UML — UNIFIED MODELING LANGUAGE

ACH 2003 — COMPUTAÇÃO ORIENTADA A OBJETOS

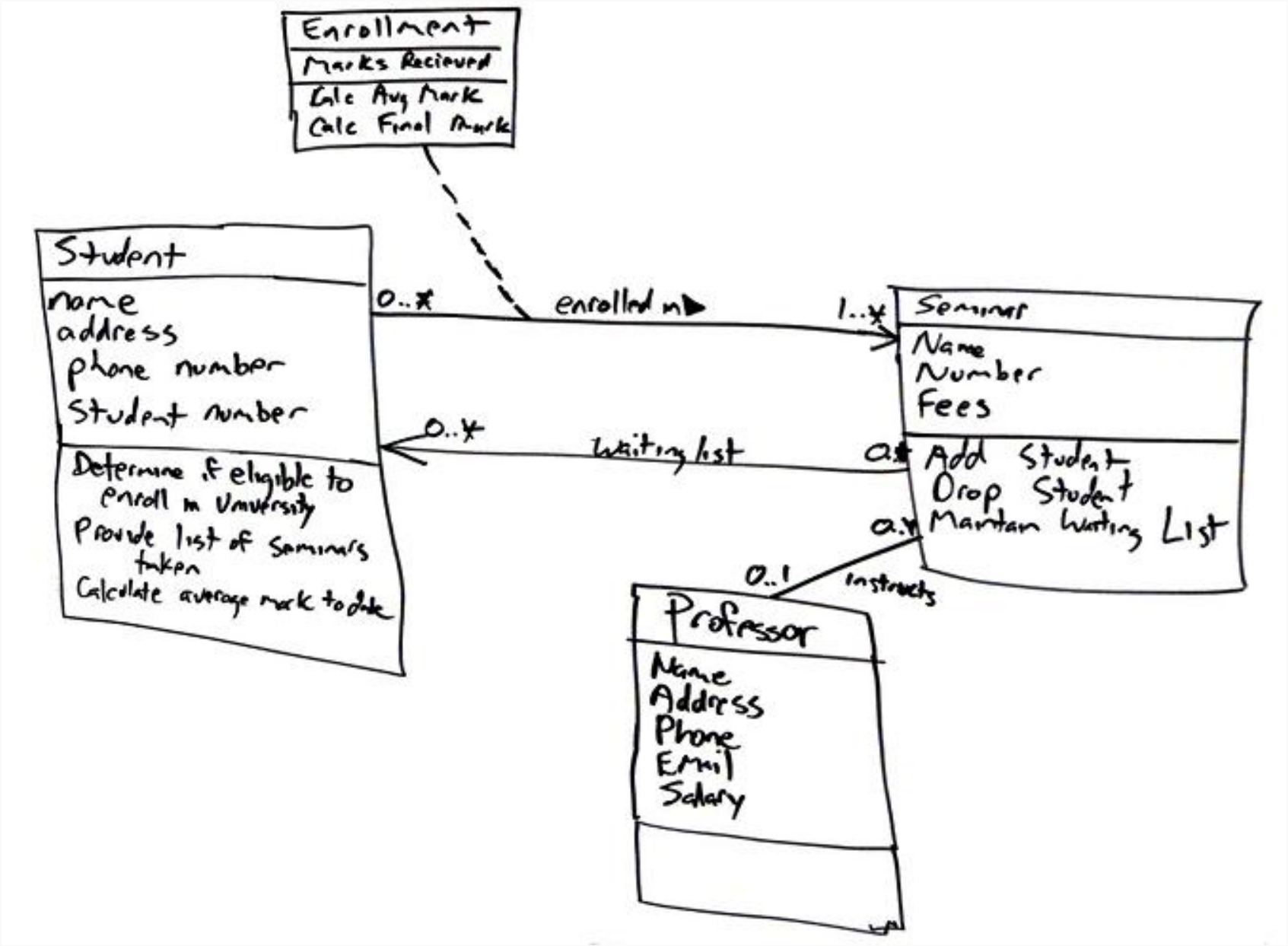
Daniel Cordeiro[a]

May the 4th be with you 

Escola de Artes, Ciências e Humanidades | EACH | USP

[a] Baseado nos slides de Laëtitia Matignon, Université Claude Bernard Lyon 1

EXEMPLO DE UML



O QUE É UML?

- Linguagem Unificada de Modelagem (*Unified Modeling Language*)
- Padrão definido pelo *Object Management Group* (OMG)

Objetivos segundo o OMG

O objetivo de UML é fornecer a arquitetos, desenvolvedores e engenheiros de software ferramentas (visuais) para a análise, projeto e implementação de sistemas de software, bem como para a modelagem de negócios e de processos.

O QUE É UML?

- Linguagem Unificada de Modelagem (*Unified Modeling Language*)
- Padrão definido pelo *Object Management Group* (OMG)

Diferença entre Linguagem e Método

- Linguagem de modelagem: notações, gramática, semântica, etc.
- Método: como utilizar uma linguagem de modelagem (análise de requisitos, concepção, desenvolvimento do software, validação, etc.)

UML é uma linguagem universal **para a modelagem de objetos**.

POR QUE MODELAR?

- A descrição de um programa orientado a objetos requer um trabalho conceitual: definição das classes, de como elas se relacionam, de seus atributos e operações (implementadas pelos seus métodos), das interfaces, etc.
- Precisamos de algo para ajudar a organizar as ideias, documentá-las e organizar o desenvolvimento
- A modelagem é uma etapa que precede a implementação

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)
- UML é uma linguagem de modelagem de aplicações orientadas a objetos

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)
- UML é uma linguagem de modelagem de aplicações orientadas a objetos
- UML não é uma linguagem de programação

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)
- UML é uma linguagem de modelagem de aplicações orientadas a objetos
- UML não é uma linguagem de programação
- UML não é um processo de desenvolvimento

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)
- UML é uma linguagem de modelagem de aplicações orientadas a objetos
- UML não é uma linguagem de programação
- UML não é um processo de desenvolvimento
- UML é independente de linguagem de programação

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)
- UML é uma linguagem de modelagem de aplicações orientadas a objetos
- UML não é uma linguagem de programação
- UML não é um processo de desenvolvimento
- UML é independente de linguagem de programação
- UML é um padrão mantido pela OMG

O QUE É UML?

- UML é uma linguagem universal de modelagem de objetos
- UML é uma notação, uma ferramenta de comunicação visual (diagramas)
- UML é uma linguagem de modelagem de aplicações orientadas a objetos
- UML não é uma linguagem de programação
- UML não é um processo de desenvolvimento
- UML é independente de linguagem de programação
- UML é um padrão mantido pela OMG
- Descrição exata: `http://www.omg.org/uml`

POR QUE UNIVERSAL?

