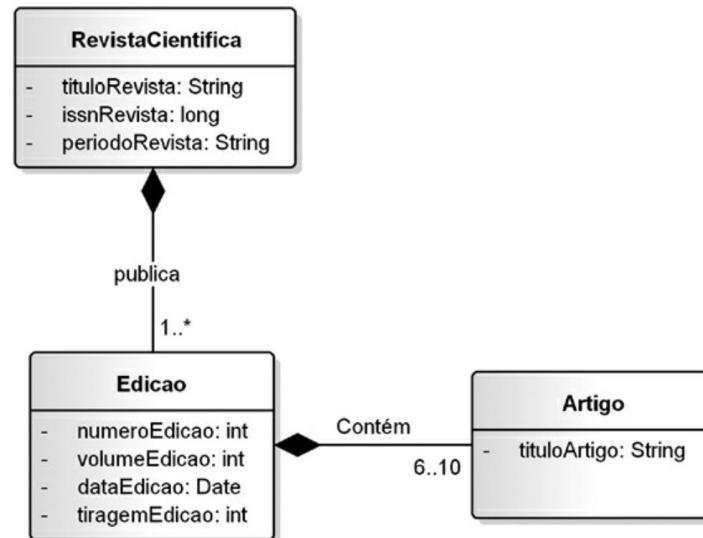
Composição

Uma associação do tipo composição constitui-se em uma variação da agregação, onde é apresentado um vínculo mais forte entre os objetos-todo e os objetos-parte, procurando demonstrar que os objetos-parte têm de estar associados a um único objeto-todo. Em uma composição, os objetos- parte não podem ser destruídos por um objeto diferente do objeto-todo ao qual estão relacionados.



Composição

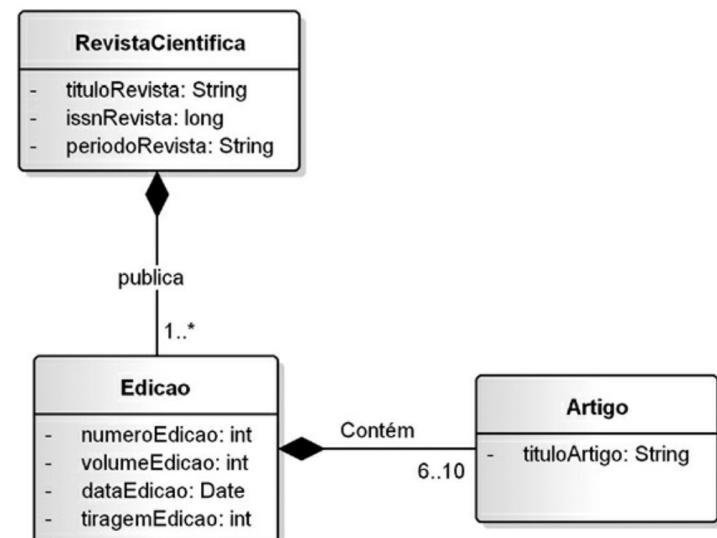
É possível perceber que um objeto da classe **RevistaCientifica** refere-se a, no mínimo, um objeto da classe **Edicao**, podendo se referir a muitos objetos dessa classe, e cada instância da classe **Edicao** relaciona-se única e exclusivamente a uma instância específica da classe **RevistaCientifica**, não podendo relacionar-se a nenhuma outra.



Composição

Verificando a imagem, é possível perceber que um objeto da classe RevistaCientifica refere-se a, no mínimo, um objeto da classe Edicao, podendo se referir a muitos objetos dessa classe, e cada instância da classe Edicao relaciona-se única e exclusivamente a uma instância específica da classe RevistaCientifica, não podendo relacionar-se a nenhuma outra.

Ainda nesse exemplo, percebemos que um objeto da classe Edicao deve se relacionar a, no mínimo, seis objetos da classe Artigo, podendo se relacionar com até 10 objetos da já citada classe. Esse tipo de informação torna-se útil como documentação e serve como forma de validação, que impede que uma revista seja publicada sem ter, no mínimo, seis artigos ou mais de 10. No entanto, um objeto da classe Artigo refere-se unicamente a um objeto da classe Edicao. Isso é também uma forma de documentação, pois uma edição de uma revista científica só deve publicar trabalhos inéditos. Assim, é lógico que não é possível a um mesmo objeto da classe Artigo relacionar-se a mais de um objeto da classe Edicao.



Generalização/Especialização

Este é um tipo especial de relacionamento, similar à associação de mesmo nome utilizada no diagrama de casos de uso. O objetivo dessa associação é representar a ocorrência de herança entre as classes, identificando as classes-mãe (ou superclasses), chamadas gerais, e classes-filhas (ou subclasses), chamadas especializadas, demonstrando a hierarquia entre as classes e, possivelmente, métodos polimórficos nas classes especializadas.

