VISTAS DE ARQUITETURA

MODELAÇÃO E ANÁLISE DE SISTEMAS | TP

ILÍDIO OLIVEIRA ico@ua.pt
v2018-04-19

universidade de aveiro
departamento de eletrónica, telecomunicações e informática
deti


http://www.kuwans.com

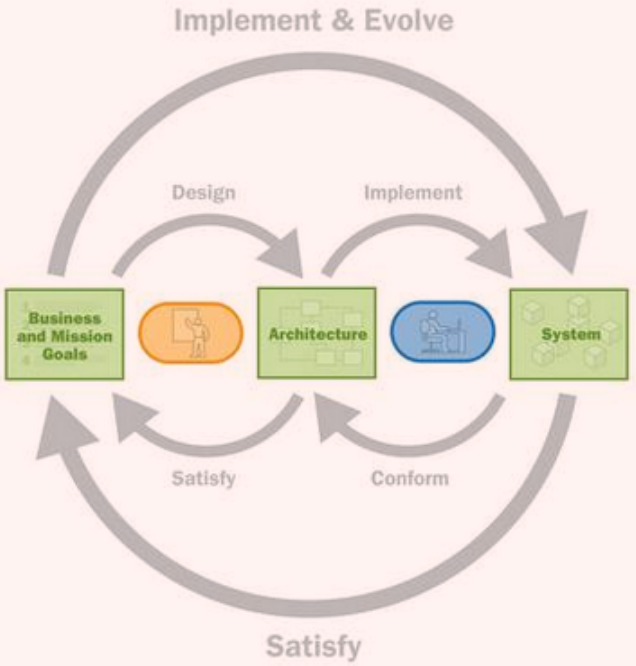
An architecture is **the set of significant decisions about the organization of a software system**, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization these elements and their interfaces, their collaborations, and their composition. [BRJ99]

→ Big ideas on system organization

Why Architecture?

The software architecture of a program or computing system is a depiction of the system that aids in understanding how the system will behave.

Software architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by design and implementation teams. The architecture is the primary carrier of system qualities such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. Architecture is an artifact for early analysis to make sure that a design approach will yield an acceptable system. By building an effective architecture, you can identify design risks and mitigate them early in the development process.



https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=21328

Assuntos da Arquitetura do sistema

Organização estrutural do sistema em grandes blocos

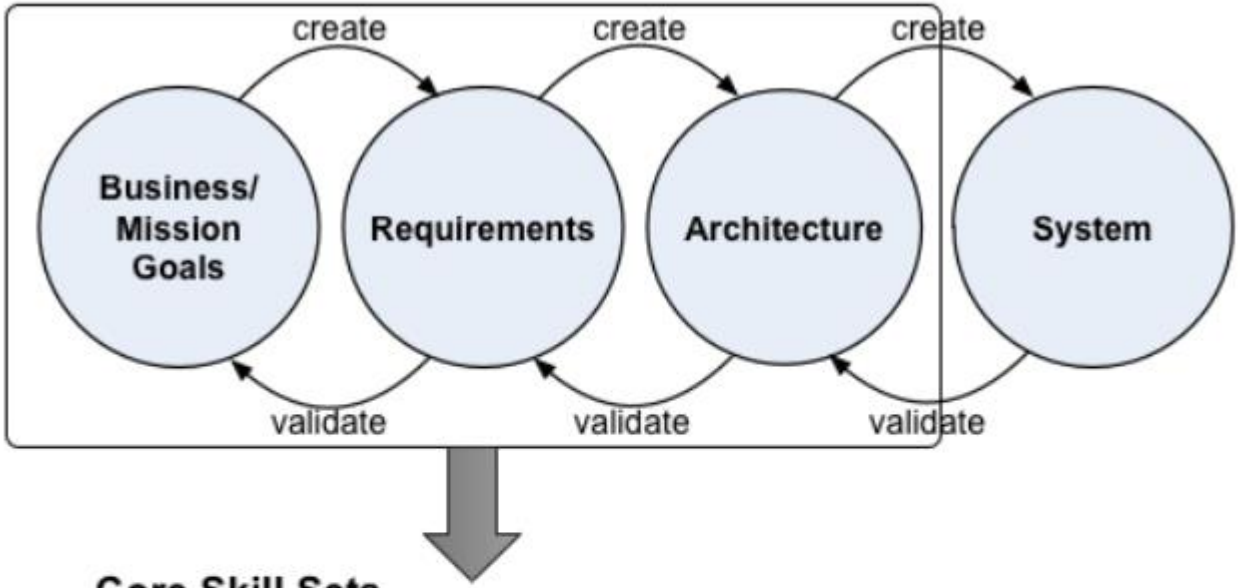
- Componentes desenvolvidos e/ou integrados
- Topologia física: servidores, rede,...

Uma arquitetura é definida para satisfazer os requisitos

- Requisitos não funcionais e restrições de operação são determinantes
- E.g.: sessões simultâneas? Licenciamento? Tolerância a falhas?

Compromissos e decisões!

Papel do arquiteto (de software)



Core Skill Sets

- Design - create and evolve
- Analysis - will the design provide the needed functions and qualities?
- Models and representations - "documentation"
- Evaluation - are we satisfying stakeholders?

- Communication – with technical and business teams
- Technical Leadership

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=455074

Elaboration: Know How to Build It by Building Some

Elaboration can be a day long or several iterations

Balance

mitigating key technical and business risks with producing value (tested code)

Produce (and validate) an **executable architecture**

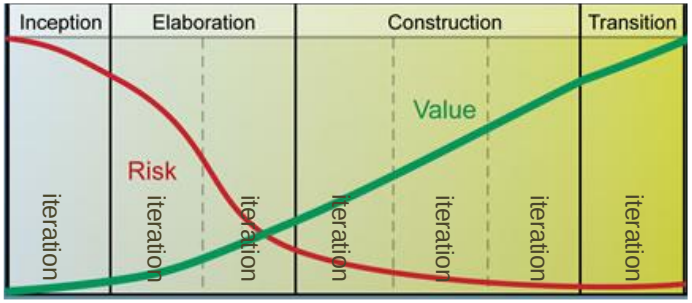
Define, implement and test interfaces of major components. Partially implement some key components.

Identify dependencies on external components and systems. Integrate shells/proxies of them.

Roughly 10% of code is implemented.

Drive architecture with key use cases

20% of use cases drive 80% of the architecture



Credit: Per Kroll (IBM)

A arquitetura do sistema aborda diferentes perspetivas de análise

❶ Arquitetura lógica do software

Organização geral dos blocos de software

Independente da tecnologia de implementação

❷Arquitetura de componentes do software

Peças construídas com uma tecnologia concreta

Construção “modular”

E.g.: existem pré-feitos?

❸ Arquitetura de instalação

Visão dos equipamentos e configuração de produção (conectividade, distribuição,...)

Arquitetura lógica

Organização geral da solução em blocos (*packages)*

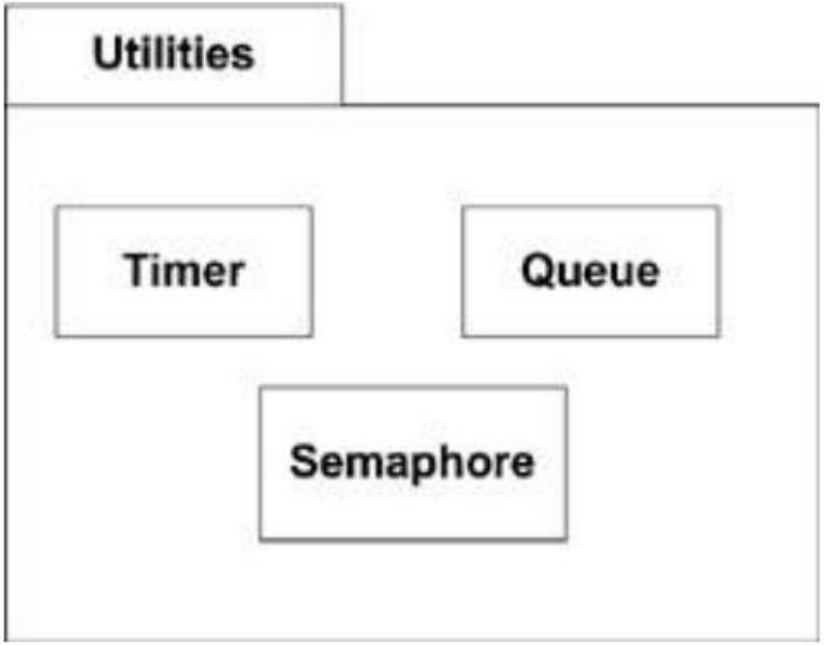
Os *packages* podem representar agrupamentos muito diferentes.

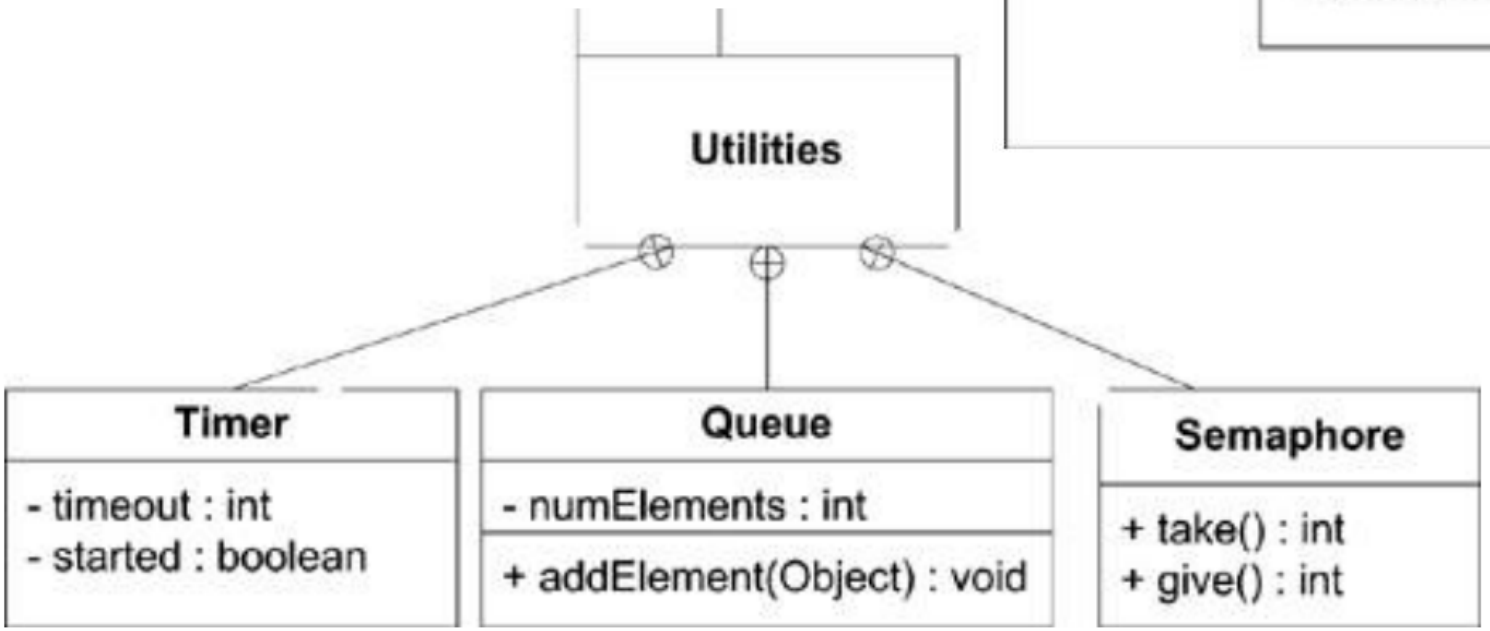
E.g.:

