Engenharia de Software

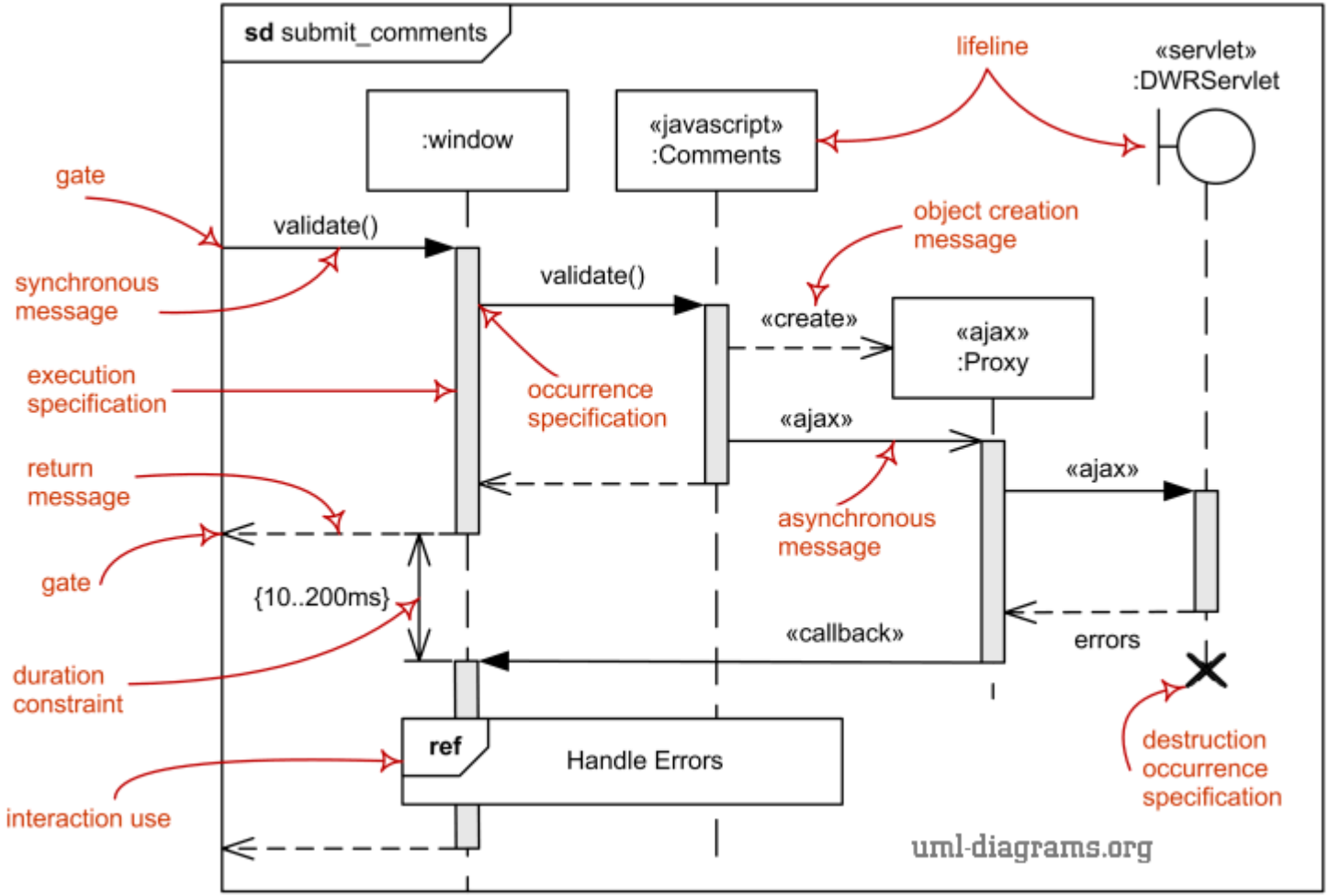
Linguagem UML
Parte 2

Luís Morgado

Instituto Superior de Engenharia de Lisboa
Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

