



# UML (Unified Modelling Language)

---

**Universidade Federal do Maranhão - UFMA**

**Departamento de Informática**

**Processo de Desenvolvimento de Software**

Prof<sup>a</sup>.MSc Simara Rocha

[simararocha@gmail.com/simara@deinf.ufma.br](mailto:simararocha@gmail.com/simara@deinf.ufma.br)

Referências: Booch, G. et al. The Unified Modeling Language User Guide

Medeiros, E. Desenvolvendo Software com UML 2.0: Definitivo, Makron Books, 2006.

Sommerville, I. Engenharia de Software, 8ª edição, 2007.



# Sumário

---

- Introdução
- Descrição Arquitetônica
- Razões para Modelar
- Modelos
- Visões
- Diagramas
- Observações



# Introdução

---

- É uma linguagem para especificação, construção, visualização e documentação de artefatos de um sistema de software
- É mantida pelo Object Management Group (OMG), com contribuições e direitos de autoria das seguintes empresas: Hewlett-Packard, IBM, ICON Computing, i-Logix, IntelliCorp, Electronic Data Services, Microsoft, ObjecTime, Oracle, Platinum, Ptech, Rational, Reich, Softeam, Sterling, Taskon A/S e Unisys.



# Introdução

---

- A ênfase da UML é na definição de uma linguagem de modelagem padrão, e por conseguinte, a UML é independente das linguagens de programação, das ferramentas CASE, bem como dos processos de desenvolvimento.
- O objetivo da UML é que, dependendo do tipo de projeto, da ferramenta de suporte, ou da organização envolvida, devem ser adotados diferentes processos/metodologias, mantendo-se contudo a utilização da mesma linguagem de modelagem.



