Engenharia de Software

Linguagem UML
Parte 1

Luís Morgado

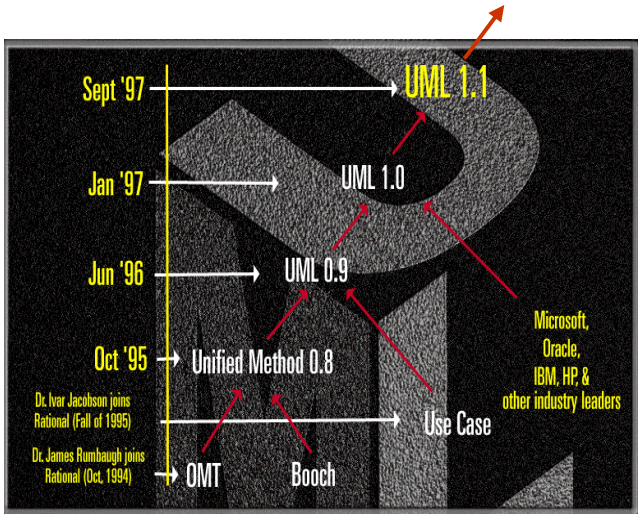
Instituto Superior de Engenharia de Lisboa

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Linguagem UML

A linguagem UML (*Unified Modeling Language*) é uma linguagem de modelação geral de sistemas, orientada em particular para a modelação de software, com as seguintes características:

- **É uma linguagem orientada para a concepção e modelação de software de forma organizada e sistemática**, facilitando a compreensão e descrição dos modelos desenvolvidos
- **É uma linguagem normalizada** - no âmbito do OMG (*Object Management Group*) - que facilita a compreensão e comunicação dos modelos, quer para quem concebe os modelos, quer para quem os utiliza
- **Facilita a produção de código**, possibilitando uma tradução organizada e sistemática dos modelos para código, incluindo a geração automática de código
- **Adopta uma abordagem orientada a objectos**, o que contribui para lidar com a complexidade, bem como para facilitar a reutilização de código
  - Resultou da combinação de várias metodologias de modelação orientadas a objectos
    - OMT (*Jim Rumbaugh*)
    - Booch Method (*Grady Booch*)
    - OOSE (*Ivar Jacobson*)



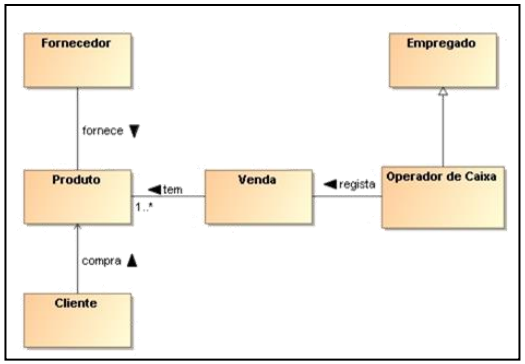
[Rational]

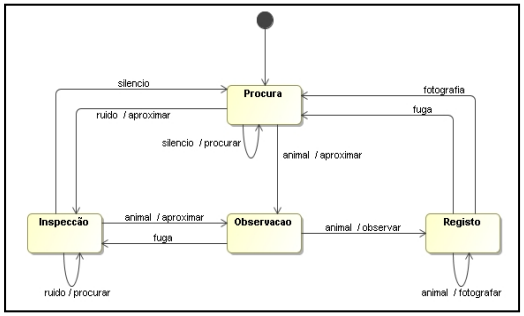
Linguagem UML

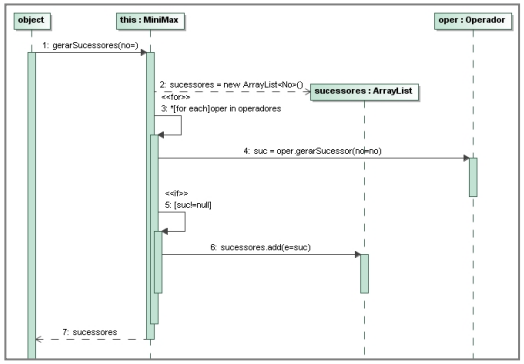
- Áreas de aplicação
  - Modelação de arquitectura lógica e física
    - Estrutura
      - Descrição da estrutura estática de um sistema
    - Comportamento
      - Descrição de como o sistema age e reage na relação com o seu ambiente ao longo do tempo
  - Modelação de processos de negócio
  - Modelação de dados
  - Modelação geral de sistemas
- Modos de utilização
  - Esboços
  - Especificações a diferentes nível de detalhe
  - Geração / análise de código (*forward / reverse engineering*)

Linguagem UML

- Linguagem gráfica
  - Conceitos
    - Representados por símbolos gráficos
    - Organizados em diagramas
  - Eficácia
    - Descrição
    - Comunicação
    - Compreensão
- Elementos base
  - Classificadores
    - Meta-elementos da linguagem UML que definem tipos de elementos com características comuns, por exemplo, *classe*
  - Mecanismos de extensão
    - Mecanismo de adaptação da linguagem para atender a necessidades específicas de um domínio ou plataforma
  - Diagramas
    - Representação gráfica de elementos e relações de um sistema em diferentes perspectivas

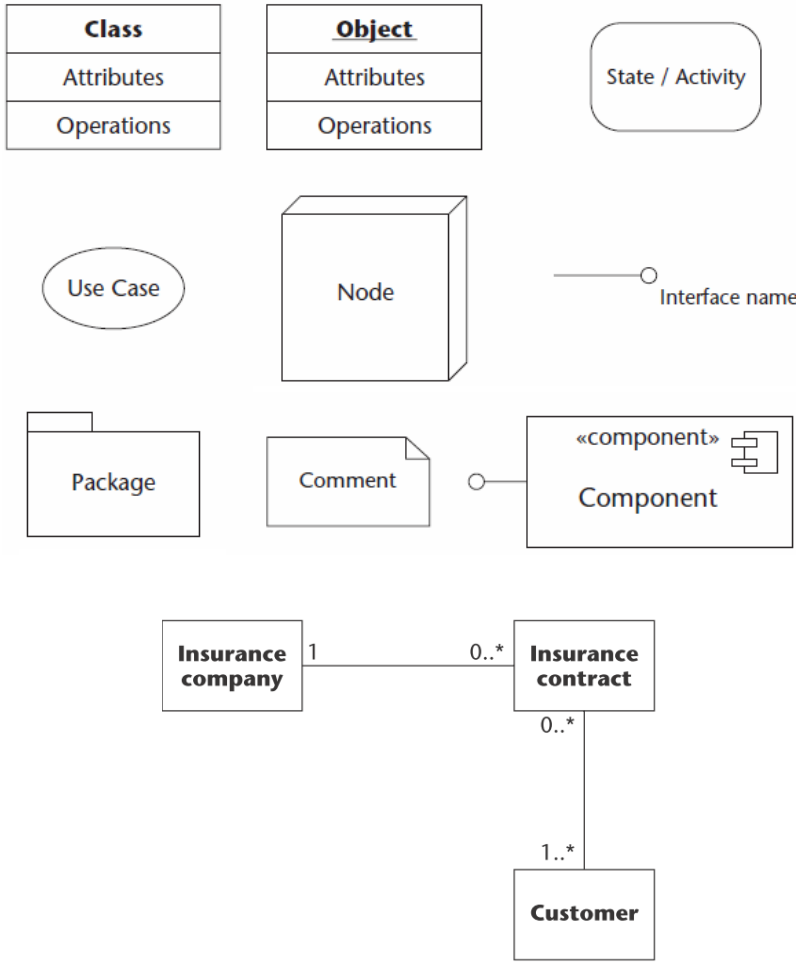






Linguagem UML

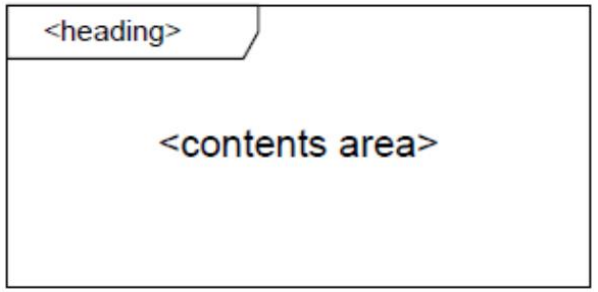
- Classificadores (tipos de elementos)
  - Actor
  - Class
  - Component
  - Datatype
  - Interface
  - Node
  - Signal
  - Subsystem
  - Use Case
  - Association



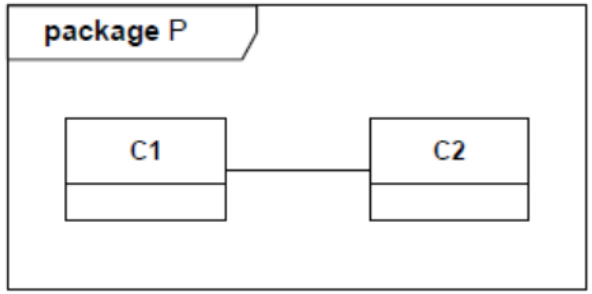
[Eriksson et al., 2004]

Linguagem UML

- Diagramas
  - Na linguagem UML os modelos são organizados em diagramas que permitem representar diferentes perspectivas de modelação de um sistema
  - A representação gráfica facilita a percepção das relações entre elementos
- Moldura (*Frame*)
  - Define o tipo de diagrama
  - Enquadra os elementos em diferentes perspectivas de modelação de um sistema



Representação de um diagrama com cabeçalho (*heading*) que indica o tipo e nome do diagrama, e a área de conteúdo onde são representados os elementos do diagrama

