Resumo: Unified Modeling Language (UML)

Síntese

Este documento sintetiza os conceitos, a estrutura e a aplicação da Unified Modeling Language (UML), uma linguagem gráfica padrão para a modelagem de sistemas. Nascida da união de metodologias proeminentes na década de 1990 por James Rumbaugh, Grady Booch e Ivar Jacobson ("The Three Amigos") na Rational Software, a UML foi padronizada pela Object Management Group (OMG) para unificar a modelagem orientada a objetos.

A UML é uma ferramenta versátil, independente de processo e de linguagem de programação, utilizada para especificar, visualizar, construir e documentar artefatos de sistemas. Embora primariamente associada ao software, sua expressividade permite a modelagem de processos de negócio e sistemas em diversas áreas, como finanças e aeroespacial. Sua principal força, segundo Martin Fowler, reside em facilitar a comunicação e o entendimento entre as equipes, oferecendo um meio-termo entre a ambiguidade da linguagem natural e o detalhamento excessivo do código.

A linguagem é organizada em 14 tipos de diagramas, divididos em duas categorias principais:

1. **Diagramas Estruturais:** Representam os aspectos estáticos de um sistema, como classes, componentes e a distribuição física do hardware. Os diagramas mais importantes desta categoria são o **Diagrama de Classes**, que serve como espinha dorsal da modelagem orientada a objetos, e o **Diagrama de Componentes**.
2. **Diagramas Comportamentais:** Descrevem os aspectos dinâmicos, mostrando como o sistema se comporta e interage ao longo do tempo. Dentro desta categoria, destacam-se:
    - **Diagrama de Casos de Uso:** Essencial para o levantamento de requisitos funcionais a partir da perspectiva do usuário.
    - **Diagrama de Atividades:** Modela fluxos de trabalho e processos de negócio, suportando comportamento paralelo.
    - **Diagrama de Máquina de Estados:** Descreve o ciclo de vida de um objeto através de suas mudanças de estado.
    - **Diagrama de Sequência e de Comunicação:** Detalham a interação entre objetos, com foco na ordem temporal (Sequência) ou na organização estrutural (Comunicação).

Mecanismos como Estereótipos permitem a extensão da semântica da UML, adaptando-a a domínios ou plataformas específicas, o que confere grande flexibilidade à linguagem.

1. Fundamentos e Origem da UML

1.1. Histórico e Criação

A Unified Modeling Language (UML) surgiu em meados da década de 1990 como uma solução para a proliferação de linguagens de modelagem que dificultava a adoção da tecnologia de Orientação a Objetos. A empresa Rational Software Corporation reuniu três dos mais proeminentes pesquisadores da área, conhecidos como "The Three Amigos":

- **James Rumbaugh:** Criador da Técnica de Modelagem de Objetos (TMO).
- **Grady Booch:** Criador do método de Projeto Orientado a Objetos (POO).

- **Ivar Jacobson:** Criador do método de Engenharia de Software Orientada a Objetos (ESOO).

O objetivo era consolidar seus respectivos métodos em uma linguagem de modelagem unificada e não proprietária.

1.2. Padronização e Evolução

Para evitar que um padrão controlado pela Rational gerasse vantagens competitivas desleais, outros fornecedores (IBM, HP, Oracle, etc.) incitaram a **Object Management Group (OMG)** — um consórcio industrial responsável por padrões de interoperabilidade — a assumir o processo de padronização.

- **UML 1.1:** Publicada em agosto de 1997 e adotada como padrão oficial pela OMG. Posteriormente, tornou-se o padrão internacional ISO/IEC 19501:2005.
- **UML 2.0:** Lançada em 2005, representou uma grande maturação da linguagem, corrigindo inconsistências e integrando novos conceitos.
- **UML 2.5:** É a versão atual, resultado de diversas pequenas atualizações desde a versão 2.0.

1.3. Definição e Propósito

A UML é definida como uma **linguagem gráfica** para especificar, visualizar e documentar artefatos de um sistema, primariamente de software. No entanto, sua aplicabilidade transcende essa área, sendo utilizada em telecomunicações, defesa, setor bancário e até para modelar sistemas não computacionais, como fluxogramas do sistema judiciário.

Características Centrais:

- **Independente de Processo:** Pode ser aplicada em diferentes contextos de desenvolvimento, como o RUP (Rational Unified Process).
- **Independente de Tecnologia:** Não está atrelada a uma linguagem de programação específica, podendo ser usada com linguagens estruturadas (C, Cobol) ou orientadas a objetos.
- **Foco em Comunicação:** Conforme destacado por Martin Fowler, seu principal valor é melhorar a comunicação e o entendimento dentro de uma equipe. A notação gráfica atua como um intermediário eficaz entre a imprecisão da linguagem natural e a complexidade do código-fonte.

2. Arquitetura e Mecanismos da UML

A UML é definida por quatro especificações inter-relacionadas e possui mecanismos gerais que permitem sua extensão e clareza.

2.1. Especificações Fundamentais

Especificação	Descrição
Infraestrutura	Contém o núcleo da arquitetura, perfis e estereótipos.
Superestrutura	Contém os elementos de modelagem estáticos e dinâmicos (os diagramas).

Object Constraint Language (OCL)	Linguagem formal e declarativa para descrever regras e restrições nos modelos UML.
Intercâmbio de Diagramas	Define os formatos para o intercâmbio de diagramas entre ferramentas.

2.2. Mecanismos de Uso Geral

Mecanismo	Descrição
Estereótipo	Estende o significado de um elemento do modelo, permitindo a adaptação da UML a domínios específicos. Pode ser predefinido (<<interface>>) ou definido pelo usuário (<<roteador>>), e representado textualmente (<<nome>>) ou graficamente (ícone).
Notas Explicativas	Comenta ou esclarece uma parte de um diagrama sem alterar a semântica do modelo. Graficamente, é um retângulo com uma "orelha", ligado ao elemento por uma linha tracejada.
Tagged Values	Define propriedades adicionais (metadados) para elementos de um modelo. A partir da UML 2.0, seu uso está associado a estereótipos.
Restrições	Especifica condições ou regras que um ou mais elementos do modelo devem satisfazer. São delimitadas por chaves ({restrição}) e podem ser escritas em OCL (formal) ou texto livre (informal).
Pacotes	Agrupa elementos semanticamente relacionados (classes, casos de uso, etc.) em unidades de mais alto nível, ajudando a organizar modelos complexos.

2.3. Visões Arquiteturais (Modelo 4+1)

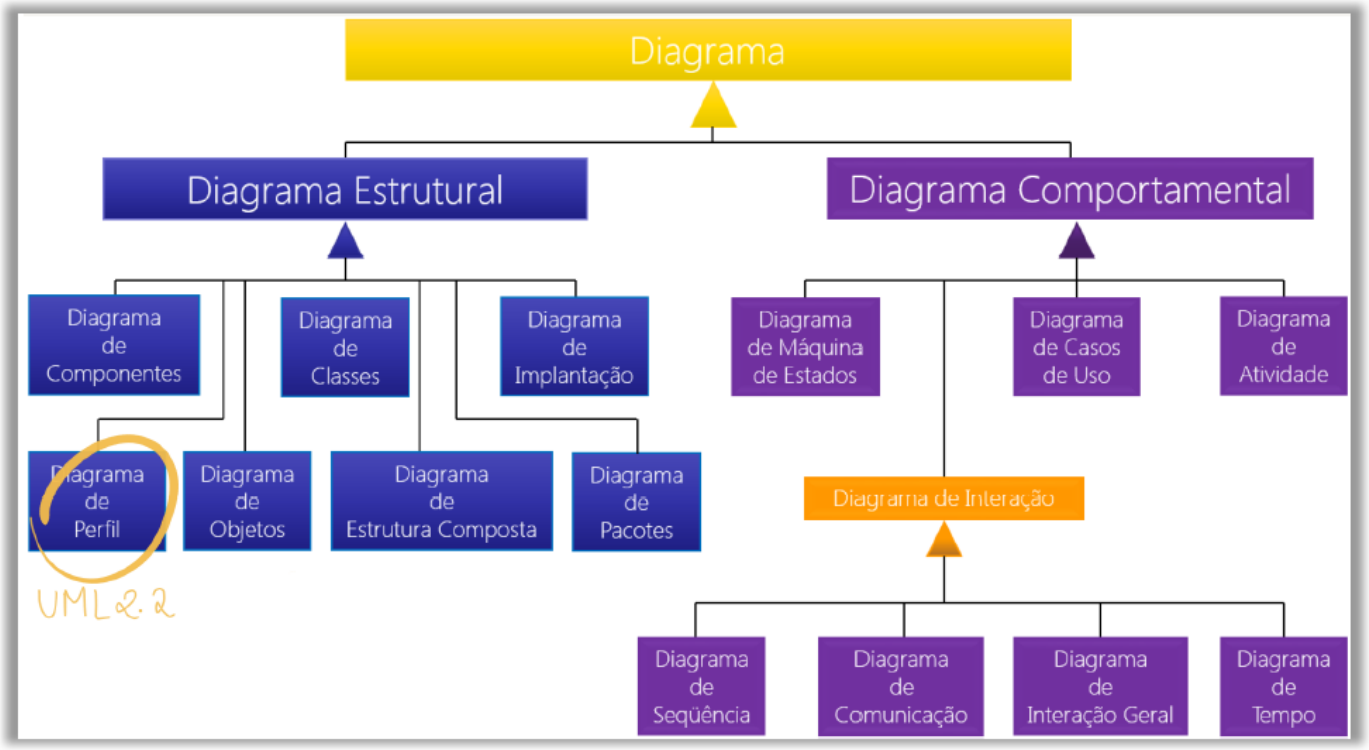
A arquitetura de um software pode ser descrita a partir de cinco visões concorrentes, cada uma focada em um conjunto específico de interesses.

