40431: Modelação e Análise de Sistemas

Arquitetura Evolutiva

Ilídio Oliveira

v2020-01-08 | TP10b (disponível em videoaula)

universidade
departamento de e
telecomunicações e in

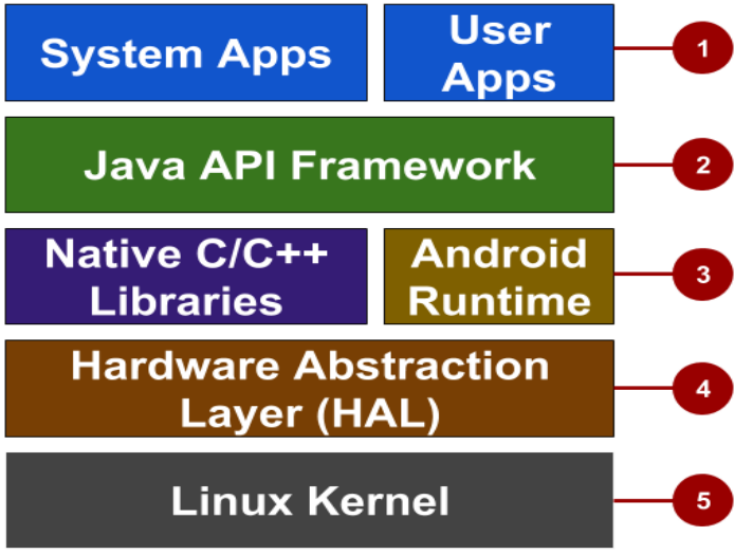
Objetivos de aprendizagem

- Explicar as atividades do desenvolvimento associadas à arquitetura de software
- Definir a prática de Arquitetura Evolutiva, conforme especificado no OpenUP
- Identificar os elementos abstratos de uma arquitetura de software
- Identificar requisitos de arquitetura significativos num domínio e relacionar com atributos de qualidade
- Descreva os conceitos de camadas e partições (numa arquitetura em camadas)
- Construir um diagrama de pacotes para ilustrar uma arquitetura lógica
- Interpretar um diagrama de componentes para descrever as partes tangíveis do software
- Construir um diagrama de instalação para descrever a configuração de um sistema

Android *stack* (visão geral)

Android stack

1. System and user apps
2. Android OS API in Java framework
3. Expose native APIs; run apps
4. Expose device hardware capabilities
5. Linux Kernel