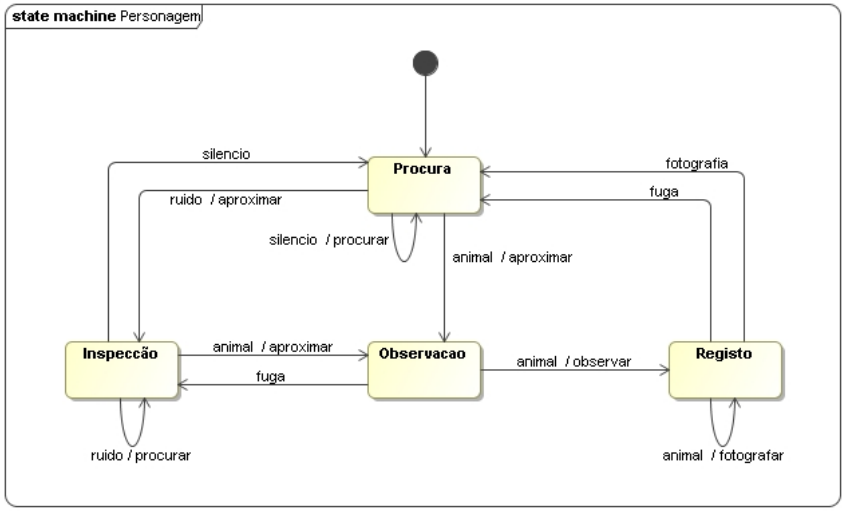


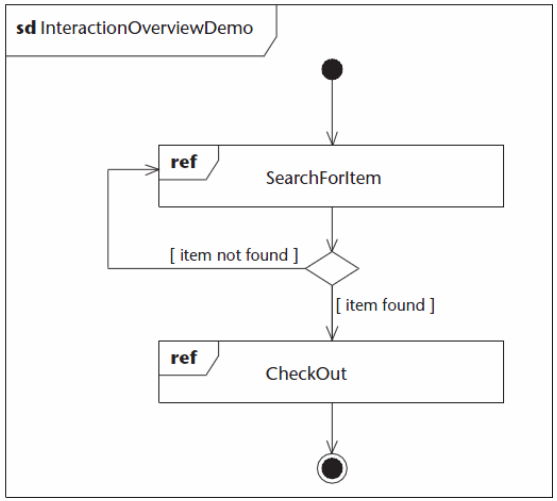
Exemplo de um diagrama que representa uma pasta (*package*) cujo nome é P, o qual é composto por duas classes com uma associação entre as classes

Linguagem UML

- Tipos de molduras
  - **class**: diagrama de classes
  - **act** (**activity**): diagrama de actividade
  - **cmp** (**component**): diagrama de componentes
  - **dep** (**deployment**): diagrama de implantação
  - **sd** (**sequence**): Representa todos os tipos de diagramas de interacção
  - **pkg** (**package**): diagrama de pastas
  - **stm** (**state machine**): diagrama de máquina de estados
  - **uc** (**use case**): diagrama de casos de utilização

Exemplos



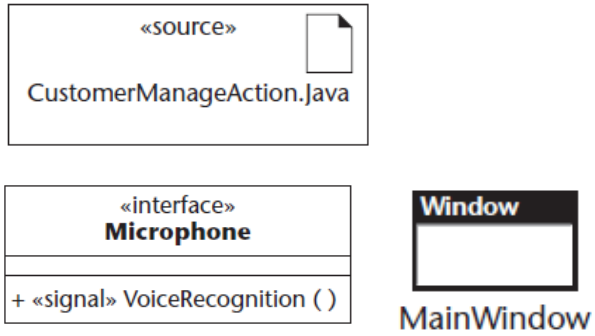


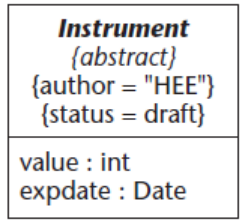
[Eriksson et al., 2004]

Linguagem UML

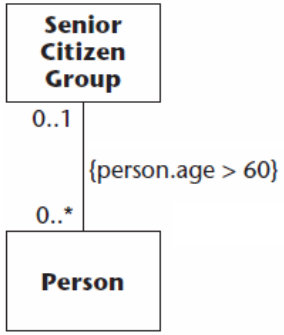
- Mecanismos de Extensão (*Extensibility Mechanisms*)
  - Estereótipos (*Stereotypes*)
    - Permitem estender o vocabulário UML de modo a criar novos elementos de modelação, com características específicas, a partir dos já existentes
    - São utilizados para criar novos elementos de modelação vocacionados para um domínio de representação específico
  - Anotações (*Tagged values*)
    - Propriedades associadas a elementos representadas por pares *palavra_chave – valor*
    - São utilizadas para estender as propriedades dos elementos da linguagem de modo a ser possível a associação de informação específica
  - Restrições (*Constraints*)
    - Propriedades que especificam a semântica e/ou condições que se devem verificar em relação a elementos de um modelo

Notação: «estereótipo»





Notação: {propriedade}

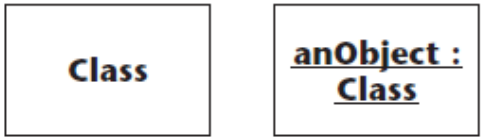


Notação: {restrição}

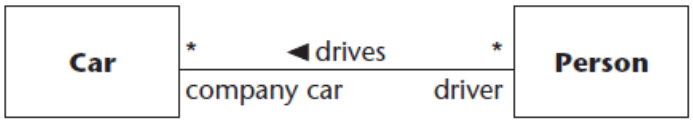
[Eriksson et al., 2004]

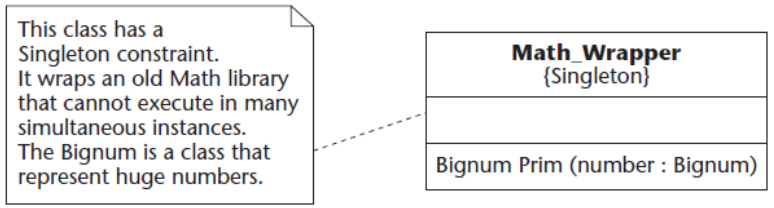
Linguagem UML

- Complementos de Informação
  - Adornos
    - Informação textual ou gráfica indicativa de aspectos específicos de um elemento
  - Comentários
    - Elemento gráfico contendo informação textual



Classe: **Negrito**
Objecto: Sublinhado



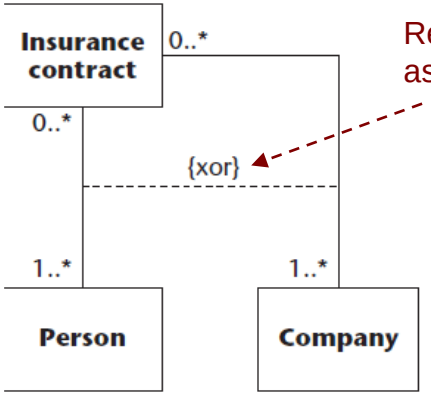


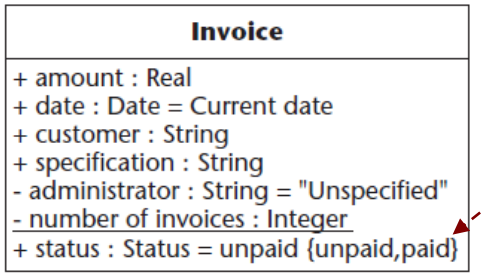
[Eriksson et al., 2004]

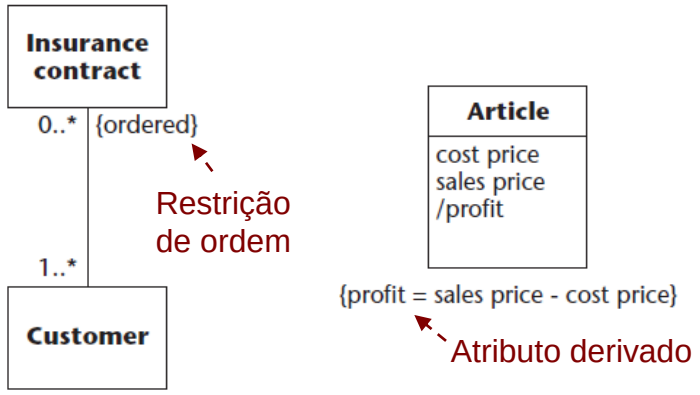
Linguagem UML

- Restrições
  - Descrevem constrições aos elementos de um modelo
  - Múltiplas formas
    - Por exemplo, multiplicidade
    - Pré-condições / Pós-condições
  - Anexação de notas aos elementos do modelo
  - Notação base UML
    - **{** *Especificação da restrição* **}**
    - Notas associadas aos elementos
  - Especificação da restrição
    - Linguagem natural
      - Problema: **Ambiguidade**
    - OCL (*Object Constraint Language*)

Redução de multiplicidade







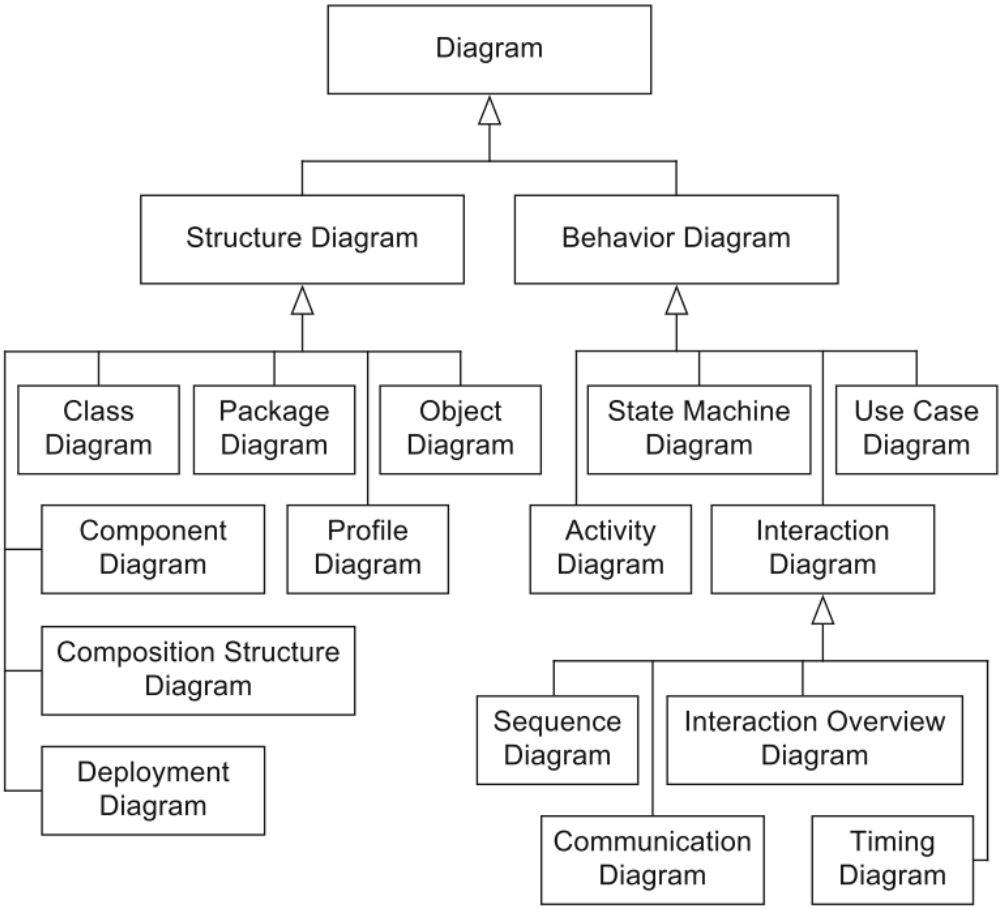
PERSPECTIVAS DE MODELAÇÃO

Diagramas

Na linguagem UML os modelos são organizados em diagramas que permitem representar diferentes perspectivas de modelação de um sistema, organizados em duas perspectivas principais:

- ***Perspectiva estrutural***, referente à organização das partes e relações entre partes que constituem a estrutura de um sistema
- ***Perspectiva comportamental***, referente à organização funcional e dinâmica de um sistema que determina o seu comportamento

Tipos de diagramas



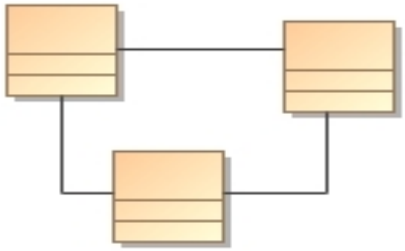
[Seidl, 2012]

Tipos de Diagramas UML

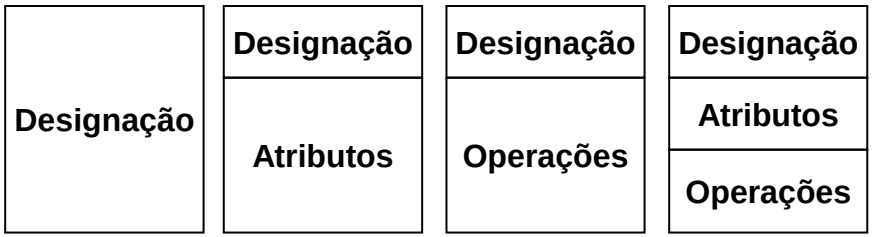
- **Funcionalidade**
  - Diagramas de Casos de Utilização
- **Estrutura**
  - Diagramas de Classes
  - Diagramas de Objectos
  - Diagramas de Estrutura Composta
- **Interacção**
  - Diagramas de Sequência
  - Diagramas de Comunicação
  - Diagramas Temporais (*Timing Diagrams*)
  - Diagramas de Enquadramento de Interacção (*Interaction Overview Diagrams*)
- **Dinâmica**
  - Diagramas de Transição de Estado
  - Diagramas de Actividade
- **Implementação**
  - Diagramas de Componentes
- **Implantação**
  - Diagramas de Implantação
- **Organização do modelo**
  - Diagramas de Pastas (*Package Diagrams*)

Diagramas de Classes

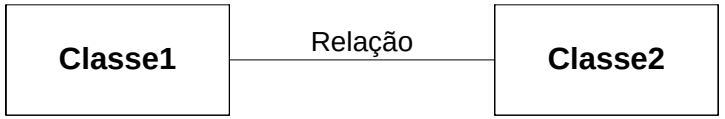
- Descrevem uma abstracção das partes e das relações entre partes de um sistema
  - Organização estática do sistema
  - Foco na estrutura



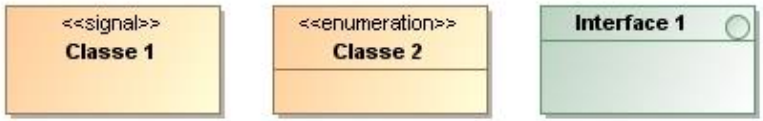
- Classes
  - Atributos
    - Definição de estrutura
  - Operações
    - Encapsulamento de comportamento



- Relações
  - Representação de interdependências

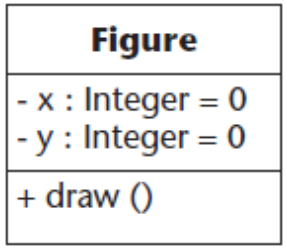


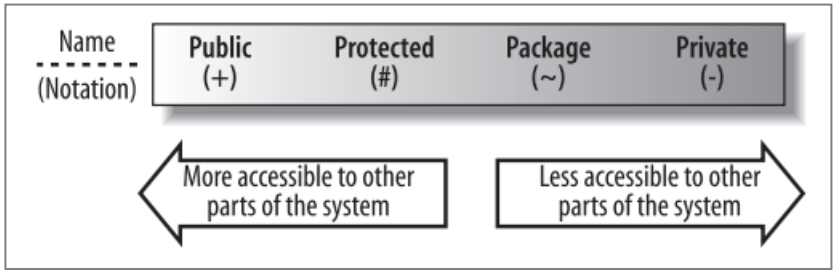
- Estereótipos de classes
  - Criação de novos elementos de modelação, com semântica específica



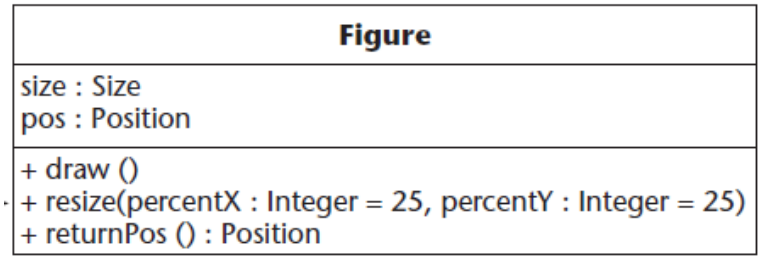
Diagramas de Classes

- Atributos
  - Representação de estrutura
  - Caracterizados por:
    - Designação
    - Tipo
    - Visibilidade
      - Público (+)
      - Privado (-)
      - Protegido (#)
      - Pacote (~)
  - Sintaxe:
    - *visibility name*:*type* = *init_value* {*property_string*}
- Operações
  - Representação de comportamento



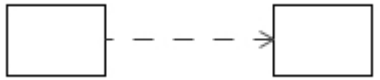


[Miles & Hamilton, 2006]

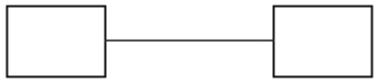


Diagramas de Classes

- Relações entre classes
  - Dependência
    - Relação na qual uma parte utiliza ou depende de outra parte, por exemplo, uma classe tem um método com um parâmetro que é uma instância de outra classe, ou cria localmente uma instância de outra classe

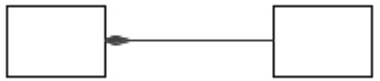


  - Associação
    - Relação na qual uma parte está estruturalmente associada a outra parte através de um dos seus atributos, ou seja, um dos seus atributos tem informação acerca de outra parte, por exemplo, uma classe tem um atributo cujo tipo é uma instância de outra classe

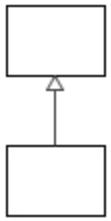


  - Agregação
    - Caso particular de associação que indica que uma parte agrega outras partes, as quais podem existir independentemente da parte agregadora, por exemplo, a relação entre turma e aluno

  - Composição
    - Caso particular de associação que indica que uma parte é composta por outras partes que só existem no contexto da parte composta, por exemplo, a relação entre apartamento e divisão (o apartamento é composto por várias divisões que não existem separadas do todo)



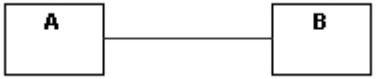
  - Generalização
    - Relação estrutural que indica que uma parte é uma especialização de outra parte mais geral



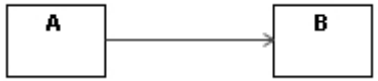
Diagramas de Classes

- Propriedades das relações
  - Direcção
    - Indica a navegabilidade de uma associação, ou seja, que partes contêm informação acerca de outras partes
    - Pode ser *bidirecional* ou *unidirecional*
  - Multiplicidade
    - Indica quantas instâncias de uma parte estão associadas a instâncias de outras partes
  - Papel
    - Indica o papel de uma parte numa associação, ou seja, o significado concreto ou a responsabilidade que lhe corresponde
  - Qualificação
    - O qualificador de uma associação designa uma chave utilizada para obter um item da colecção respectiva

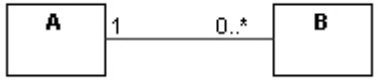
Associação *bidirecional*, as instâncias de ambas as partes conhecem-se reciprocamente



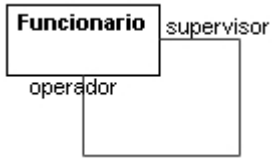
Associação *unidirecional*, apenas as instâncias da parte A conhecem instâncias da parte B



Por cada instância de A existem zero ou mais de B, por cada instância de B existe uma instância de A



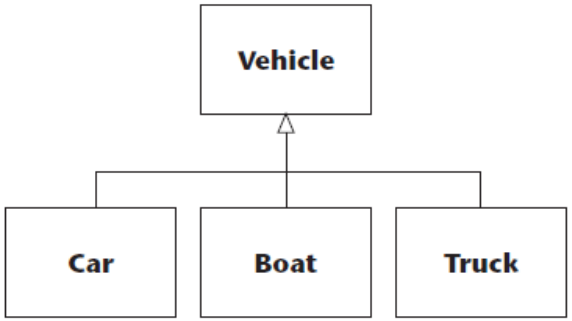
As instâncias de funcionário desempenham os papéis de *operador* ou *supervisor* (cada operador tem associado um supervisor, cada supervisor tem associado um operador



O acesso aos produtos de uma encomenda é realizado através do atributo *codigoProduto*

Diagramas de Classes

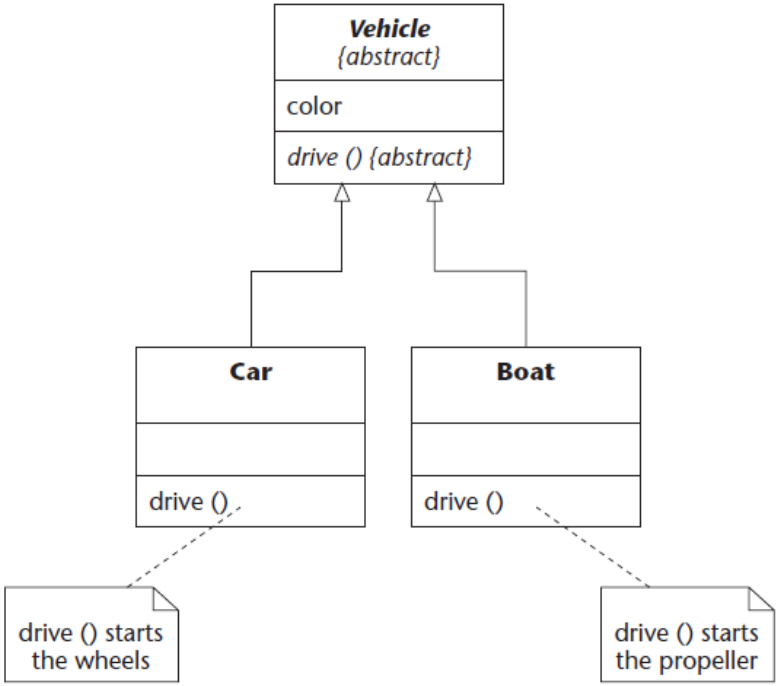
Generalização



[Eriksson et al., 2004]

A *generalização* é uma relação estrutural que indica que uma parte é uma especialização de outra parte mais geral