Visão	Perspectiva	Foco	Diagramas Principais
Lógica (ou de Projeto)	Usuário Final	Requisitos funcionais e estrutura do sistema (classes, objetos).	Classe, Objetos, Pacotes
Desenvolvimento (ou de Implementação)	Programador	Organização estática dos módulos de software.	Componentes
Processo	Integrador	Requisitos não-funcionais (desempenho, escalabilidade) e concorrência.	Sequência, Estrutura Composta, Máquina de Estados, Atividade
Física (ou de Implantação)	Engenheiro de Sistemas	Topologia física do sistema (hardware e distribuição de software).	Implantação, Componentes

Casos de Uso (ou de Cenários)	Todos os Interessados	Consistência e validação do sistema através de cenários de uso.	Casos de Uso

3. Classificação dos Diagramas UML

A UML 2.x define 14 tipos de diagramas, agrupados em três categorias (sendo a terceira um subconjunto da segunda).



Mapa-Minemônico para memorização dos diagramas da UML:

3.1. Diagramas Estruturais

Representam os aspectos **estáticos** do sistema, ou seja, sua estrutura em um determinado momento, sem considerar a passagem do tempo.

- Diagrama de Classes

Tipos de relacionamentos no Diagrama de Classes:

- Diagrama de Objetos
- Diagrama de Componentes
- Diagrama de Implantação
- Diagrama de Pacotes
- Diagrama de Estrutura Composta
- Diagrama de Perfil

3.2. Diagramas Comportamentais

Representam os aspectos **dinâmicos** do sistema, descrevendo como os processos e funcionalidades se relacionam e evoluem ao longo do tempo.

- Diagrama de Casos de Uso
- Diagrama de Atividades
- Diagrama de Máquina de Estados
- Diagrama de Sequência
- Diagrama de Comunicação

- Diagrama de Tempo
- Diagrama de Interação Geral

3.3. Diagramas de Interação

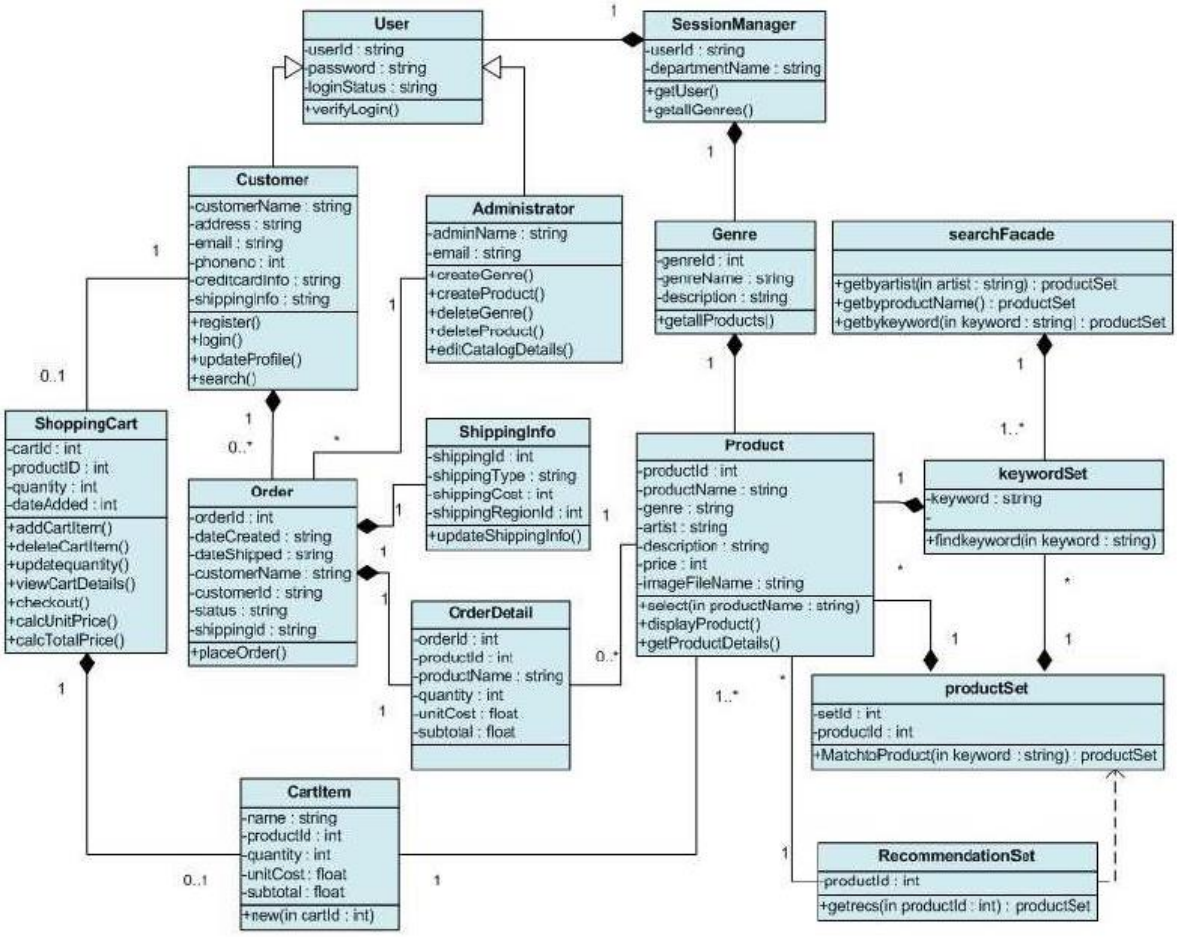
São um subconjunto dos diagramas comportamentais que focam na **troca de informações e no fluxo de controle** entre os objetos do sistema.

- Diagrama de Sequência
- Diagrama de Comunicação
- Diagrama de Tempo
- Diagrama de Interação Geral

4. Análise Detalhada dos Diagramas Estruturais

4.1. Diagrama de Classes

É o diagrama mais utilizado e cobrado. Descreve a estrutura estática do sistema em termos de classes, interfaces, seus atributos, operações e os relacionamentos entre elas.



- **Classe:** Abstração de um conjunto de objetos com características similares. Pode ser representada com diferentes níveis de detalhe (apenas nome; nome e atributos; nome, atributos e operações). Uma **Classe Ativa** (com borda dupla) representa objetos que possuem seus próprios fluxos de controle (threads).

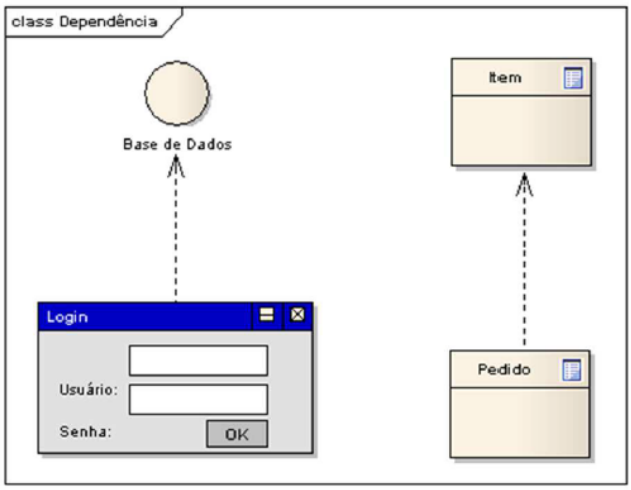
- **Atributos e Operações:**
    - Um atributo/operação **estático** é sublinhado.
    - Uma operação **abstrata** é escrita em itálico.

Professor, só existe uma única forma de representar classes? Não! **Pode-se representar de diversas formas, dependendo do nível de abstração.** A imagem seguinte apresenta maneiras distintas de se representar uma classe: primeiro, apenas com nome da classe (mais abstrata); segundo, com nome da classe e suas propriedades; e terceiro, com nome da classe, suas propriedades e suas operações (mais concreta).

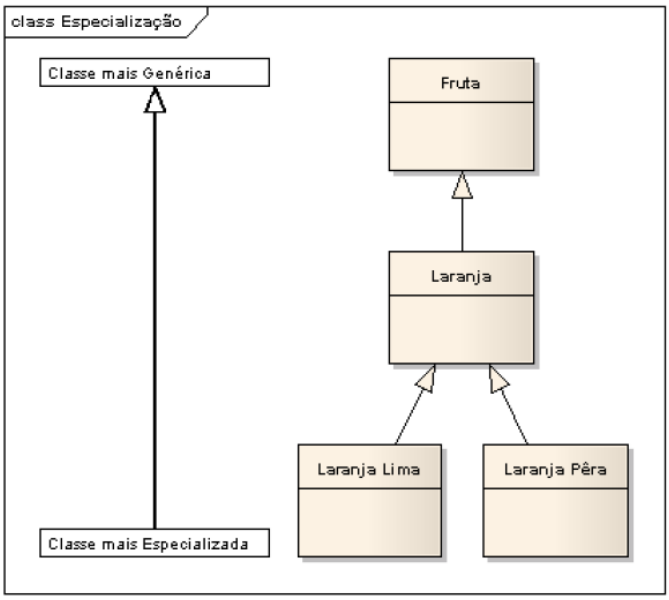
- **Visibilidade:** Define o nível de acesso a atributos e operações.