- fim dos anos 80 / começo dos 90: tendência em direção a métodos de análise OO
- até metade dos anos 90 vários métodos foram propostos (> 50):
  - **Booch**'91 e Booch'93, linguagem de modelagem OO criada por Grady Booch (pioneiro no uso de técnicas orientadas a objetos). Distingue dois níveis:

Lógico	diagramas de classes, de instâncias, de estados/transições
Físico	diagramas de módulos, de processos

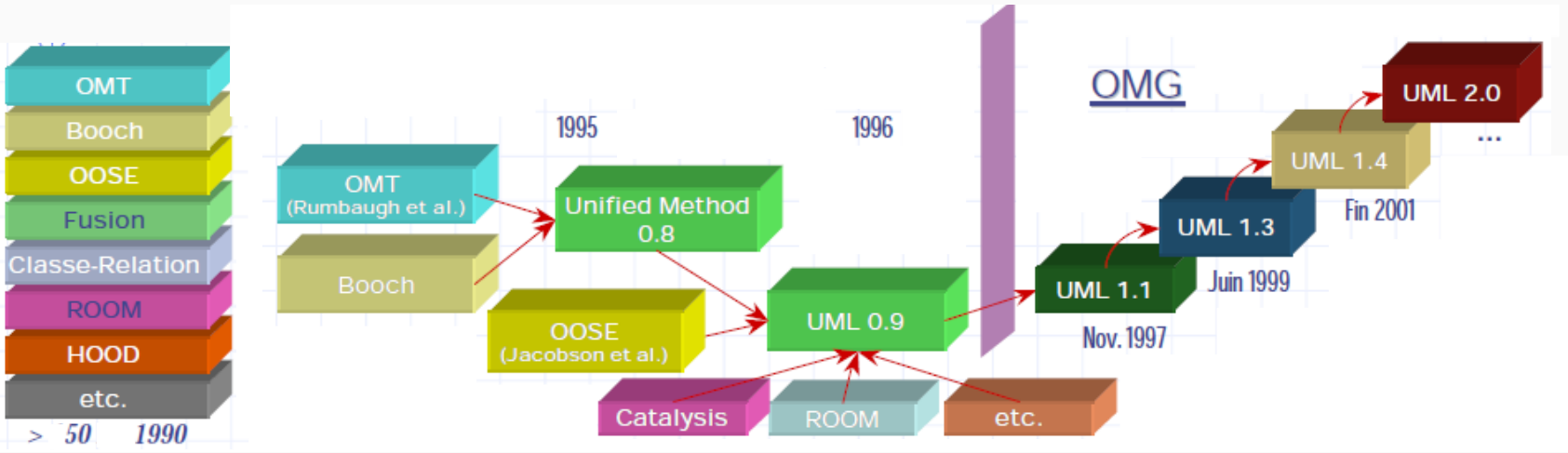
  - **Object Modeling Technique** (OMT), criada por James Rumbaugh em 91 e 93, define três tipos de modelos: estáticos (ou de objeto), dinâmicos e funcionais
  - **Object-Oriented Software Engineering** (OOSE) de Ivar Jacobson em 92. Introduziu a noção de casos de uso

NECESSIDADE DE UNIFICAÇÃO

- Conceitos OO próximos, mas notações gráficas diferentes
- essa “guerra de metodologias” atrapalhava o avanço das tecnologias orientadas a objetos
- A indústria tinha necessidade de um padrão
- Busca por uma linguagem comum única:
  - utilizável em todos os métodos
  - adaptado a todas as fases do desenvolvimento
  - compatível com todas as técnicas de desenvolvimento

UNIFICAÇÃO

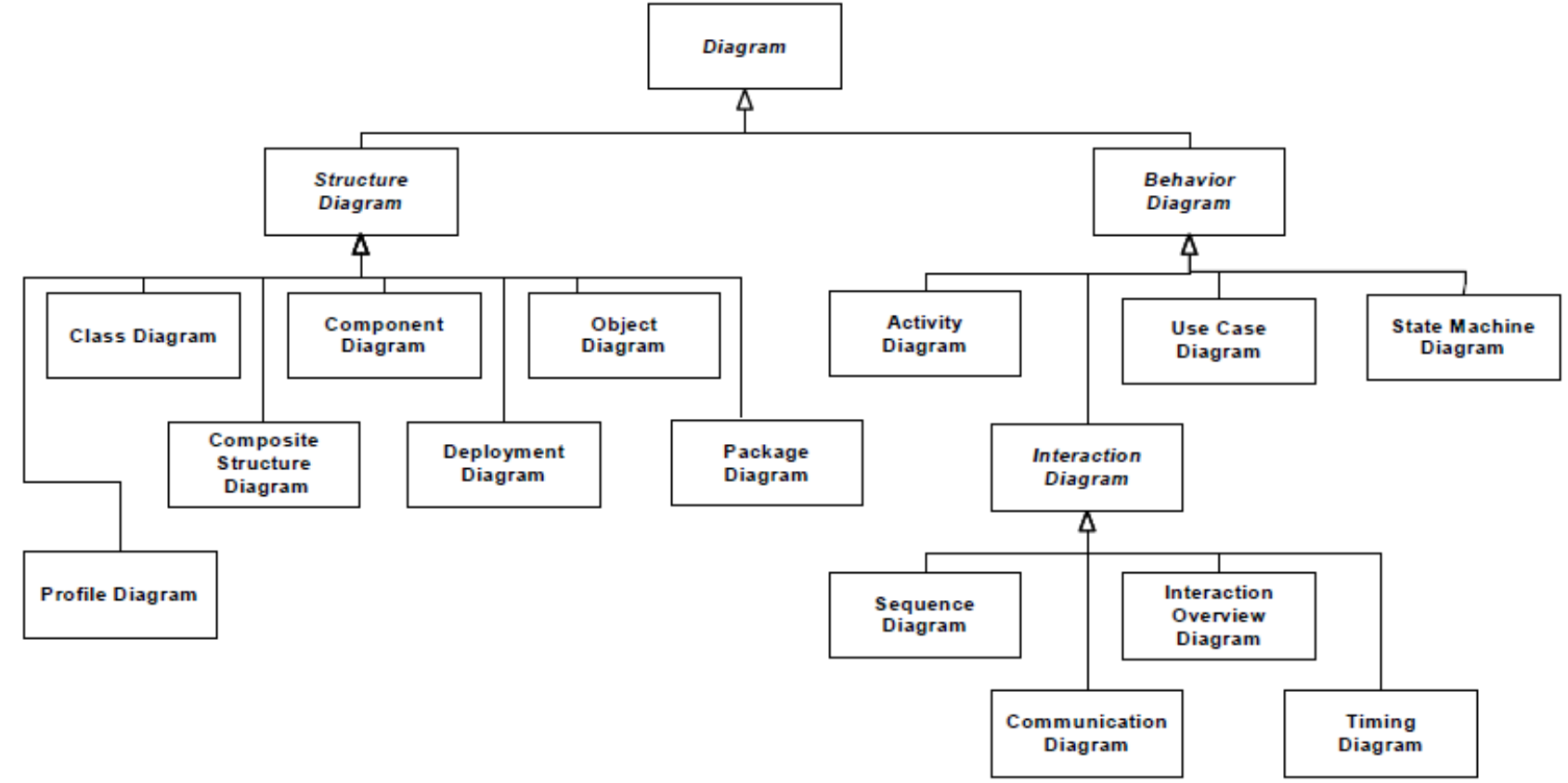
- Outubro de 1995: unificação de 2 métodos (Método Unificado v0.8)
- Fim de 1995: Jacobson (OOSE) se junta a eles (UML 0.9)
- Criação de um consórcio (IBM, Microsoft, Oracle, ...)
- Janeiro de 1997: submissão ao OMG da versão UML 1.0
- Novembro de 1997: UML 1.1 adotado pela OMG como um padrão oficial
- Versão 2.5 em março de 2015