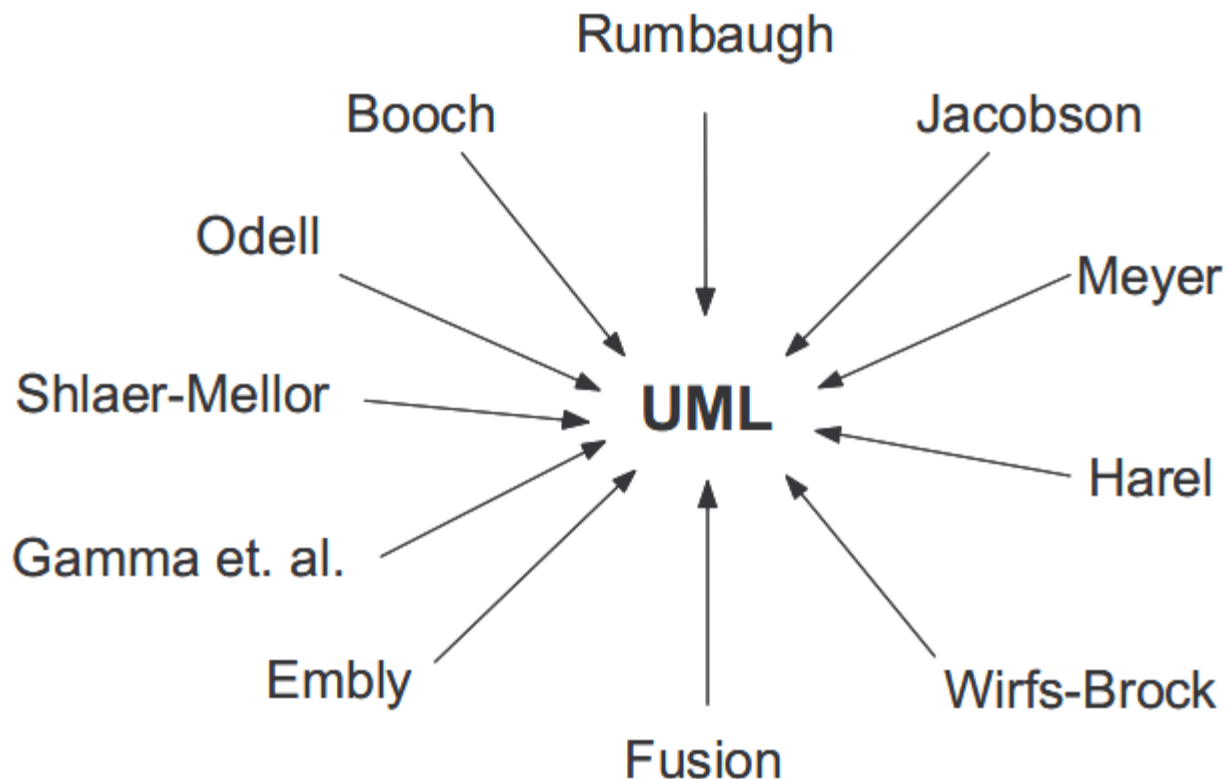
# Introdução

---

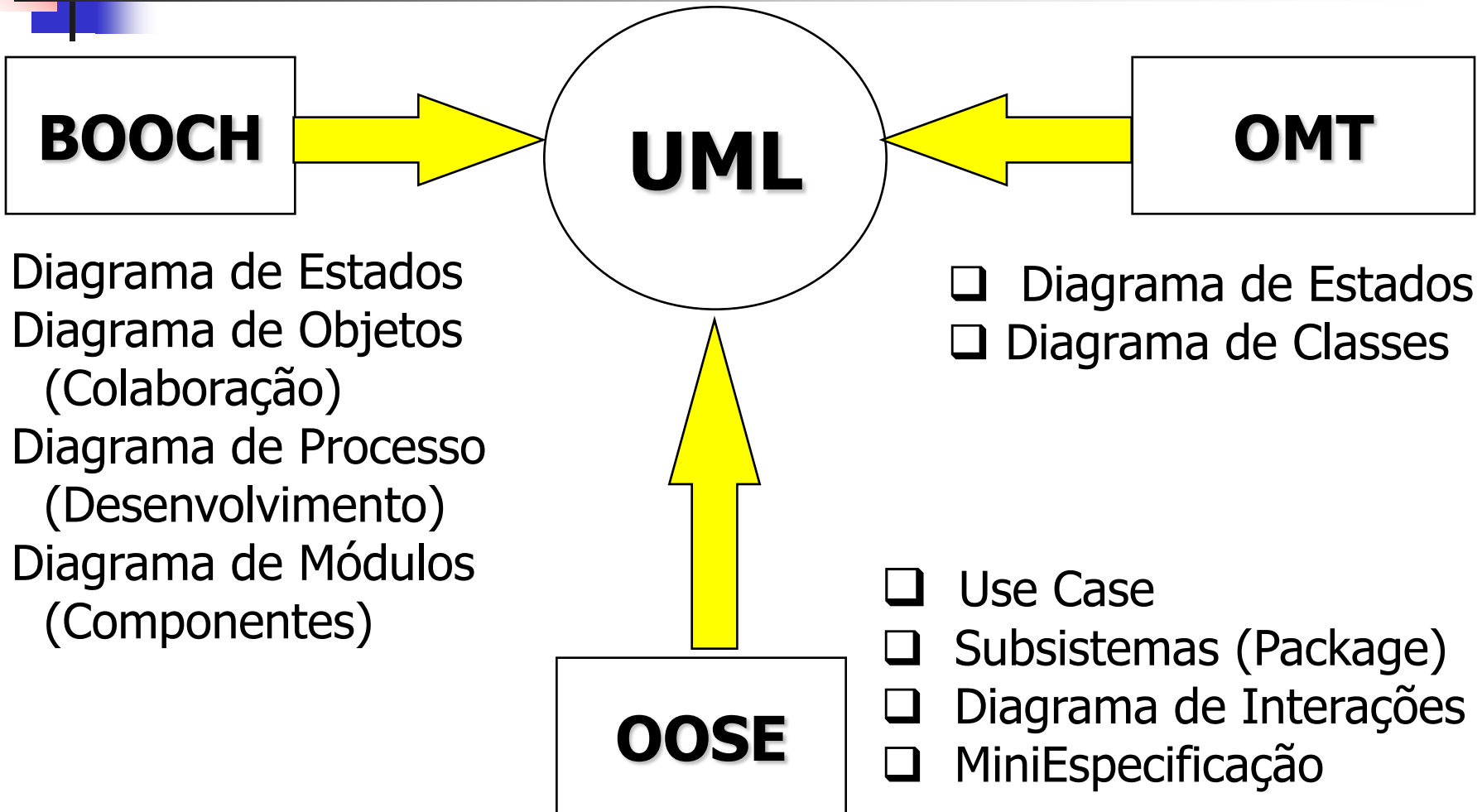
- A UML é usada no desenvolvimento dos mais diversos tipos de sistemas.
- Ela abrange sempre qualquer característica de um sistema em um de seus diagramas
- Aplicada em diferentes fases do desenvolvimento de um sistema, desde a especificação da análise de requisitos até a finalização com a fase de testes.
- O objetivo da UML é descrever qualquer tipo de sistema, em termos de diagramas orientado a objetos.

# Introdução

- Vantagem é a unificação de todas as notações anteriores



# Introdução





# Introdução

---

- A UML não é
  - um processo
  - uma metodologia
  - análise e projeto OO
  - regras de projeto
  - linguagem de programação





# Introdução

---

- UML é uma linguagem padrão da OMG para
  - Visualização,
  - Especificação,
  - Construção e
  - Documentação de software orientado a objetos.



# Visualização

---

- A existência de um modelo visual facilita a comunicação e faz com que os membros de um grupo tenham a mesma idéia do sistema.
- Cada símbolo gráfico tem uma semântica bem definida.
- O que modelamos?
  - Dimensões: dados, função, comportamento



# Especificação

---

- É uma ferramenta poderosa para a especificação de diferentes aspectos arquiteturais e de uso de um sistema.



# Construção

---

- Geração automática de código a partir do modelo visual
- Geração do modelo visual a partir do código
- Ambientes de desenvolvimento de software atuais permitem:
  - movimentações em ambos sentidos e
  - manutenção da consistência entre as duas visões.



# Documentação

---

- Pode incluir artefatos como:
  - Deliverables (documentos como especificação de requisitos, especificações funcionais, planos de teste, etc.).
  - Materiais que são importantes para controlar, medir, e refletir sobre um sistema durante o seu desenvolvimento e implantação.



# Descrição Arquitetônica

---

- UML oferece uma forma padrão de se desenhar as “plantas” (como em arquitetura) de um sistema de forma a incluir
  - aspectos abstratos (processos de negócio, funcionalidades do sistema)
  - aspectos concretos (classes C++/Java, esquemas de bancos de dados, componentes de software reutilizáveis)



# Razões para Modelar

---

- Comunicar a estrutura e o comportamento desejado de um sistema.
- Visualizar e controlar a arquitetura de um sistema.
- Para melhorar o nosso entendimento de um sistema e, assim, expor oportunidades para melhorias e reutilização.
- Para administrar os riscos



# Razões para Modelar

---

- A UML permite modelar:
  - Elementos;
  - Relacionamentos;
  - Mecanismos de extensibilidade;
  - Diagramas





# Elementos

---

- Para formar um modelo conceitual da linguagem é necessário aprender três elementos principais
  - Blocos de construção
  - Regras que determinam como esses blocos poderão ser combinados
  - Mecanismos comuns aplicados na UML



# Blocos de Construção

---

- Três tipos:
  - Itens: são abstrações
  - Relacionamentos: os relacionamentos reúnem esses itens
  - Diagramas: agrupam coleções interessantes de item



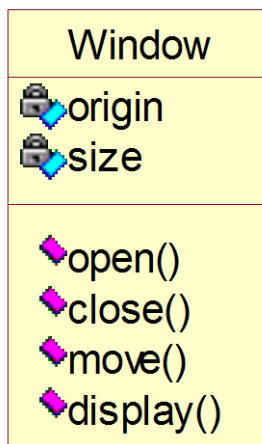
# Itens da UML

---

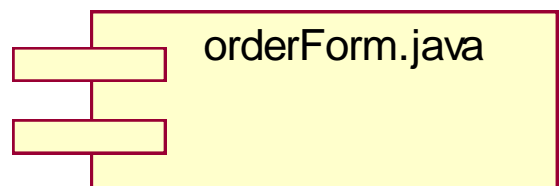
- Estruturais
- Comportamentais
- De agrupamento
- Anotacionais

# Itens estruturais

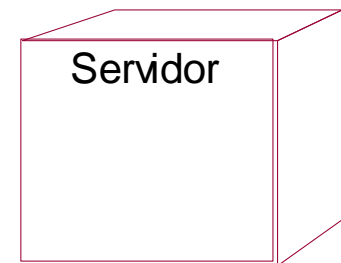
- São os substantivos dos modelos. São a parte estática, representando elementos conceituais ou físicos
- Sete tipos: classes, interfaces, colaborações, casos de uso, classes ativas, componentes e nós