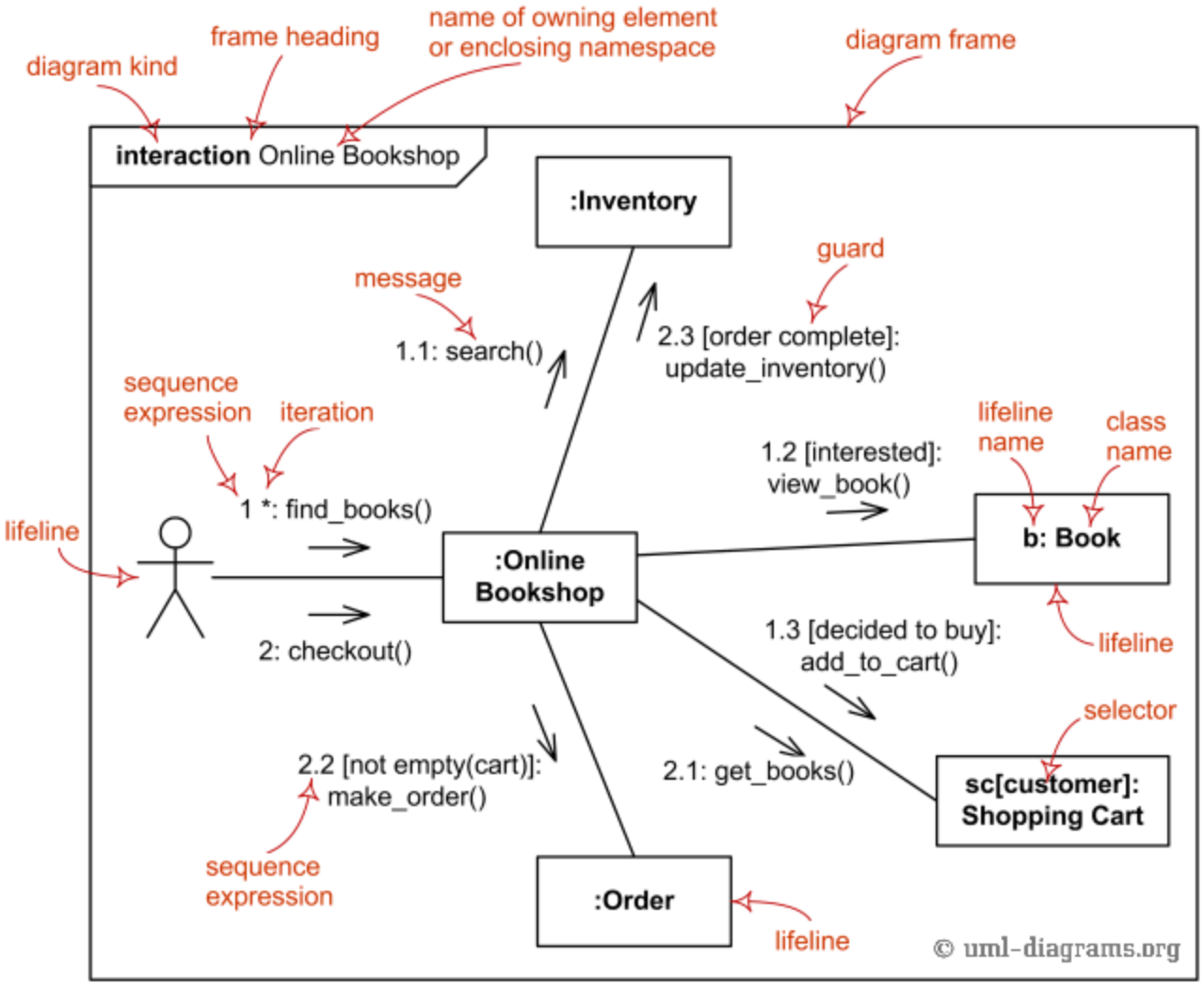
Diagramas de Sequência

Descrevem a sequência temporal de interacções entre partes do sistema e/ou com o exterior do sistema



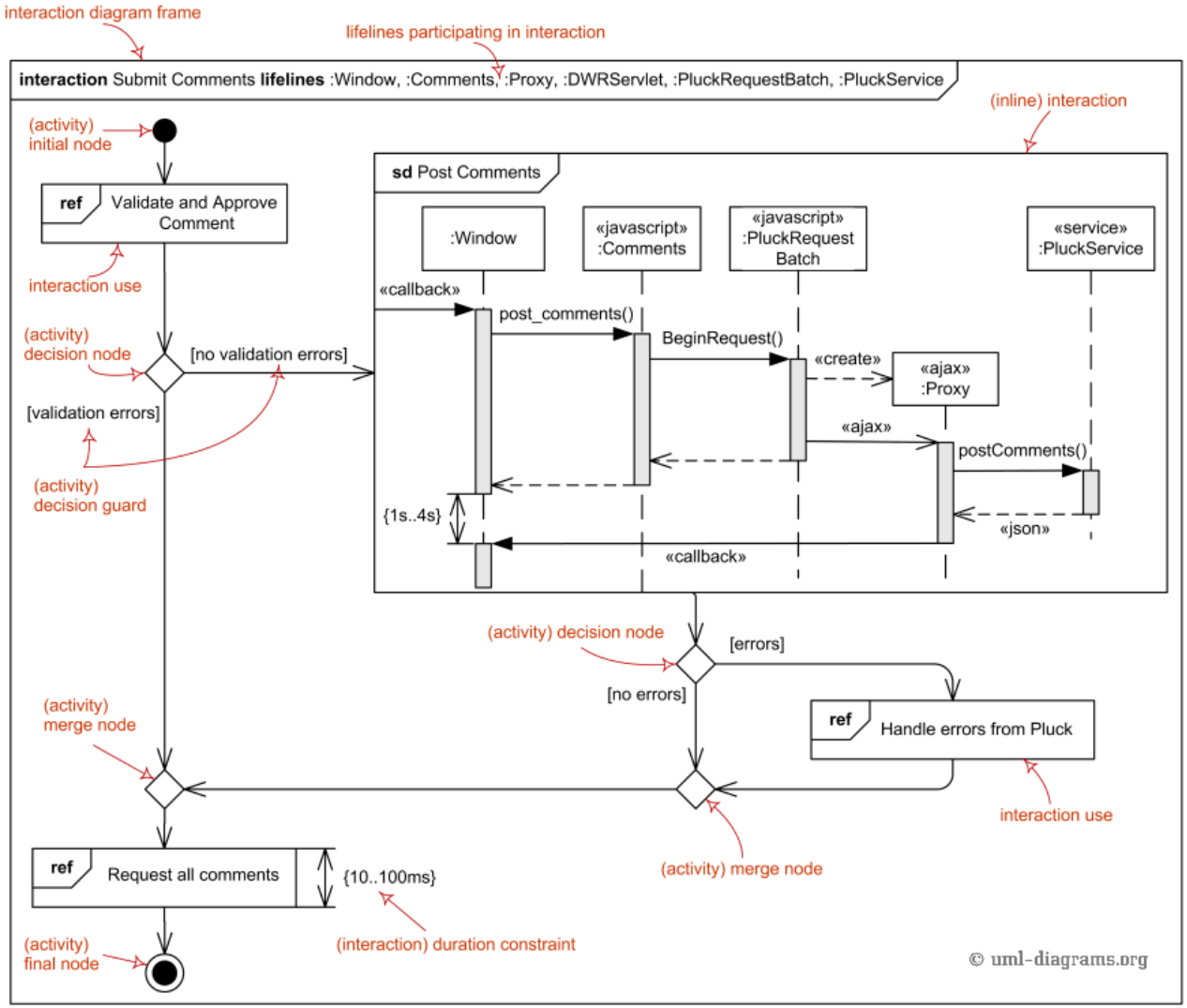
Diagramas de Comunicação

Descrevem a comunicação entre objectos em interacções entre partes do sistema e/ou com o exterior do sistema