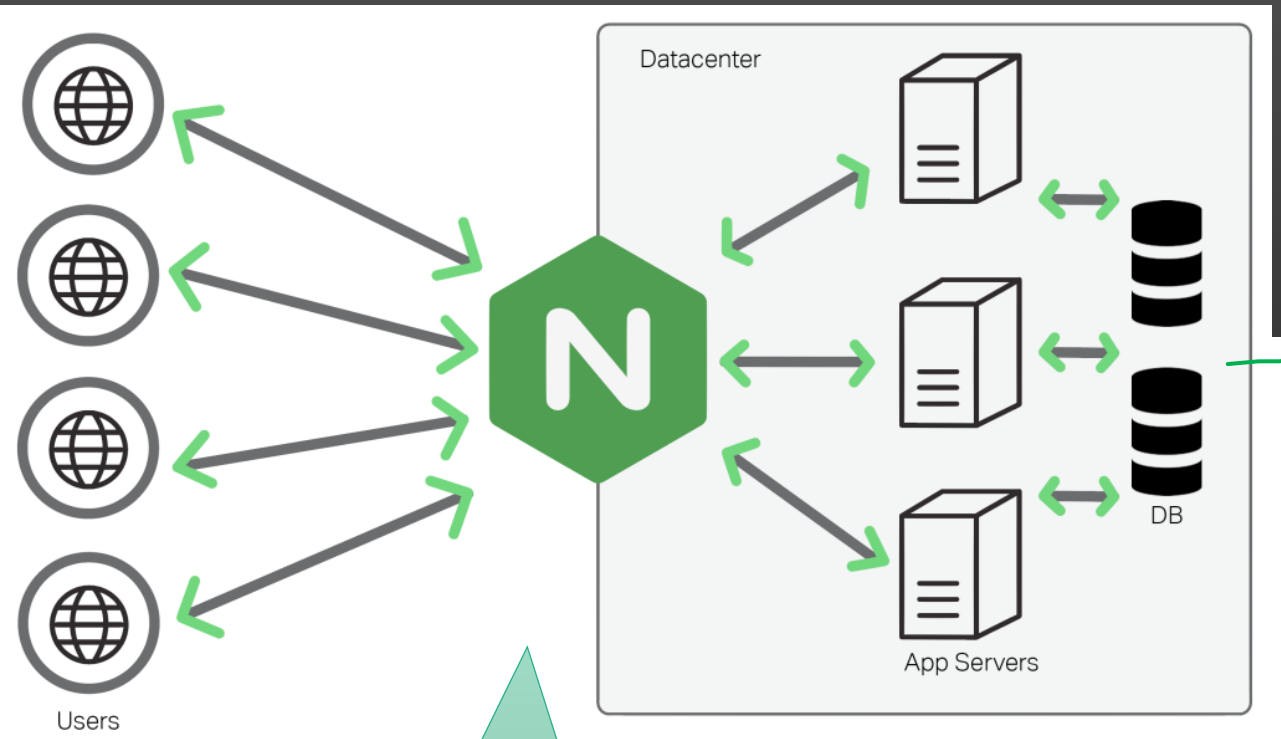


Um exemplo de arquitetura: a organização do sistema Android.

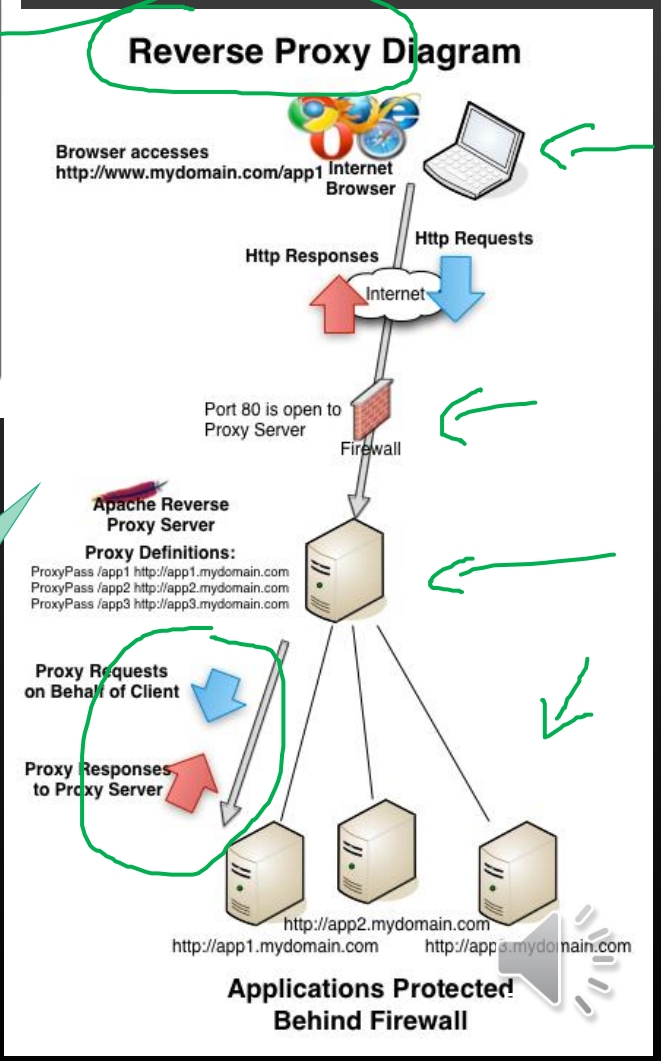
Android *stack*

Módulos em cada camada (partições).