MODIFICADOR/ESPECIFICADOR			CLASSE	SUBCLASSE	PACOTE	TODOS
UML	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X	X		
	PACOTE	~	X		X	
	PRIVADO	–	X			
JAVA	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X	X	X	
	DEFAULT	~	X		X	
	PRIVADO	-	X			

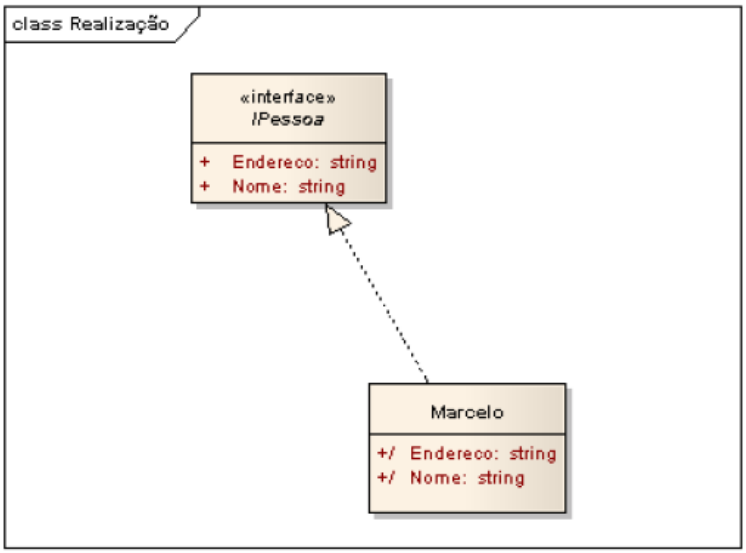
- **Relacionamentos:**
    - **Dependência:** Um relacionamento "usa-um", onde uma mudança em um elemento (independente) pode afetar outro (dependente). Representado por uma **seta tracejada apontando para o elemento independente**.



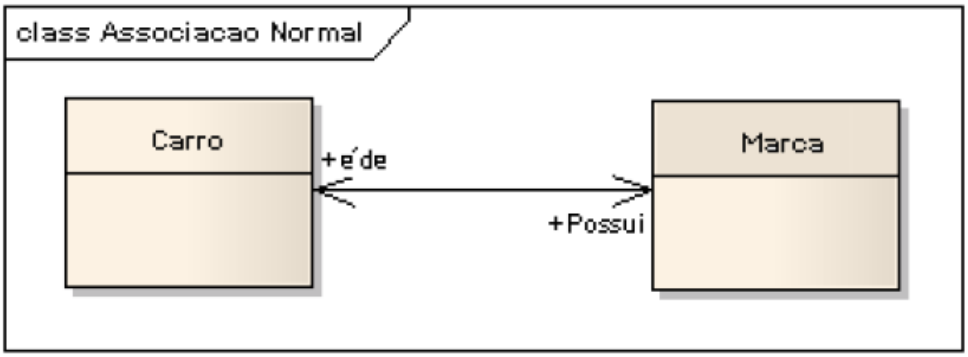
- **Generalização (Herança):** Um relacionamento "é-um", onde uma subclasse herda características de uma superclasse. Representado por **uma linha sólida com uma seta triangular vazia** apontando para a superclasse.



- **Realização:** Um relacionamento onde um elemento (ex: uma classe) implementa o comportamento especificado por outro (ex: uma interface). Representado por uma linha tracejada com uma seta triangular vazia apontando para o elemento que especifica o contrato.

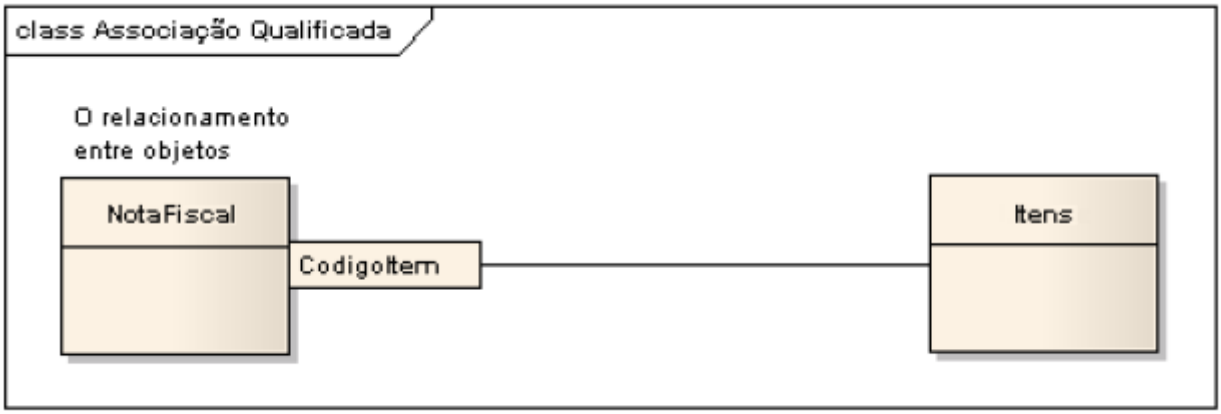


- **Associação:** Relacionamento estrutural que descreve conexões entre instâncias de classes.
    - **Simples: Simples:** é um tipo de relacionamento mais forte que o relacionamento de dependência e **indica que uma instância de um elemento está ligada à instância de outro elemento**. São representados por **uma linha sólida com ou sem setas de navegabilidade**. Ademais, pode haver nomes para a associação e indicação de multiplicidade



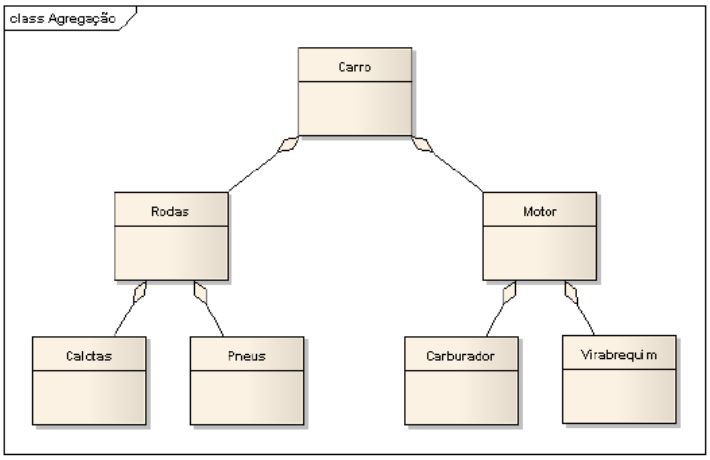
Na imagem acima, podemos observar uma associação simples entre Carro e Marca. **Trocando em miúdos, significa que o carro é de uma marca e a marca possui diversos carros**. Observem que, se se excluir um carro, a marca continua a existir. Assim como se se excluir uma marca, o carro continua a existir. Em outras palavras, esse é um exemplo clássico relacionamento de associação simples. *Certinho?*

- **Qualificada:** é um tipo de relacionamento similar à associação simples, contudo possui um **qualificador, que é um atributo do elemento-alvo capaz de identificar uma instância dentre as demais**. Ela ocorre em associações um-para-muitos ou muitos-para-muitos em que se deseja encontrar um elemento específico dada uma chave.



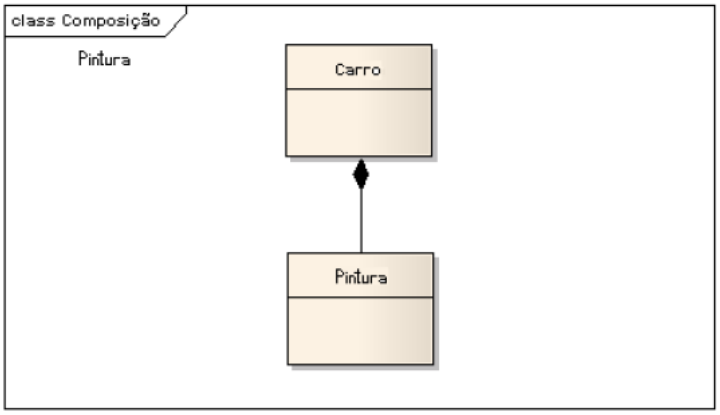
Na imagem acima, podemos observar uma associação qualificada entre NotaFiscal e Itens. Pensem comigo: *uma nota fiscal pode ter milhares de itens, mas e se – por acaso – eu quiser encontrar um item específico?* Ora, eu posso utilizar um identificador de itens como qualificador de NotaFiscal. Dito de outra forma, cada item está associado a um código de item e a uma nota fiscal.

- **Agregação:** Associação "tem-um" onde as partes podem existir independentemente do todo (ex: um carro e suas rodas). Representada por **uma linha sólida com um losango vazio** no lado do todo.

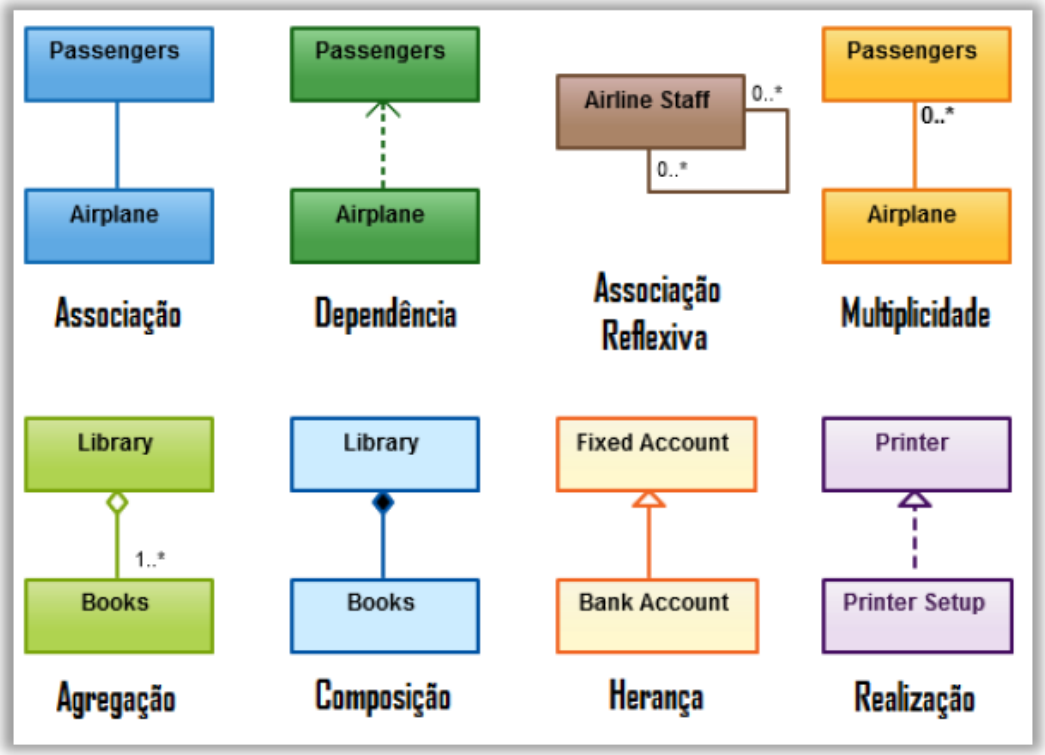


Na imagem acima, podemos observar uma agregação entre Carro e Rodas, Carro e Motor, Rodas e Calotas, Rodas e Pneus, Motor e Carburador, e Motor e Virabrequim. Vamos interpretar apenas a agregação entre Carro (Todo) e Rodas (Partes). ***Respondam-me: a roda pode existir sem um carro?*** É claro que pode! Inclusive, eu posso retirar a roda de um carro e colocá-la em outro carro. Logo, esse é um relacionamento de agregação.