Diagramas elementos gráficos que representam o problema de acordo com a definição de seus pontos de vista

Pontos de vista descrevem um ponto de vista do sistema

Modelos dos elementos elementos básicos dos diagramas

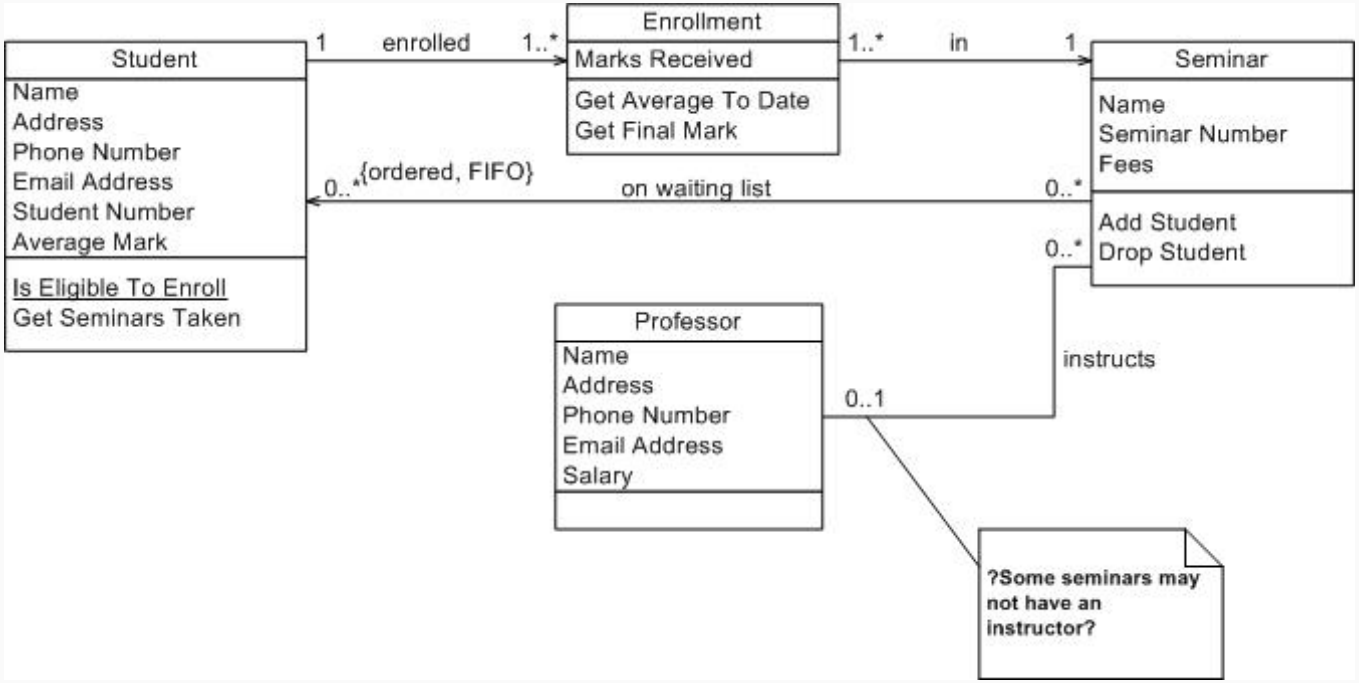


DIAGRAMAS ESTRUTURAIS

Usados para visualizar, especificar, construir e documentar aspectos estáticos de um sistema.

- diagrama de classes
- diagrama de pacotes
- diagrama de objetos
- diagrama de componentes
- diagrama de implantação

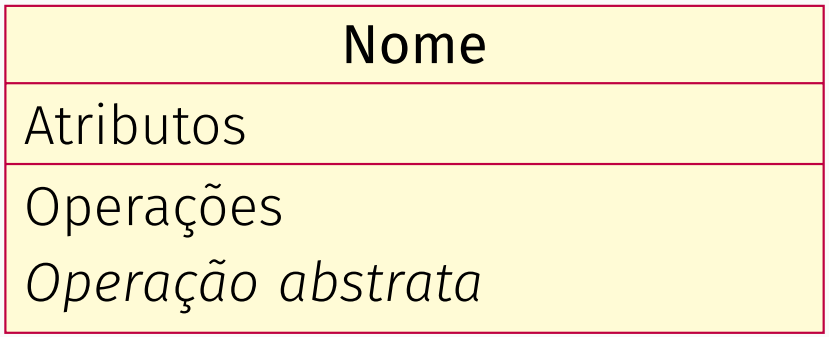
DIAGRAMA DE CLASSES



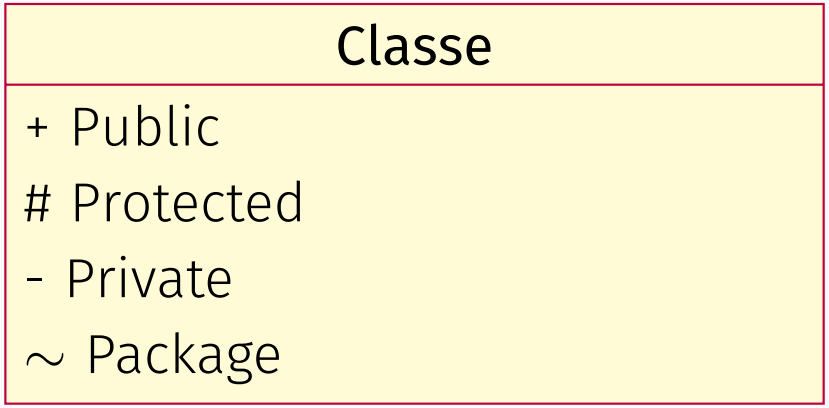
Usos comuns

- Modelar o vocabulário do sistema, em termos de quais abstrações fazem parte do sistema e quais caem fora de seus domínios
- Modelar as colaborações/interações (sociedades de elementos que trabalham em conjunto oferecendo algum comportamento cooperativo)
- Modelagem lógica dos dados manipulados pelo sistema (servindo de base para a definição formal do modelo da base de dados)

NOTAÇÃO PARA CLASSES



ESPECIFICAÇÃO DE ACESSO



ESPECIFICAÇÃO DE ACESSO

Classe
+ Public # Protected - Private ~ Package

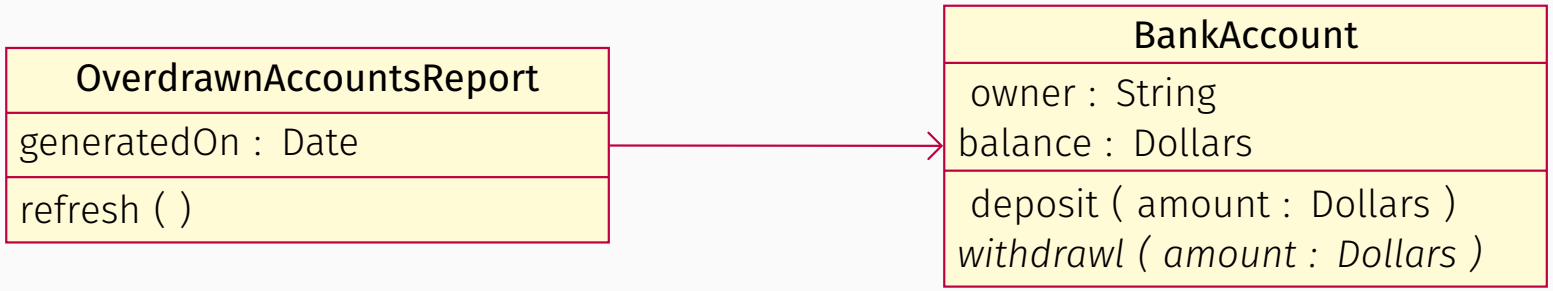
BankAccount
+ owner : String + balance : Dollars
+ deposit (amount : Dollars) + withdrawal (amount : Dollars) # updateBalance (newBalance : Dollars)

