# UML – Visão Geral



## Índice

# O que é a UML?

- UML = *Unified Modeling Language*
- UML é uma linguagem (notação com semântica associada) para
  - visualizar
  - especificar
  - construir
  - documentar

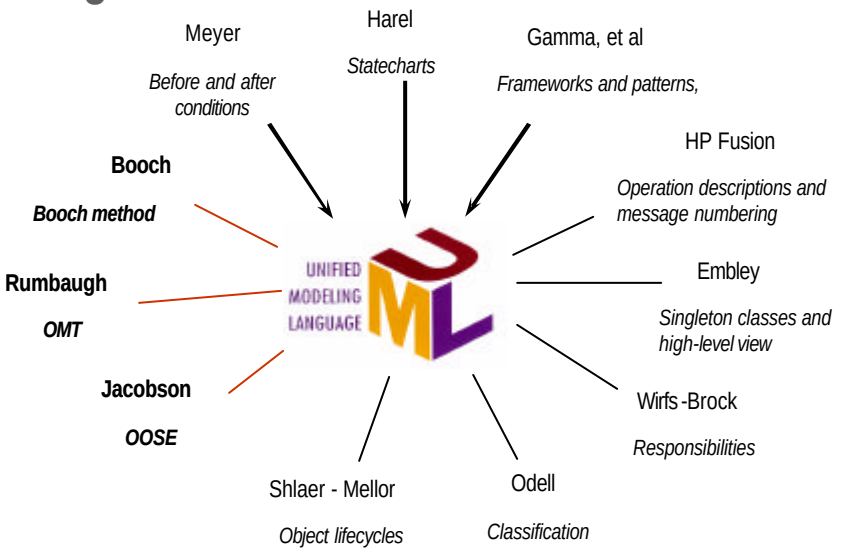
  os artefactos de um sistema com uma componente intensiva de software (*software intensive system*)
- UML não é uma metodologia
  - não diz quem deve fazer o quê, quando e como
  - UML pode ser usado segundo diferentes metodologias, tais como RUP (*Rational Unified Process*), FDD (*Feature Driven Development*), etc.
- UML não é uma linguagem de programação



# Valor da UML

- É um standard aberto
  - versão 1.1 aprovada pelo OMG (*Object Management Group)* em Novembro de 1997
  - versão 1.3 aprovada em Junho de 1999
- Suporta todo o ciclo de vida do software
  - modelação do negócio (processos e objectos do negócio)
  - modelação de requisitos alocados ao software
  - modelação da solução de software
- Suporta diversas áreas de aplicação
- É baseado na experiência e necessidades da comunidade de utilizadores
- É suportado por muitas ferramentas

# Origens da UML






# Parceiros da UML

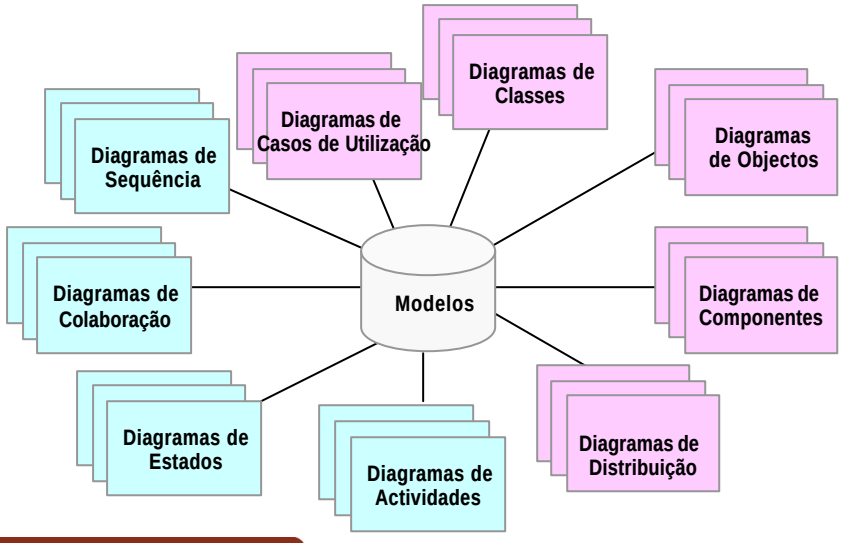
- Rational Software Corporation
- Hewlett-Packard
- I-Logix
- IBM
- ICON Computing
- Intellicorp
- MCI Systemhouse
- Microsoft
- ObjecTime
- Oracle
- Platinum Technology
- Taskon
- Texas Instruments/Sterling Software
- Unisys