**Classe**



**Componente**



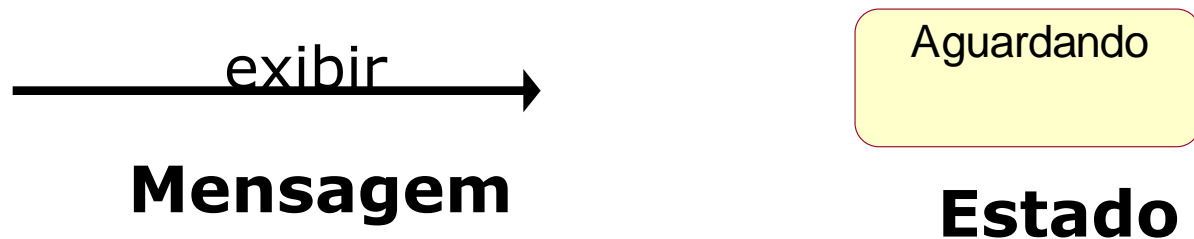
**Nó**



# Itens comportamentais

---

- Representam as partes dinâmicas dos modelos. São os verbos, representando comportamentos no tempo e no espaço
- Dois tipos: interação e máquina de estado

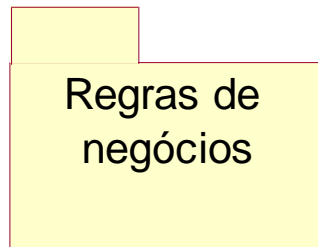




# Itens de agrupamento

---

- São as partes organizacionais dos modelos de UML. São os blocos em que os modelos podem ser decompostos – pacotes
- Um pacote é um mecanismo de propósito geral para a organização de elementos em grupos



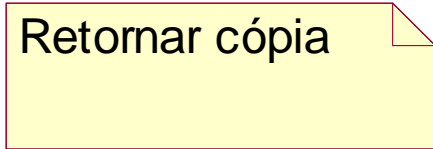
**Pacote**



# Itens anotacionais

---

- Partes explicativas dos modelos UML. São comentários, incluídos para descrever, esclarecer e fazer alguma observação importante sobre qualquer elemento do modelo - notas



Retornar cópia

Nota



# Relacionamentos

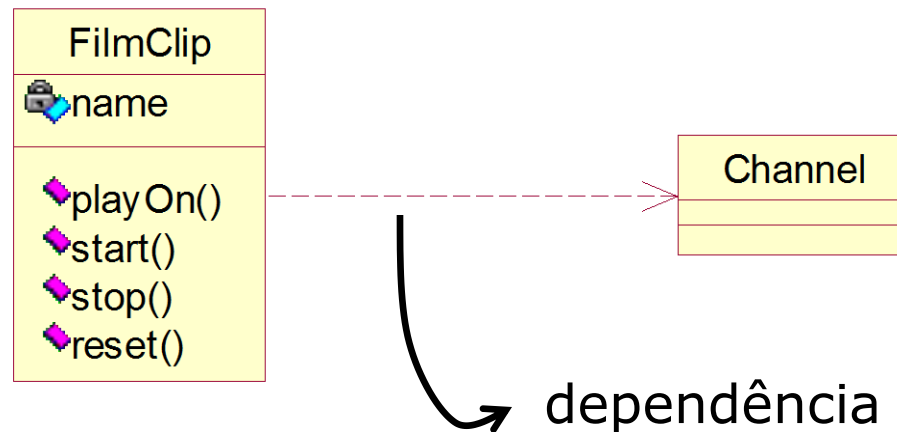
---

- Dependências
- Associações
- Generalização
- Realização



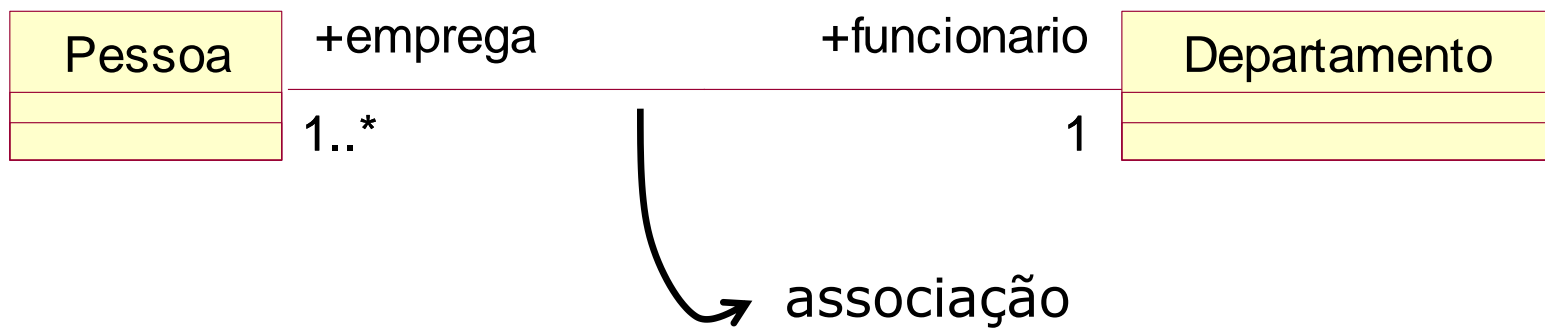
# Dependência

- Relacionamento semântico entre dois itens, nos quais a alteração de um (o item independente) pode afetar a semântica do outro (o item dependente)