São conexões entre classes:

1. dependência
2. generalização
3. associação

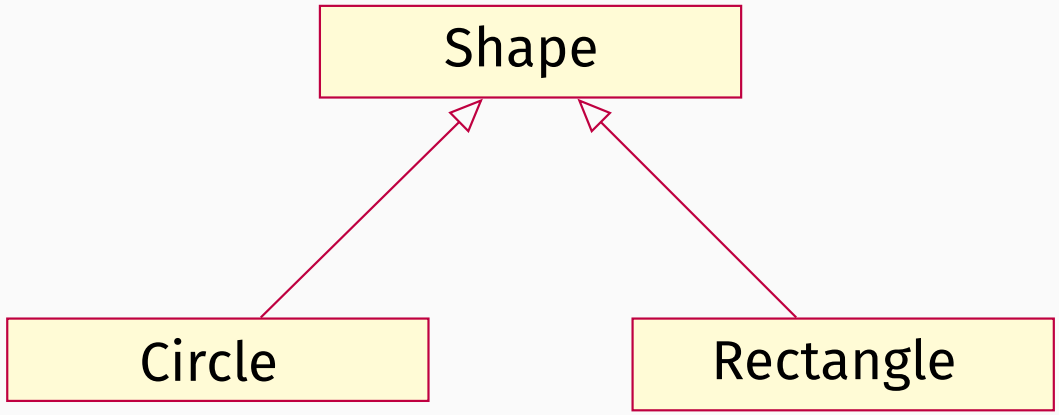
É uma relação do tipo “usa” na qual mudanças na implementação de uma classe podem causar efeitos em outra classe que a usa.

Exemplo: uma classe usa a outra



GENERALIZAÇÃO

É uma relação do tipo “é um” entre uma coisa geral (superclasse) e uma coisa mais específica (subclasse)



ASSOCIAÇÃO

É uma relação estrutural na qual classes ou objetos estão interconectados.

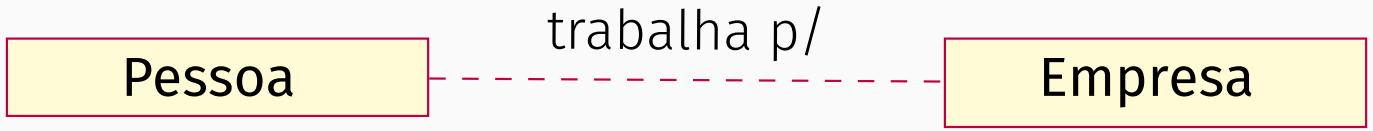
Uma associação entre objetos é chamada de ligação (*link*)

ORNAMENTOS PARA ASSOCIAÇÕES

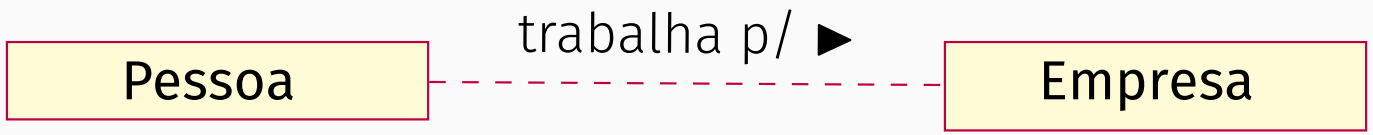
- nome
- papel
- multiplicidade
- agregação
- composição

NOME DA ASSOCIAÇÃO

descreve a natureza da relação:



pode indicar a direção:



PAPÉIS

Classes e objetos podem assumir papéis diferentes em diferentes momentos.

MULTIPLICIDADE

Valores possíveis: valor exato, intervalo ou * para “muitos”

Exemplo:

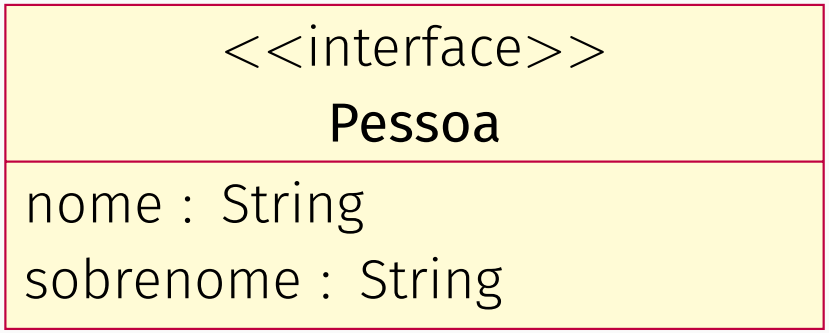
AGREGAÇÃO

É uma relação do tipo “todo/parte” ou “possui um” na qual uma classe representa uma coisa grande que é composta por coisas menores. Indicada por um diamante vazio.

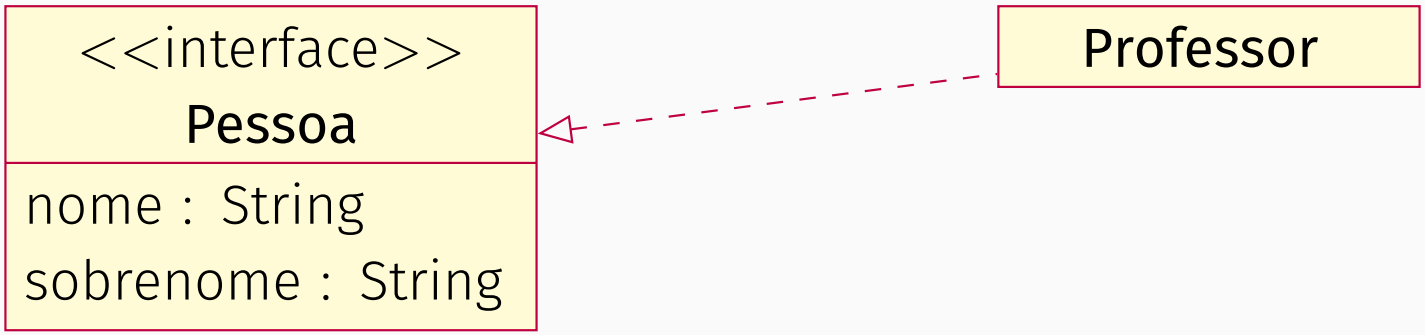
COMPOSIÇÃO

É um tipo especial de agregação na qual as partes são inseparáveis do todo. Indicada por um diamante cheio.