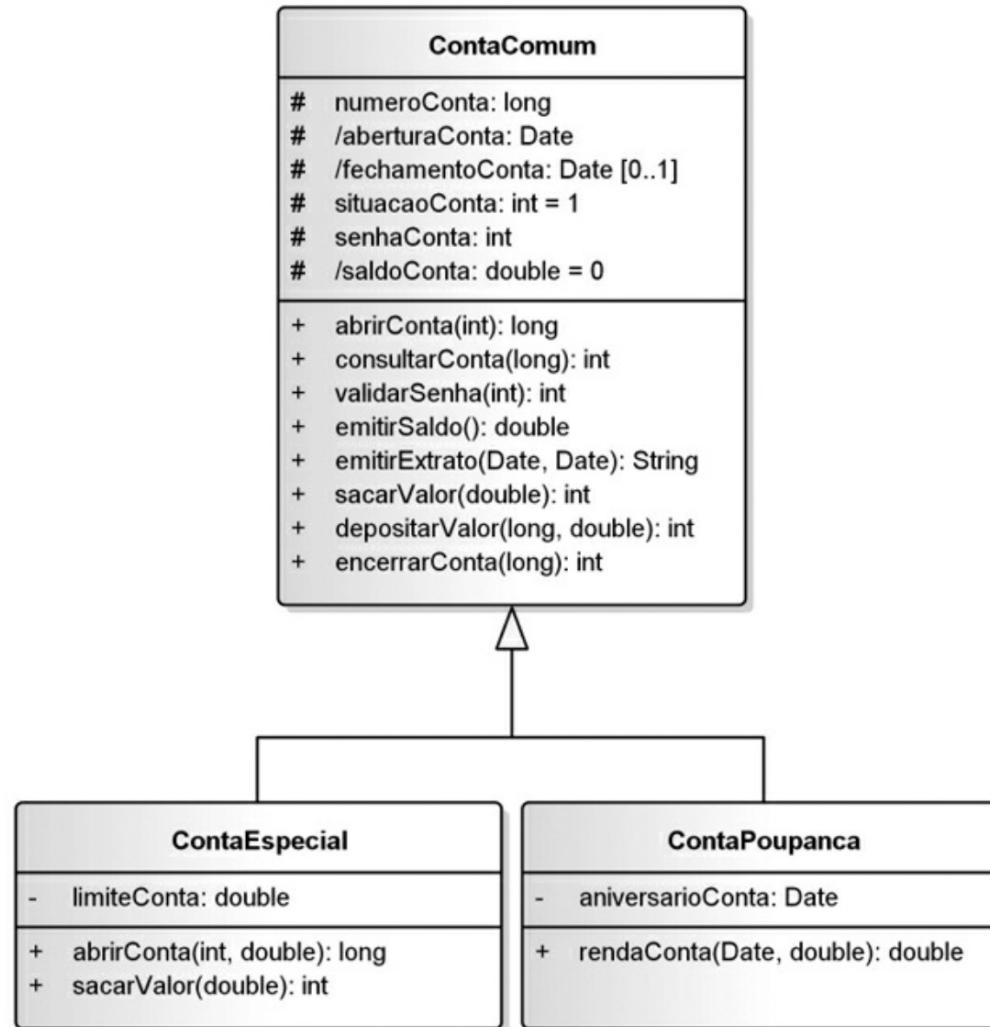


Generalização/Especialização

Esse tipo de relacionamento permite representar classes derivadas a partir de classes mais antigas e, ao mesmo tempo que as novas classes herdam todos os atributos e métodos das classes das quais foram derivadas, é possível adicionar novos atributos e/ou métodos a essas classes, dessa forma especializando-as. Isso permite maior rapidez no desenvolvimento, uma vez que não é necessário adicionar os atributos e métodos já existentes às classes anteriores, apenas os novos atributos ou métodos, o que também impede que erros de codificação sejam cometidos desnecessariamente. Além disso, métodos podem ser redeclarados em uma classe especializada, com o mesmo nome, mas comportando-se de forma diferente, não sendo, portanto, necessário modificar o código-fonte do sistema em relação às chamadas de métodos das classes especializadas, pois o nome do método não mudou, somente foi redeclarado em uma classe especializada e só se comportará de maneira diferente quando for chamado por objetos dessa classe.