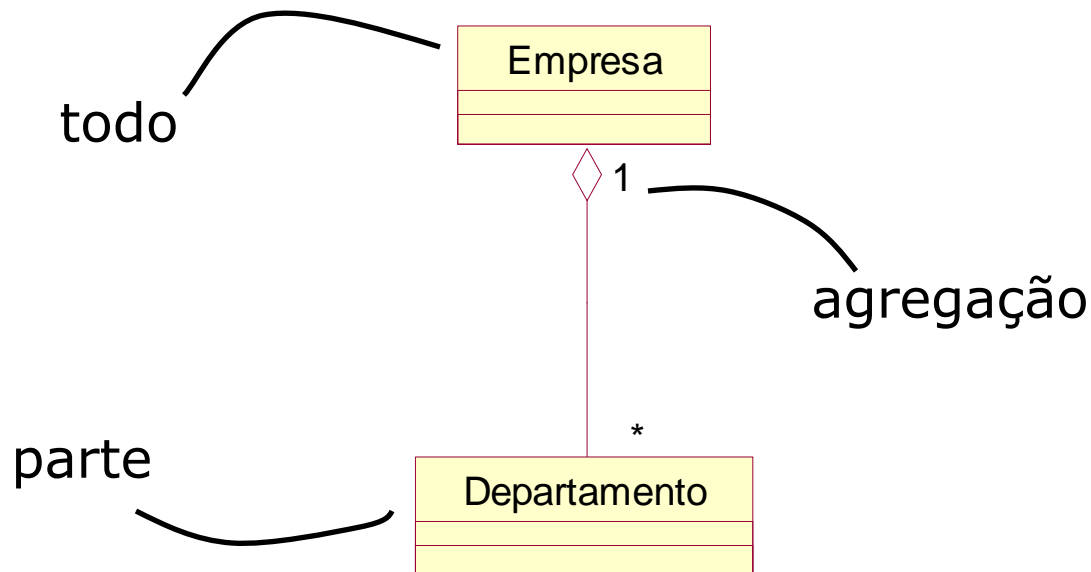
# Associação

- É um relacionamento estrutural que descreve um conjunto de ligações, em que as ligações são conexões entre objetos



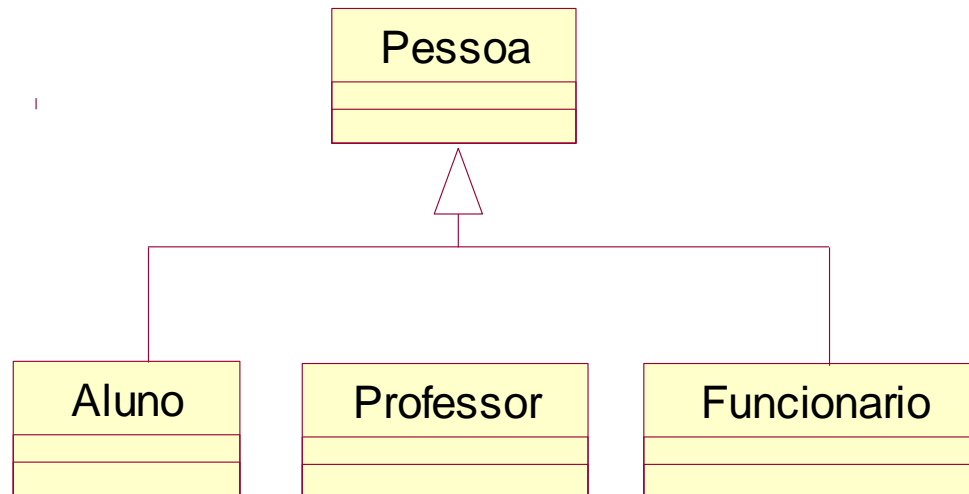
# Associação: Agregação

- A agregação é um tipo especial de associação representando um relacionamento estrutural entre o todo e sua parte



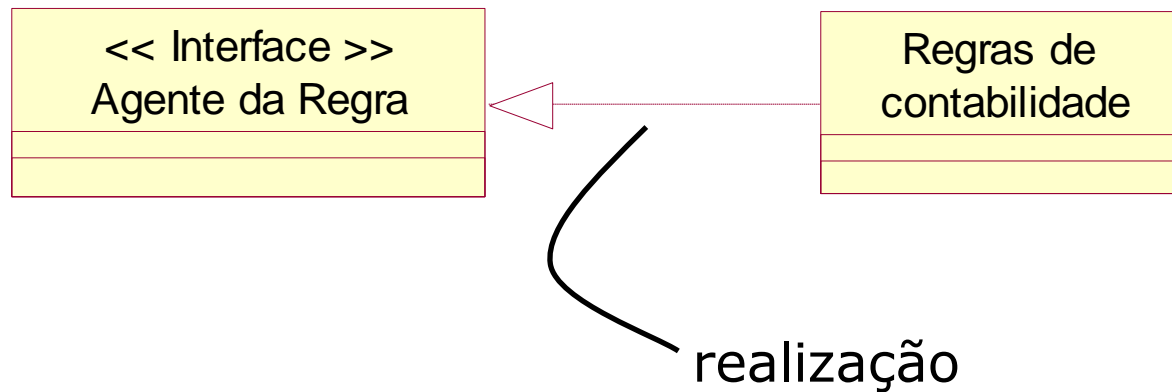
# Generalização

- É um relacionamento de especialização/generalização, nos quais os objetos dos elementos especializados (os filhos) são substituíveis por objetos do elemento generalizado (os pais)



# Realização

- É um relacionamento semântico entre classificadores, em que um classificador especifica um contrato que outro classificador garante executar





# Mecanismos de extensibilidade

---

- Esteriótipos
- Regras
- Valores marcados (tagged values)
  - {aluno = "placido neto"; versao = 2.3}



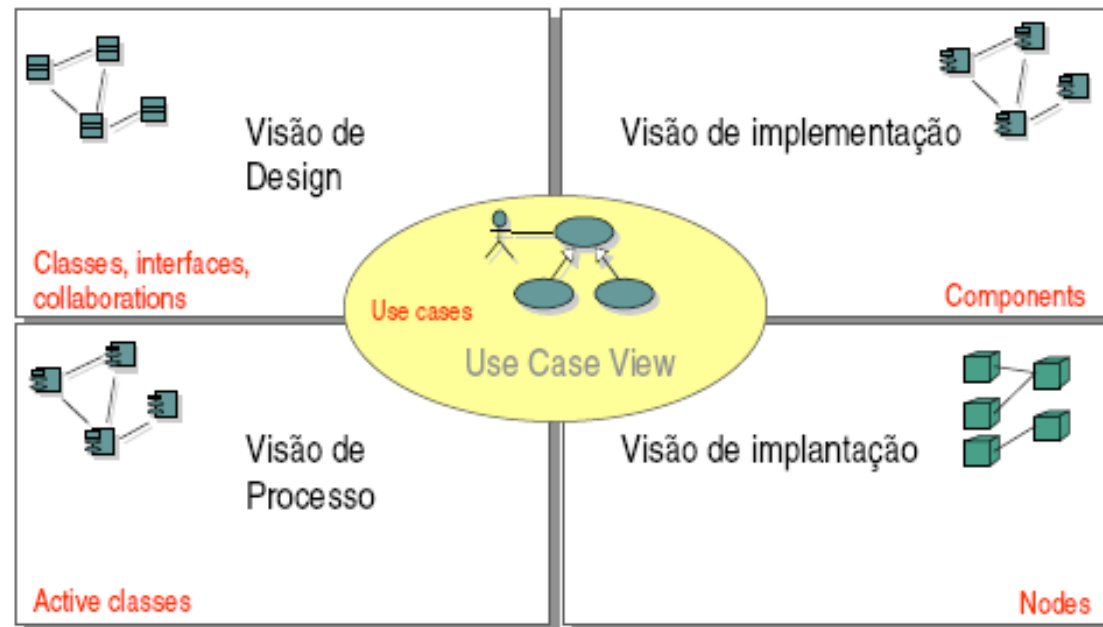
# Modelos

---

- Um modelo é uma simplificação da realidade
- Um modelo é representado por um ou mais diagramas. Desta forma, um diagrama pode ser visto como uma visão dentro de um modelo.
- Um diagrama pode ser representado de várias formas, dependendo de quem irá interpretá-lo.