- *Packages* num programa em Java
- *Packages* num modelo UML
- Subsistemas/divisões do sistema sob especificação

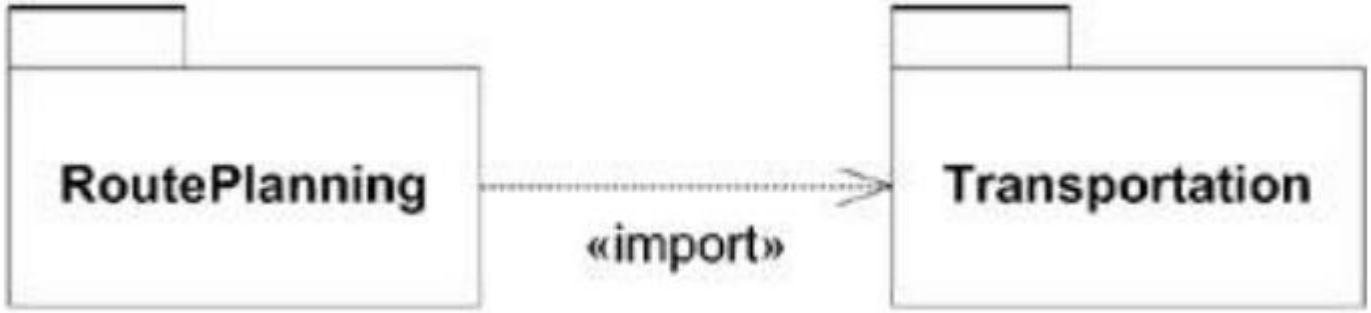
Comunicar a arquitetura lógica com pacotes (*packages)*