# Modelos e Diagramas

- Um modelo é uma representação em **pequena escala**, numa perspectiva particular, de um sistema existente ou a criar
  - Atitude de **abstracção** (omissão de detalhes) fundamental na construção de um modelo
  - Modelos são a linguagem por excelência do projectista (*designer*)
  - Modelos são veículos para comunicação com vários interessados (*stakeholders*)
  - Modelos permitem raciocinar acerca do sistema real, sem o chegar a construir
- Ao longo do ciclo de vida de um sistema são construídos vários modelos, sucessivamente refinados e enriquecidos
- Um modelo é constituído por um conjunto de **diagramas** (desenhos) consistentes entre si, acompanhados de descrições textuais dos **elementos** que aparecem nos vários diagramas
  - Um **diagrama** é uma vista sobre um modelo
  - O mesmo **elemento** (exemplo: classe) pode aparecer em vários diagramas de um modelo
- No UML, há nove diagramas *standard*
  - Diagramas de visão estática: casos de utilização (*use case*), classes, objectos, componentes, distribuição (*deployment*)
  - Diagramas de visão dinâmica: sequência, colaboração, estados (*statechart*), actividades

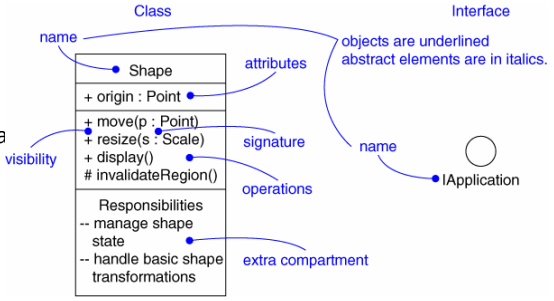


# Modelos e Diagramas



# Elementos de Modelação (1)

- Elementos estruturais
  - classe, interface, colaboração, caso de utilização, classe activa componente, nó



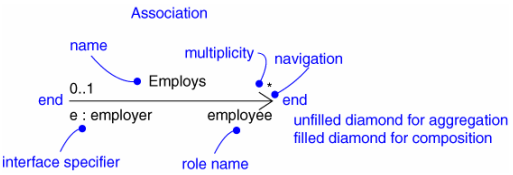

Fonte: Grady Booch

- Elementos de comportamento
  - interacção, máquina de estados
- Elementos de agrupamento
  - pacote (*package*), subsistema
- Outros elementos
  - nota



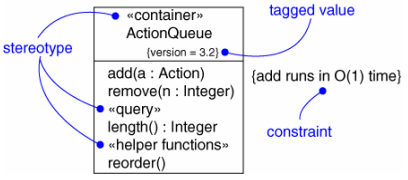
# Elementos de Modelação (2)

- Relações
  - Dependência
  - Associação
  - Generalização
  - Concretização (*realization*)




Fonte: Grady Booch

- Mecanismos de extensibilidade
  - Estereótipos
  - Propriedades (*tagged values*)
  - Restrições (*constraints*)