# Visão

- Um diagrama é uma visão sobre um modelo



**Organização**

**Comportamento**



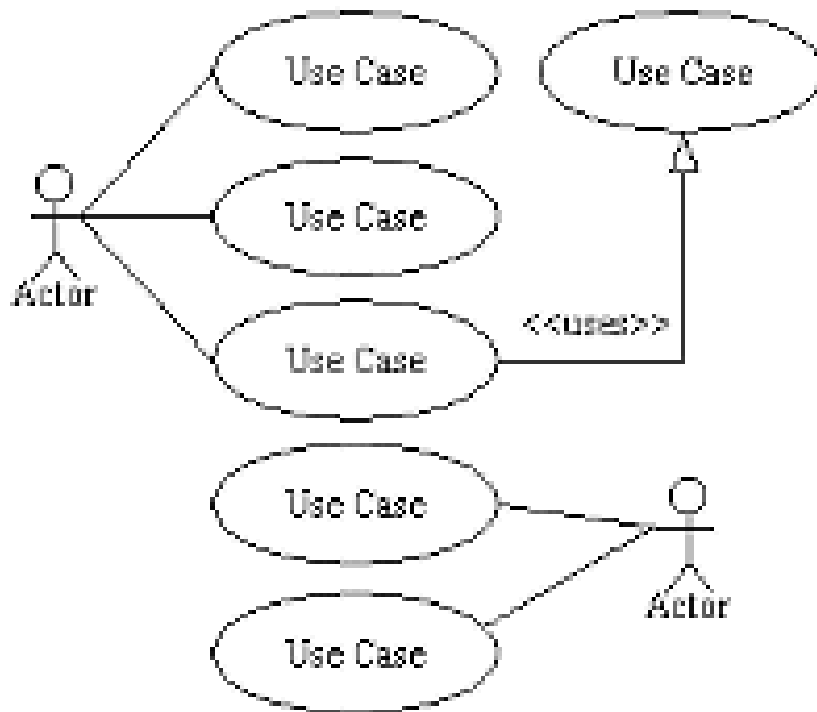


# Diagrama

---

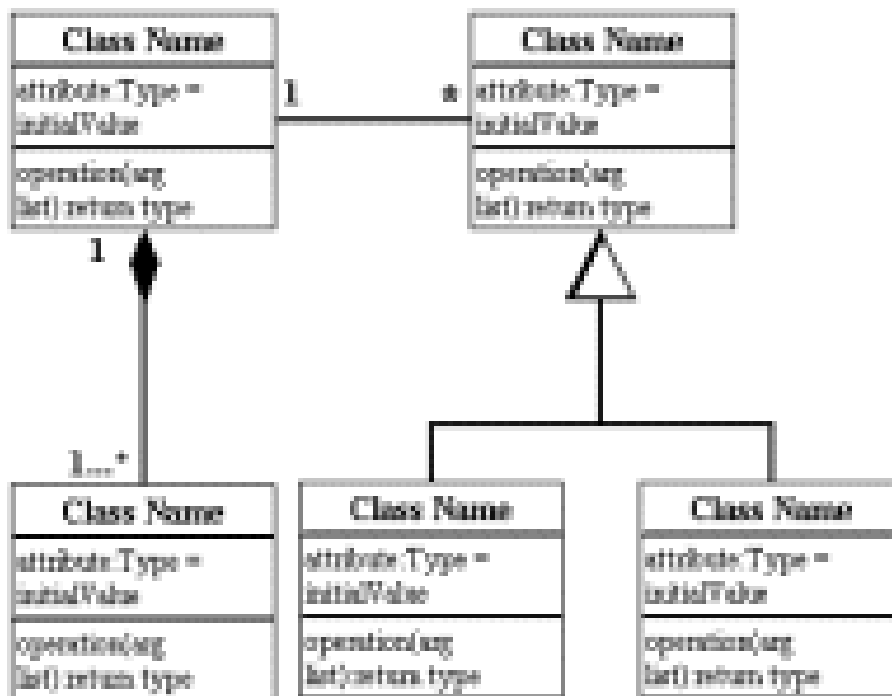
- Um diagrama provê uma parcial representação do sistema.
- Ele ajuda a compreender a arquitetura do sistema em desenvolvimento.
- Os diagramas:
  - Caso de uso, classes, sequência, objeto, colaboração, atividade, estado, implantação, pacotes, componentes