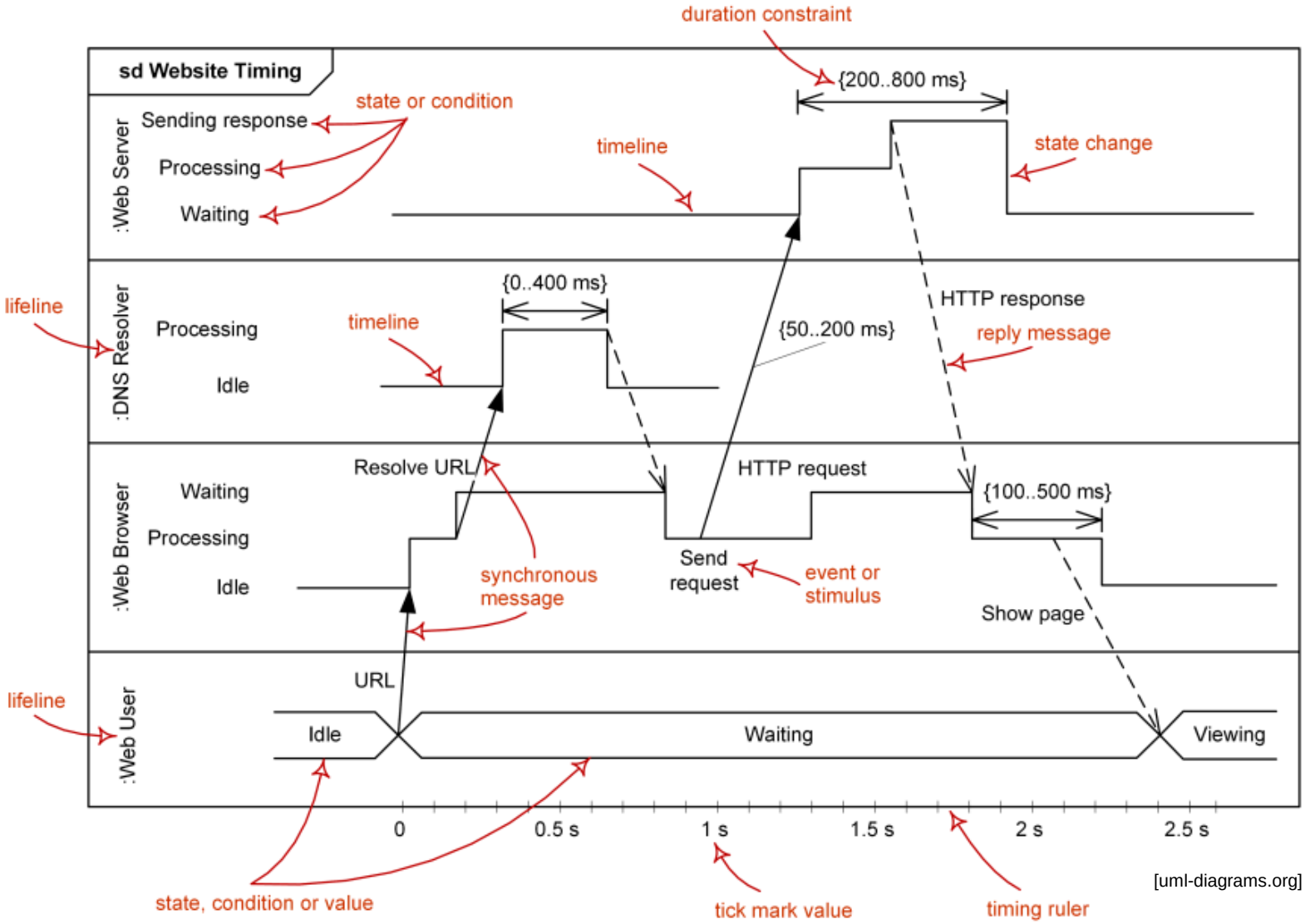
Diagramas de Enquadramento de Interacção

Descrevem as interacções entre objectos numa perspectiva global, enquadrando e relacionando representações parciais



Diagramas Temporais

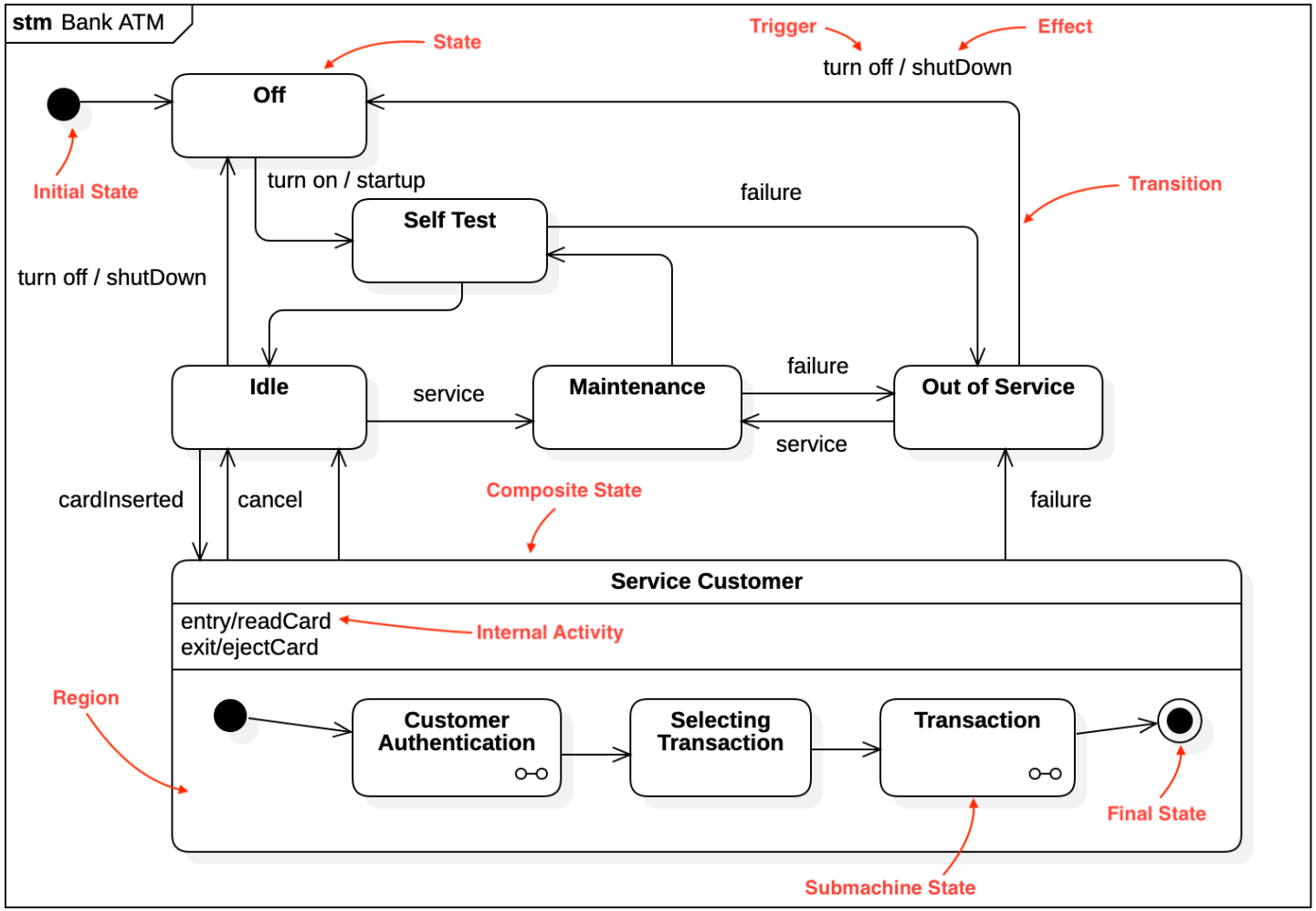
Descrevem restrições temporais de partes e interacções



[uml-diagrams.org]