- **Composição:** Uma agregação forte onde as partes **não** podem existir sem o todo (ex: um carro e sua pintura). A existência da parte é dependente do todo. Representada por uma linha sólida com um losango **preenchido** no lado do todo.



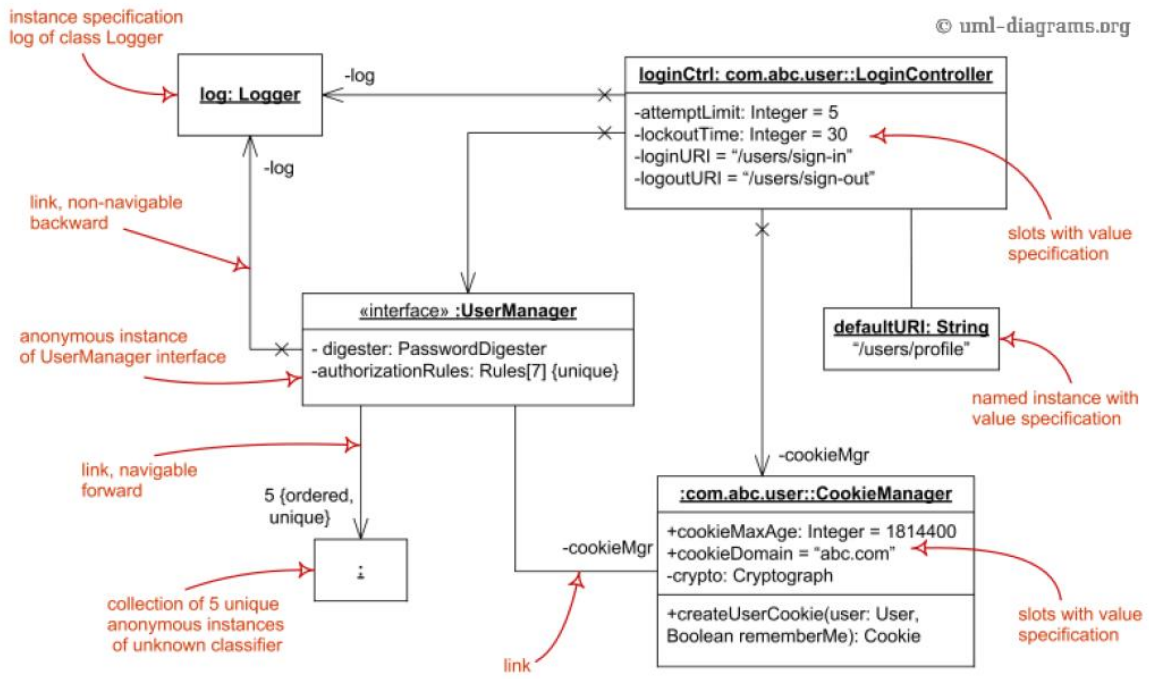
Na imagem acima, podemos observar uma composição entre Carro e Pintura. *Respondam-me: a pintura pode existir sem o carro?* Não, isso não faz o menor sentido! A existência do carro é requisito fundamental, essencial, obrigatório para a existência da pintura. Além disso, essa pintura é exclusiva daquele carro e não pode ser compartilhada. Logo, na composição, **o todo tem sempre cardinalidade 1**!



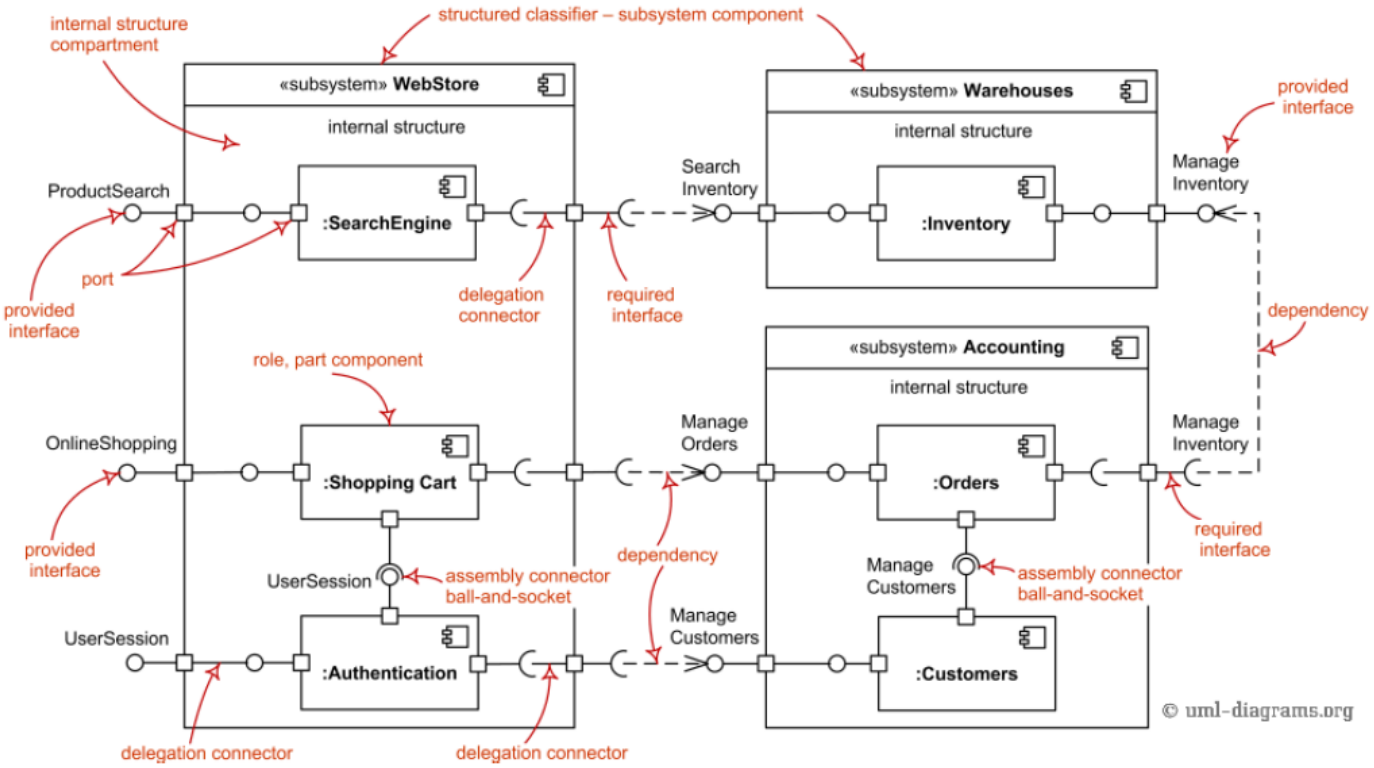
- **Associação:** mostra que os passageiros e o avião possuem alguma ligação;
- **Dependência:** mostra que o avião depende de passageiros;
- **Associação Reflexiva:** mostra que a equipe do avião se relaciona entre si;
- **Multiplicidade:** mostra que um avião possui zero ou mais passageiros;
- **Agregação:** mostra que uma biblioteca tem um ou mais livros (independentes);
- **Composição:** mostra que uma biblioteca tem livros (necessariamente)
- **Herança:** mostra que uma conta bancária é filha de conta fixa;
- **Realização:** *setup* da impressora implementa funcionalidades da impressora.

4.2. Outros Diagramas Estruturais

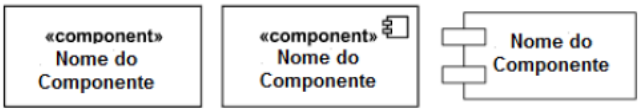
- **Diagrama de Objetos:** Mostra uma "fotografia" do sistema em um momento específico, representando instâncias concretas de classes (joao:Pessoa) e seus relacionamentos. Útil para exemplificar estruturas complexas do diagrama de classes.



- **Diagrama de Componentes:** Apresenta a organização e as dependências entre os componentes de software (módulos, bibliotecas, arquivos executáveis). Foca na visão de implementação e utiliza interfaces **fornecidas** (o que o componente oferece) e **requeridas** (o que ele precisa).