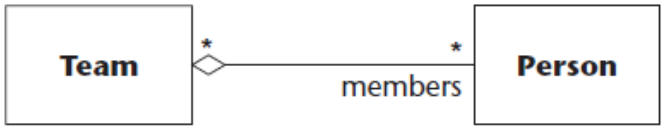
Polimorfismo



O *polimorfismo* é uma característica da modelação e programação orientada a objectos que possibilita que objectos de uma mesma classe assumam formas distintas, nomeadamente expressões comportamentais distintas, por exemplo, operações com diferentes implementações que podem ser utilizadas de forma homogénea independentemente do tipo dos objectos, isso é conseguido com uma definição comum das operações através de uma *classe abstracta* ou de uma *interface*

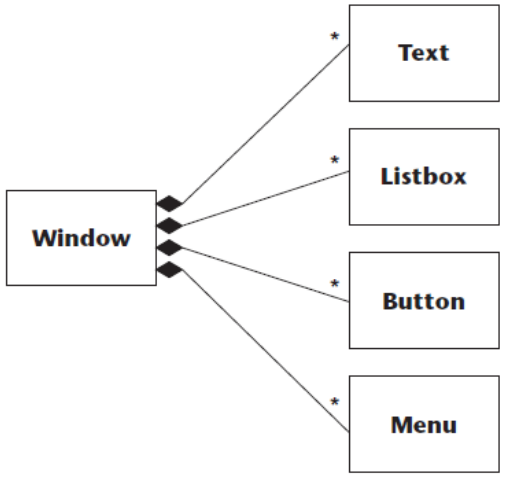
Diagramas de Classes

- **Agregação**
  - Relação Parte - Todo
  - Denota que uma parte contém outra em termos lógicos ou físicos
  - Forma fraca de composição de partes
    - Não define restrições semânticas acerca de:
      - Pertença das partes
      - Criação e destruição das partes
      - Períodos de existência das partes



Relação *parte-todo*

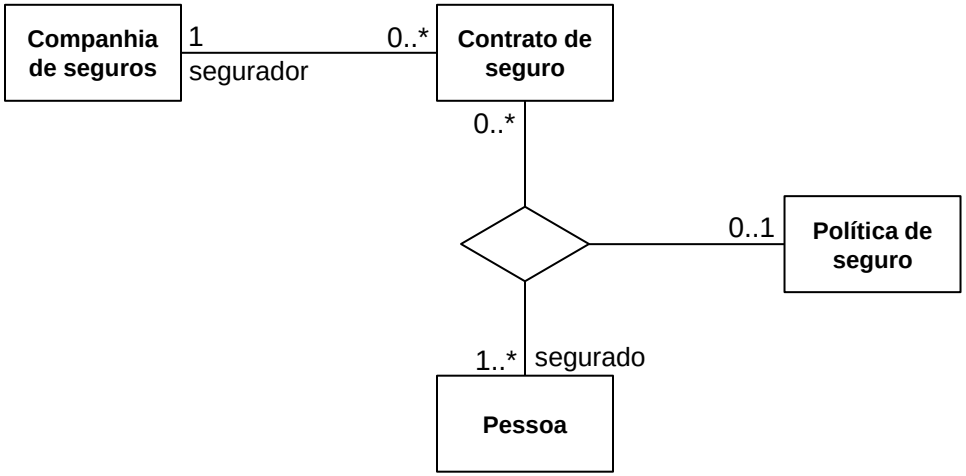
- **Composição**
  - Forma forte de composição de partes
    - Define restrições semânticas específicas
      - O todo cria e destrói as partes
      - Sobreposição de períodos de existência das partes
      - Implica exclusividade na composição das partes
    - Organização hierárquica em árvore
      - Níveis de abstracção



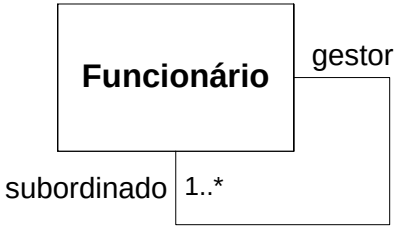
Regra da *não-partilha*

Diagramas de Classes

- Associações ternárias (*n*-árias)
  - A interpretação deve ter em conta as dependências funcionais envolvidas, ou seja, as partes que determinam uma parte específica, por exemplo, "*por cada contrato de seguro e por cada política de seguro existe uma ou mais pessoas seguradas*"

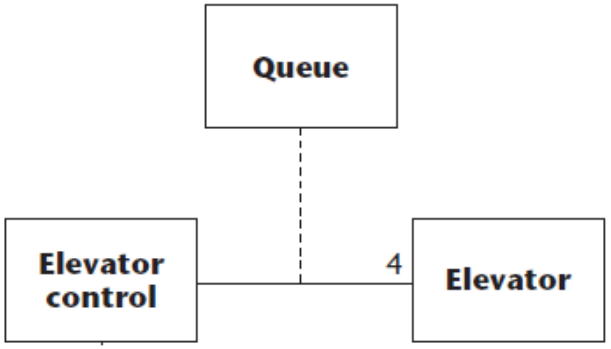


- Associações reflexivas
  - Ocorre quando os objectos de uma classe estão relacionados entre si, podendo desempenhar mais de um papel

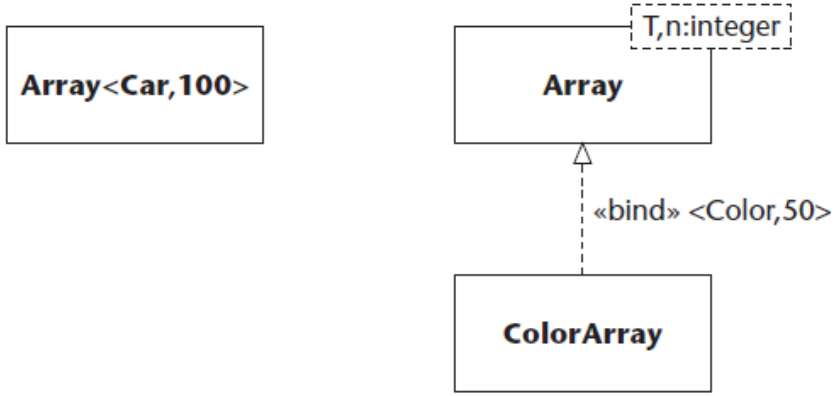


Diagramas de Classes

- Classes associativas
  - Atributos de associações
  - Podem estar associadas a outras classes



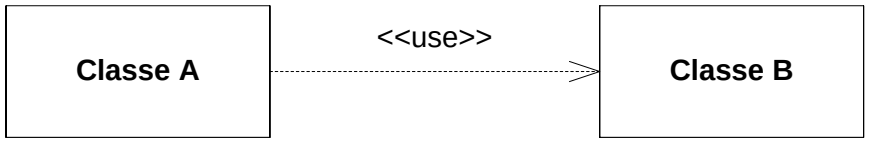
- Classes paramétricas
  - Definição de tipos genéricos (parametrizáveis)



[Eriksson et al., 2004]

Diagramas de Classes

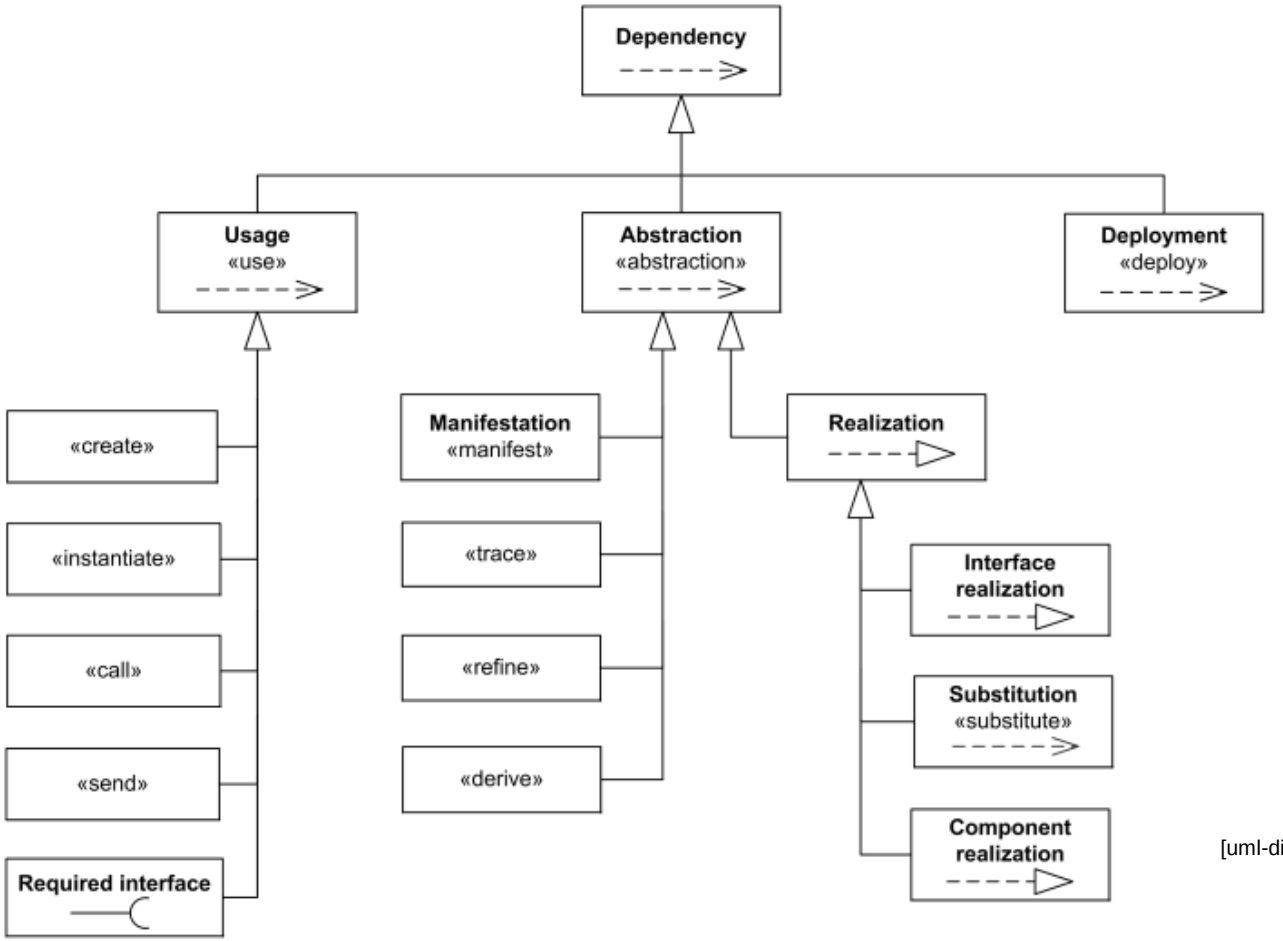
- Dependências
  - Utilização quando uma classe não tem uma relação estrutural com outra classe mas depende dela de alguma forma
  - Alterações numa classe podem implicar alterações noutra classe
  - Rastreabilidade
    - Alterações numa classe podem afectar outras classes
  - Definidas como estereótipos de associações
  - Utilização de estereótipos para representar tipos de dependência