É uma coleção de operações que possui um nome. É usada para especificar um tipo de serviço sem ditar a sua implementação. Também podem participar de generalizações, associações e dependências.

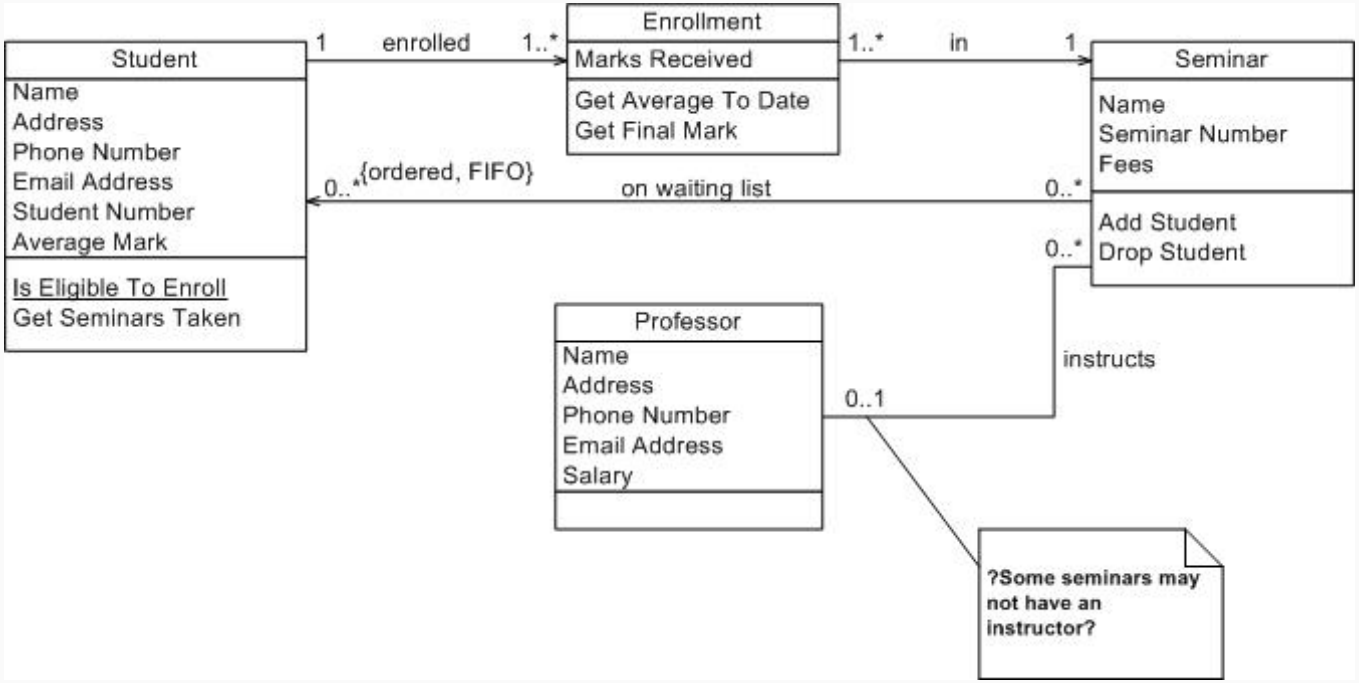


É uma relação entre uma interface e a classe que a implementa, i.e., que provê o serviço definido pela interface.



Uma classe pode realizar (implementar) várias interfaces.

DIAGRAMA DE CLASSES



PACOTES

- Um mecanismo para organizar elementos de um modelo (classes, diagramas, etc.) em grupos
- Cada elemento de um modelo pertence a um único pacote. O seu nome dentro do pacote deve ser único