Classes contidas num pacote
- duas representações alternativas

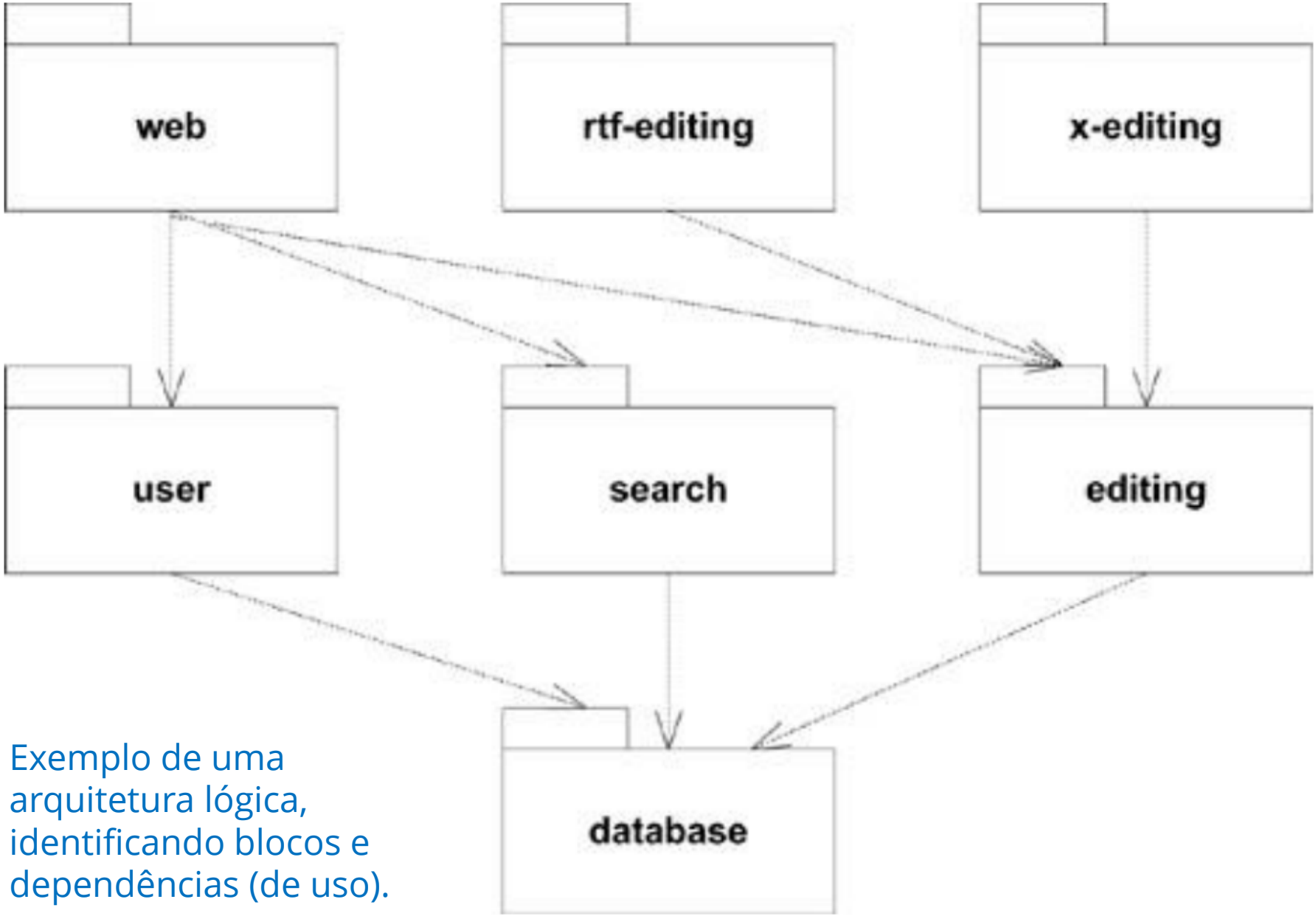




Associações entre pacotes

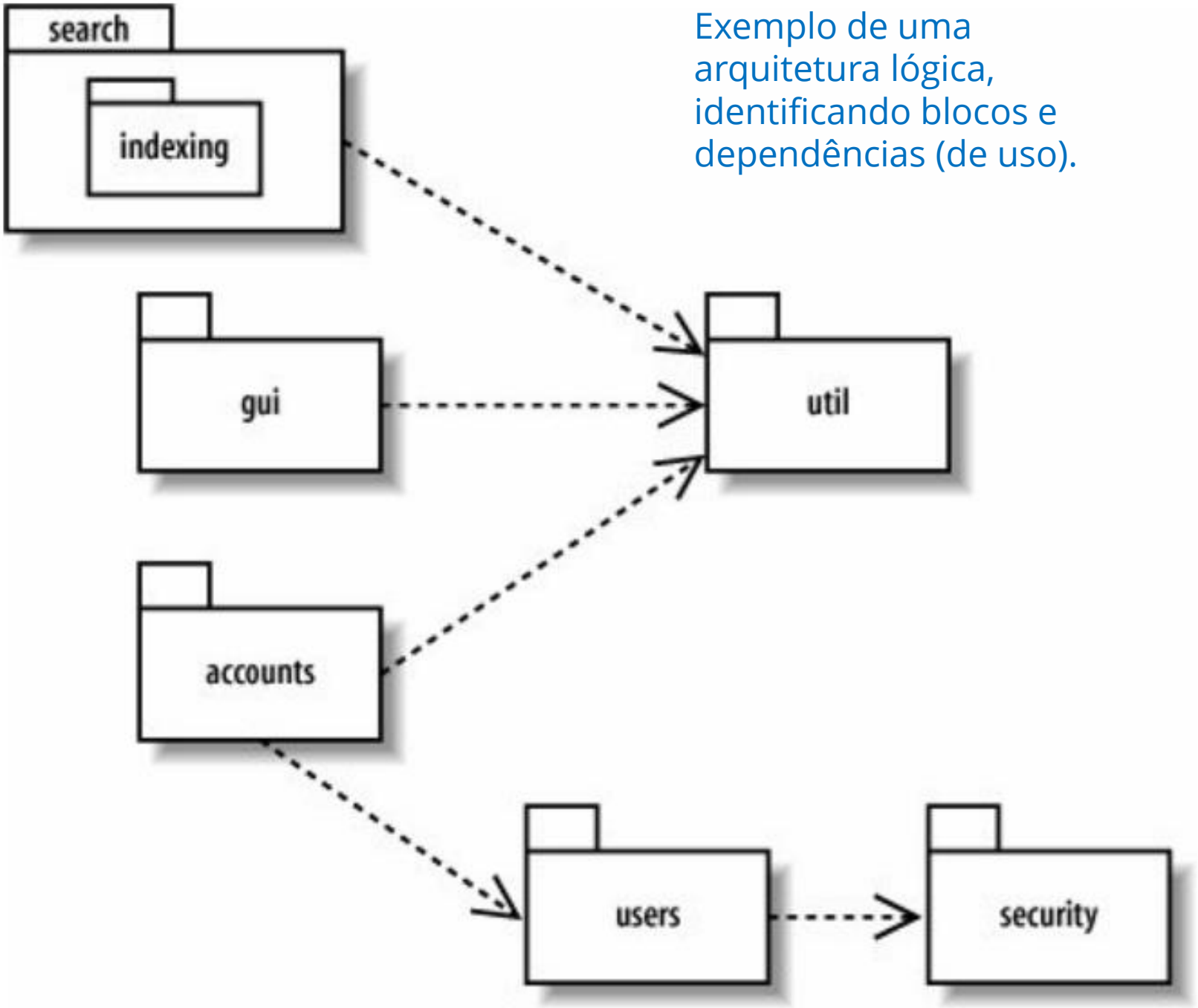


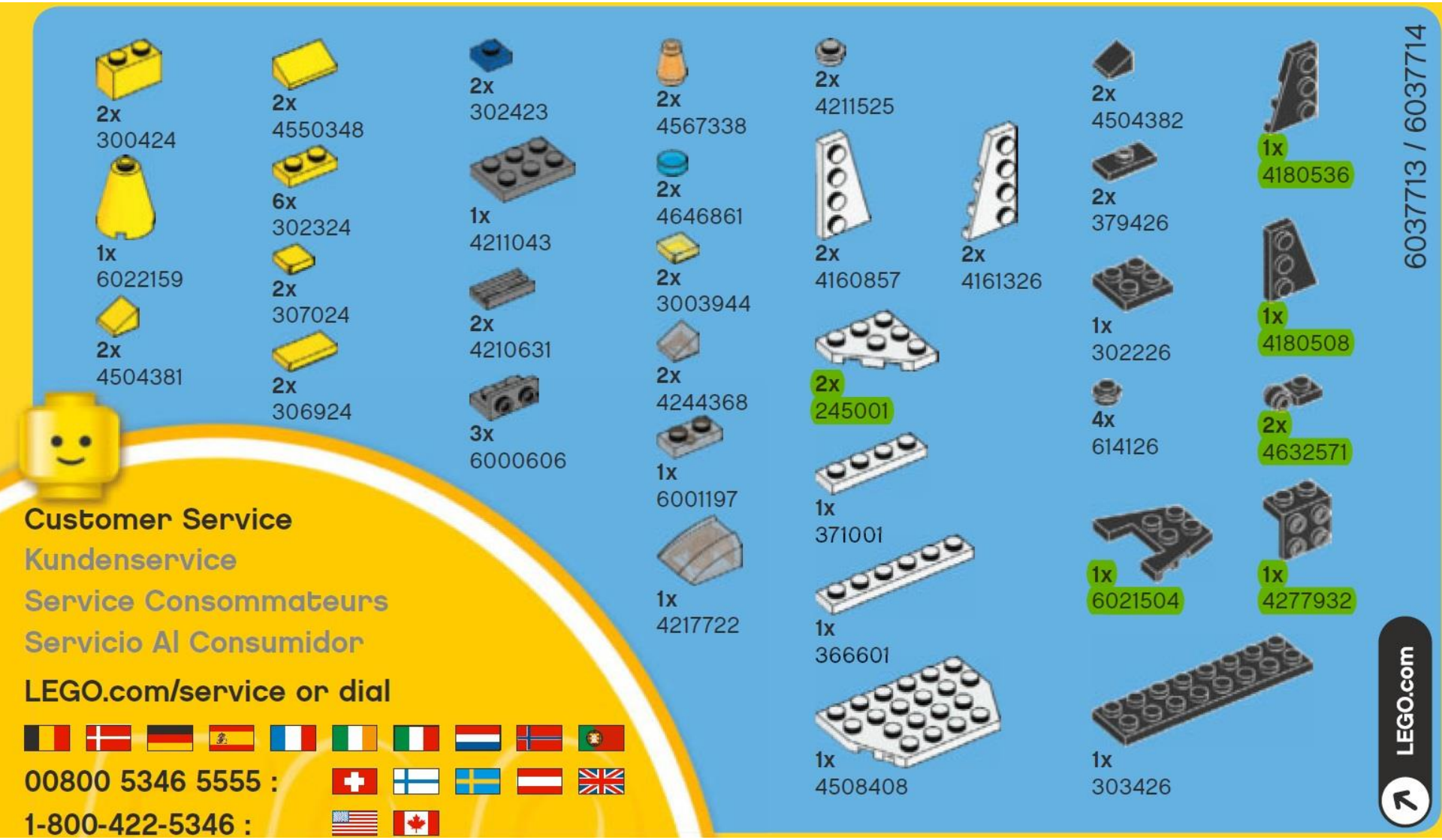
Dependência do tipo “import”



Exemplo de uma arquitetura lógica, identificando blocos e dependências (de uso).

Exemplo de uma arquitetura lógica, identificando blocos e dependências (de uso).




2x 300424
2x 4550348
6x 302324
2x 307024
2x 306924
1x 6022159
2x 4504381
2x 302423
1x 4211043
2x 4210631
3x 6000606
2x 4567338
2x 4646861
2x 3003944
2x 4244368
1x 6001197
1x 4217722
2x 4211525
2x 4160857
2x 4161326
2x 245001
1x 371001
1x 366601
1x 4508408
2x 4504382
2x 379426
1x 302226
4x 614126
1x 6021504
1x 303426
1x 4180536
1x 4180508
2x 4632571
1x 4277932
6037713 / 6037714
Customer Service
Kundenservice
Service Consommateurs
Servicio Al Consumidor
LEGO.com/service or dial
00800 5346 5555 :
1-800-422-5346 :
LEGO.com

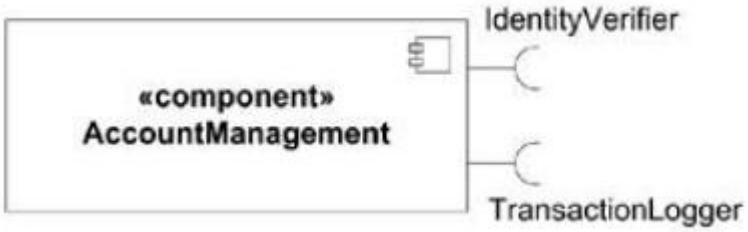
Componente

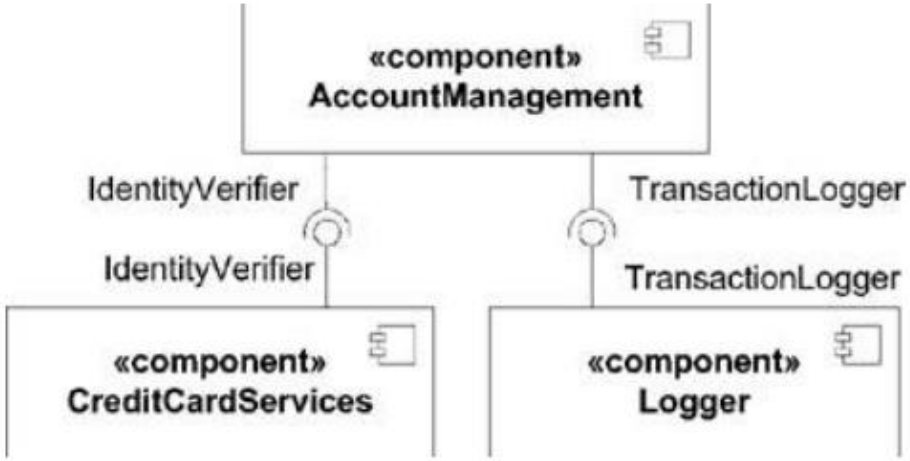
É normal dividir sistemas complexos em subsistemas mais geríveis

O componente é uma peça substituível, reusável de um sistema maior, cujos detalhes de implementação são abstraídos

A funcionalidade de um componente é descrita por um conjunto de interfaces fornecidos

Para além de implementar, o componente pode requerer funcionalidades de outros





A *component* is a self-contained, encapsulated piece of software that can be plugged into a system to provide a specific set of required functionalities. Today, there are many components available for purchase. A component has a well-defined *API* (application program interface). An API is essentially a set of method interfaces to the objects contained in the component. The internal workings of the component are hidden behind the API. Com-

Dennis, Alan, Barbara Wixom, David Tegarden. *Systems Analysis and Design: An Object Oriented Approach with UML, 5th Edition*. Wiley.

Arquitetura de componentes do software

Ao contrário do *package*, o componente é **uma peça tangível** da solução (e.g.: ficheiro, arquivo)

Os componentes são implementados com tecnologia concreta

Propriedades desejáveis:

Encapsulamento

> Garante uma certa funcionalidade e expõe uma interface/contrato

Reutilizável (em vários projetos)

Substituível

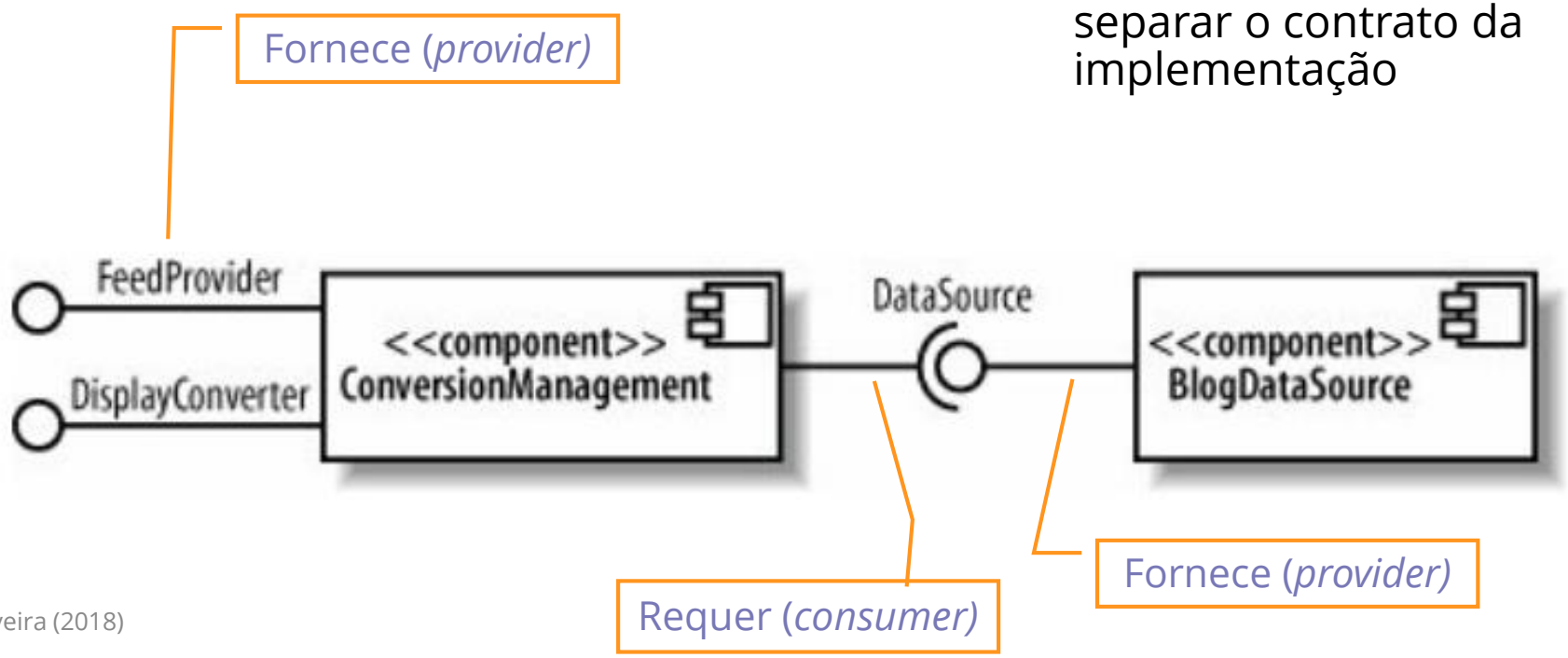
Candidatos naturais:

Aspetos recorrentes em vários projetos

Módulos que se podem obter pré-feitos ou disponibilizar

Módulos definidos para ir de encontro às regras dos ambientes de execução (e.g.: módulos para a*pplication servers*)