Em sistemas complexos é importante manter registo de dependências

Diagramas de Classes

- Estereótipos de dependências
  - Especializam a relação geral de dependência

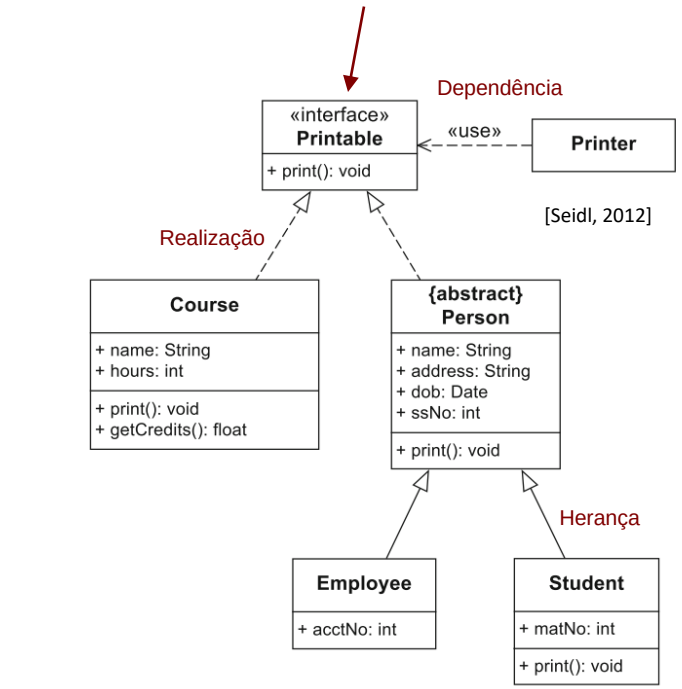


[uml-diagrams.org]

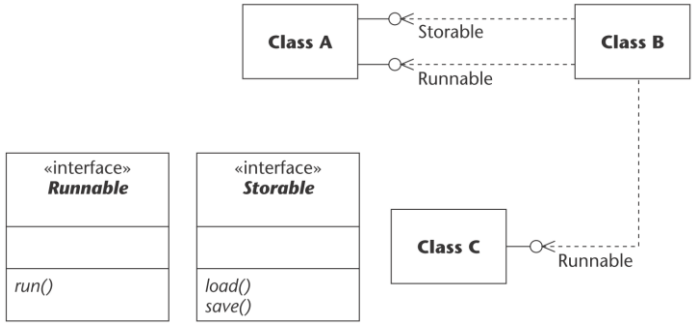
Diagramas de Classes

- Interface
  - Classificador que define as características visíveis de uma parte
  - Conjunto **coeso** de características
  - Define um contrato
    - Não implementa essas características
  - **Encapsulamento**
- Realização
  - Representa implementação
  - Relação entre uma interface e uma classe ou um componente
  - Uma classe ***realiza*** as características de uma interface implementando-as
  - Uma classe pode implementar mais que uma interface

A interface define um contracto de prestação de serviços independente da implementação



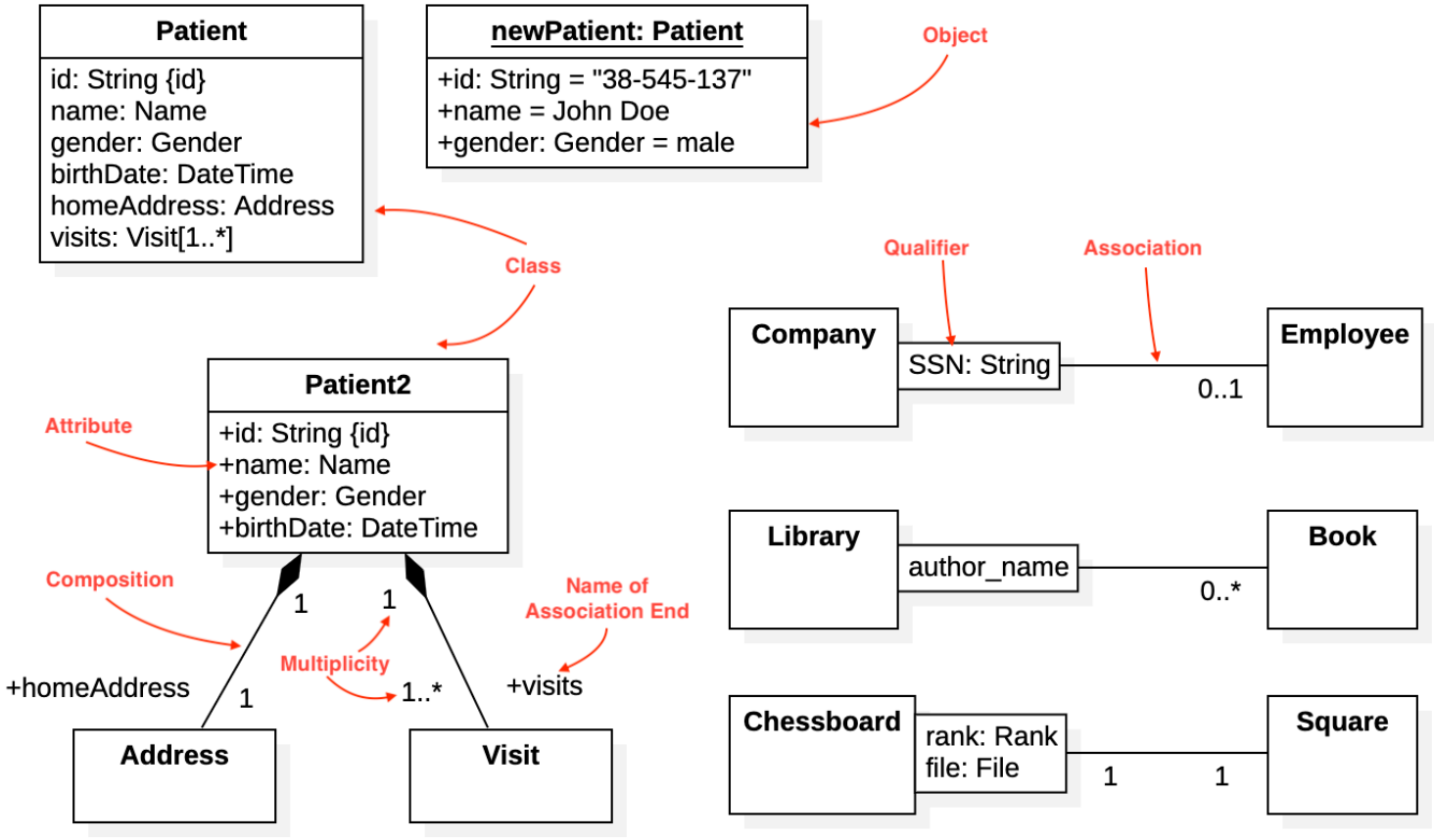
Realização (notação alternativa)



[Eriksson et al., 2004]

Diagramas de Classes

Exemplo



[StarUML]

Diagramas de Objectos

- Descrevem a forma como instâncias de classes se relacionam ou operam em conjunto para realizar a funcionalidade do sistema
  - Elementos são instâncias de classes
  - Também designados diagramas de instâncias
  - Descrevem a concretização da estrutura num **instante específico de tempo**
- Utilização
  - Identificação de objectos e de relações entre objectos
  - Compreensão das formas de colaboração entre objectos para produzir a função do sistema

Diagrama de classes

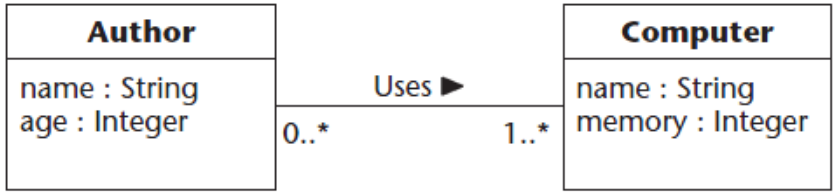
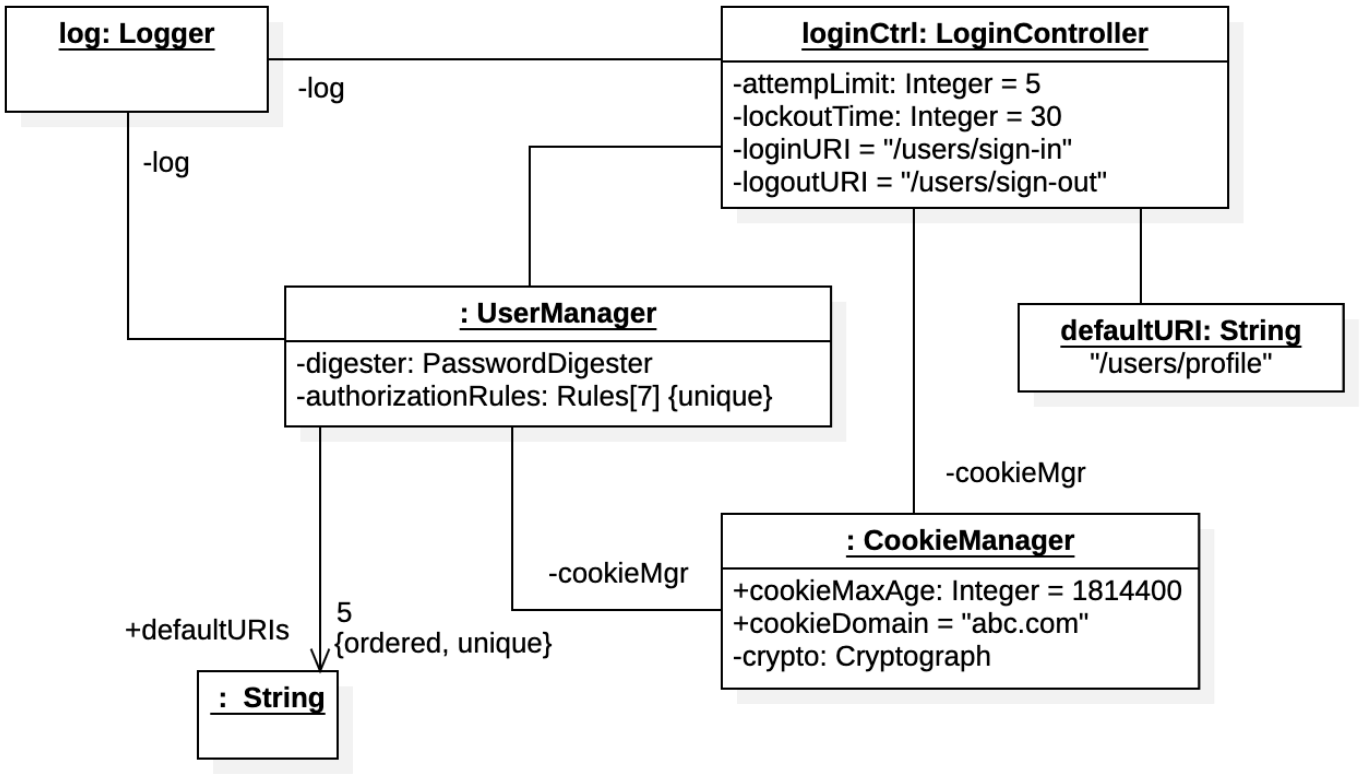