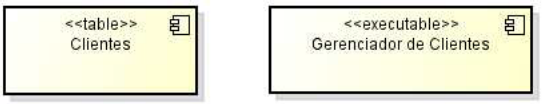


Frequentemente, eles vêm com estereótipos para definir seu tipo e seus relacionamentos – e sua grande vantagem é a modularidade! Galera, é possível representá-los das seguintes formas:

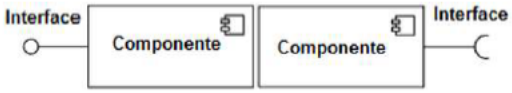


Um componente pode apresentar um estereótipo, isto é, uma definição do que é este componente. Os principais estereótipos são:

- **Arquivo:** determina que o componente é um arquivo de dados do sistema;
- **Biblioteca:** determina que o componente é uma biblioteca de código;
- **Documento:** determina que o componente é um documento de sistema;
- **Executável:** determina que o componente é um arquivo executável;
- **Tabela:** determina que o componente é uma tabela de um banco de dados.

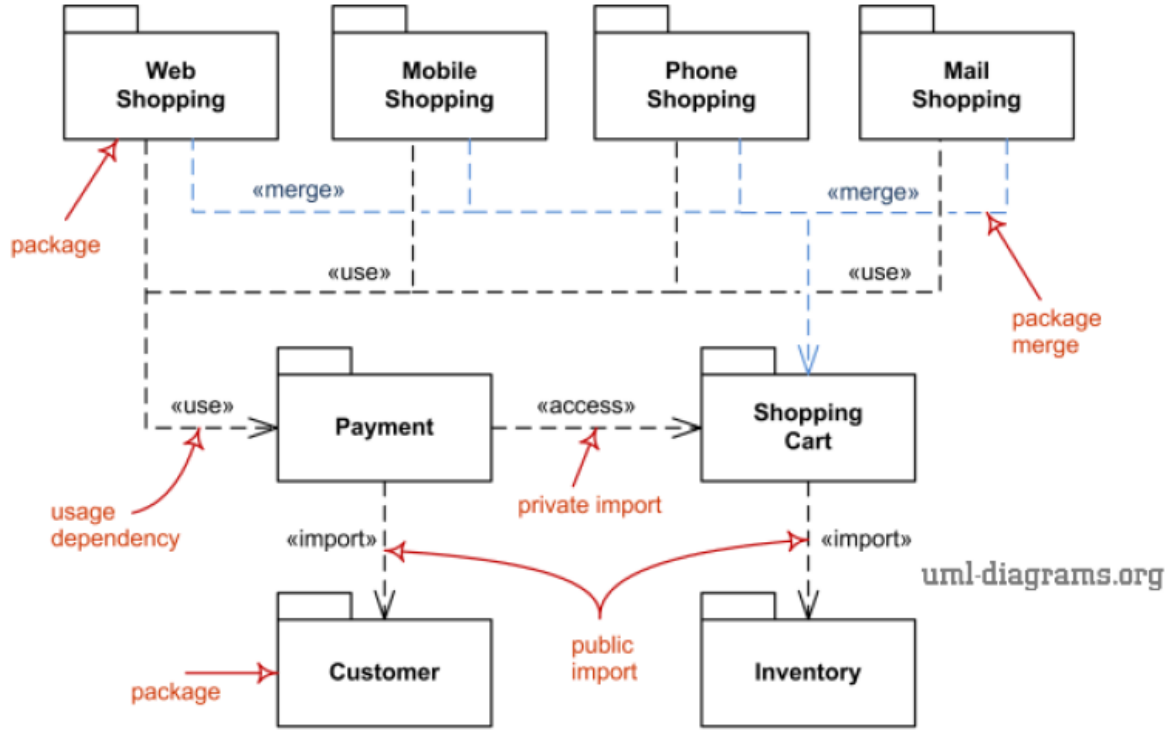


Temos, também, a Interface Fornecida, que designa uma interface que o próprio componente possui e oferece para outros componentes – o componente só pode ser acessado pela interface fornecida; e Interface Requerida, que designa uma interface necessária para que componente se comunique com outros componentes. Esta interface será conectada em uma interface fornecida de outro sistema. *Legal?*

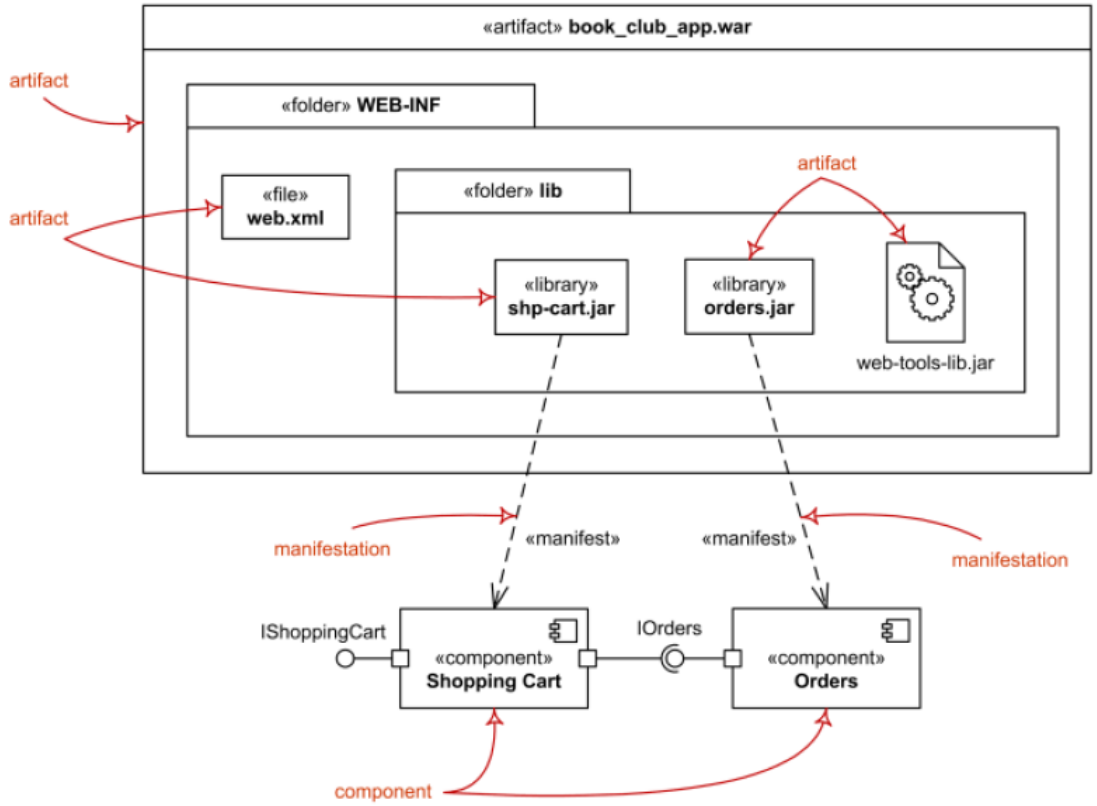


Quando utilizar Diagramas de Componentes? Use Diagramas de Componentes quando você estiver dividindo seu sistema em componentes e **quiser representar seus relacionamentos por intermédio de interfaces ou também quando quiser representar a decomposição de componentes em uma estrutura de nível mais baixo**. *Entendido, galera?*

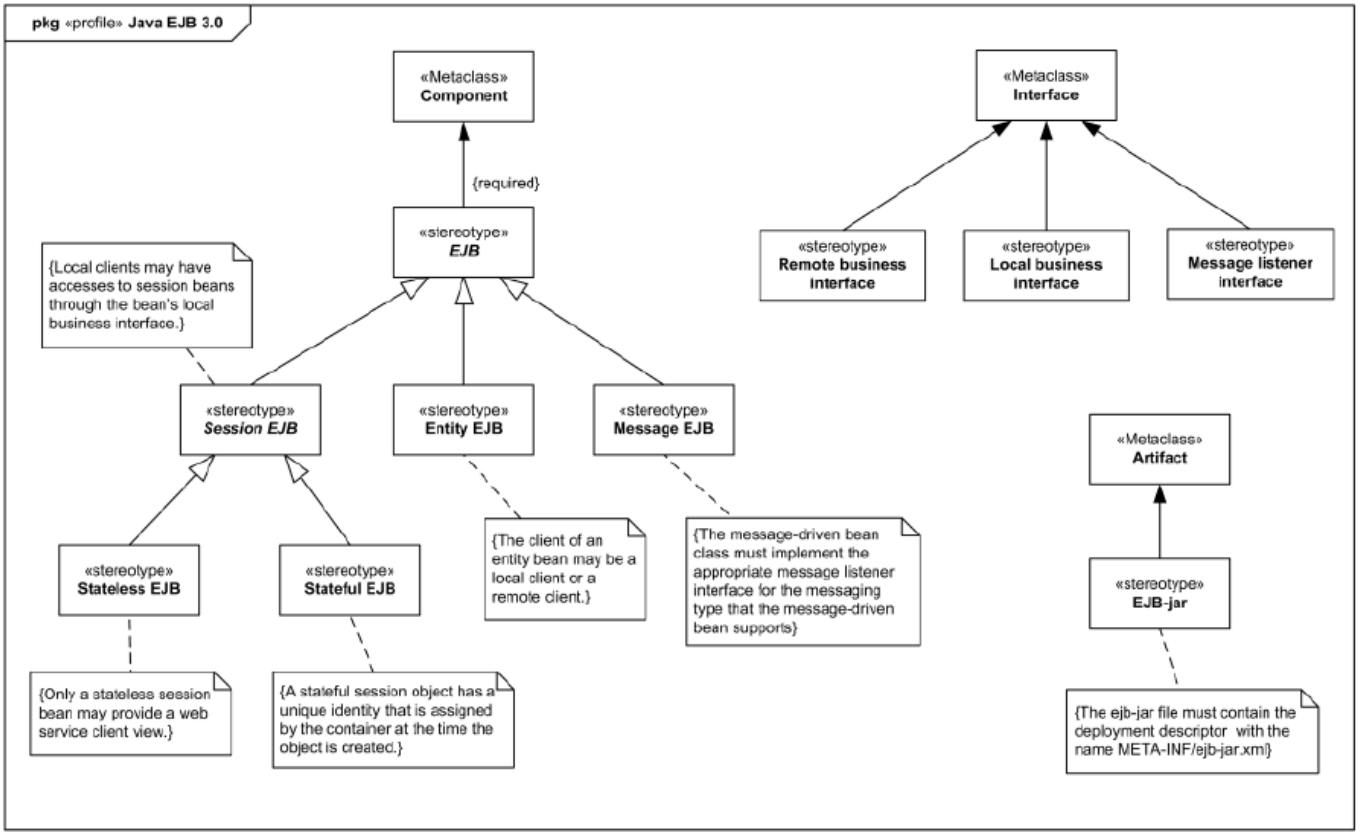
- **Diagrama de Pacotes:** Organiza os elementos do modelo em grupos lógicos (pacotes), mostrando as dependências entre eles. É crucial para gerenciar a complexidade de sistemas grandes.