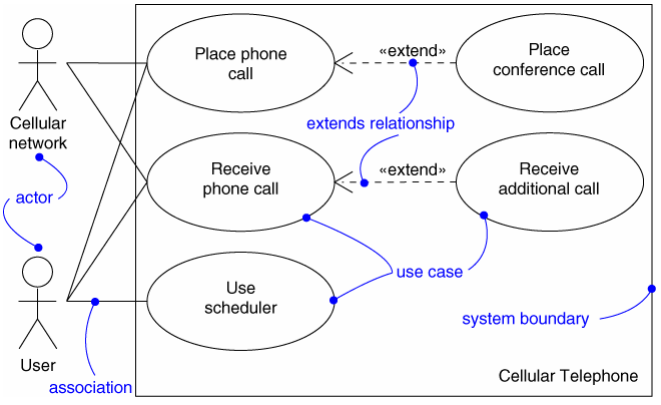

Fonte: Grady Booch

# Diagrama de Casos de Utilização (*Use Case Diagram*)

- Captura a funcionalidade do sistema tal como é visto pelos utilizadores
- Construído nos primeiros estágios do desenvolvimento
- Objectivo
  - Especificar o contexto de um sistema
  - Capturar os requisitos funcionais de um sistema
  - Validar a arquitectura de um sistema
  - Dirigir a implementação e gerar casos de teste
- Desenvolvido por analistas e especialistas de domínio



# Diagrama de Casos de Utilização (*Use Case Diagram*)



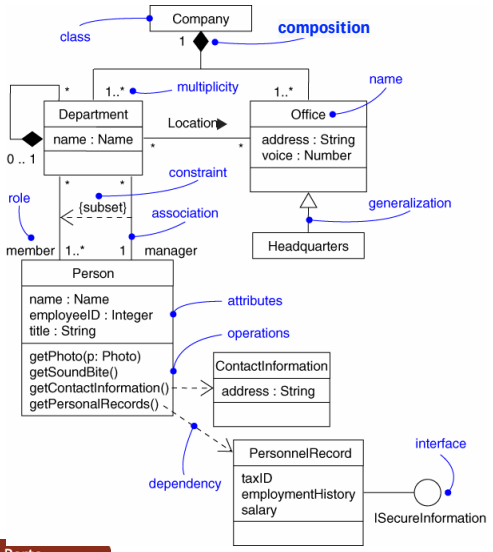

Fonte: Grady Booch

# Diagrama de Classes

- Captura o vocabulário de um sistema
- Construído e refinado ao longo do desenvolvimento
- Objectivo
  - Nomear e modelar conceitos no sistema
  - Especificar colaborações
  - Especificar esquemas lógicos de bases de dados
- Desenvolvido por analistas, *designers* e implementadores



# Diagrama de Classes



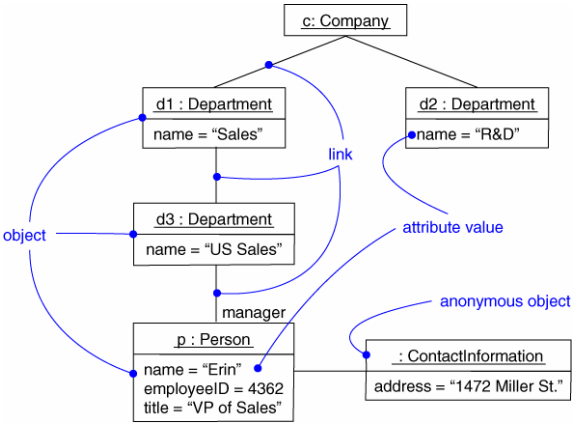

Fonte: Grady Booch

# Diagrama de Objectos

- Mostra objectos (instâncias de classes) e ligações (instâncias de associações)
- Construído durante a análise e *design*
- Objectivo
  - Ilustrar estruturas de dados/objectos
  - Especificar instantâneos (*snapshots*)
- Desenvolvido por analistas, *designers* e implementadores