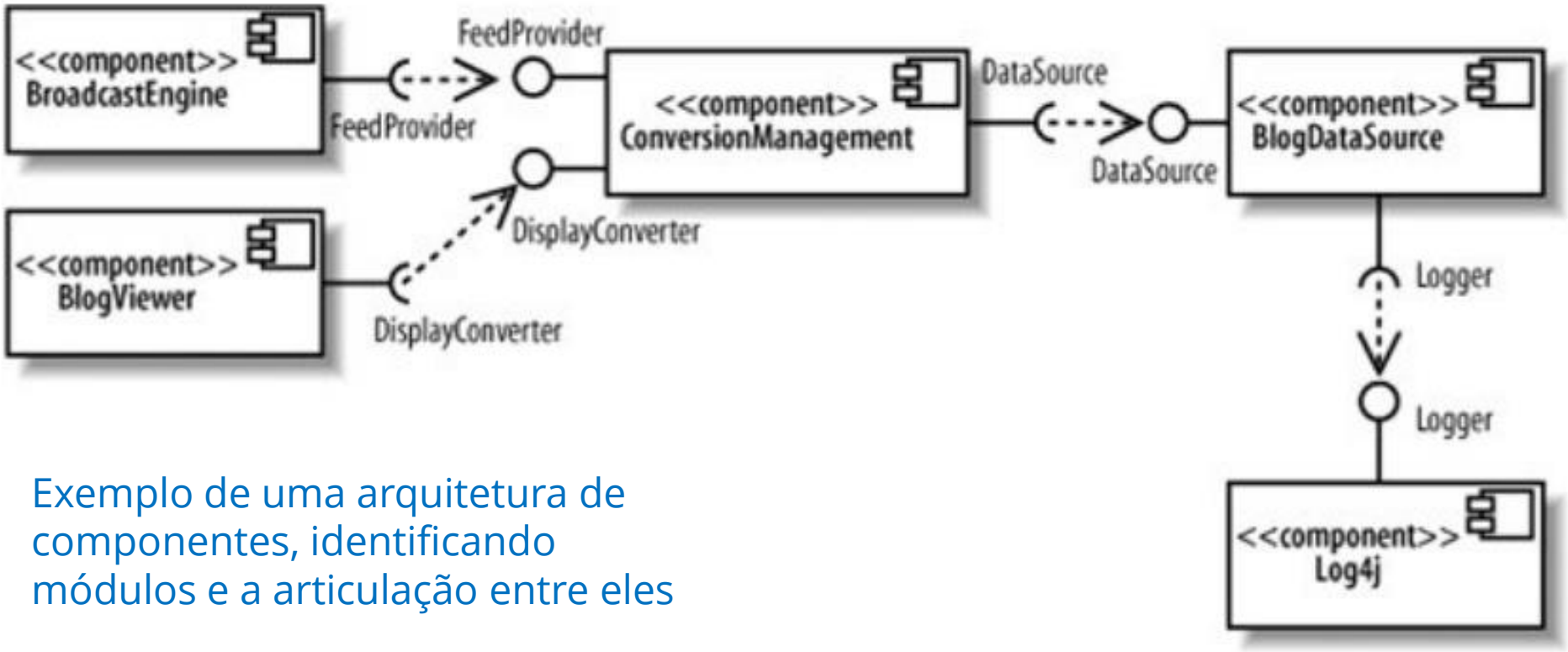
Figure



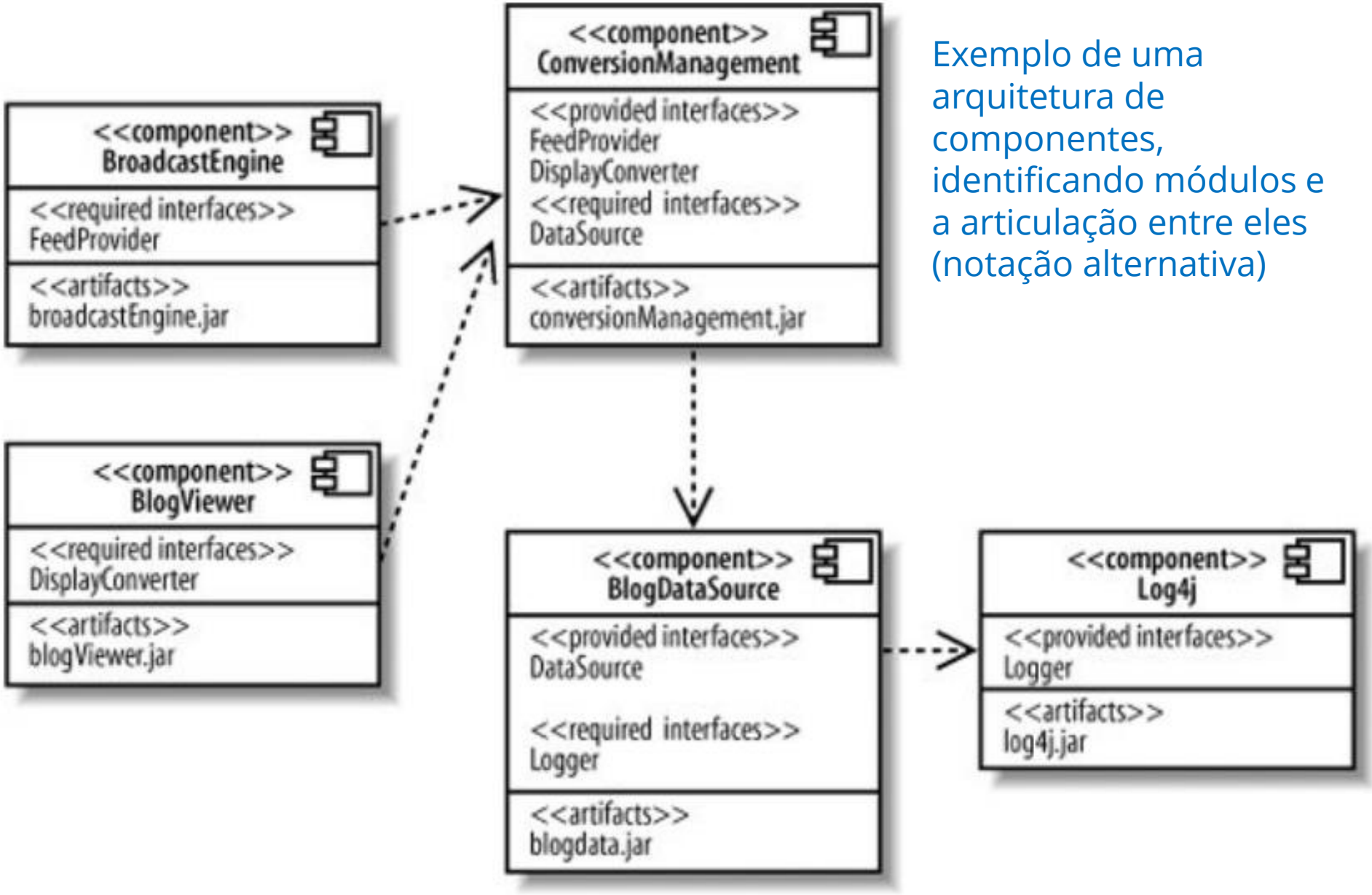
Solução modular com componentes

Com os componentes, pretende-se arquiteturas com baixo "coupling"

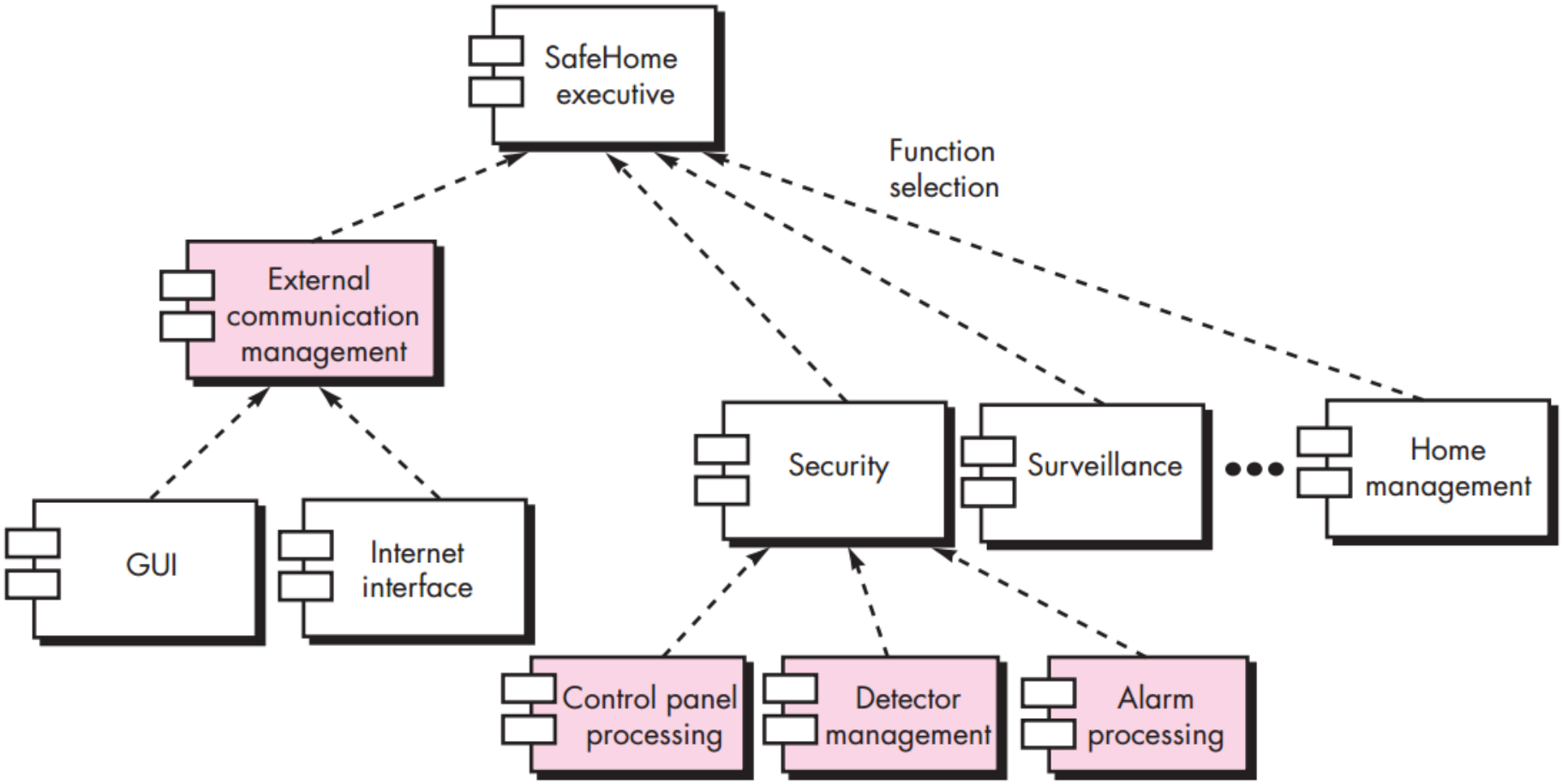
A exposição da funcionalidade através de interfaces ajuda a separar o contrato da implementação



Exemplo de uma arquitetura de componentes, identificando módulos e a articulação entre eles