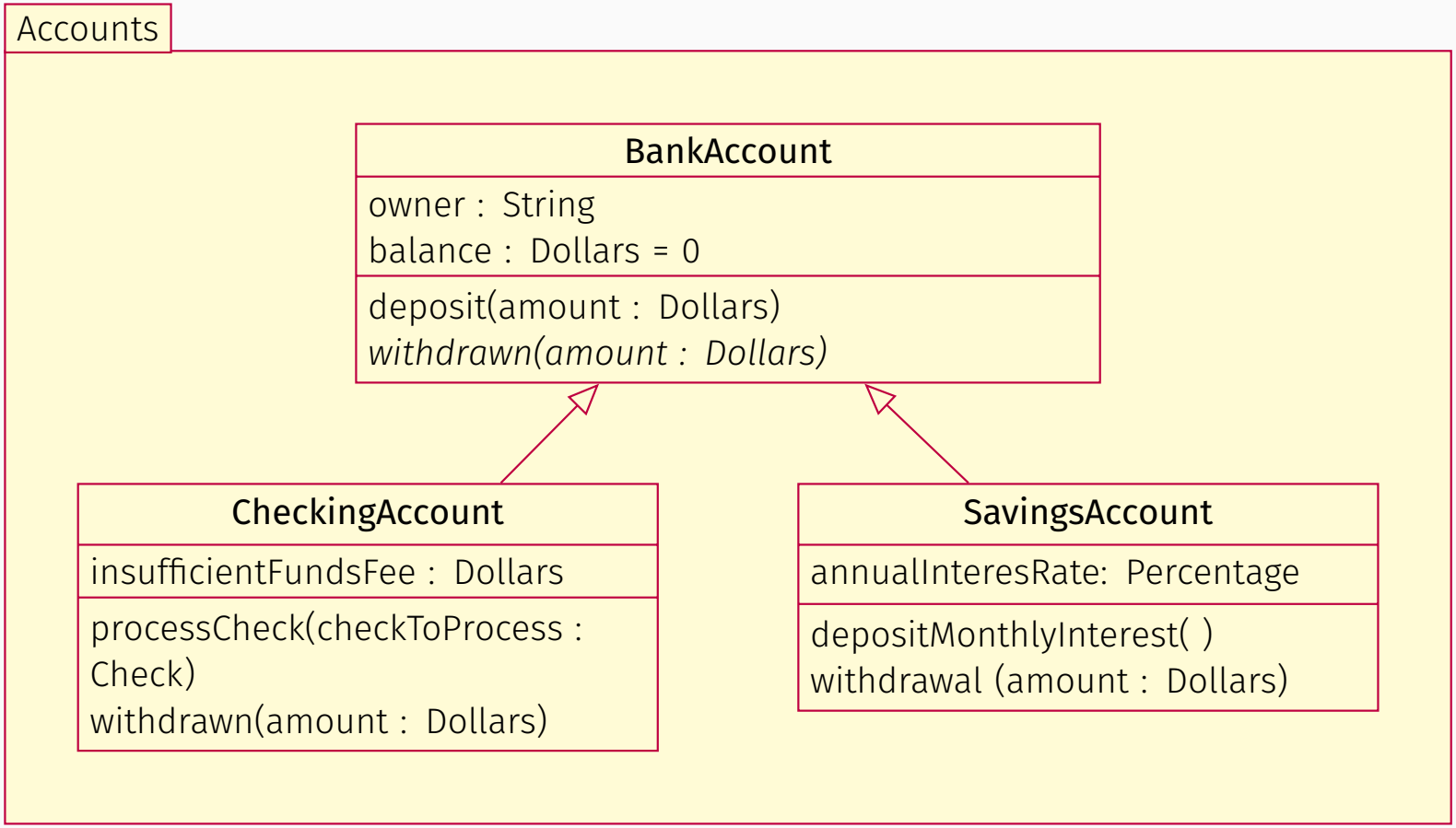
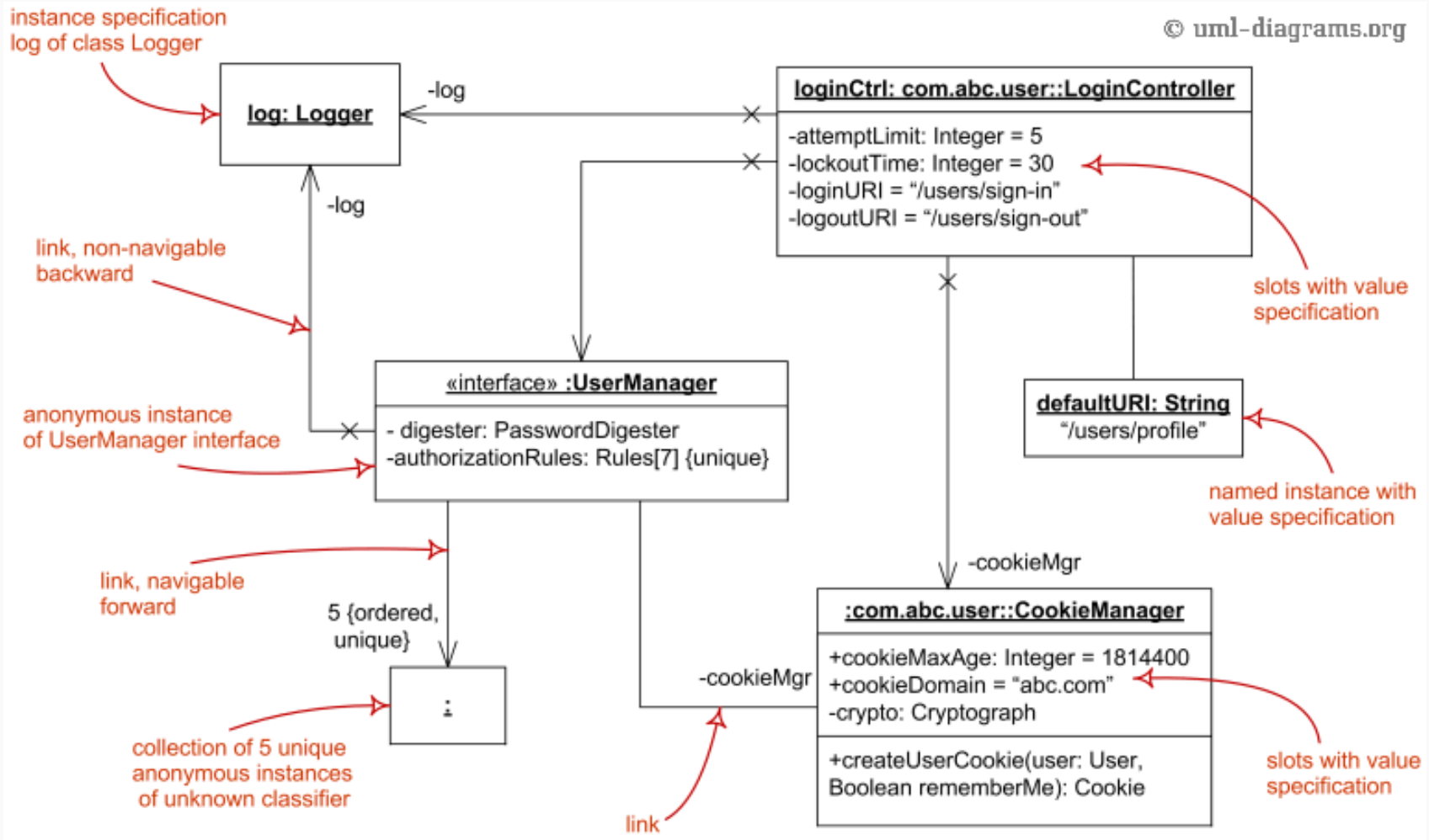