# Diagramas de casos de uso



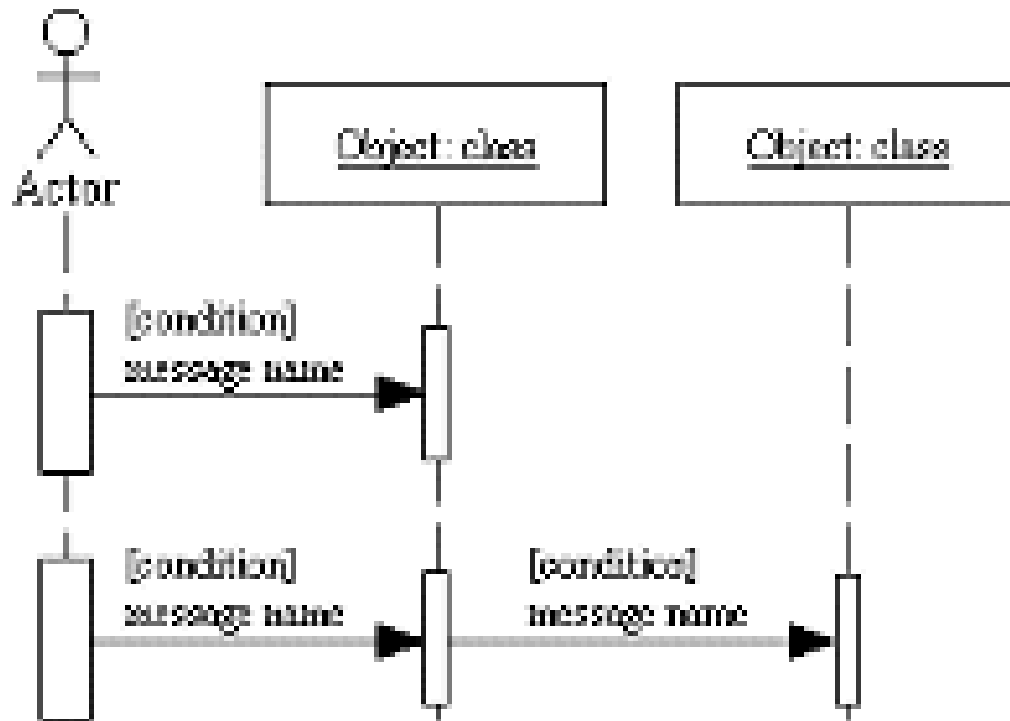
- Modelam a funcionalidade do sistema através de atores e casos de uso
- Casos de uso são serviços ou funções fornecidas pelo sistema aos seus usuários

# Diagramas de classes



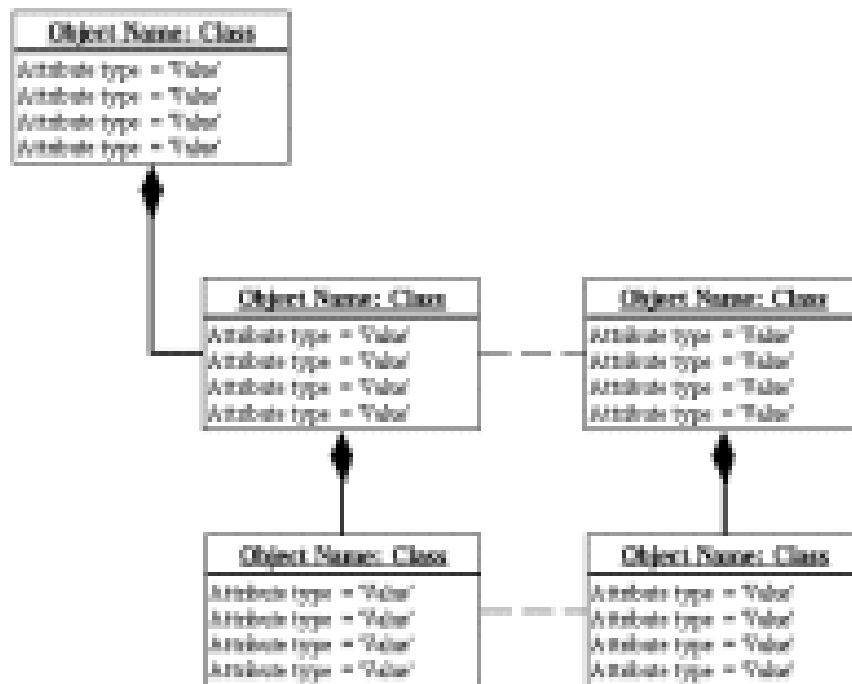
- Diagramas de classe são a espinha dorsal da maioria dos métodos orientados a objeto, inclusive UML
- Descrevem a estrutura estática do sistema (entidades e relacionamentos)

# Diagramas de seqüências



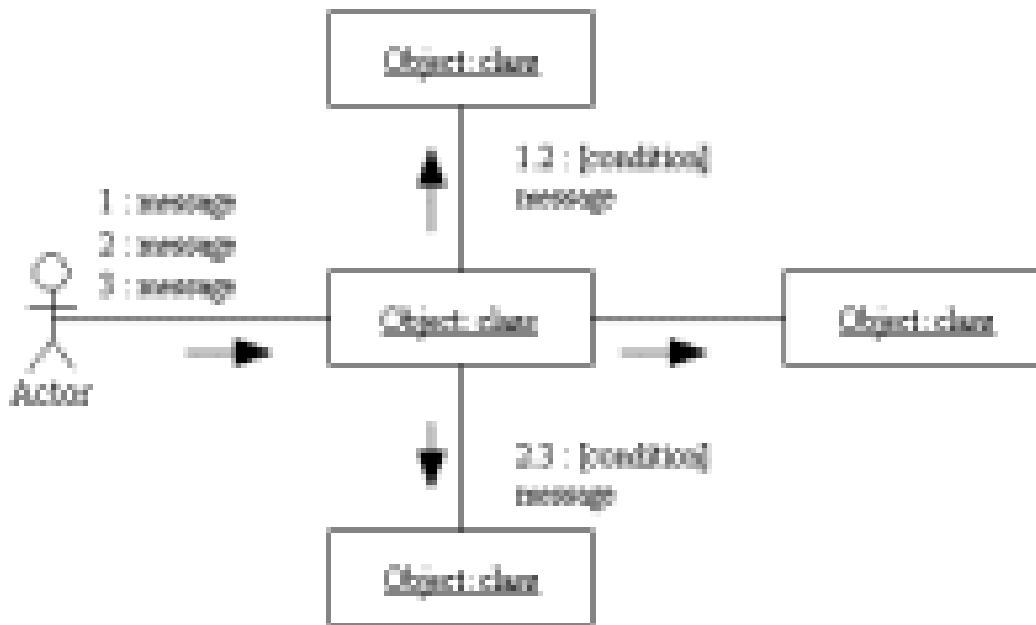
- Descreve as interações entre as classes através das trocas de mensagens ao longo do tempo