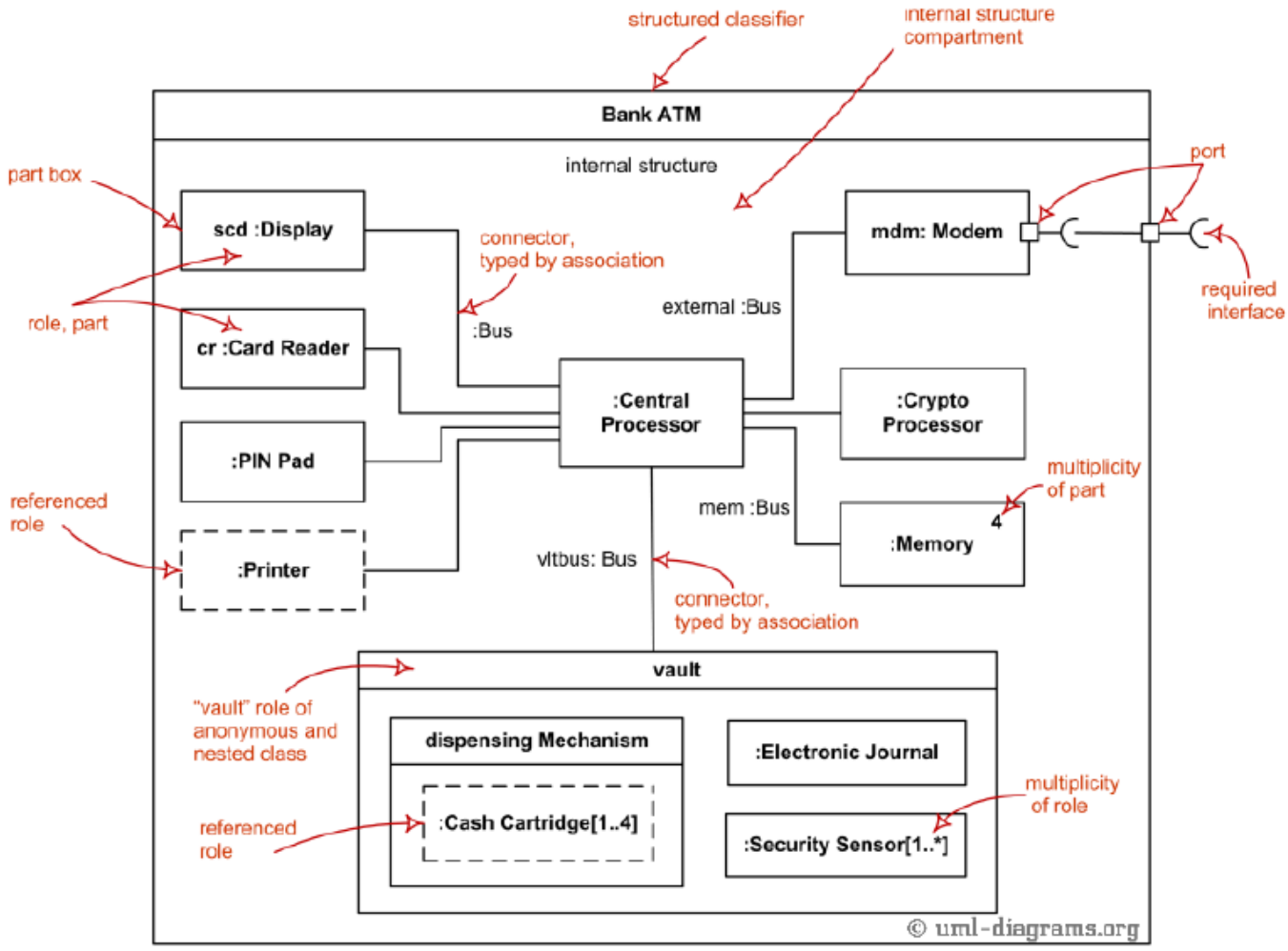
- **Diagrama de Implantação:** Descreve a arquitetura física do sistema, mostrando como os artefatos de software são distribuídos nos nós de hardware (servidores, dispositivos).



- **Diagrama de Perfil:** Opera no nível de metamodelo e é usado para criar customizações da UML para domínios específicos através da definição de um conjunto de estereótipos, tagged values e restrições.



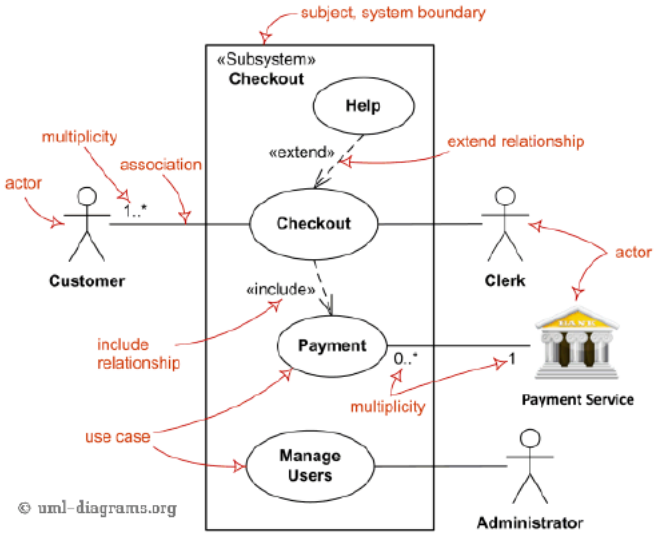
- **Diagrama de Estrutura Composta:** Detalha a estrutura interna de um classificador (como uma classe ou componente), mostrando suas partes internas e como elas colaboram para realizar uma funcionalidade.

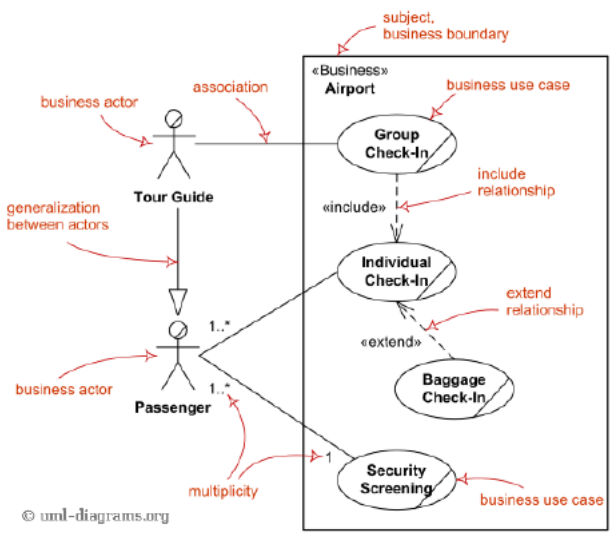


5. Análise Detalhada dos Diagramas Comportamentais

5.1. Diagrama de Casos de Uso

Captura os requisitos funcionais do sistema a partir da perspectiva do usuário. Descreve as interações entre atores e o sistema para atingir um objetivo.

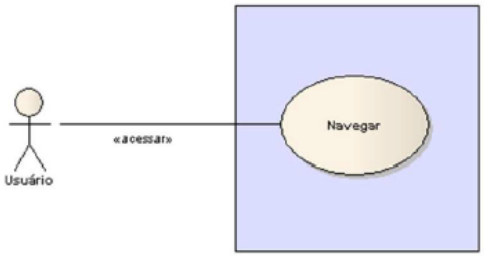




- **Ator:** Representa um papel desempenhado por uma entidade externa (humano, outro sistema, hardware) que interage com o sistema.

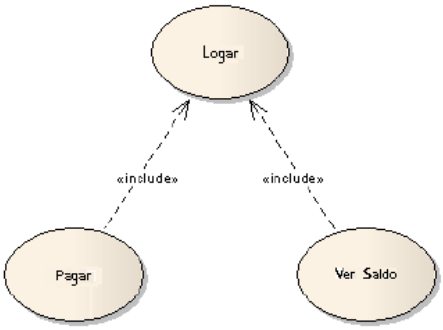
- **Caso de Uso:** Descreve uma sequência de ações que o sistema executa para produzir um resultado de valor observável para um ator.
- **Relacionamentos:**
    - **Comunicação (Associação):** Liga um ator a um caso de uso, indicando interação.

Relacionamento de Comunicação: também chamada de relacionamento de associação, o ator se comunica com o sistema por meio do envio e recebimento de mensagens. A imagem abaixo mostra a comunicação entre um ator e um caso de uso, representado por uma linha sólida no sentido do ator (Usuário) para o caso de uso (Navegar).



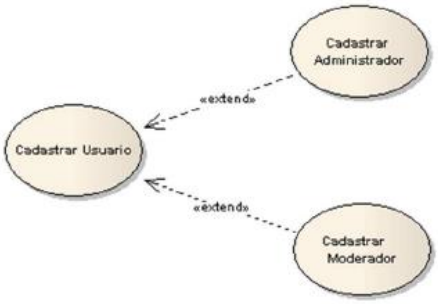
- **Inclusão (<<include>>):** Um caso de uso base **obrigatoriamente** inclui a funcionalidade de outro. Usado para reutilizar comportamento comum. A seta aponta do caso de uso base para o incluído.