Diagramas de Transição de Estado

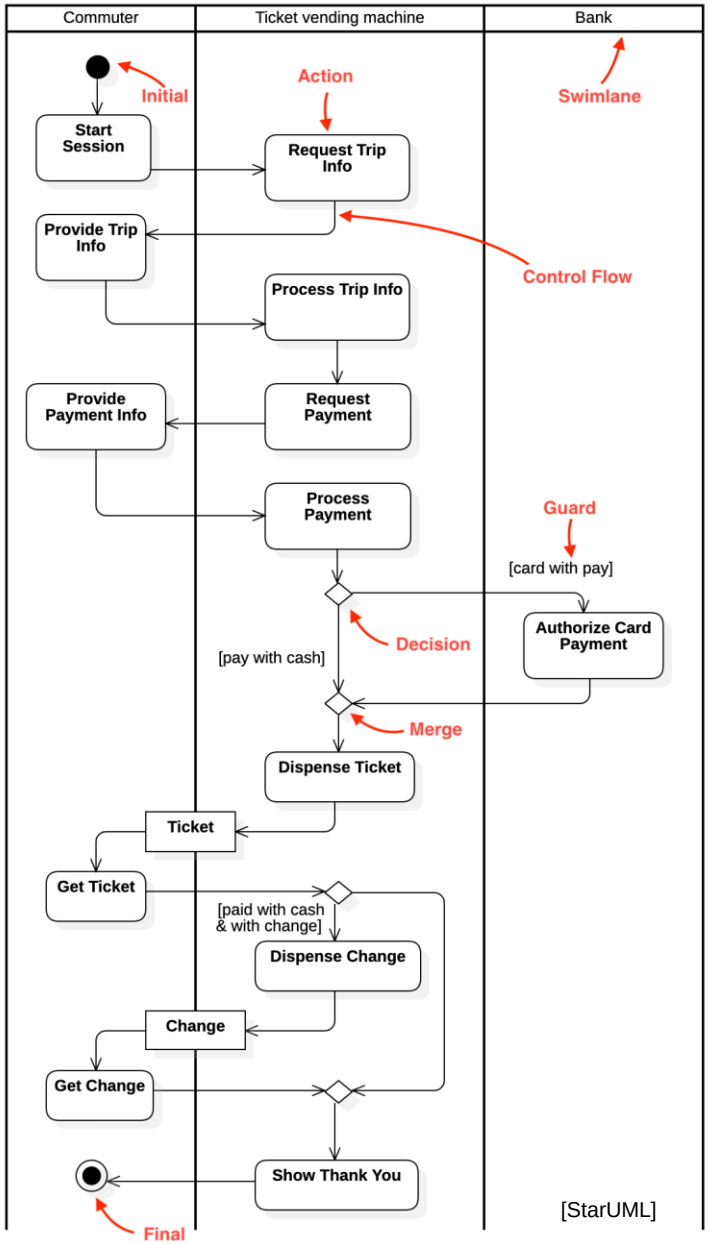
Descrevem a dinâmica de um sistema ou parte de um sistema em termos de estados e transições entre estados



[StarUML]

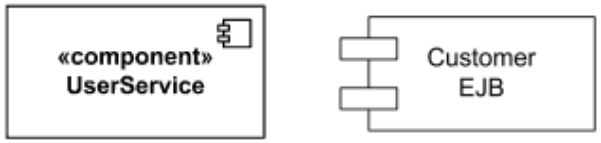
Diagramas de Actividade

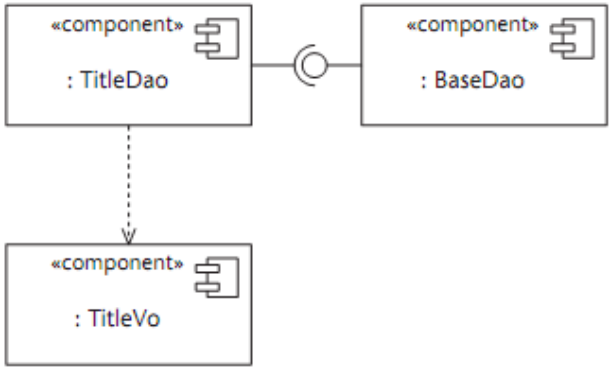
Descrevem a dinâmica de um sistema ou parte de um sistema em termos da sequência de acções de actividades ou processos

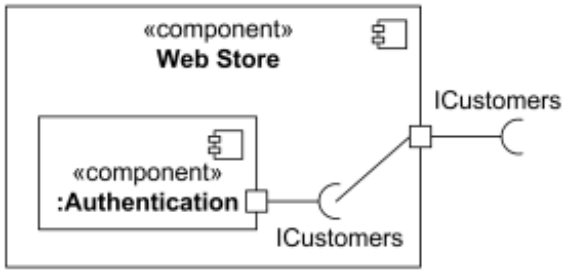


Diagramas de Componentes

- Organização lógica e física
  - Descrição das partes reutilizáveis de um sistema (componentes) e das relações entre essas partes
- Componente
  - Módulo ou parte autónoma e encapsulada de um sistema que pode ser reutilizada ou substituída
  - Disponibiliza uma ou mais interfaces que permitem a interacção entre componentes
  - Organização modular de um sistema
- Perspectivas de modelação
  - Perspectiva de caixa fechada (*black-box*)
    - Perspectiva de alto-nível com foco nos componentes e nas respectivas relações e configurações
  - Perspectiva de caixa aberta (*white-box*)
    - Perspectiva detalhada com foco na arquitectura interna dos componentes

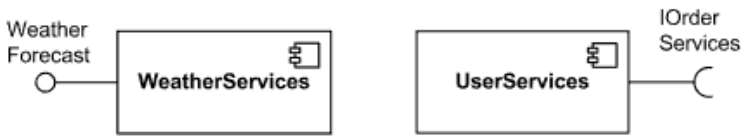


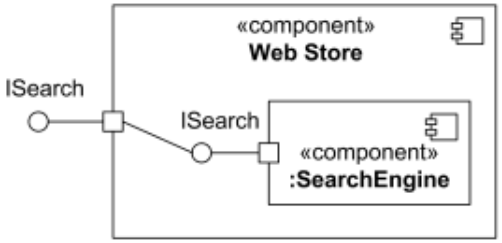


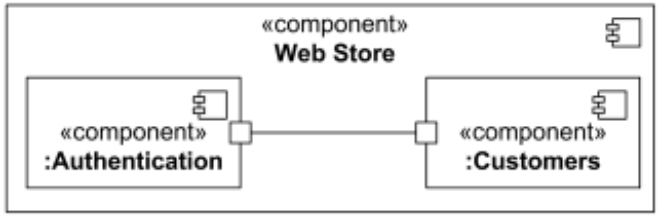


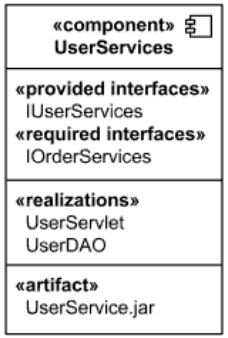
Diagramas de Componentes

- Interfaces
  - Encapsulamento de funcionalidade
  - Redução de acoplamento
    - Alterações internas num componente não se propagam a outros componentes
    - Controlo de dependências entre componentes
  - **Disponibilizadas** (*provided*)
  - **Requeridas** (*required*)
- Portos (*Ports*)
  - Pontos de interacção com o exterior
    - Ligação a partes internas que implementam ou utilizam interfaces
    - Agrupamento de interfaces
  - Conectores de delegação
    - Permitem relacionar interfaces com as partes internas que as implementam
- Compartimentos
  - Forma de representação onde são indicados aspectos específicos ou artefactos necessários à disponibilização do componente