DIAGRAMA DE OBJETOS



DIAGRAMAS COMPORTAMENTAIS

DIAGRAMA DE CASOS DE USO

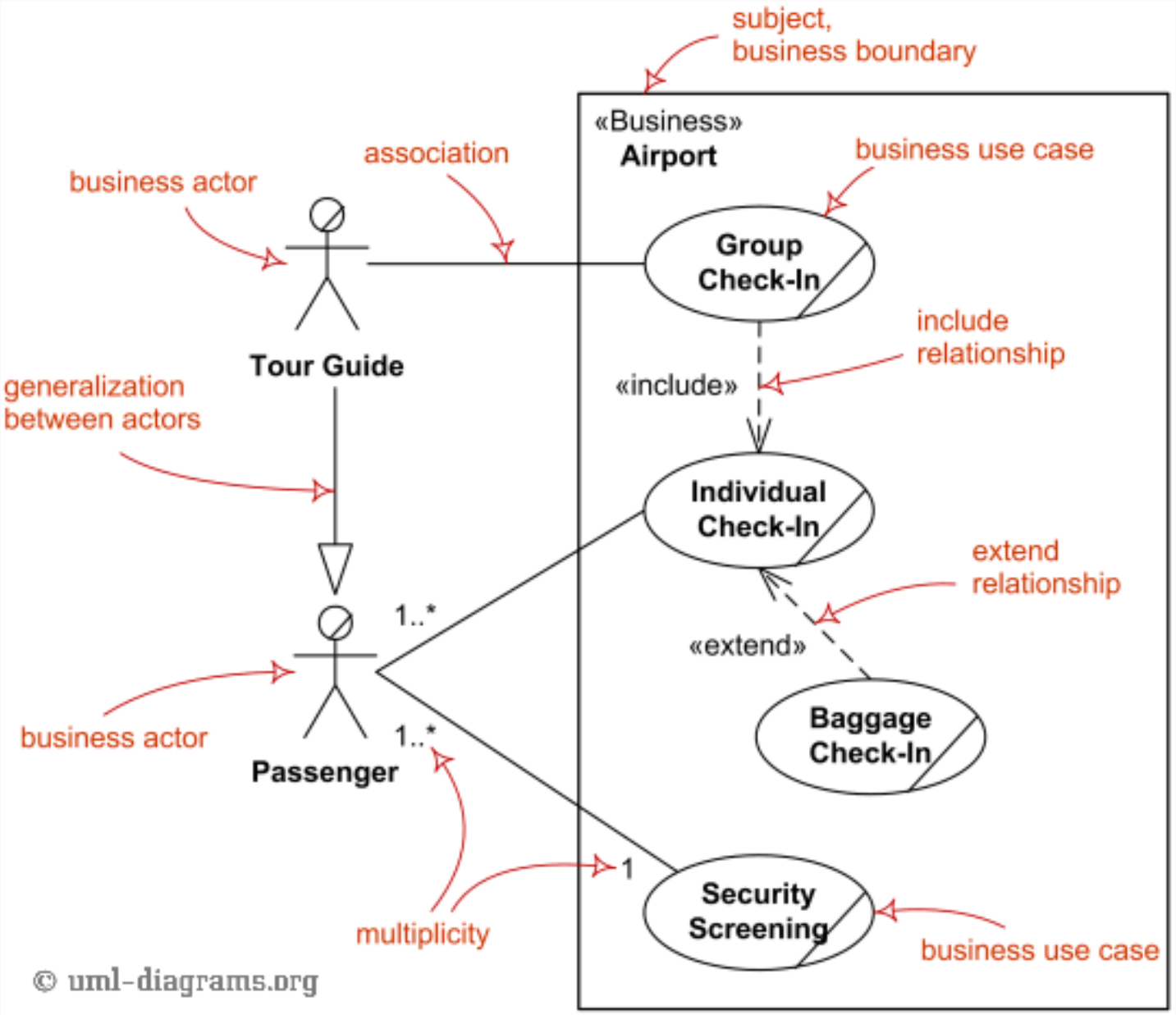
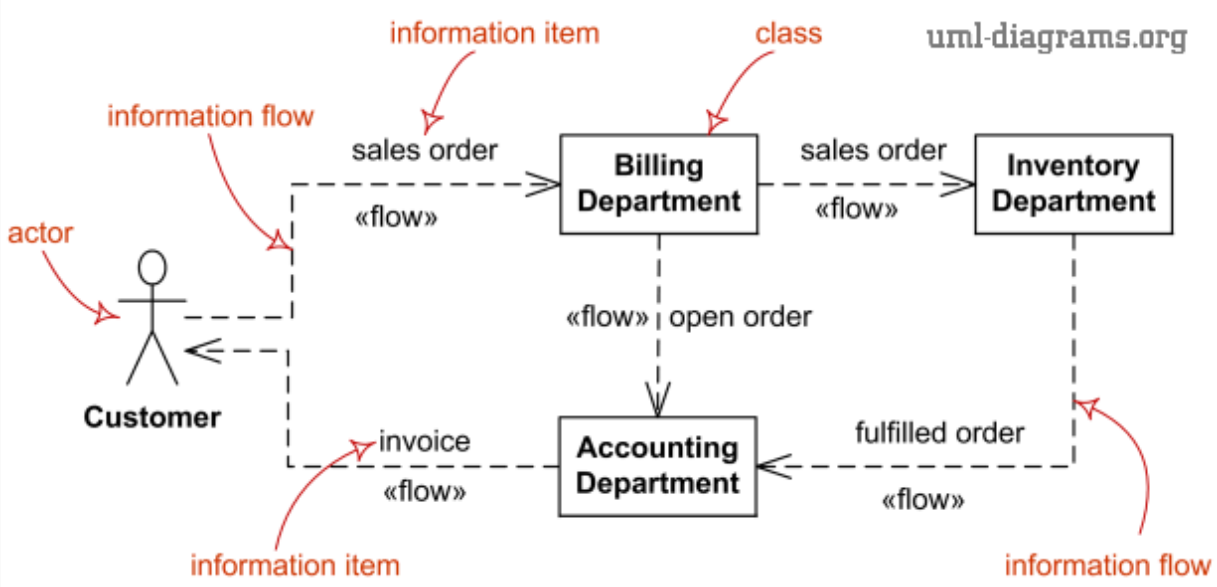


DIAGRAMA DE FLUXOS



- O fluxo de eventos principal descreve o caso em que tudo corre bem.
- Fluxos de eventos excepcionais cobrem as variações que podem ocorrer quando diferentes coisas dão errado ou quando algo pouco comum acontece.

DIAGRAMA DE SEQUÊNCIA

- É um diagrama de interações que enfatiza a ordem temporal das mensagens
- Uma linha de vida é uma linha tracejada vertical que representa o tempo de vida de um objeto
- Um foco de controle é um retângulo fino vertical sobreposto à linha de vida que mostra o período durante o qual um objeto está realizando uma ação