Relacionamento de Inclusão: utilizado quando um mesmo comportamento se repete em mais de um caso de uso. A imagem abaixo apresenta o domínio de um *internet banking*. Observem que, para realizar um pagamento ou visualizar o saldo, é obrigatório fazer *Login*. **Logo, é um relacionamento obrigatório, representado por uma linha tracejada com uma seta na ponta**[1].



- **Extensão (<<extend>>):** Um caso de uso **opcionalmente** estende o comportamento de outro em um ponto específico (ponto de extensão). Usado para modelar fluxos alternativos ou excepcionais. A seta aponta do caso de uso extensor para o estendido.

Relacionamento de Extensão: utilizado quando se deseja modelar um relacionamento alternativo. A imagem abaixo apresenta o contexto de um fórum de discussões. Observem que para cadastrar um usuário, há duas opções: moderador ou administrador. **Logo, é um relacionamento opcional, representado por uma linha tracejada com uma seta na ponta**[2].



- INCLUSÃO: A ------> B SIGNIFICA QUE A INCLUI B, OU SEJA, PARA REALIZAR A, EU DEVO REALIZAR B;
- EXTENSÃO: A ------> B SIGNIFICA QUE A ESTENDE B, OU SEJA, PARA REALIZAR B, EU POSSO REALIZAR A;

- **Herança (Generalização):** Um ator/caso de uso especializado herda o comportamento de um mais genérico.

Relacionamento de Herança: relacionamento entre atores ou entre casos de uso, utilizado para reaproveitar comportamentos e estruturas. É útil para definir sobreposição de papéis entre atores e é representado com uma linha sólida com um triângulo no ator genérico. Na imagem seguinte, Vendedor é especialização de Pessoa.

Abaixo segue uma tabela com as possibilidades de relacionamentos entre os elementos do modelo de casos de uso.

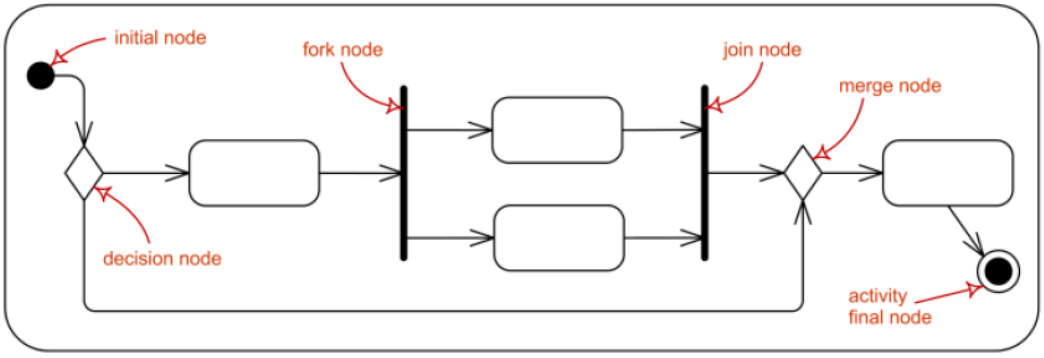
RELAÇÕES	COMUNICAÇÃO	EXTENSÃO	INCLUSÃO	HERANÇA
ENTRE CASOS DE USO		X	X	X
ENTRE ATORES				X
ENTRE CASOS DE USO E ATORES	X			

Quando utilizar Diagramas de Casos de Uso? Os casos de uso são uma ferramenta valiosa para ajudar no entendimento dos requisitos funcionais de um sistema. **Uma primeira passagem nos casos de uso deve ser feita no início**. Versões mais detalhadas dos casos de uso devem ser elaboradas apenas antes do desenvolvimento desse caso de uso.

É importante lembrar que casos de uso representam uma visão externa do sistema. Como tal, não espere quaisquer correlações entre eles e as classes dentro do sistema. **Com os casos de uso, concentra-se energia mais no texto do que no diagrama**. A despeito de a UML não dizer nada sobre o texto do caso de uso, é esse texto que contém todo o valor da técnica. Um grande perigo dos casos de uso é que as pessoas os tornam complicados demais e não conseguem prosseguir.

5.2. Diagrama de Atividades

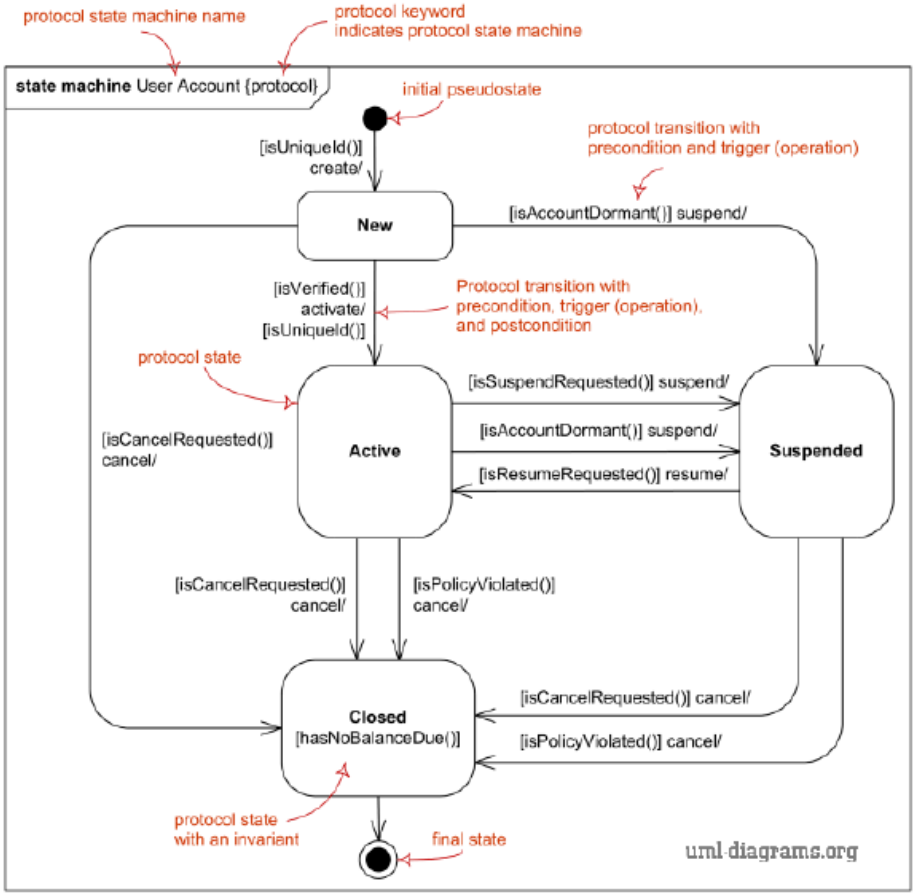
Modela fluxos de trabalho e processos de negócio, descrevendo a sequência de ações. É semelhante a um fluxograma, mas com suporte nativo a paralelismo.



- **Ação:** Um passo único dentro de uma atividade.
- **Nó de Decisão (Ramificação):** Representado por um losango, indica um ponto onde o fluxo se divide com base em uma condição.
- **Nó de Fork/Join (Bifurcação/União):** Representado por uma barra, divide o fluxo em múltiplos caminhos paralelos (fork) ou sincroniza múltiplos caminhos em um único fluxo (join).
- **Partições (Swimlanes):** Organiza as ações em colunas ou linhas que representam as responsabilidades de diferentes atores ou componentes.

5.3. Diagrama de Máquina de Estados

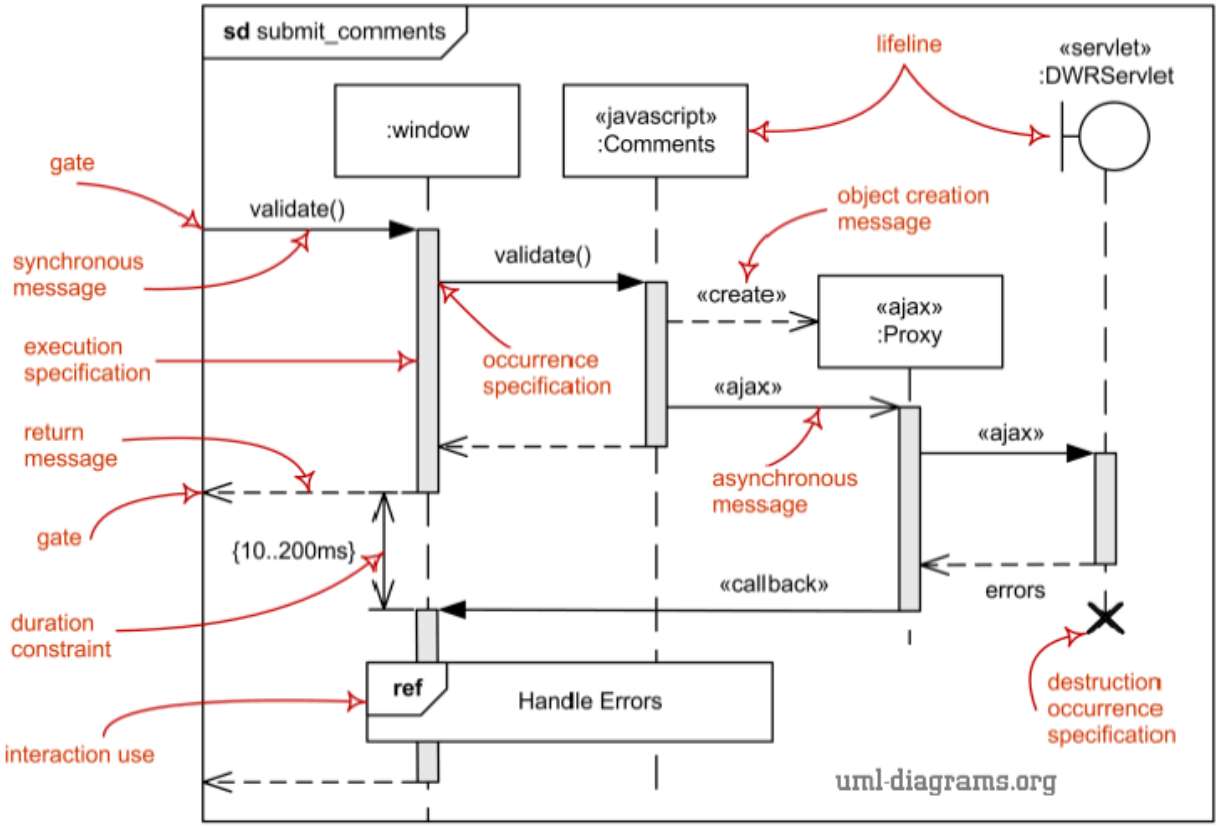
Descreve o ciclo de vida de um único objeto, mostrando os diferentes estados em que ele pode se encontrar e as transições entre esses estados em resposta a eventos.