[uml-diagrams.org]

Diagramas de Componentes

Estereótipos de componentes

Especializações com semântica específica

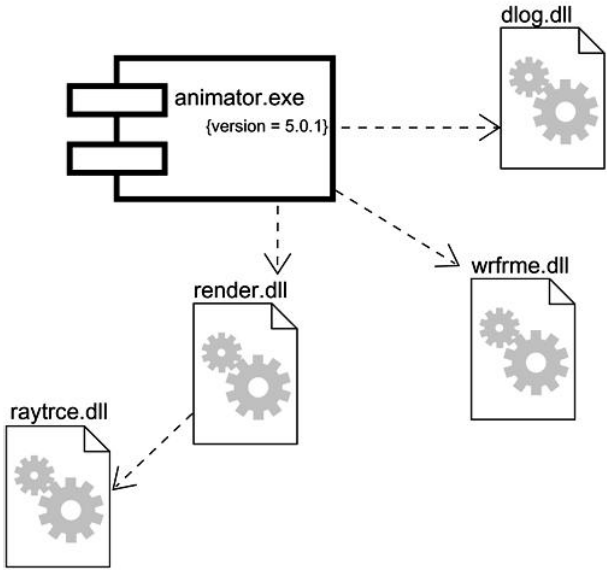
Entity	O componente representa um conceito do domínio do problema
Specification	O componente representa uma especificação, inclui interfaces mas não as implementa
Realization	O componente representa a realização de uma especificação, surge em conjunto com o estereótipo *«specification»*
Process	Componente transaccional ou com fluxo autónomo de execução (*thread*)
Executable	Componente de software que pode ser executado num nó físico
Subsystem	Parte de um sistema maior que agrega partes e mecanismos relacionados
Service	Componente sem manutenção de estado
Database	Base de dados
Table	Tabela de uma base de dados
Source code	Código fonte

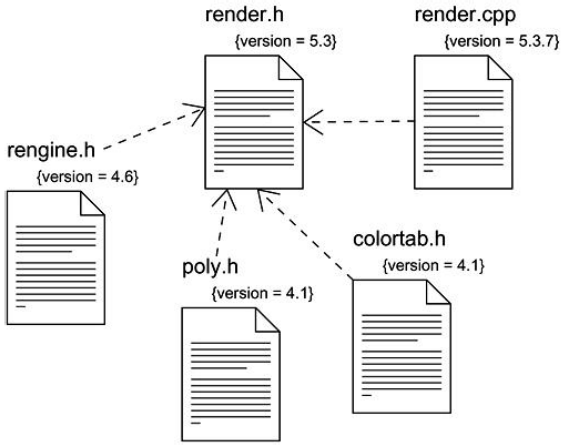
Diagramas de Componentes

Utilização

- Modelação de arquitectura lógica e física
- Modelação de executáveis e bibliotecas
- Modelação de dados e documentos
- Modelação de serviços
- Modelação de código fonte
- Representação da relação entre modelos lógicos e físicos

Exemplo: modelos de implementação e de código fonte

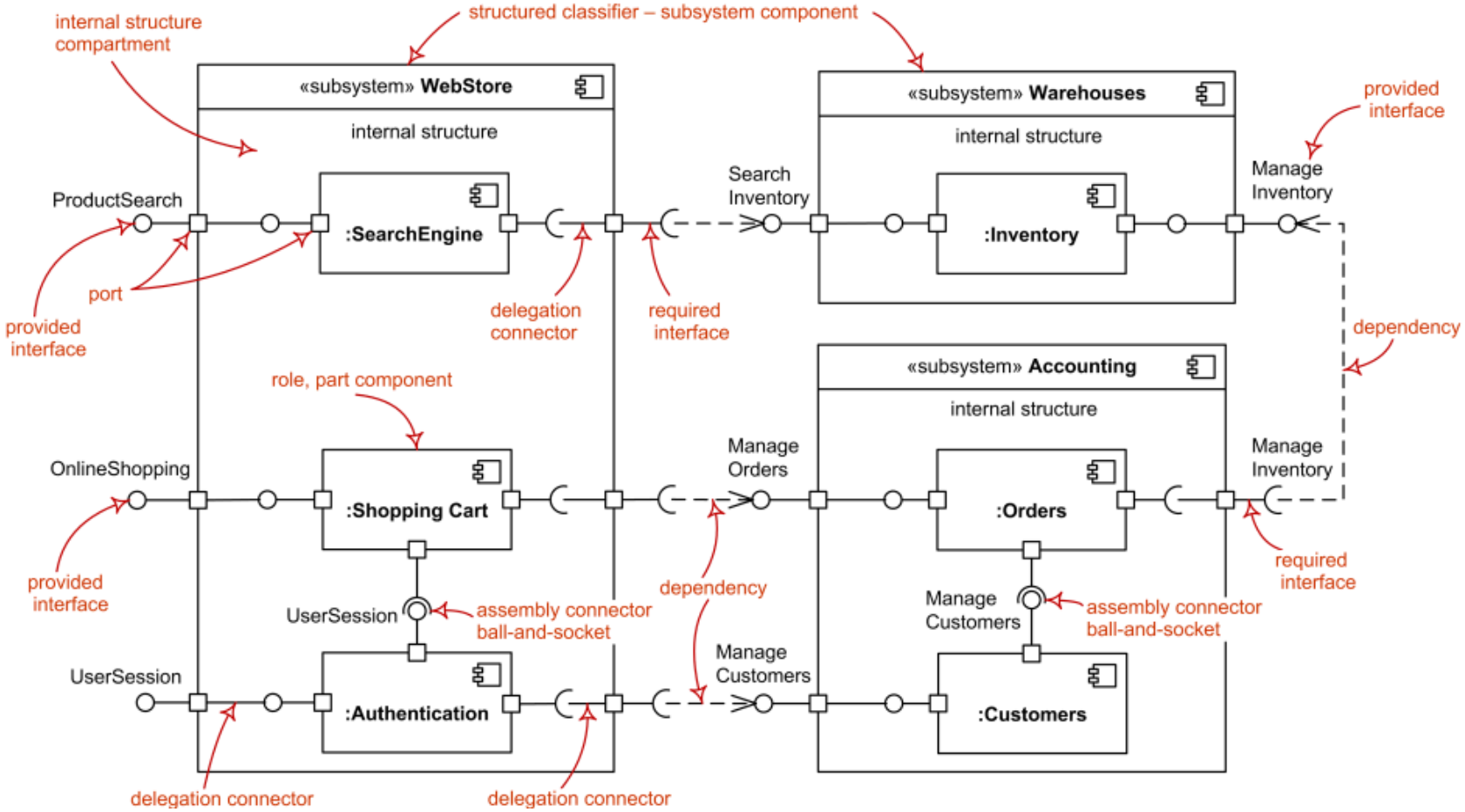




[Booch *et al.* 1998]