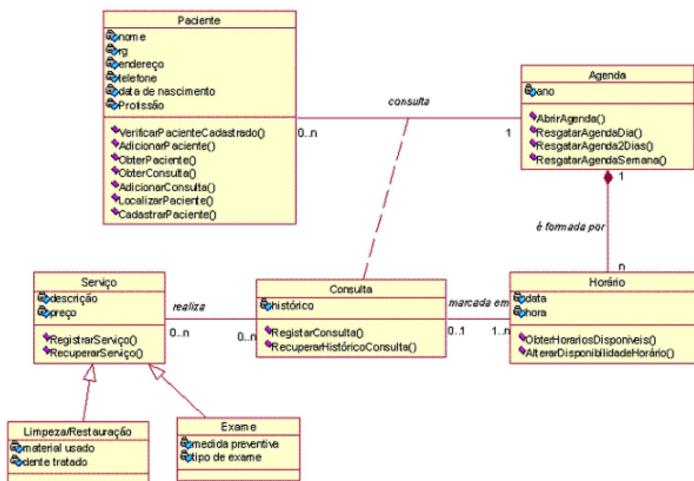


Generalização/Especialização



Chegou a vez do vocês!

https://docs.google.com/document/dz1dIF__XFcNEehfwREJnX1BWgP-4VDgBuhstQyUbiGBdw/edit?usp=sharing



Técnicas para identificação de classes

Várias técnicas (de uso não exclusivo) são usadas para identificar classes:

Categorias de Conceitos

Análise Textual de Abbott (Abbot Textual Analysis)

Categorização BCE

Padrões de Análise (Analisis Patterns)

Identificação Dirigida a Responsabilidades



Categorias de Conceitos

Estratégia: usar uma lista de conceitos comuns.

Conceitos concretos. Por exemplo, edifícios, carros, salas de aula, etc.

Papéis desempenhados por seres humanos. Por exemplo, professores, alunos, empregados, clientes, etc.

Eventos, ou seja, ocorrências em uma data e em uma hora particulares. Por exemplo, reuniões, pedidos, aterrissagens, aulas, etc.

Lugares: áreas reservadas para pessoas ou coisas. Por exemplo: escritórios, filiais, locais de pouso, salas de aula, etc.

Organizações: coleções de pessoas ou de recursos. Por exemplo: departamentos, projetos, campanhas, turmas, etc.

Conceitos abstratos: princípios ou idéias não tangíveis. Por exemplo: reservas, vendas, inscrições, etc.



Análise Textual de Abbott

Estratégia: identificar termos da narrativa de casos de uso e documento de requisitos que podem sugerir classes, atributos, operações.

Neste técnica, são utilizadas diversas fontes de informação sobre o sistema: documento e requisitos, modelos do negócio, glossários, conhecimento sobre o domínio, etc.

Para cada um desses documentos, os nomes (substantivos e adjetivos) que aparecem no mesmo são destacados. (São também consideradas locuções equivalentes a substantivos.)

Após isso, os sinônimos são removidos (permanecem os nomes mais significativos para o domínio do negócio em questão).



Análise Textual de Abbott

Cada termo remanescente se encaixa em uma das situações a seguir:

O termo se torna uma classe (ou seja, são classes candidatas);

O termo se torna um atributo;

O termo não tem relevância alguma com ao SSOO.

Abbott também preconiza o uso de sua técnica na identificação de operações e de associações.

Para isso, ele sugere que destaquemos os verbos no texto.

Verbos de ação (e.g., calcular, confirmar, cancelar, comprar, fechar, estimar, depositar, sacar, etc.) são operações em potencial.

Verbos com sentido de “ter” são potenciais agregações ou composições.

Verbos com sentido de “ser” são generalizações em potencial.

Demais verbos são associações em potencial.

Análise Textual de Abbott

Cada termo remanescente se encaixa em uma das situações a seguir:

O termo se torna uma classe (ou seja, são classes candidatas);

O termo se torna um atributo;

O termo não tem relevância alguma com ao SSOO.

Abbott também preconiza o uso de sua técnica na identificação de operações e de associações.

Para isso, ele sugere que destaquemos os verbos no texto.

Verbos de ação (e.g., calcular, confirmar, cancelar, comprar, fechar, estimar, depositar, sacar, etc.) são operações em potencial.

Verbos com sentido de “ter” são potenciais agregações ou composições.

Verbos com sentido de “ser” são generalizações em potencial.

Demais verbos são associações em potencial.

Análise Textual de Abbott

Parte do Texto	Componente	Exemplo
Nome próprio	Objeto	Eduardo Bezerra
Nome simples	Classe	aluno
Verbos de Ação	Operação	registrar
Verbo Ser	Herança	é um
Verbo Ter	Todo-partes	tem um



Análise Textual de Abbott

A ATA é de aplicação bastante simples.

No entanto, uma desvantagem é que seu resultado (as classes candidatas identificadas) depende de os documentos utilizados como fonte serem completos.