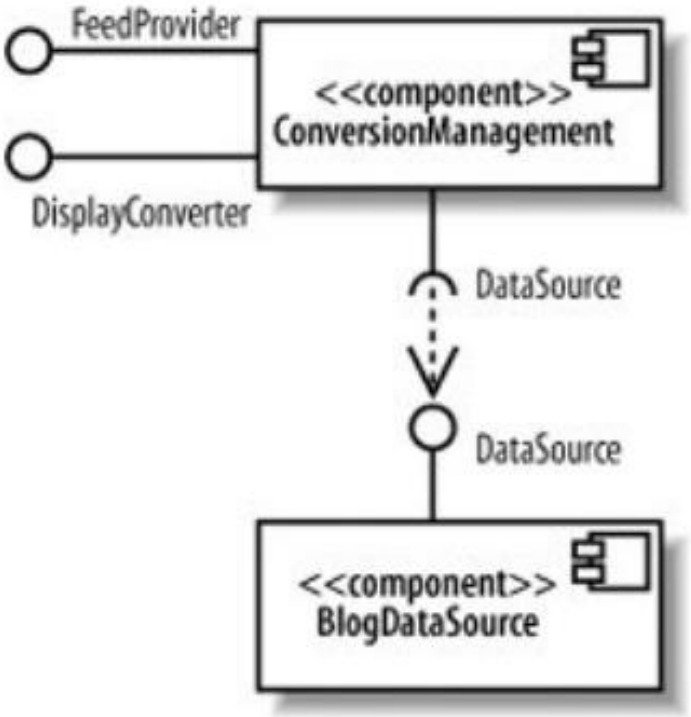


Exemplo de uma arquitetura de componentes, identificando módulos e a articulação entre eles (notação alternativa)

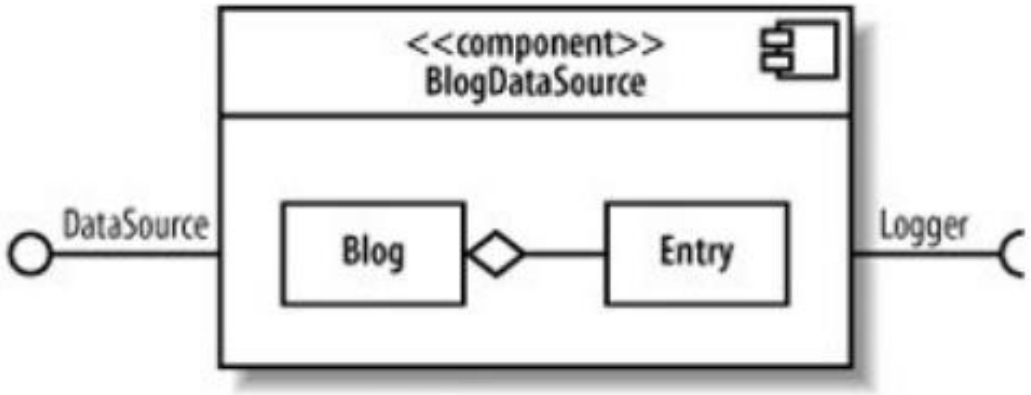

SafeHome executive
Function selection
External communication management
Security
Surveillance
Home management
GUI
Internet interface
Control panel processing
Detector management
Alarm processing

Notação “caixa fechada” / “caixa aberta”

Example Black-Box Component View



Example White-Box Component View



Arquitetura “física” de instalação

Explicita a configuração concreta do sistema, no ambiente de produção pretendido

Questões da arquitetura física:

Que servidores são necessários?
Onde se instala cada módulo?...

Diagramas de instalação da UML

Nós (*node*)

Um equipamento de hardware

Ambiente de execução

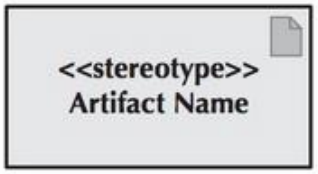
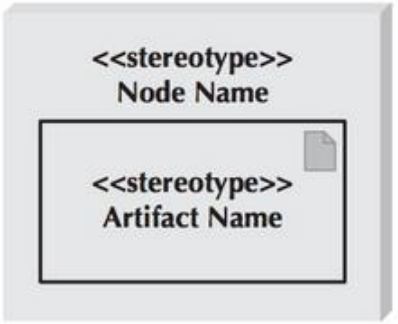
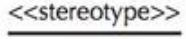
Um ambiente externo à solução que proporciona o contexto necessário à sua execução

E.g.: SO, servidor web, servidor aplicacional

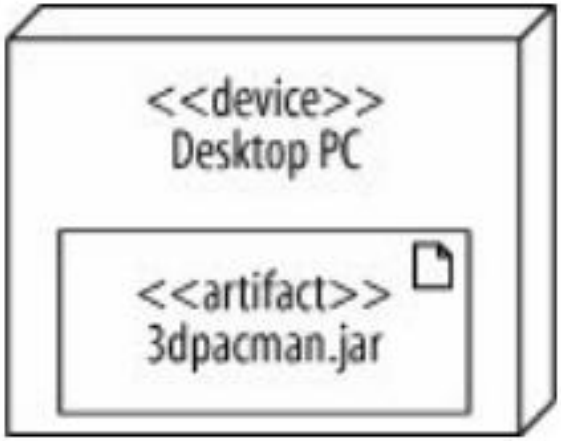
Artefactos (*artifact)*

Ficheiros concretos que são executados ou utilizados pela solução

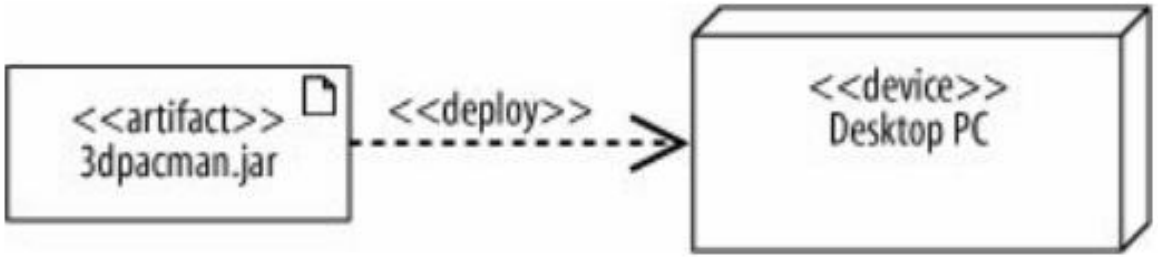
E.g.: executáveis, bibliotecas, configurações, scripts

A node: ■ Is a computational resource, e.g., a client computer, server, separate network, or individual network device. ■ Is labeled by its name. ■ May contain a stereotype to specifically label the type of node being represented, e.g., device, client workstation, application server, mobile device, etc.	
An artifact: ■ Is a specification of a piece of software or database, e.g., a database or a table or view of a database, a software component or layer. ■ Is labeled by its name. ■ May contain a stereotype to specifically label the type of artifact, e.g., source file, database table, executable file, etc.	
A node with a deployed artifact: ■ Portrays an artifact being placed on a physical node.	
A communication path: ■ Represents an association between two nodes. ■ Allows nodes to exchange messages. ■ May contain a stereotype to specifically label the type of communication path being represented, (e.g., LAN, Internet, serial, parallel).	

Os artefactos são executados em nós

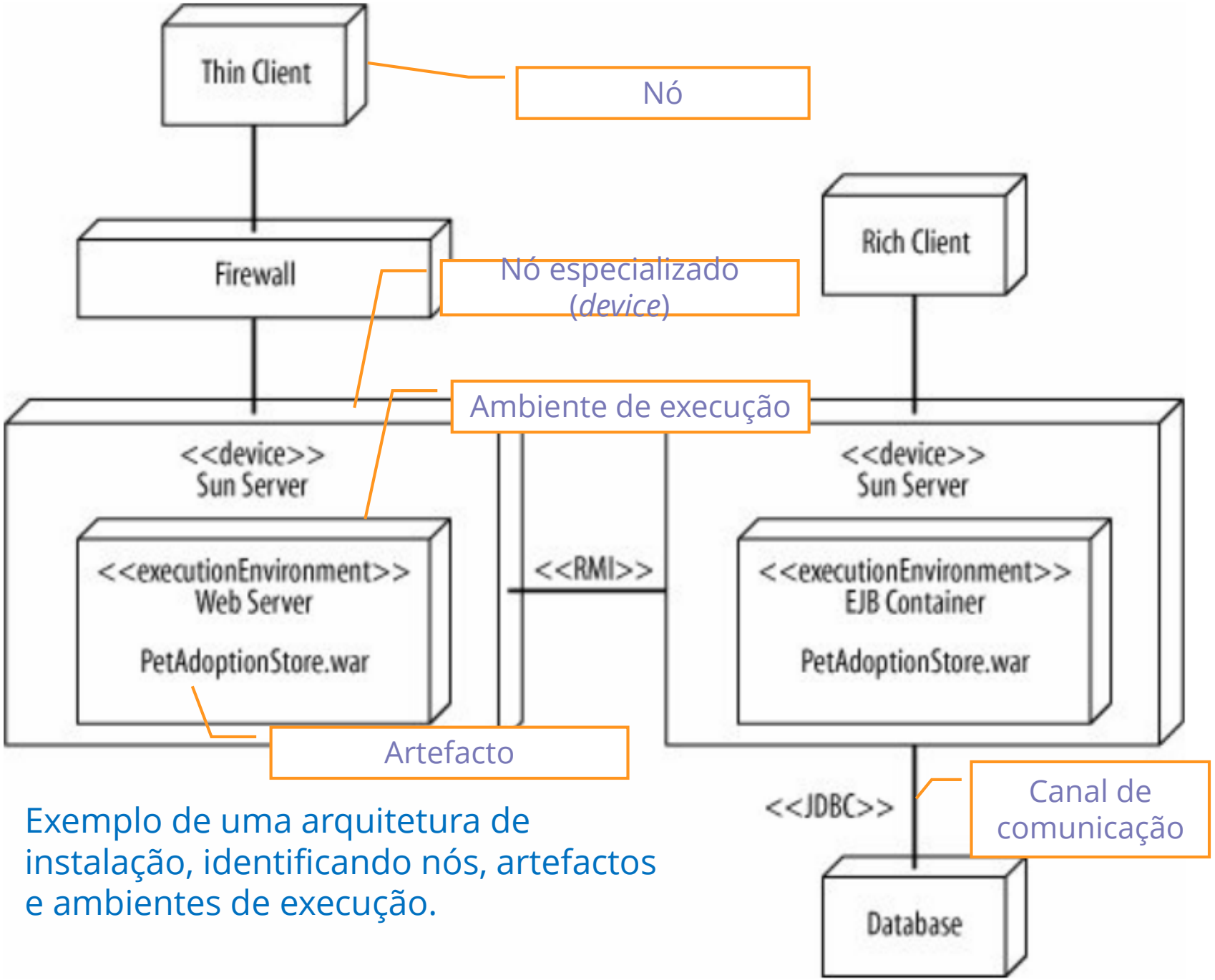


Notações alternativas.