Diagramas de Componentes

Exemplo

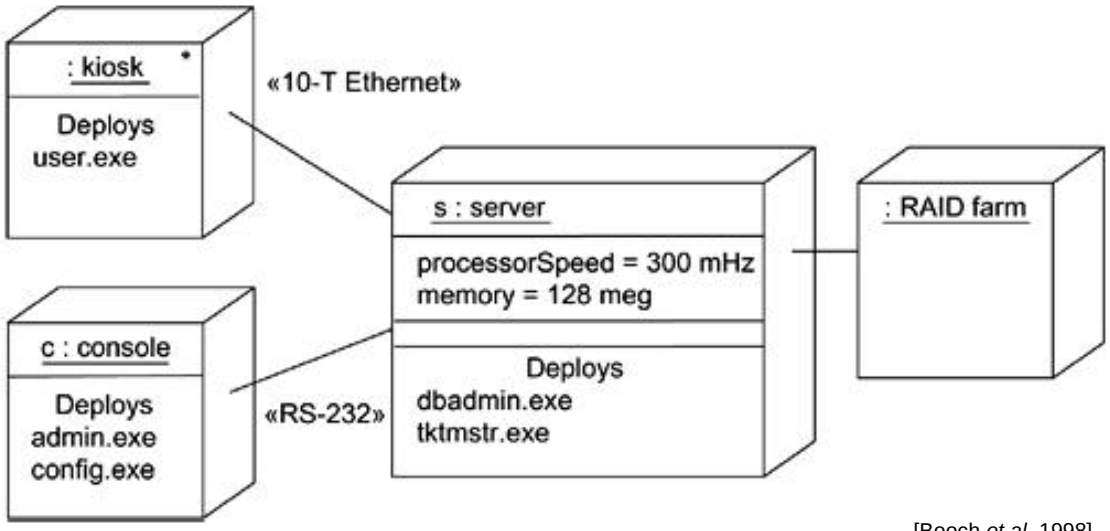


[uml-diagrams.org]

Diagramas de Implantação

Modelo de implantação

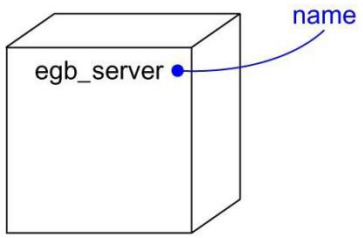
- Descreve a arquitetura de um sistema em termos da distribuição de *artefactos* de software para alvos de implantação
- Os *artefactos* representam elementos concretos resultantes do processo de desenvolvimento, por exemplo, ficheiros executáveis, bibliotecas, arquivos, esquemas de bases de dados, ficheiros de configuração, etc.
- O alvo de implantação é representado por um nó que corresponde a um dispositivo de hardware ou um ambiente de execução de software
- Os nós podem ser ligados através de canais de comunicação



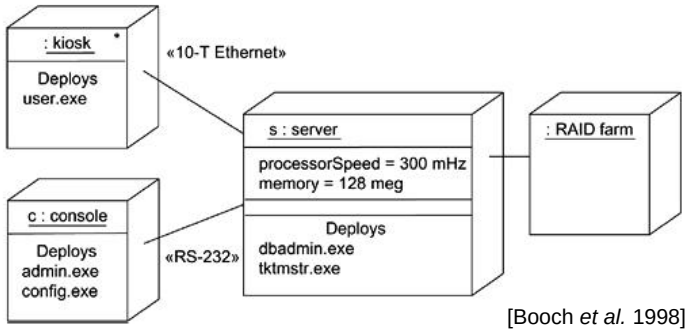
[Booch *et al.* 1998]

Diagramas de Implantação

- Perspectiva física do sistema
  - Relacionam os artefactos de software com o hardware que os executa
    - Mostram uma perspectiva estática em termos da sua configuração de execução, localização física e formas de comunicação
    - Sistema descrito em termos de software e de hardware
    - Representação da organização de implantação do sistema
  - Evoluem ao longo do ciclo de desenvolvimento
    - Identificação e exploração de dependências entre o sistema e outros sistemas no seu ambiente



Nó físico

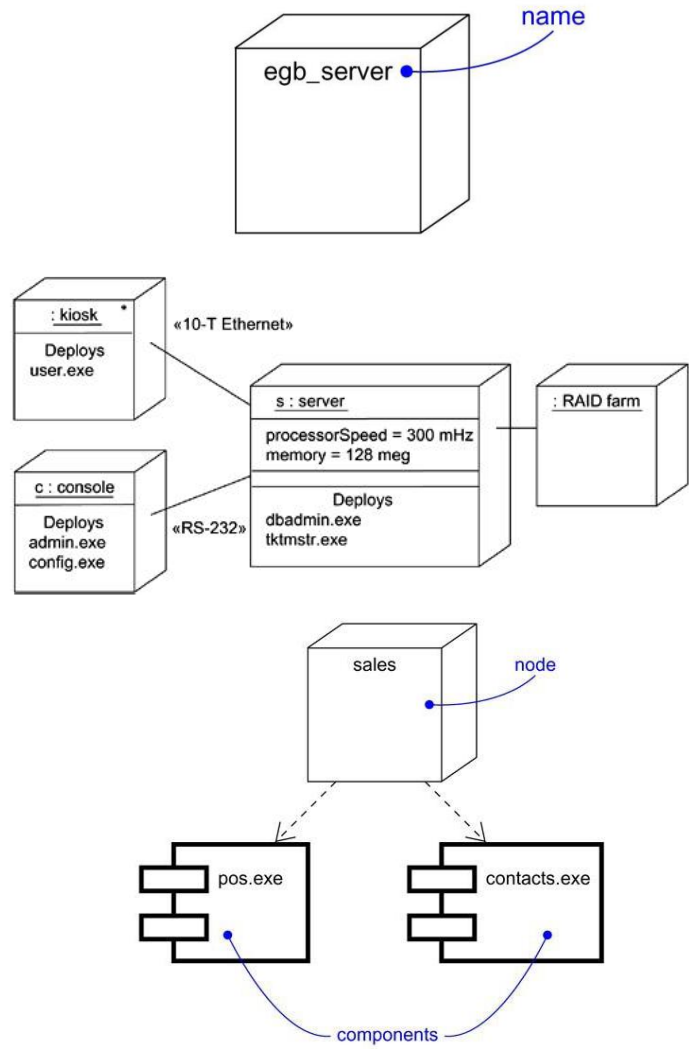


[Booch *et al.* 1998]