- **Estado:** Uma condição ou situação na vida de um objeto.
- **Transição:** A passagem de um estado para outro, geralmente disparada por um evento.
- **Ação:** Uma atividade executada durante uma transição.

5.4. Diagrama de Sequência

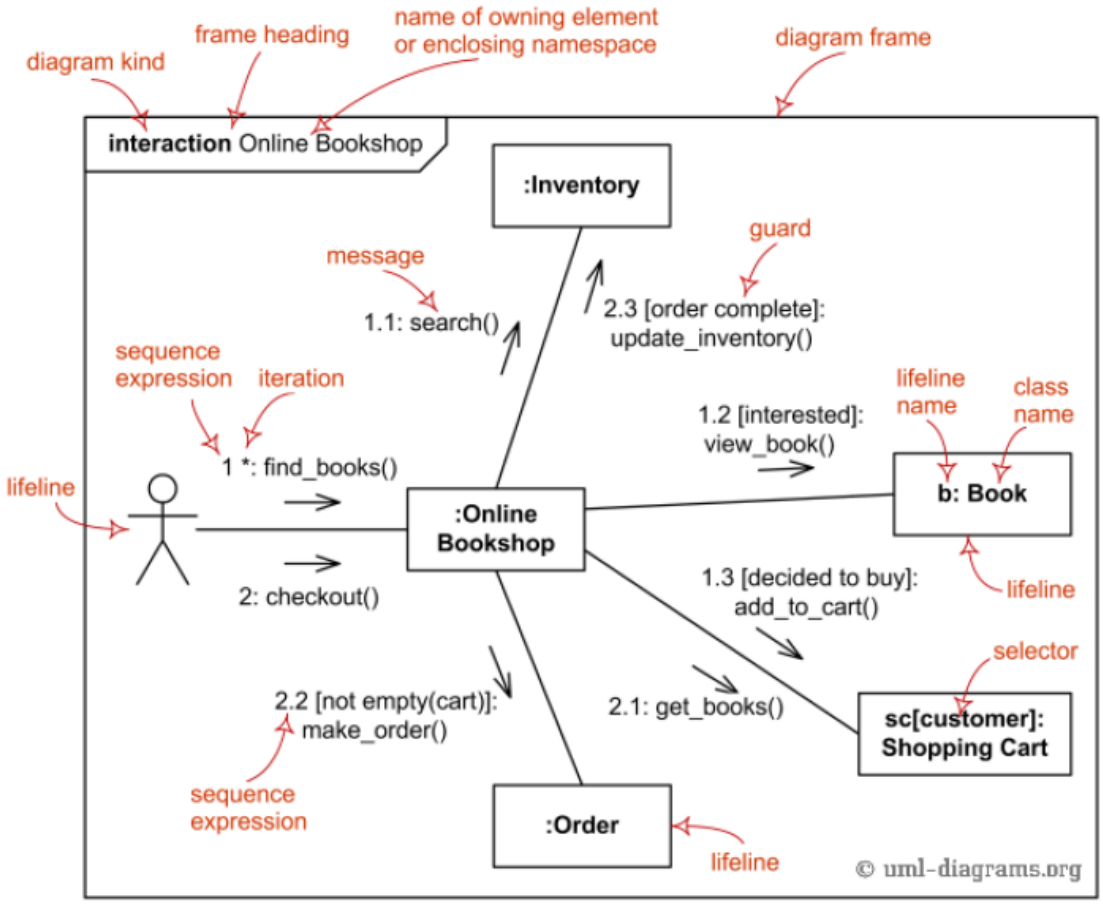
É um diagrama de interação que enfatiza a **ordem temporal** das mensagens trocadas entre objetos.



- **Eixo Vertical:** Representa o tempo (passa de cima para baixo).
- **Eixo Horizontal:** Representa os objetos participantes.
- **Linha de Vida (Lifeline):** Uma linha tracejada vertical que representa a existência de um objeto ao longo do tempo.
- **Mensagem:** Uma comunicação entre objetos, representada por uma seta.

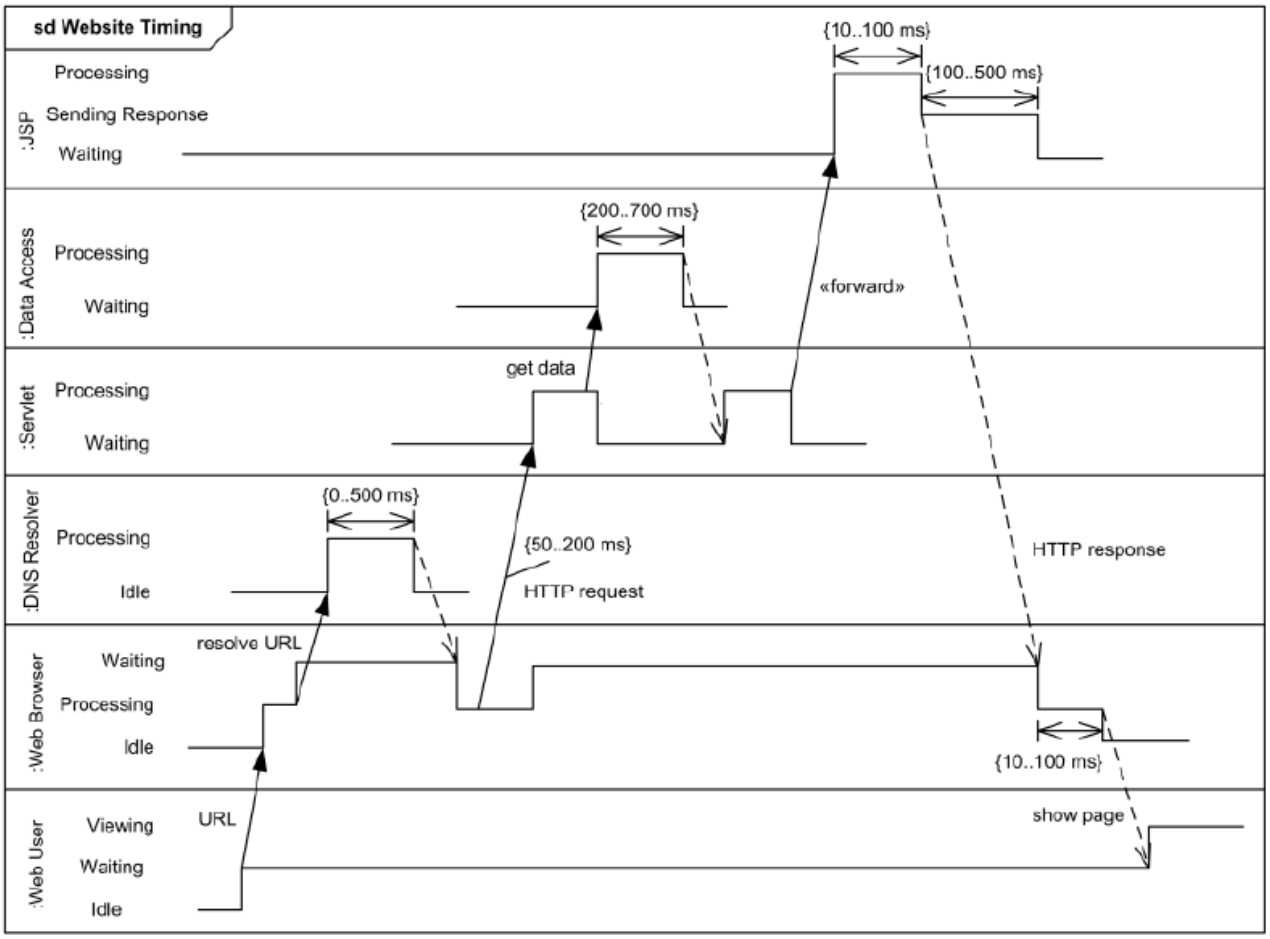
5.5. Diagrama de Comunicação

Anteriormente chamado de Diagrama de Colaboração, é um diagrama de interação que enfatiza a **organização estrutural** dos objetos e os links entre eles. Mostra as mesmas informações que um diagrama de sequência, mas foca nos relacionamentos em vez da cronologia. A ordem das mensagens é indicada por numeração sequencial.



5.6. Outros Diagramas Comportamentais

- **Diagrama de Tempo:** Foca em restrições de tempo, mostrando como os estados de um ou mais objetos mudam ao longo de uma linha do tempo e a duração em que permanecem em cada estado. É muito utilizado em sistemas de tempo real e por engenheiros de hardware.



- **Diagrama de Interação Geral:** Combina elementos do Diagrama de Atividades e do Diagrama de Sequência para fornecer uma visão geral do fluxo de controle entre interações complexas, mostrando como diferentes fragmentos de interação se encaixam.