Diagrama de objectos

[Eriksson et al., 2004]

Diagramas de Objectos

Exemplo

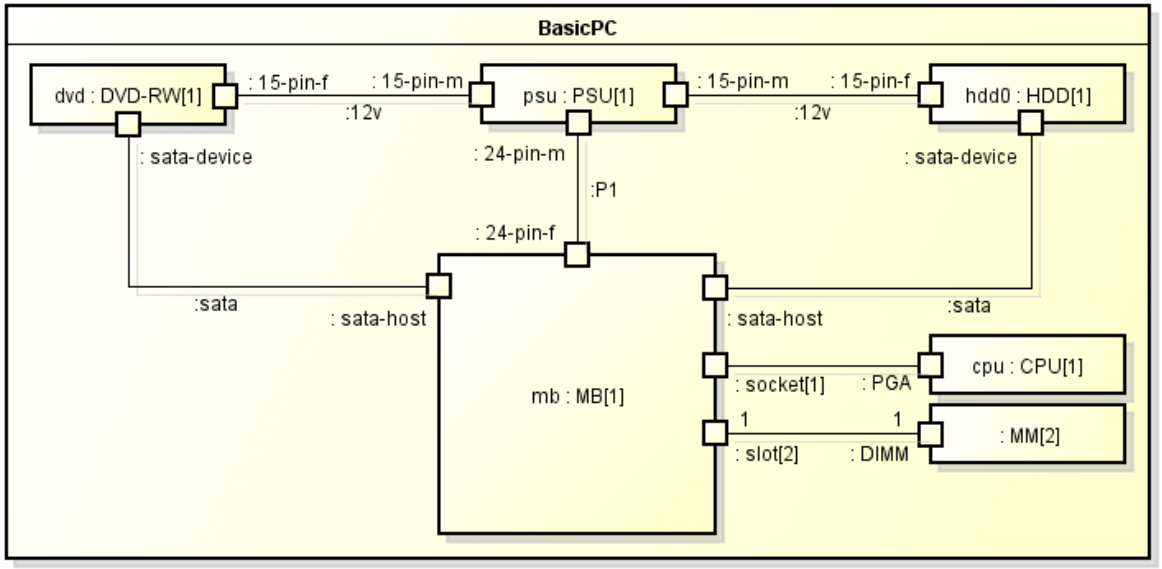


[StarUML]

Diagramas de Estrutura Composta

- Os diagramas de estrutura composta têm por objectivo apresentar uma visão lógica de como as partes de um sistema são compostas
- Descrevem a estrutura interna de classes ou componentes na forma como se relacionam para produzir a funcionalidade respectiva

Exemplo:



Diagramas de Estrutura Composta

- Focagem a diferentes níveis de detalhe
  - Essencial para lidar com a complexidade
  - Abstracção
  - Encapsulamento
- Subsistemas
- Partes
- Pontos de interacção (*port*)
  - Permite representar a ligação com a parte do classificador (classe, componente) que realiza determinada funcionalidade
  - Agrupam interfaces (uma determinada parte do classificador implementa ou utiliza uma interface)
- Interfaces
  - Disponibilizadas
  - Requeridas

Forma alternativa de representar composição

Diagrama de estrutura composta

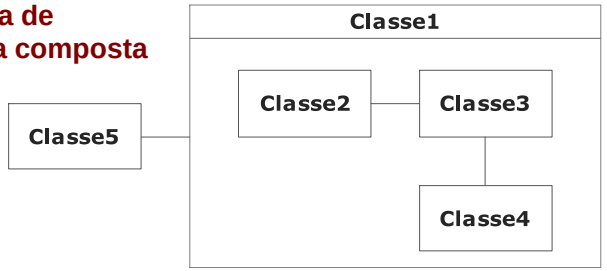
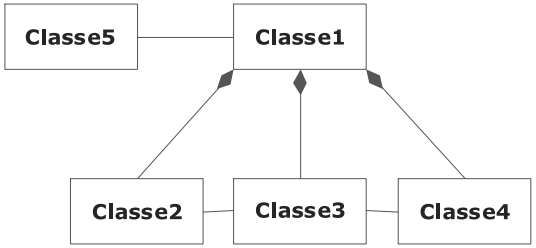
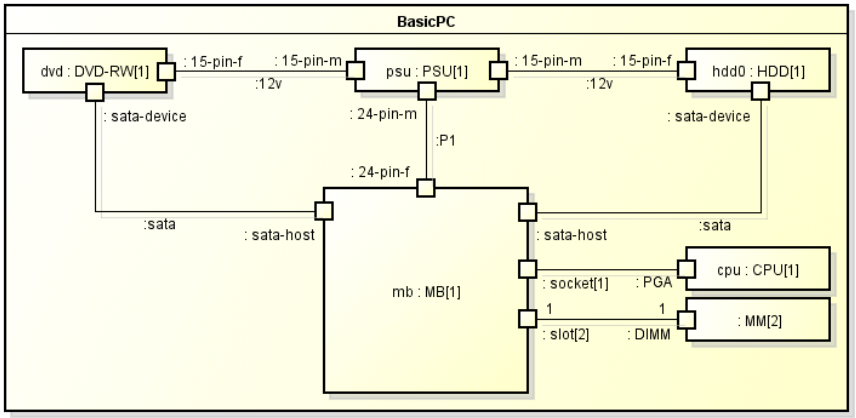