Nós físicos e respectivas ligações com indicação dos artefactos implantados

Diagramas de Implantação

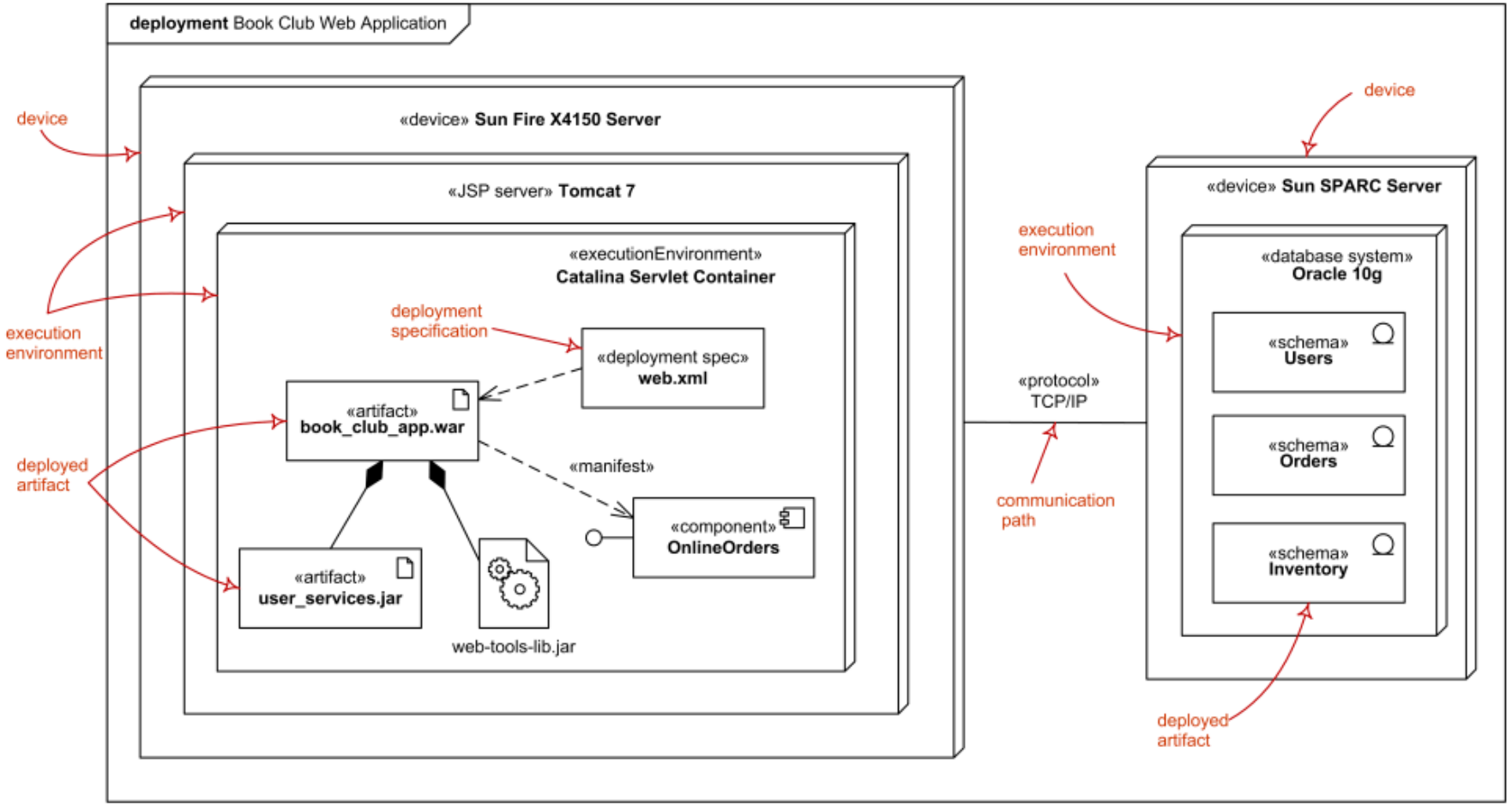
- **Nós**
  - Recursos físicos (entidades computacionais, hardware ou software) capazes de executar artefactos
    - Dispositivos (hardware de diferentes tipos)
    - Ambientes de execução (pode ser software, e.g. sistema operativo)
- **Artefactos**
  - Recursos físicos (ficheiros) que o sistema utiliza ou executa
  - Tipos de artefactos (podem corresponder a diferentes ficheiros físicos)
- **Ligações**
  - Representam canais de comunicação
    - Estereótipos: HTTP, TCP/IP, RMI, JDBC, ODBC, RPC, etc.



[Booch *et al.* 1998]

Diagramas de Implantação

Exemplo



[uml-diagrams.org]

Diagramas de Fluxo de Informação

Os *diagramas de fluxo de informação* representam a troca de informação entre entidades do sistema de forma abstracta

- Úteis para representar aspectos de modelos ainda não totalmente especificados, bem como para registar representações heurísticas menos detalhadas de aspectos de modelos complexos

Descrevem a troca de informação através de um sistema de uma forma geral

- Não especificam a natureza da informação, os mecanismos pelos quais é transmitida, as sequências de troca ou quaisquer condições de controlo

Num nível mais detalhado dos modelos, podem ser acrescentadas associações de realização para especificar quais os elementos do modelo que implementam um fluxo de informação e para mostrar como a informação é transmitida