# Diagramas de objetos



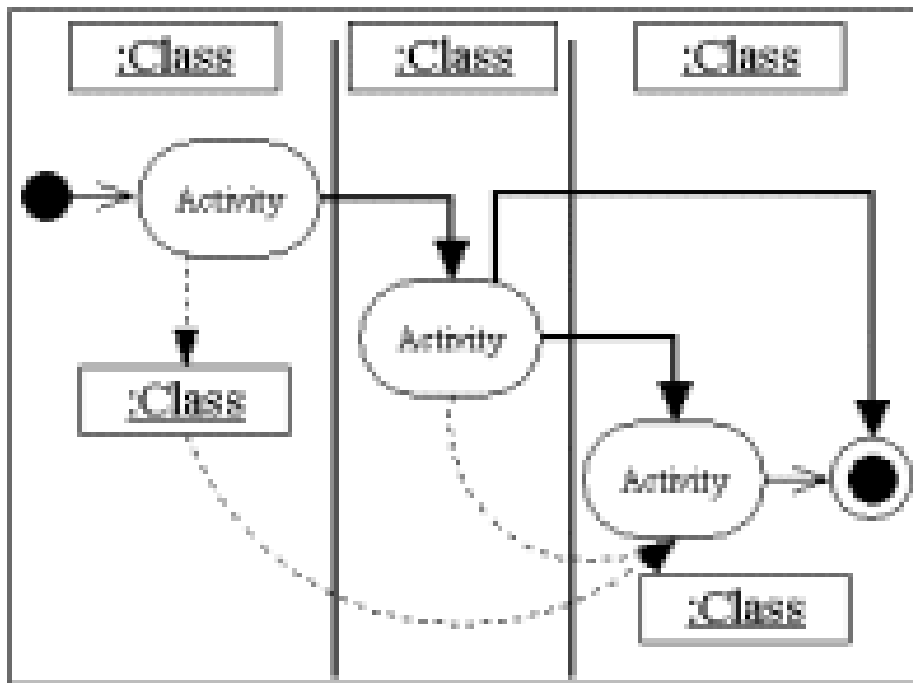
- Descrevem a estrutura estática de um sistema em um determinado momento
- Podem ser usados para testar a precisão dos diagramas de classe

# Diagramas de Comunicação



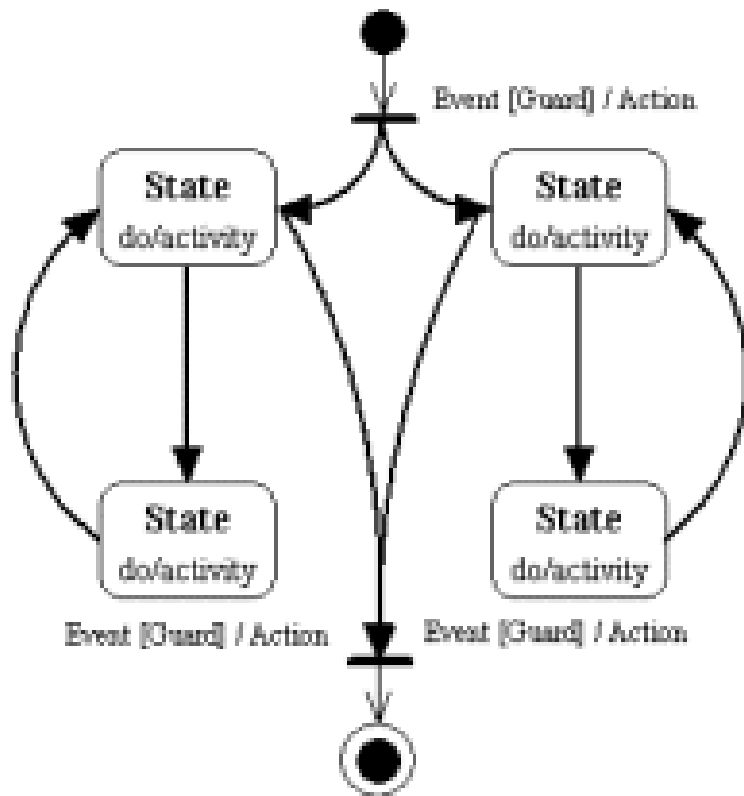
- Representam as interações entre objetos em termos de mensagens em seqüência
- Descrevem tanto a estrutura estática como o comportamento dinâmico do sistema

# Diagramas de atividades



- Ilustram a natureza dinâmica de um sistema modelando o fluxo de controle de uma atividade para outra
- Uma atividade representa uma operação em uma classe do sistema que resulta na mudança do estado do sistema
- Tipicamente, são usados para modelar fluxo de trabalho ou processos de negócio e funcionamento interno

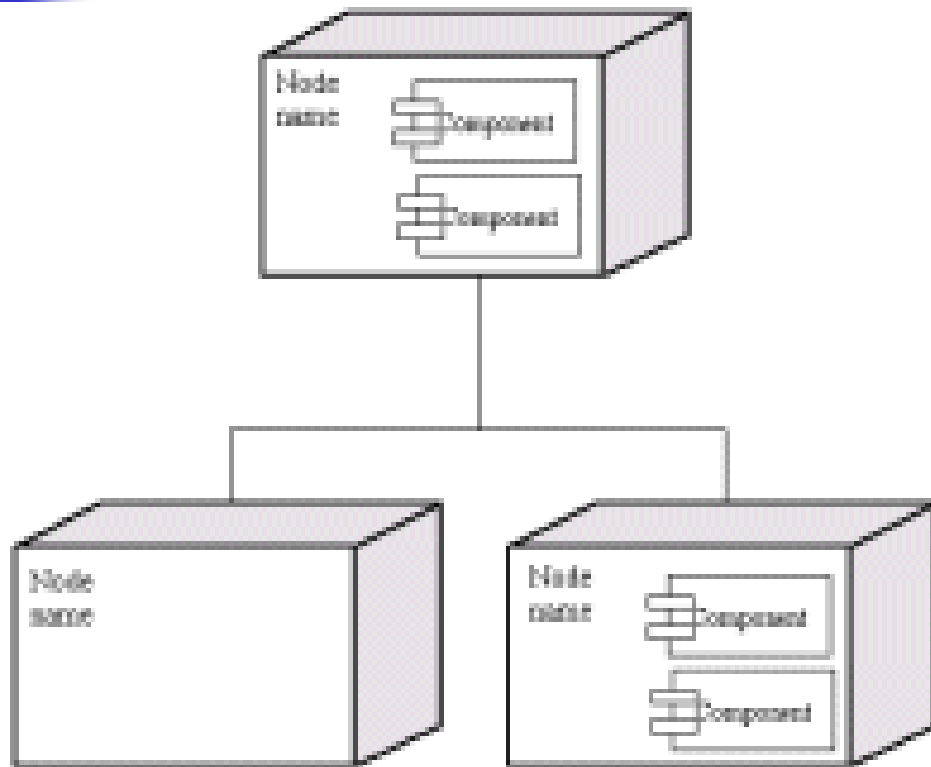
# Diagramas de estados



- Descrevem o comportamento dinâmico do sistema em resposta a estímulos externos
- São especialmente úteis para modelar objetos reativos cujos estados são disparados por eventos específicos