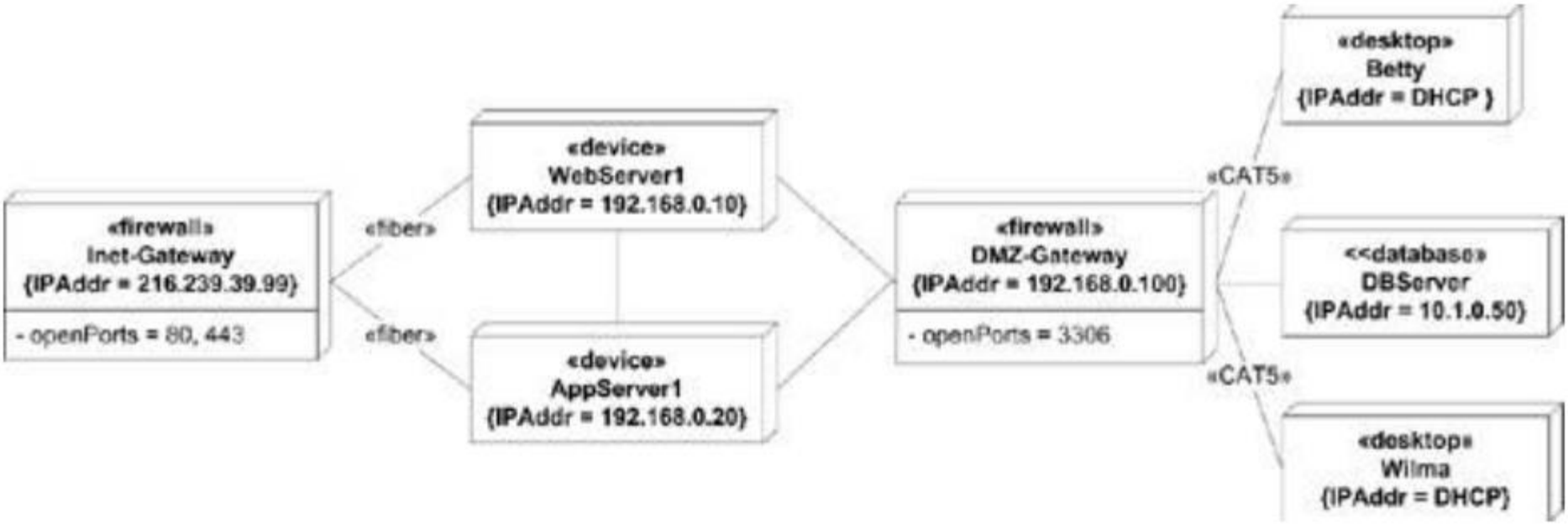


Rastreabilidade até aos componentes

A relação “manifest” permite associar componentes a artefactos.



Exemplo de uma arquitetura de instalação, identificando nós, artefactos e ambientes de execução.

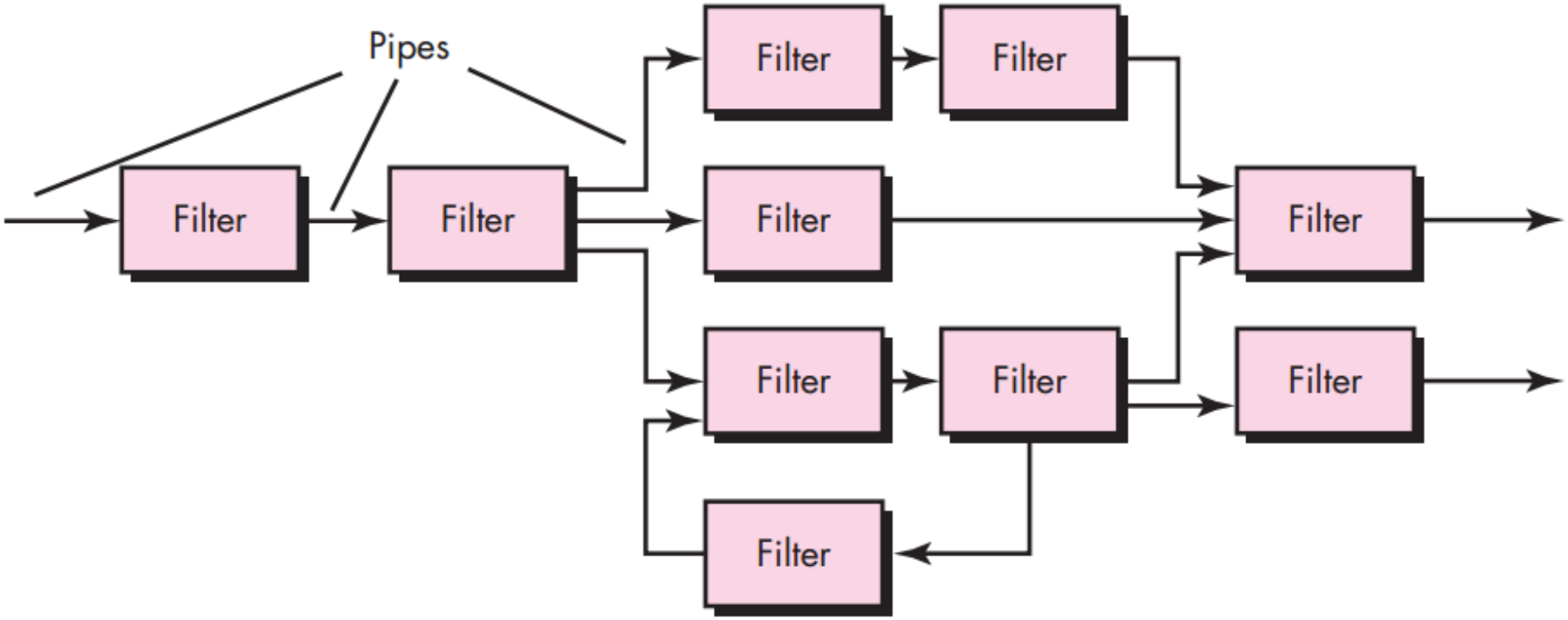


Exemplo de uma arquitetura de instalação para descrever uma topologia de rede.

Estilos de arquitetura

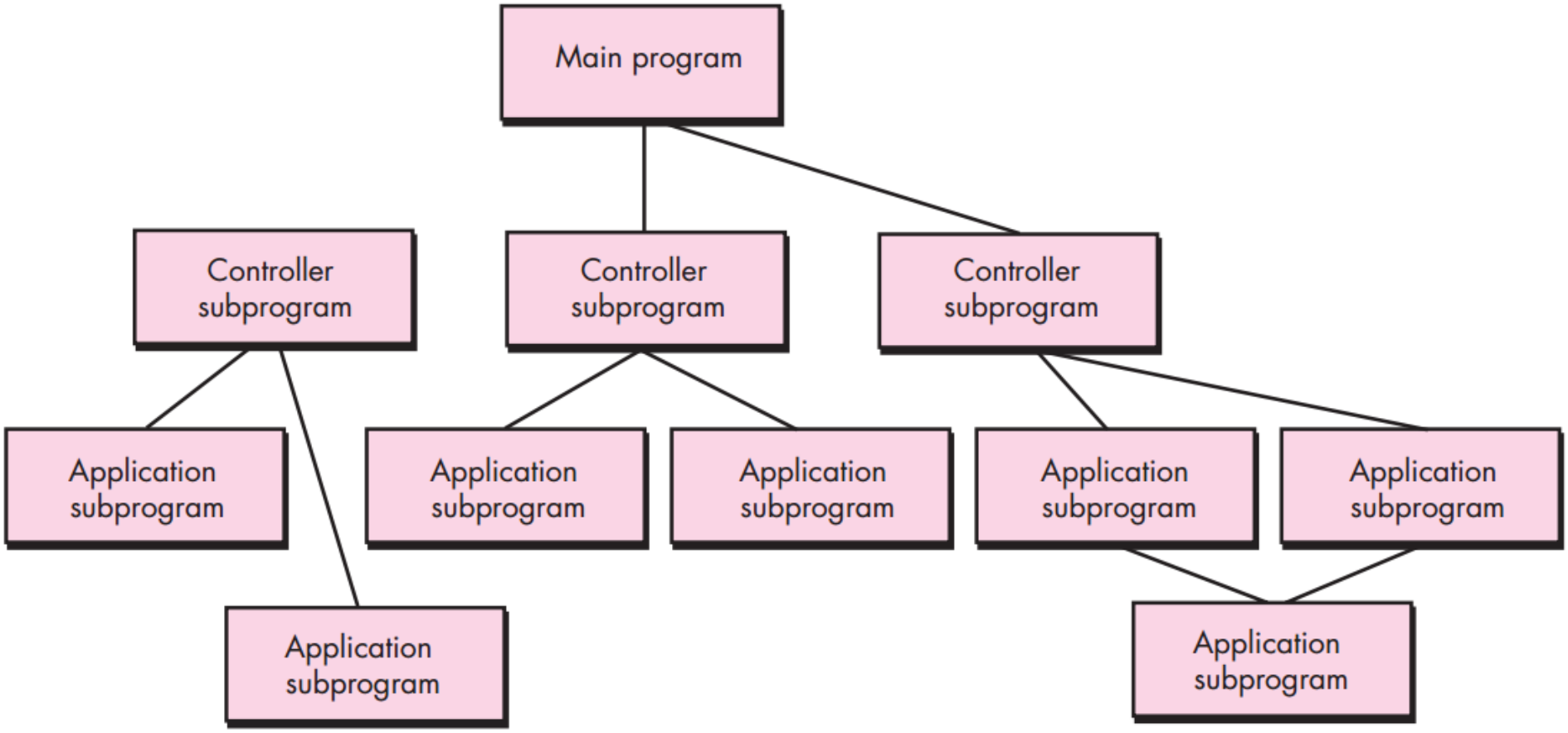
Apesar do vastíssimo número de soluções de software, é possível identificar um conjunto de padrões para a arquitetura do software