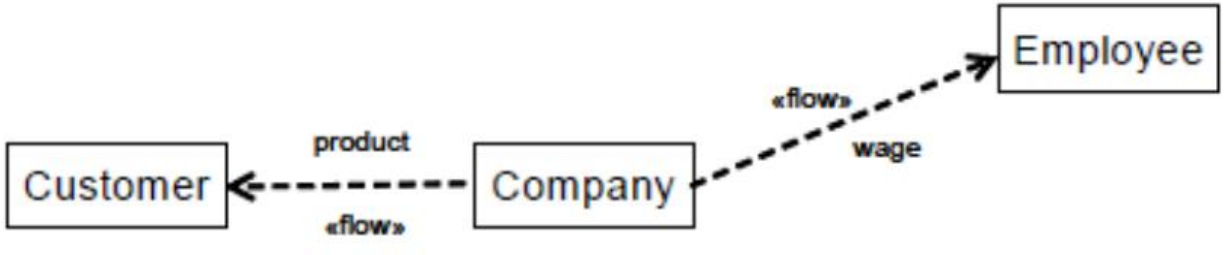
Podem ser utilizados *itens de informação* para representar a informação que circula nos fluxos de informação, nomeadamente, de forma abstracta antes dos pormenores da sua realização terem sido definidos

Diagramas de Fluxo de Informação

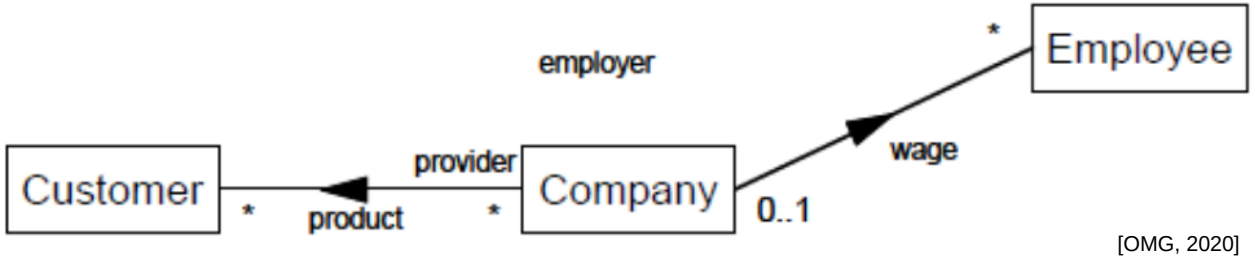
Os *fluxos de informação* são representados como dependências entre partes com o estereótipo **<<flow>>**, através dos quais fluem *itens de informação*

Exemplo

Informação sobre produtos e salários (elementos de informação) que flui de uma empresa para os seus clientes e empregados em dois fluxos distintos



Sem representação de canais de informação



[OMG, 2020]

Com representação de canais de informação

Diagramas de Fluxo de Informação

Os *itens de informação* podem ser representados explicitamente e representar outros itens de informação, incluindo elementos concretos

Exemplo

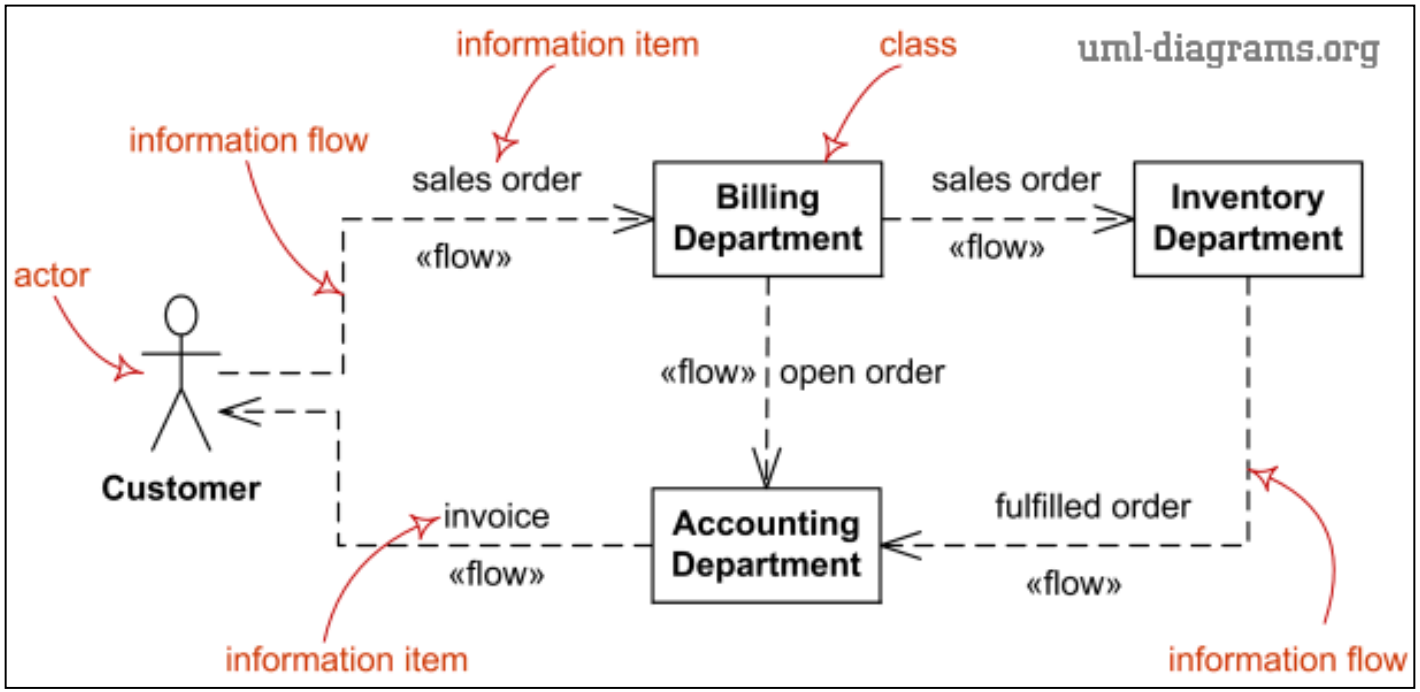
O item de informação que representa um documento de viagem ("*travel document*") representa tanto passaportes como bilhetes de avião (também itens de informação)

O item de informação que representa o vencimento ("*Wage*") representa as classes concretas "*Salary*" e "*Bonus*"

[OMG, 2020]

Diagramas de Fluxo de Informação

Exemplo



Bibliografia

[Pressman, 2003]

R. Pressman, *Software Engineering: a Practitioner's Approach*, McGraw-Hill, 2003.

[Gamma et al., 1995]

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[Shaw & Garlan, 1996]

M. Shaw, D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

[Vernon, 2013]

V. Vernon, *Implementing Domain Driven Design*, Addison-Wesley, 2013.

[Parnas, 1972]

D. Parnas, *On the Criteria to Be Used in Decomposing Systems into Modules*, Communications of the ACM 15-12, 1968.

[Kruchten, 1995]

F. Kruchten, *Architectural Blueprints - The "4+1" View Model of Software Architecture*, IEEE Software, 12-6, 1995.

[Burbeck, 1992]

S. Burbeck; *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)*, http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html,1992

[Booch, 2004]

G. Booch, *Software Architecture*, IBM, 2004.