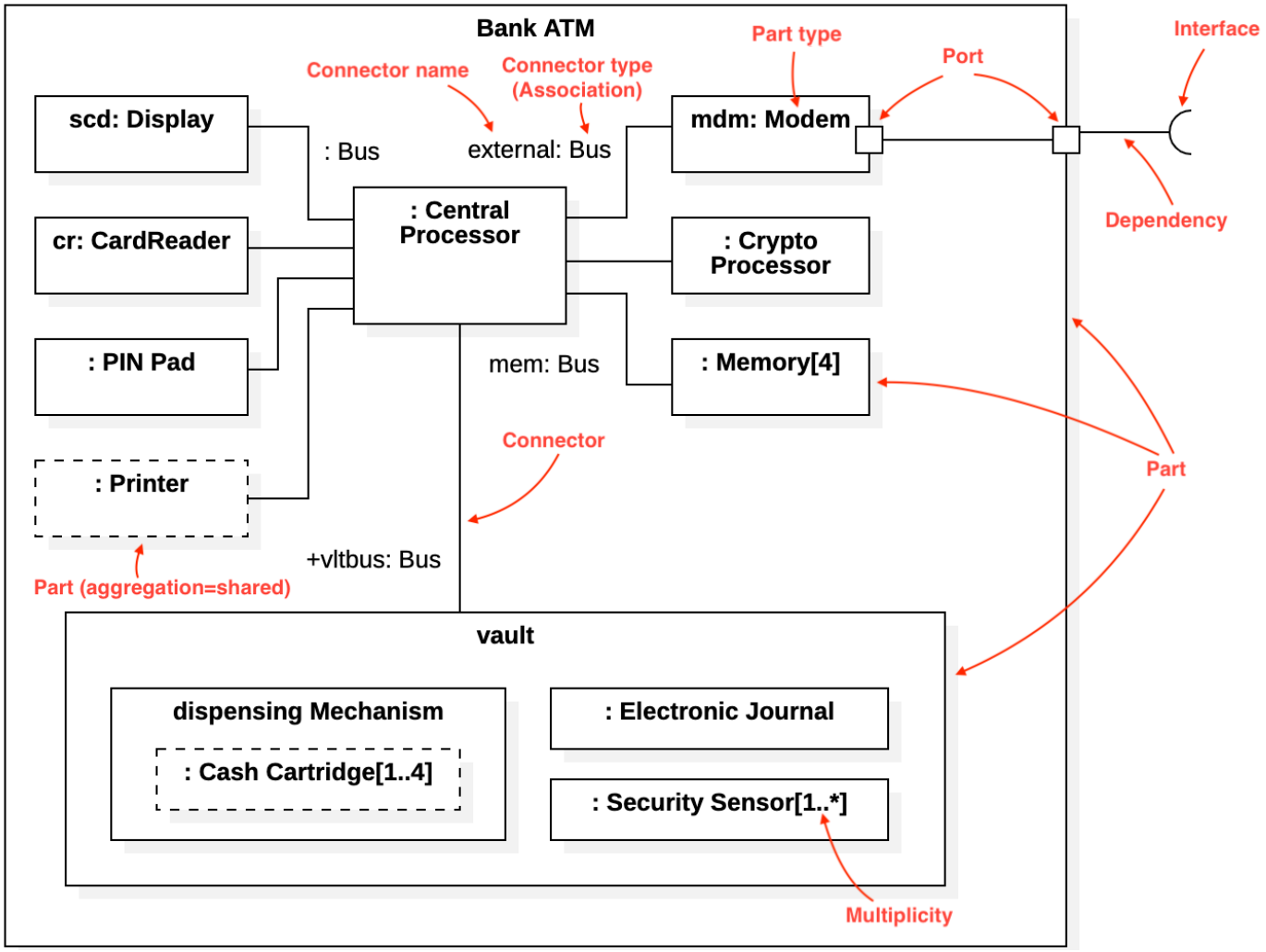
Diagrama de classes





Diagramas de Estrutura Composta

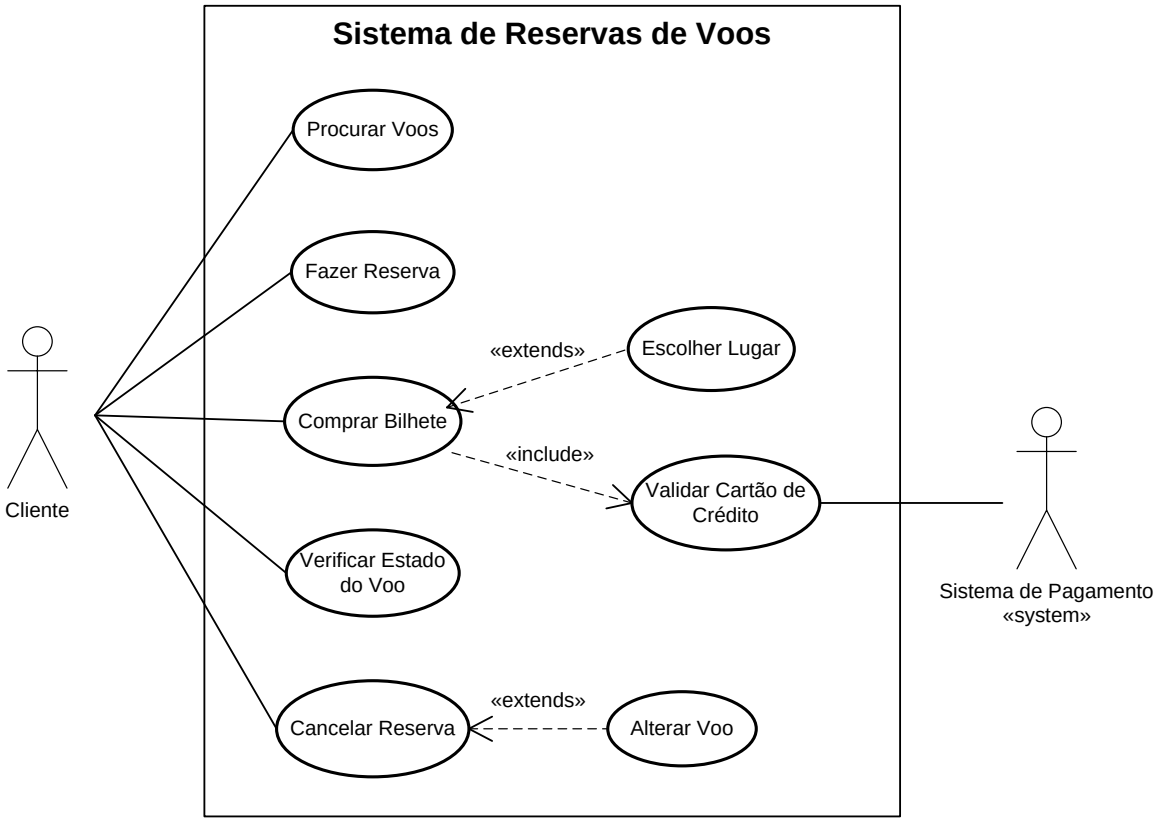
Exemplo



[StarUML]

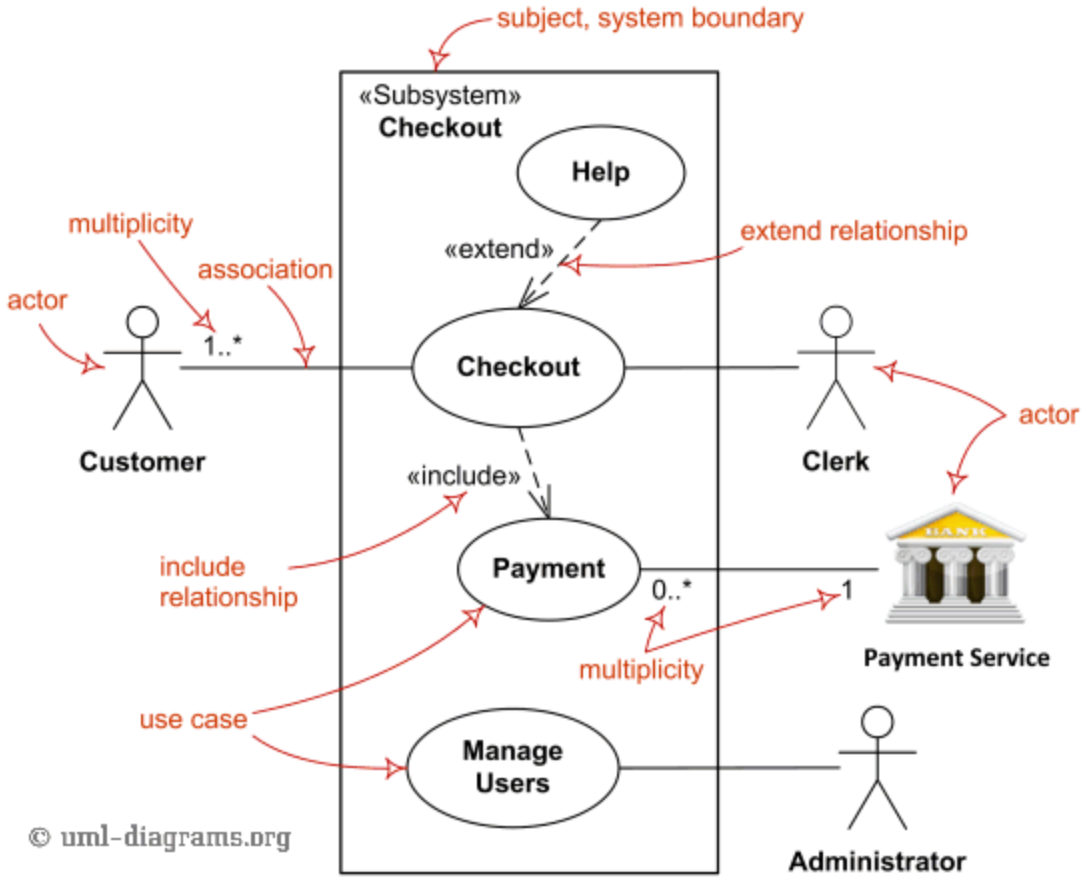
Diagramas de Casos de Utilização

- Descrevem a funcionalidade do sistema na perspectiva dos actores
  - Casos de utilização correspondem a **objectivos** dos **actores**
  - Descritos por cenários de utilização
- Forma de organização dos requisitos funcionais



Diagramas de Casos de Utilização

Elementos principais

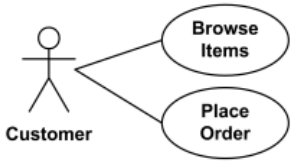


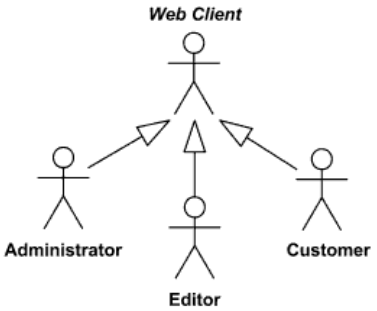
Sistemas externos que participam num caso de utilização são representados como actores (podem ter estereótipos gráficos)

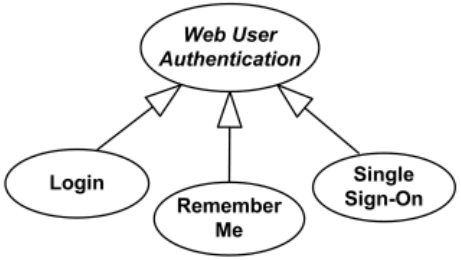
Diagramas de Casos de Utilização

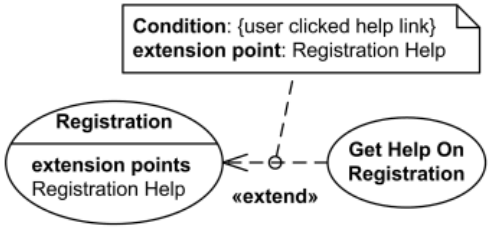
Relações entre cenários e/ou actores

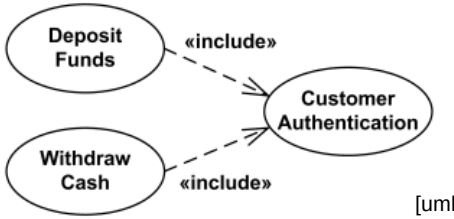
- Participação
- Inclusão
- Extensão
- Generalização/Especialização



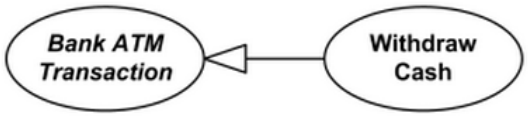
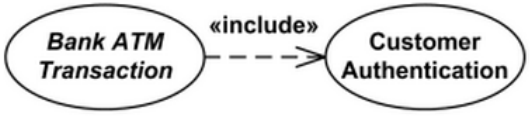






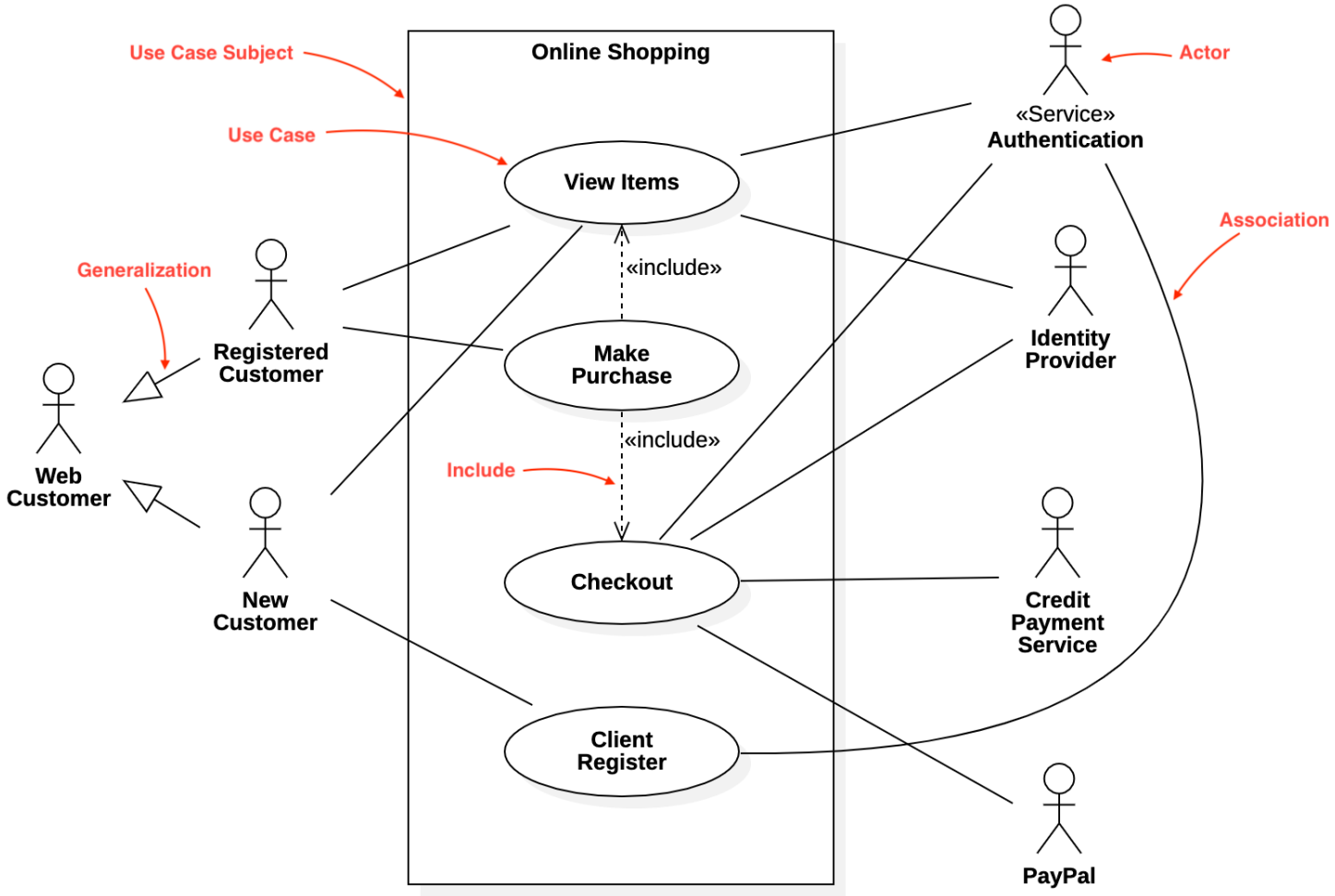


[uml-diagrams.org]