DIAGRAMA DE SEQUÊNCIA

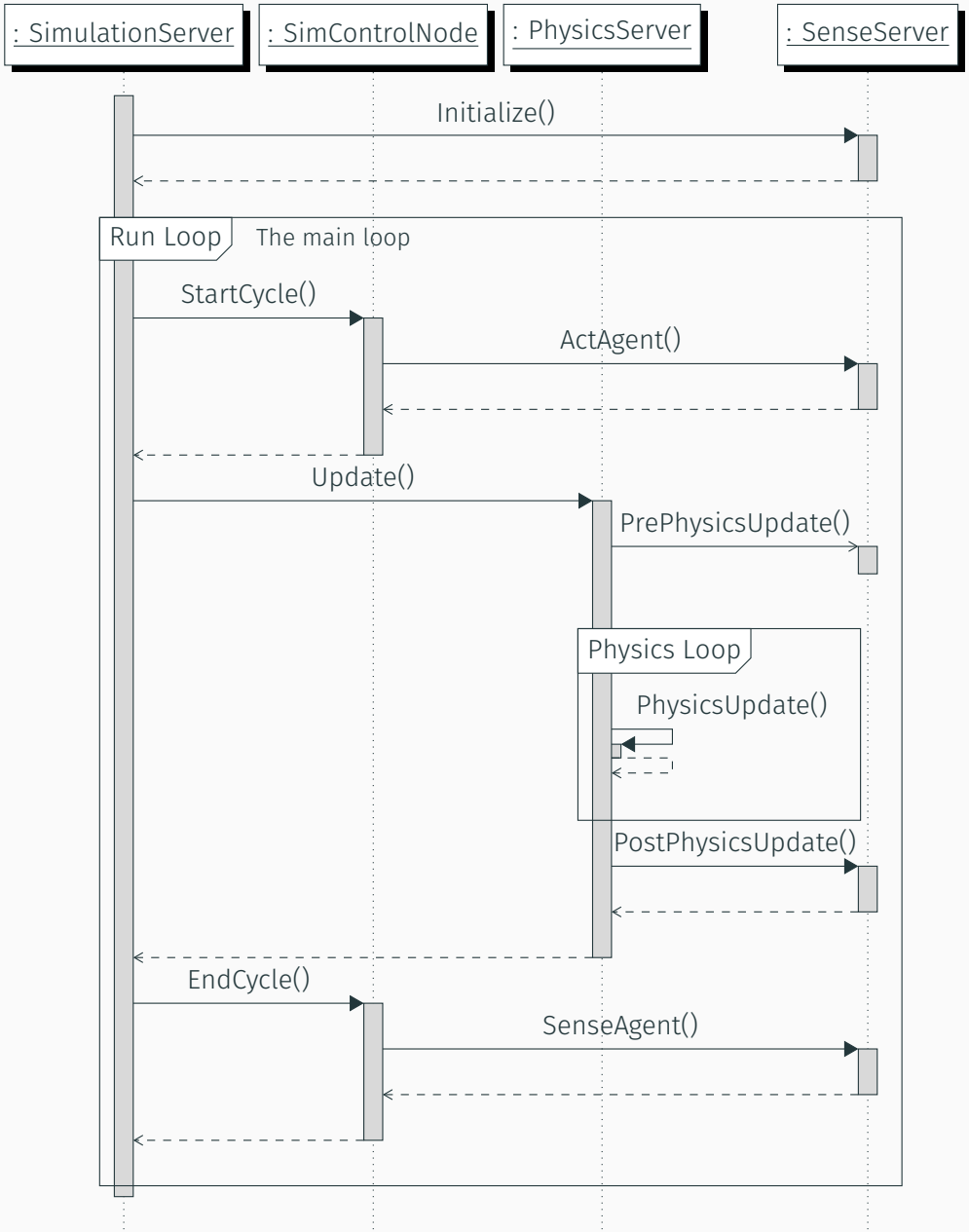
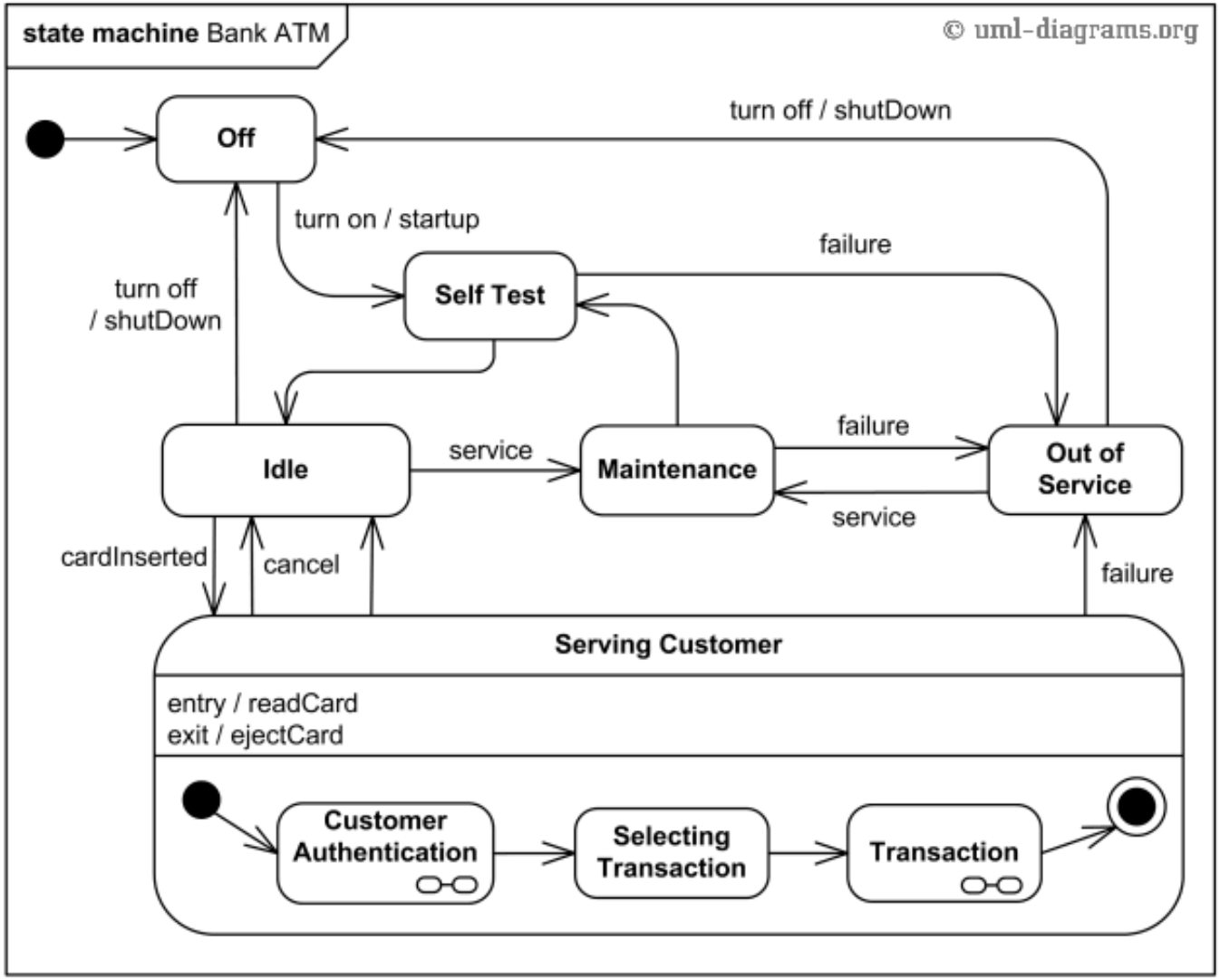


DIAGRAMA DE ESTADOS



USOS DE UML: COMO UM ESBOÇO

- Uso mais comum de UML
- Usado para comunicar algum aspecto do sistema para melhor entendê-lo
- Usado tanto para a engenharia do sistema (ou seja, desenhar os diagramas antes de escrever código) como para engenharia reversa (ou seja, desenhar os diagramas para entender melhor o código)
- Se esforça para ser informal e dinâmico
- Enfatiza apenas as classes, atributos, operações e relações de interesse
- Mais preocupado com a comunicação seletiva do que com a especificação completa

USOS DE UML: COMO O DESENHO COMPLETO DO SISTEMA

- O objetivo é completude
- Tem caráter definitivo, enquanto que o esboço tem caráter exploratório
- Usado para descrever o desenho do sistema em detalhes para que o programador possa seguir o desenho e escrever o código correspondente
- A notação tem que ser suficiente para que um programador possa seguir o desenho do sistema
- Às vezes usado por arquitetos de software para desenvolver um modelo de alto nível que mostra apenas as interfaces dos subsistemas ou classes (desenvolvedores ficam responsáveis pelos detalhes de implementação)
- Quando criado como produto de engenharia reversa, os diagramas transmitem com mais facilidade informações sobre o código-fonte

USOS DE UML: COMO LINGUAGEM DE PROGRAMAÇÃO

- Especifica o sistema completo de forma que o código possa ser gerado automaticamente
- Trata UML da perspectiva do software e não da perspectiva do modelo conceitual do problema
- Diagramas são compilados diretamente para código executável, de modo que o UML se torna o código-fonte do programa
- O desafio é fazer com que o UML seja mais produtivo do que uma linguagem de programação
  - na prática não é viável!
  - modelar comportamento é complicado (mesmo com o uso de diagramas de iteração, estado e atividades)

COMPARAÇÃO DOS USOS DE UML

- Esboços em UML são muito úteis tanto no desenvolvimento como na engenharia reversa; tanto no nível conceitual como na perspectiva de software
- UML como desenho completo de um sistema são difíceis de fazer corretamente e tendem a diminuir o ritmo de desenvolvimento; além disso, a implementação acaba mostrando novas necessidades de mudanças
- UML como desenho completo de um sistema feito com engenharia reversa pode ser útil, mas depende de auxílio de ferramentas: se for exibido dinamicamente pode ser muito útil, mas como documentação é um desperdício de tempo e recursos
- o uso de UML como linguagem de programação provavelmente nunca será significante

BIBLIOGRAFIA

- Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2003. Disponível de dentro da USP em http://proquest.safaribooksonline.com/book/software-engineering-and-development/uml/0321193687
- Scott W. Ambler. Introduction to the Diagrams of UML 2.X http://www.agilemodeling.com/essays/umlDiagrams.htm
- Fabio Kon. Uma Visão Geral de UML. http://www.ime.usp.br/~kon/presentations/UMLIntro.pdf
- http://www.uml-diagrams.org/