Data-flow

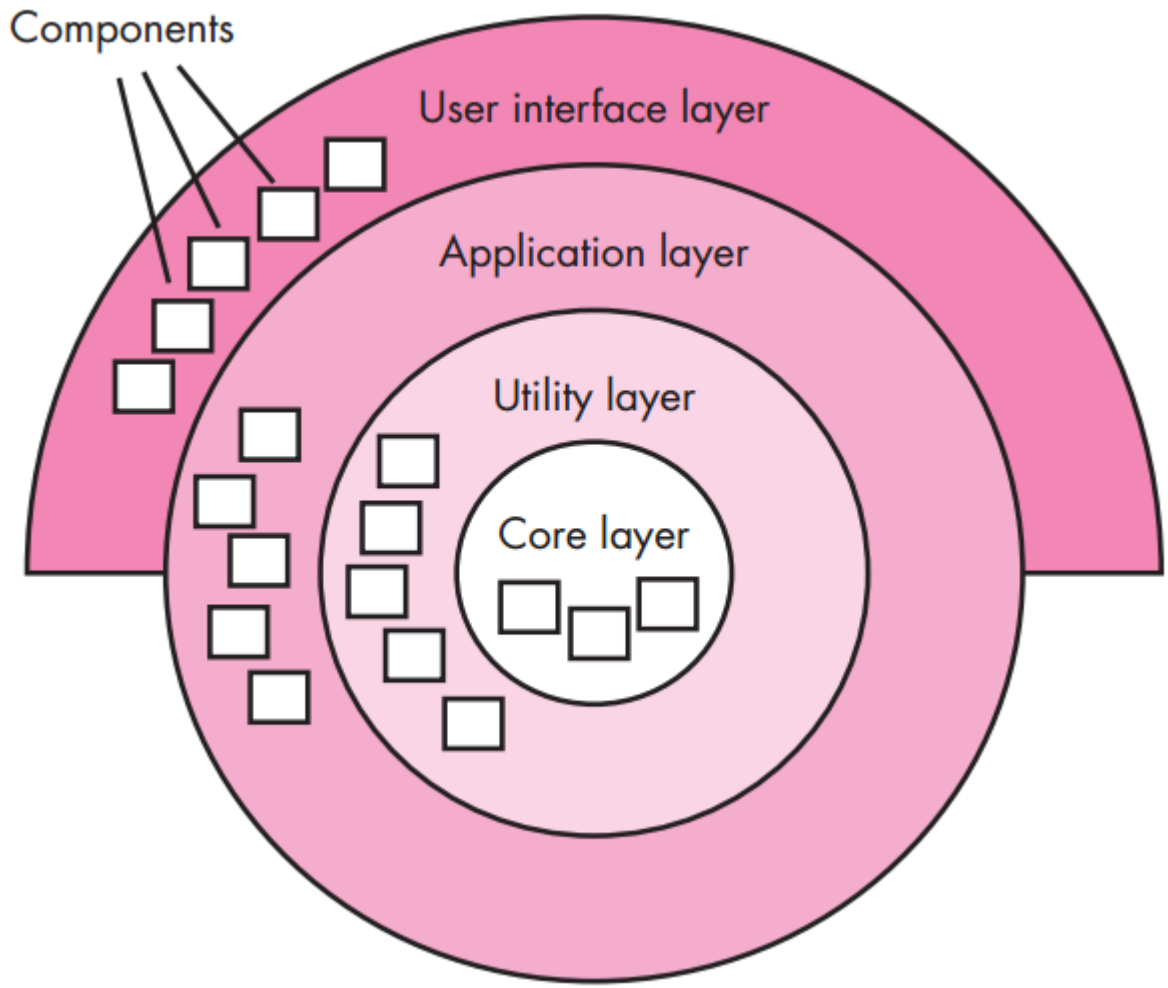


Pipes and filters

Programa principal/subprogramas



Arquitetura por camadas



Arquitetura por camadas

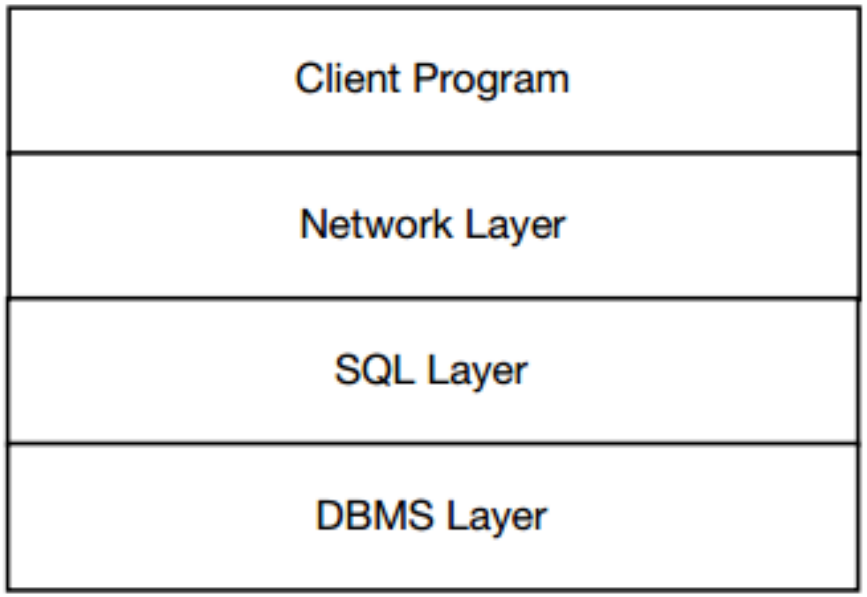
Divisão modular da solução de software em camadas/níveis

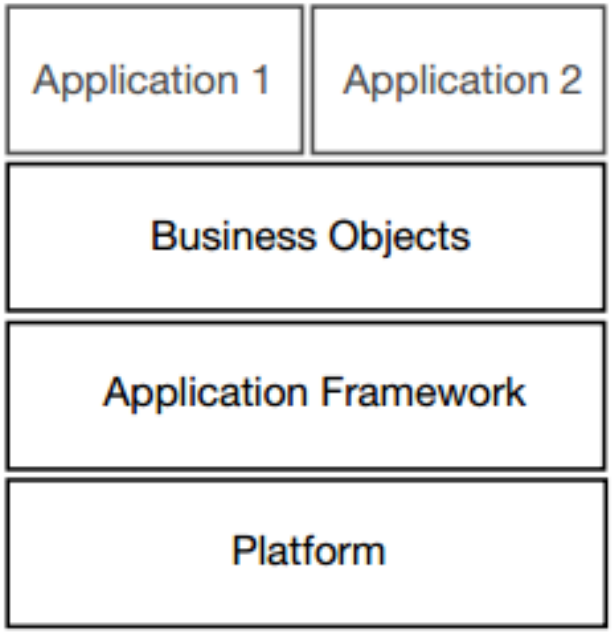
As camadas são sobrepostas

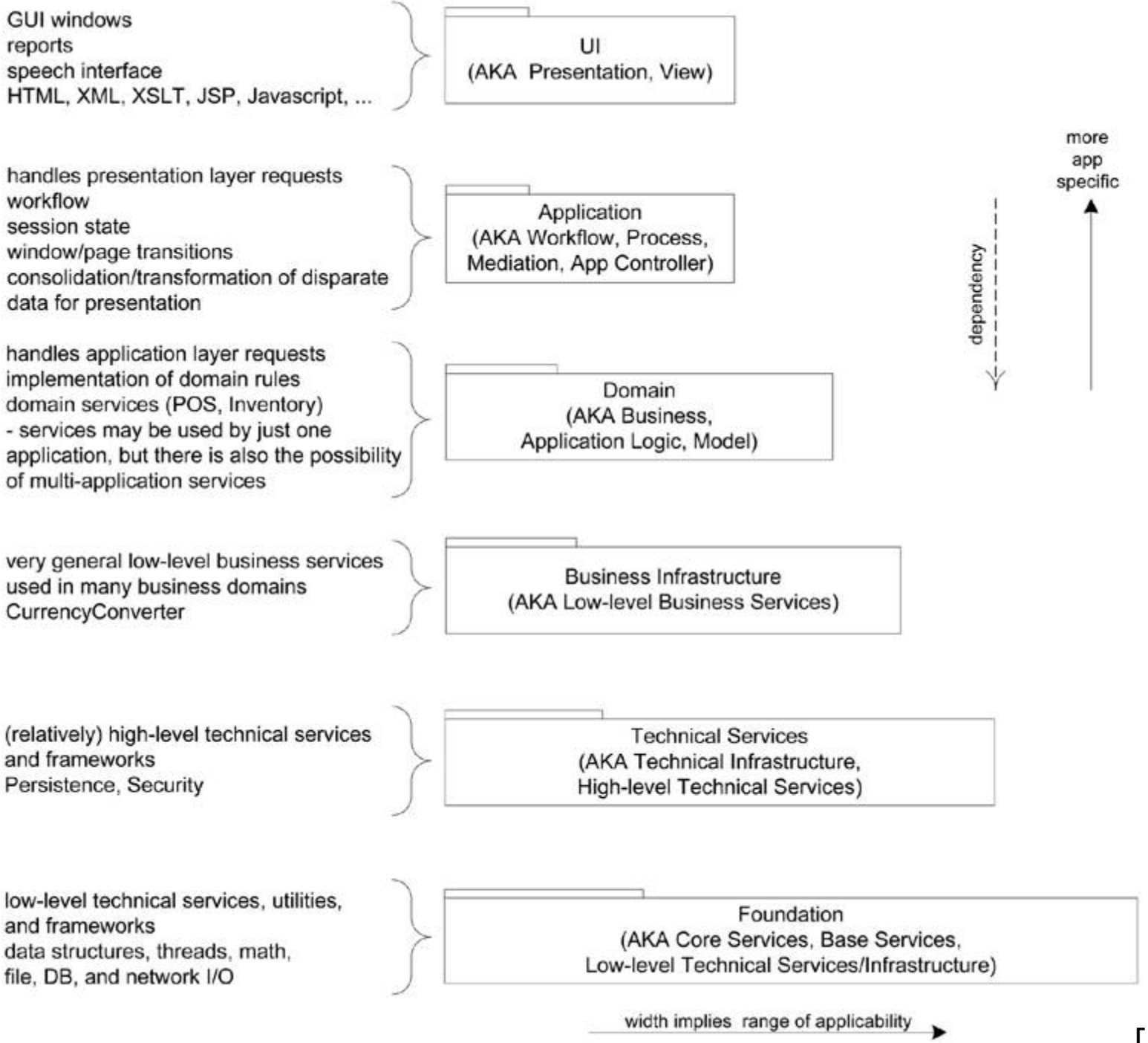
Cada camada tem uma especialização

Camadas “em cima” pedem serviços às camadas “de baixo”