Generalization	Extend	Include
		
Base use case could be **abstract use case** (incomplete) or concrete (complete).	Base use case is complete (concrete) by itself, defined independently.	Base use case is incomplete (**abstract use case**).
Specialized use case is required, not optional, if base use case is abstract.	Extending use case is optional, supplementary.	Included use case required, not optional.
No explicit location to use specialization.	Has at least one explicit extension location.	No explicit inclusion location but is included at some location.
No explicit condition to use specialization.	Could have optional extension condition.	No explicit inclusion condition.

Diagramas de Estrutura Composta

Exemplo



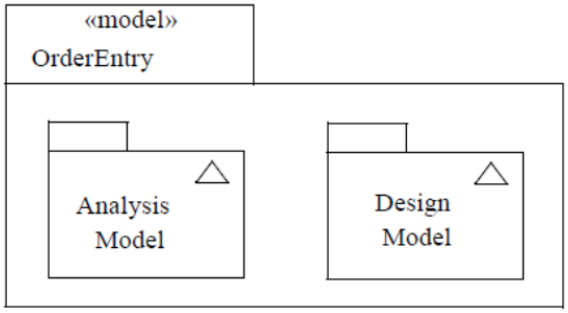
[StarUML]

Diagramas de Pastas (*Packages*)

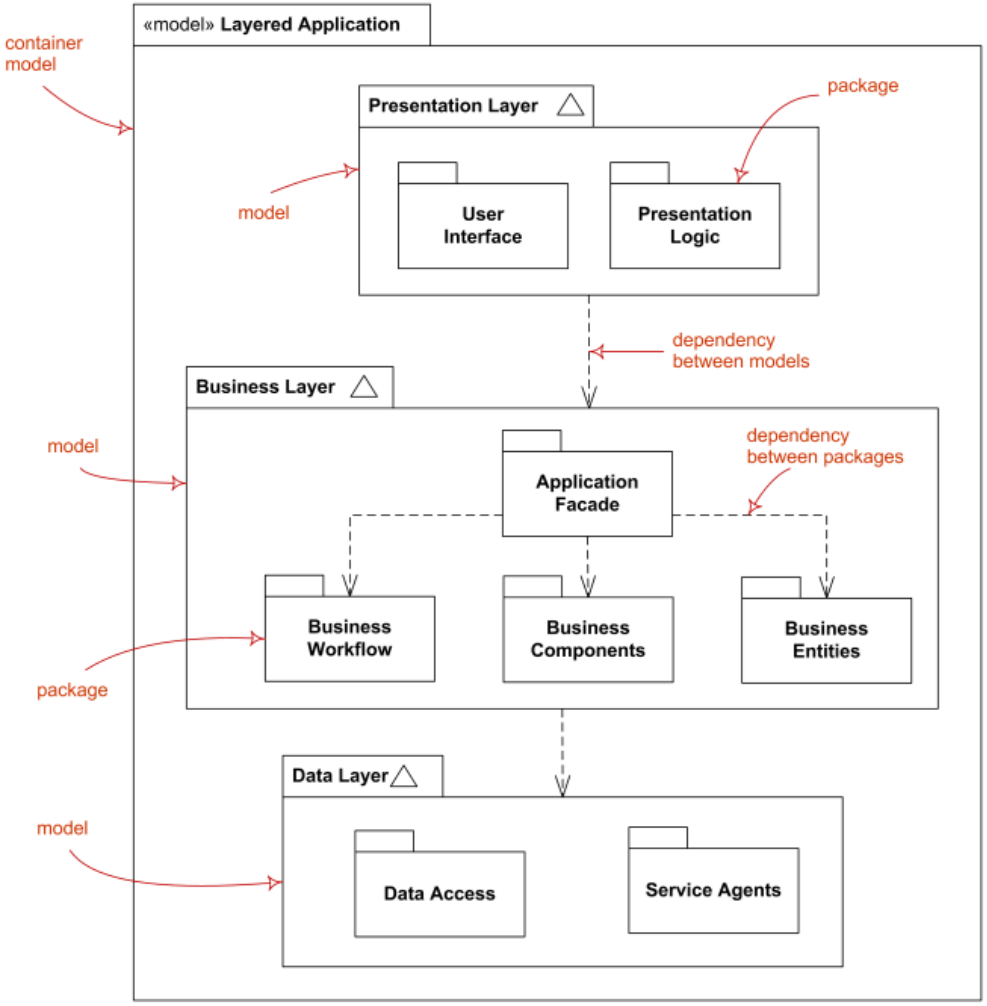
Organização estruturada dos elementos do modelo

- Gestão e organização de sistemas complexos
- Foco nas dependências
- Divisão de um sistema em subsistemas

Exemplo:
Modelo com duas perspectivas de arquitectura organizadas em pastas específicas



[OMG, 2020]

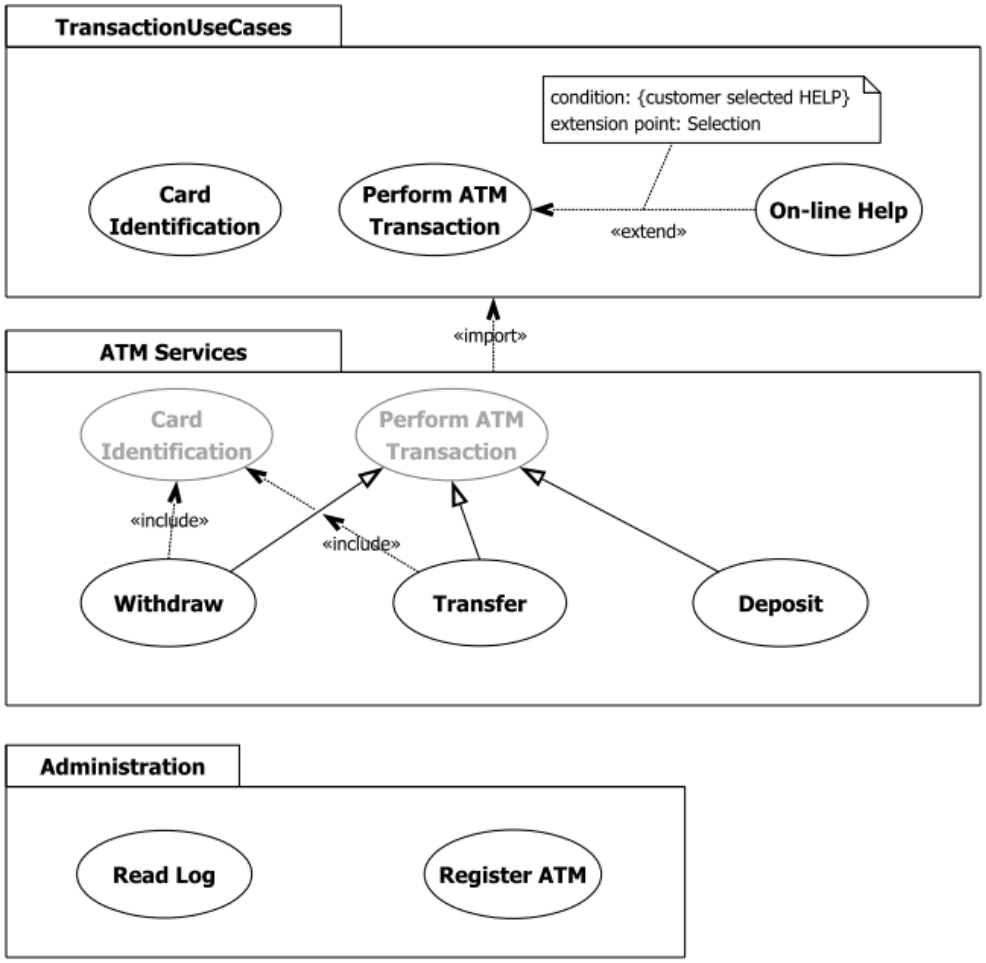


[uml-diagrams.org]

Diagramas de Pastas (*Packages*)

Exemplo: Organização de Casos de Utilização

- Utilização de pastas (*packages*) para organização de casos de utilização



[OMG, 2020]

Bibliografia

[Watson, 2008]
Andrew Watson, *Visual Modeling: past, present and future*, OMG, 2008.

[Meyer, 1997]
B. Meyer, *UML: The Positive Spin*, American Programmer - Special UML issue, 1997.

[Yelland et al., 2002]
Yelland, M. J., B. I. Moat, R. W. Pascal and D. I. Berry, *CFD model estimates of the airflow over research ships and the impact on momentum flux measurements*, Journal of Atmospheric and Oceanic Technology, 19(10), 2002.

[Selic, 2003]
B. Selic, *Brass bubbles: An overview of UML 2.0*, Object Technology Slovakia, 2003.

[Graessle, 2005]
P. Graessle, H. Baumann, P. Baumann, *UML 2.0 in Action*, Packt Publishing, 2005.

[Eriksson et al., 2004]
H. Eriksson, M. Penker, B. Lyons, D. Fado, *UML 2 Toolkit*, Wiley, 2004.

[USDT, 2005]
U.S. Department of Transportation, *Clarus: Concept of Operations*, Publication No. FHWA-JPO-05-072, 2005.

[Douglass, 2006]
B. Douglass, *Real-Time UML*, Telelogic, 2006.

[OMG, 2020]
Unified Modeling Language (Specification), OMG, 2020.