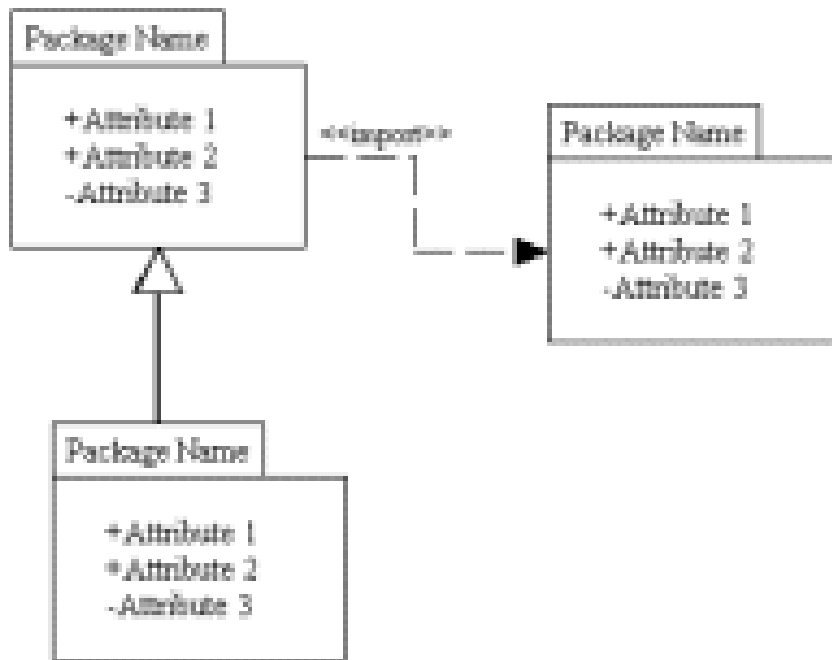


# Diagramas de implantação



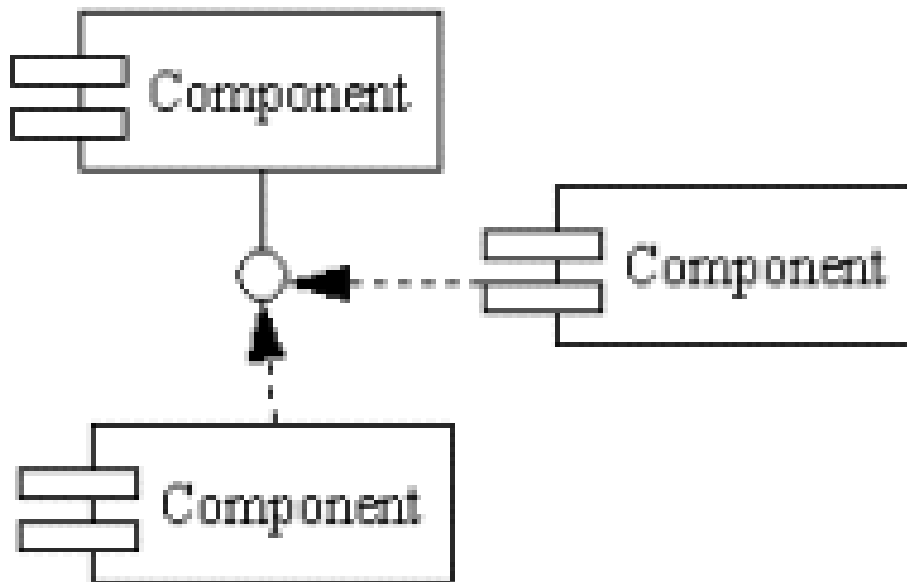
- Descrevem os recursos físicos em um sistema, incluindo nós, componentes e conexões

# Diagramas de pacotes



- Organizam elementos do sistema em grupos relacionados a fim de minimizar a dependência entre eles

# Diagramas de componente



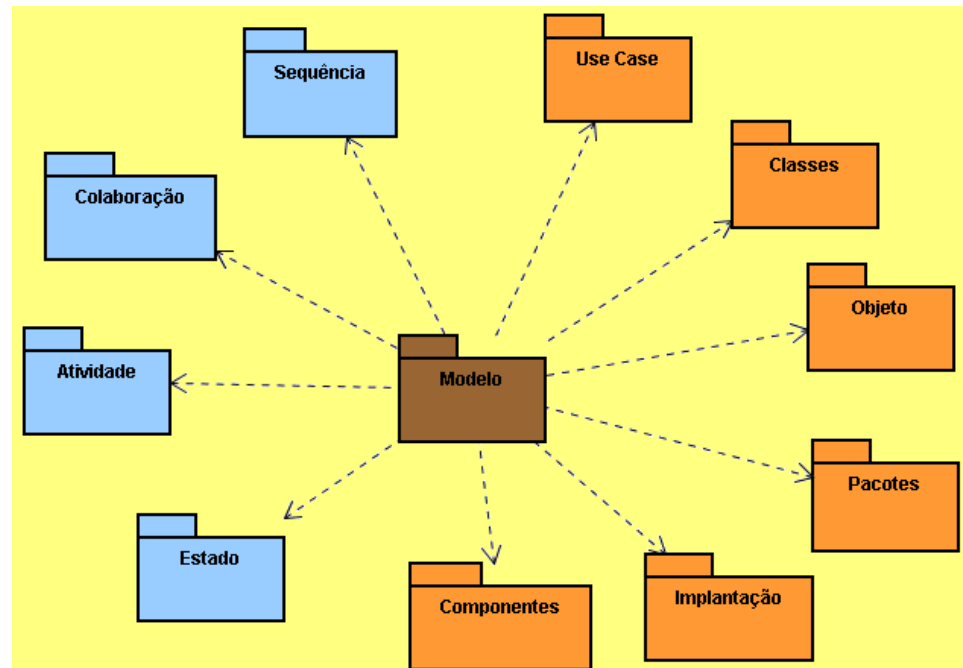
- Descreve a organização dos componentes físicos de software
- Ex.: código-fonte, código em tempo de execução (binário) e executáveis

# Diagrama

- Os diagramas UML são abordados como **Estáticos** e **Dinâmicos**.

**Estes diagramas também  
Podem ser classificados  
como:**

**Comportamentais**  
**Estruturais**





# Observações

---

- UML não é uma metodologia:
  - Não diz quem deve fazer o que, quando e como;
  - UML pode ser usado segundo diferentes metodologias, como o RUP.
- UML não é uma linguagem de programação.