# Diagrama de Objectos



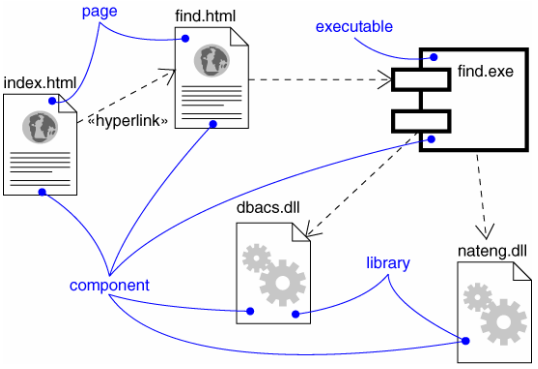

Fonte: Grady Booch

# Diagrama de Componentes

- Captura a estrutura física da implementação (tipicamente ficheiros)
- Construído como parte da especificação da arquitectura
- Objectivo
  - Organizar o código fonte
  - Construir uma *release* executável
  - Especificar uma base de dados física
- Desenvolvido por arquitectos e programadores



# Diagrama de Componentes



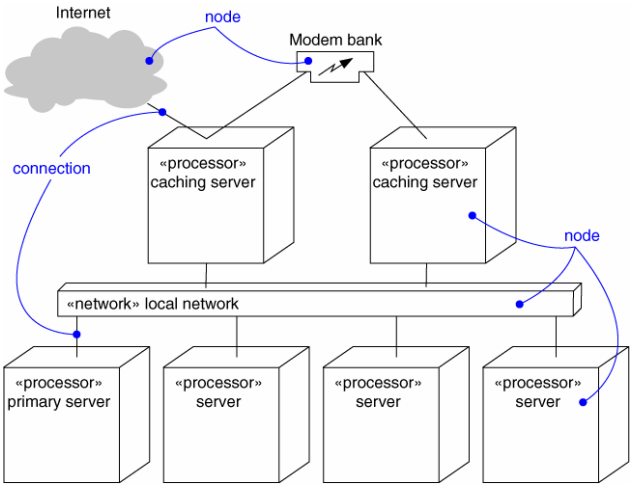

Fonte: Grady Booch

# Diagrama de Distribuição (*Deployment Diagram*)

- Captura a topologia do hardware de um sistema
- Construído como parte da especificação da arquitectura
- Objectivo
  - Especificar a distribuição de componentes
  - Identificar estrangulamentos de desempenho
- Desenvolvido por arquitectos, engenheiros de redes, e engenheiros de sistemas



# Diagrama de Distribuição (*Deployment Diagram*)



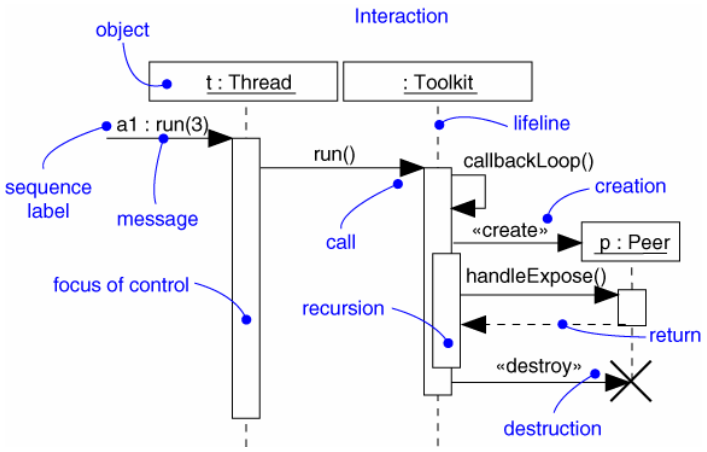

Fonte: Grady Booch

# Diagrama de Sequência

- Captura comportamento dinâmico (orientado ao tempo)
- Objectivo
  - Modelar fluxos de controlo
  - Ilustrar cenários típicos