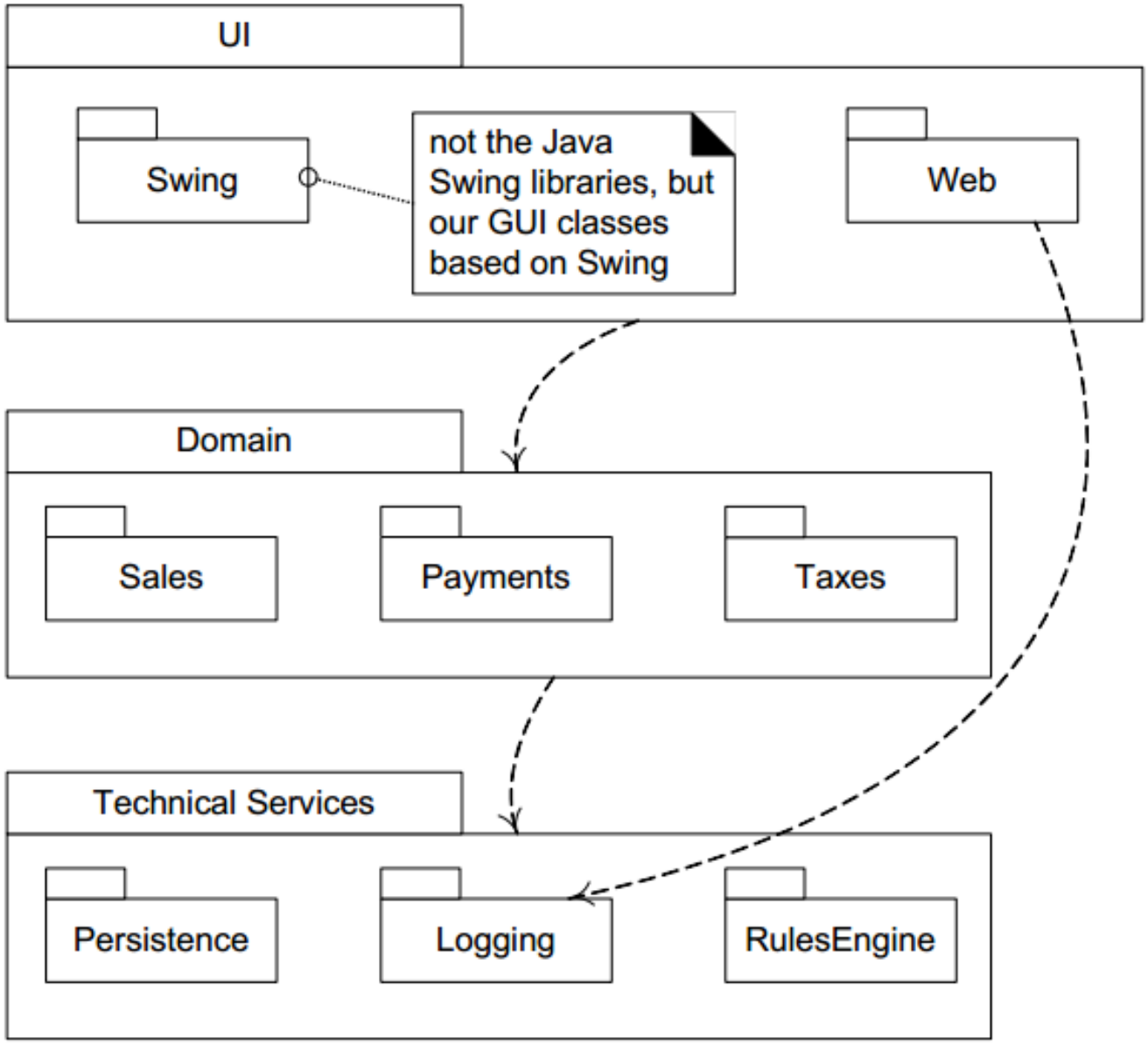
Não se pode saltar camadas.


Client Program
Network Layer
SQL Layer
DBMS Layer

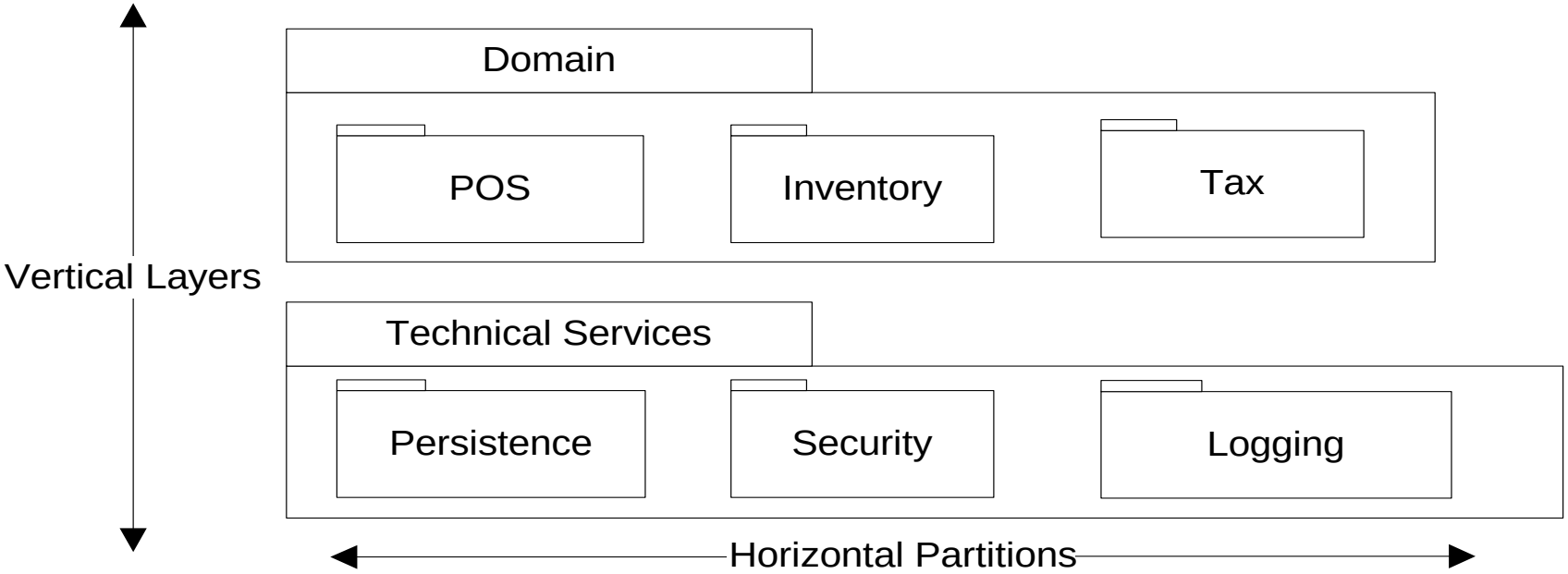

Application 1
Application 2
Business Objects
Application Framework
Platform

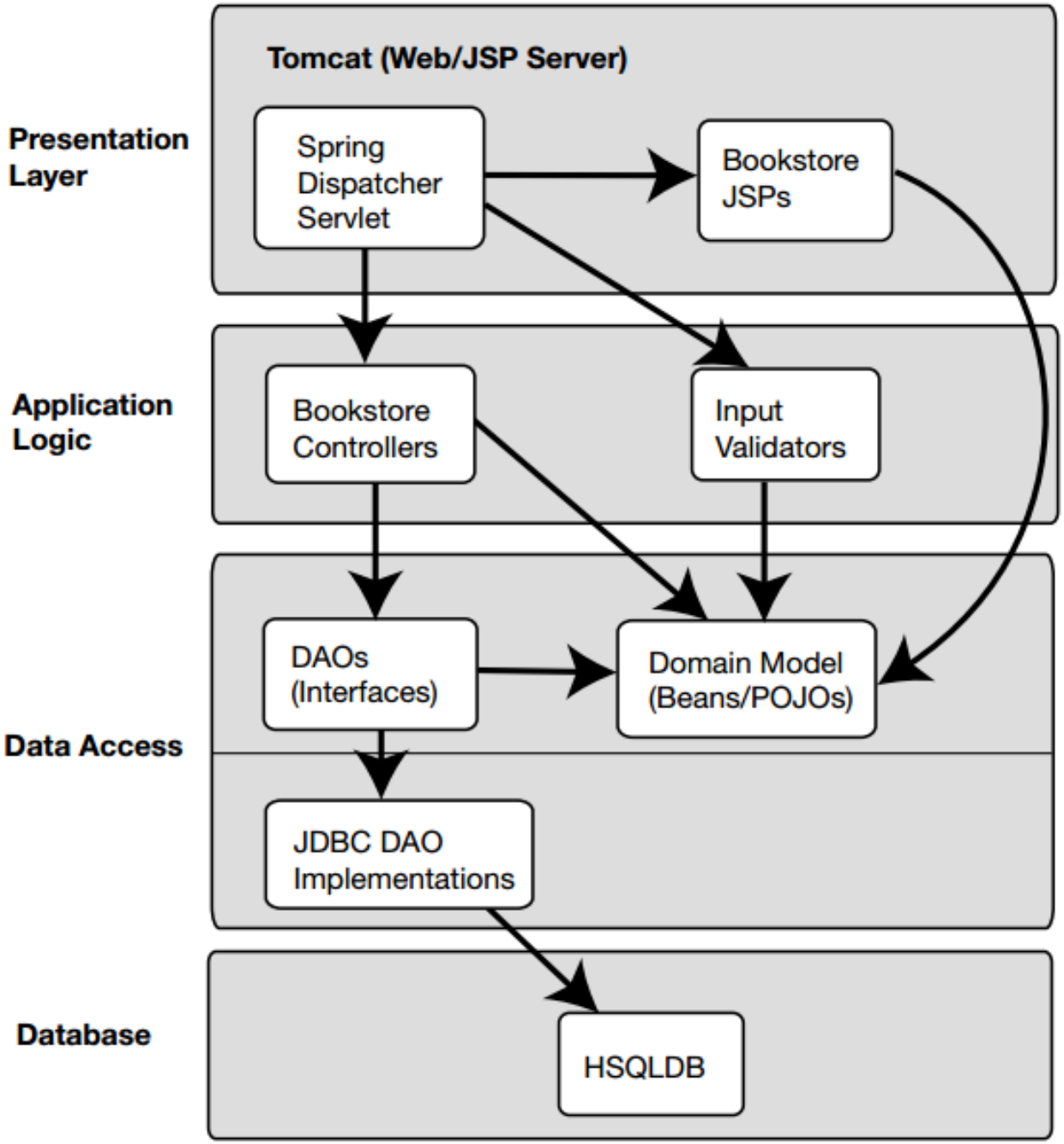

GUI windows
reports
speech interface
HTML, XML, XSLT, JSP, Javascript, ...
UI
(AKA Presentation, View)
handles presentation layer requests
workflow
session state
window/page transitions
consolidation/transformation of disparate data for presentation
Application
(AKA Workflow, Process, Mediation, App Controller)
more app specific
dependency
handles application layer requests
implementation of domain rules
domain services (POS, Inventory)
- services may be used by just one application, but there is also the possibility of multi-application services
Domain
(AKA Business, Application Logic, Model)
very general low-level business services
used in many business domains
CurrencyConverter
Business Infrastructure
(AKA Low-level Business Services)
(relatively) high-level technical services and frameworks
Persistence, Security
Technical Services
(AKA Technical Infrastructure, High-level Technical Services)
low-level technical services, utilities, and frameworks
data structures, threads, math, file, DB, and network I/O
Foundation
(AKA Core Services, Base Services, Low-level Technical Services/Infrastructure)
width implies range of applicability

[LAR'12]


UI
Swing
not the Java Swing libraries, but our GUI classes based on Swing
Web
Domain
Sales
Payments
Taxes
Technical Services
Persistence
Logging
RulesEngine

Layers and partitions




Tomcat (Web/JSP Server)
Presentation Layer
Spring Dispatcher Servlet
Bookstore JSPs
Application Logic
Bookstore Controllers
Input Validators
Data Access
DAOs (Interfaces)
Domain Model (Beans/POJOs)
JDBC DAO Implementations
Database
HSQLDB

Referências

[DEN'15] Dennis, A., Wixom, B. H., & Tegarden, D. (2015). *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons.

→ chap. 7; 11.

[LAR'12] Larman, C. (2012). *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Interative Development*. Pearson Education.

→ chap. 13, 36, 37.