Dependendo do estilo que foi utilizado para escrever esse documento, essa técnica pode levar à identificação de diversas classes candidatas que não gerarão classes.

A análise do texto de um documento pode não deixar explícita uma classe importante para o sistema.

Em linguagem natural, as variações lingüísticas e as formas de expressar uma mesma idéia são bastante numerosas



Análise de Casos de Uso

Essa técnica é também chamada de identificação dirigida por casos de uso, e é um caso particular da ATA.

Técnica preconizada pelo Processo Unificado.

Nesta técnica, o MCU é utilizado como ponto de partida.

Premissa: um caso de uso corresponde a um comportamento específico do SSOO. Esse comportamento somente pode ser produzido por objetos que compõem o sistema.

Em outras palavras, a realização de um caso de uso é responsabilidade de um conjunto de objetos que devem colaborar para produzir o resultado daquele caso de uso.

Com base nisso, o modelador aplica a técnica de análise dos casos de uso para identificar as classes necessárias à produção do comportamento que está documentado na descrição do caso de uso.



Análise de Casos de Uso

Procedimento de aplicação:

O modelador estuda a descrição textual de cada caso de uso para identificar classes candidatas.

Para cada caso de uso, se texto (fluxos principal, alternativos e de exceção, pós-condições e pré-condições, etc.) é analisado.

Na análise de certo caso de uso, o modelador tenta identificar classes que possam fornecer o comportamento do mesmo.

Na medida em que os casos de uso são analisados um a um, as classes do SSOO são identificadas.

Quando todos os casos de uso tiverem sido analisados, todas as classes (ou pelo menos a grande maioria delas) terão sido identificadas.

Na aplicação deste procedimento, podemos utilizar as categorização BCE.



Categorização BCE

Na categorização BCE, os objetos de um SSOO são agrupados de acordo com o tipo de responsabilidade a eles atribuída.

- **objetos de entidade:** usualmente objetos do domínio do problema
- **objetos de fronteira:** atores interagem com esses objetos
- **objetos de controle:** servem como intermediários entre objetos de fronteira e de entidade, definindo o comportamento de um caso de uso específico.

Categorização proposta por Ivar Jacobson em 1992.

Possui correspondência (mas não equivalência!) com o framework model-view-controller (MVC).

Ligaçāo entre análise (o que; problema) e projeto (como; solução)

Estereótipos na UML: «boundary», «entity», «control»



Objetos de Entidade

Repositório para informações e as regras de negócio manipuladas pelo sistema.

Representam conceitos do domínio do negócio.

Características

Normalmente armazenam informações persistentes.

Várias instâncias da mesma entidade existindo no sistema.

Participam de vários casos de uso e têm ciclo de vida longo.

Exemplo:

Um objeto Pedido participa dos casos de uso Realizar Pedido e Atualizar Estoque. Este objeto pode existir por diversos anos ou mesmo tanto quanto o próprio sistema.



Objetos de Fronteira

Realizam a comunicação do sistema com os atores.
traduzem os eventos gerados por um ator em eventos relevantes ao sistema **eventos de sistema.**
também são responsáveis por apresentar os resultados de uma interação dos objetos em algo inteligível pelo ator.
Existem para que o sistema se comunique com o mundo exterior.
Por consequência, são altamente dependentes do ambiente.
Há dois tipos principais de objetos de fronteira:
Os que se comunicam com o usuário (atores humanos): relatórios, páginas HTML, interfaces gráfica desktop, etc.
Os que se comunicam com atores não-humanos (outros sistemas ou dispositivos): protocolos de comunicação.



Objetos de Controle

São a “ponte de comunicação” entre objetos de fronteira e objetos de entidade.

Responsáveis por controlar a lógica de execução correspondente a um caso de uso.

Decidem o que o sistema deve fazer quando um evento de sistema ocorre.

Eles realizam o controle do processamento

Agem como gerentes (coordenadores, controladores) dos outros objetos para a realização de um caso de uso.

Traduzem eventos de sistema em operações que devem ser realizadas pelos demais objetos.



Importância da Categorização BCE

A categorização BCE parte do princípio de que cada objeto em um SSOO é especialista em realizar um de três tipos de tarefa, a saber:
se comunicar com atores (fronteira),
manter as informações (entidade) ou
coordenar a realização de um caso de uso (controle).

A categorização BCE é uma “receita de bolo” para identificar objetos participantes da realização de um caso de uso.

A importância dessa categorização está relacionada à capacidade de adaptação a eventuais mudanças.

Se cada objeto tem atribuições específicas dentro do sistema, mudanças podem ser menos complexas e mais localizadas.

Uma modificação em uma parte do sistema tem menos possibilidades de resultar em mudanças em outras partes.



Notação da UML para objetos, segunda a categorização BCE