# Diagrama de Sequência



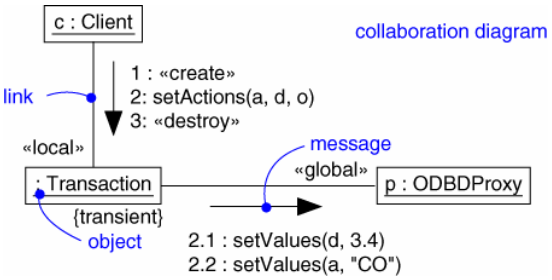

Fonte: Grady Booch

# Diagrama de Colaboração

- Captura comportamento dinâmico (orientado a mensagens)
- Objectivo
  - Modelar fluxo de controlo
  - Ilustrar a coordenação entre estrutura de objectos e controlo



# Diagrama de Colaboração



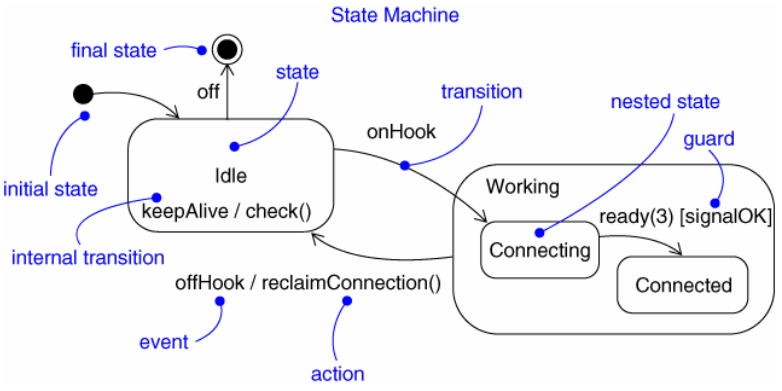

Fonte: Grady Booch

# Diagrama de Estados (*Statechart Diagram*)

- Captura comportamento dinâmico (orientado a eventos)
- Objectivo
  - Modelar ciclo de vida de objectos
  - Modelar objectos reactivos (interfaces com o utilizador, dispositivos, etc.)



# Diagrama de Estados (*Statechart Diagram*)



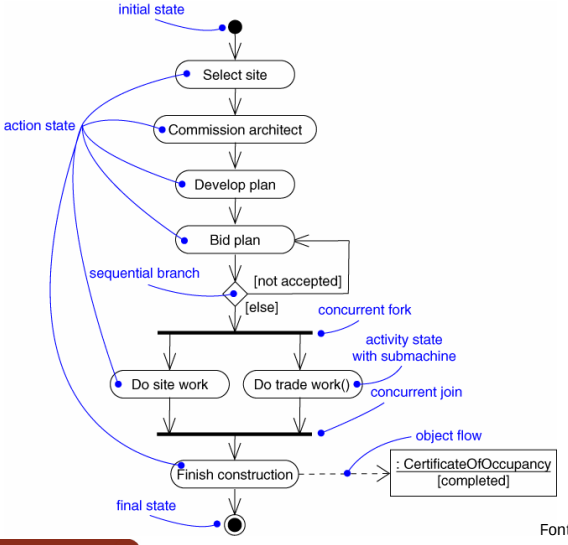

Fonte: Grady Booch

# Diagrama de Actividades

- Captura comportamento dinâmico (orientado a actividades)
- Objectivo
  - Modelar processos de negócio e *workflows*
  - Modelar operações (algoritmos)



# Diagrama de Actividades




Fonte: Grady Booch

# Referências

- Ferramentas de modelação visual
  - Rational Rose (www.rational.com)
  - Together (www.togethersoft.com) – disponível nos computadores da FEUP
  - Platinum Paradigm Plus (www.platinum.com)
  - Microsoft Visio – disponível no DEEC ao abrigo de protocolo com Microsoft
- Livros
  - The Unified Modeling Language User Guide, Grady Booch *et al*, Addison-Wesley, October, 1998
  - UML, Metodologias e Ferramentas CASE, Alberto SIlva e Carlos Videira, Centro Atlântico, 2001
- Especificações
  - www.omg.org