I Oliveira



Componentes organizados em 4 *tiers* numa solução de distribuição de carga (com Nginx).



Organização dos serviços de rede numa configuração de “reverse proxy”

Elementos comuns

O que é que as “ilustrações” anteriores têm em comum?

→ Explicar a organização de uma solução, em termos dos seus “grandes” componentes *(“high-level”)*

→ Mostrar as principais linhas de dependência (colaboração/comunicação) entre os módulos

→ Equilíbrio entre “Caixa aberta” (ver para dentro da solução) e “Caixa fechada” (sem mostrar a organização interna dos módulos)

Mas também há diferenças:

→ Vista “lógica” (não mostra instalação) vs. vista de “sistema” (onde é que correm os componentes)

Concept: Software Architecture

The software architecture represents the structure or structures of the system, which consists of software components, the externally visible properties of those components, a[illegible] among them.

Relationships

Related Elements	• Architecture Notebook • Design • Executable Architecture • How to adopt the Evolutionary A[illegible]

A arquitetura não é uma atividade separada do desenvolvimento / implementação.
Trata as grandes decisões/estratégias para a implementação.

Main Description

Introduction

Software architecture is a concept that is easy to understand, and that most engineers intuitively feel, especially with little experience, but it is hard to define precisely. In particular, it is difficult to draw a sharp line between design and architecture-architecture is one aspect of design that concentrates on some specific features.

In An Introduction to Software Architecture, David Garlan and Mary Shaw suggest that software architecture is a level of design concerned with issues: "Beyond the algorithms and data structures of the computation: designing and

Software architecture

Plano (decisões) com a estratégia para a implementação.

An architecture is **the set of significant decisions about the organization of a software system**

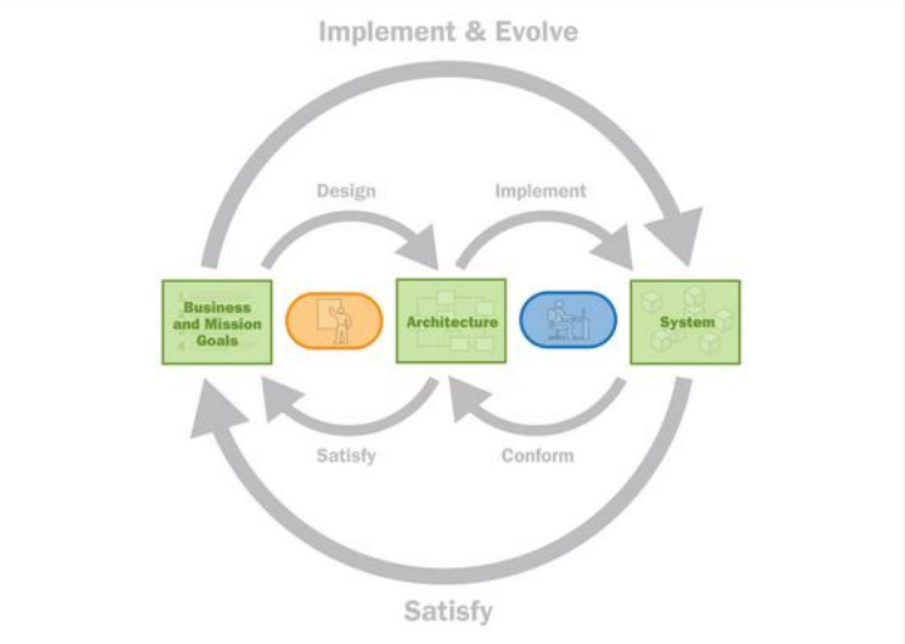
...the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural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

→ Big ideas on system organization and interactions

Why Architecture?

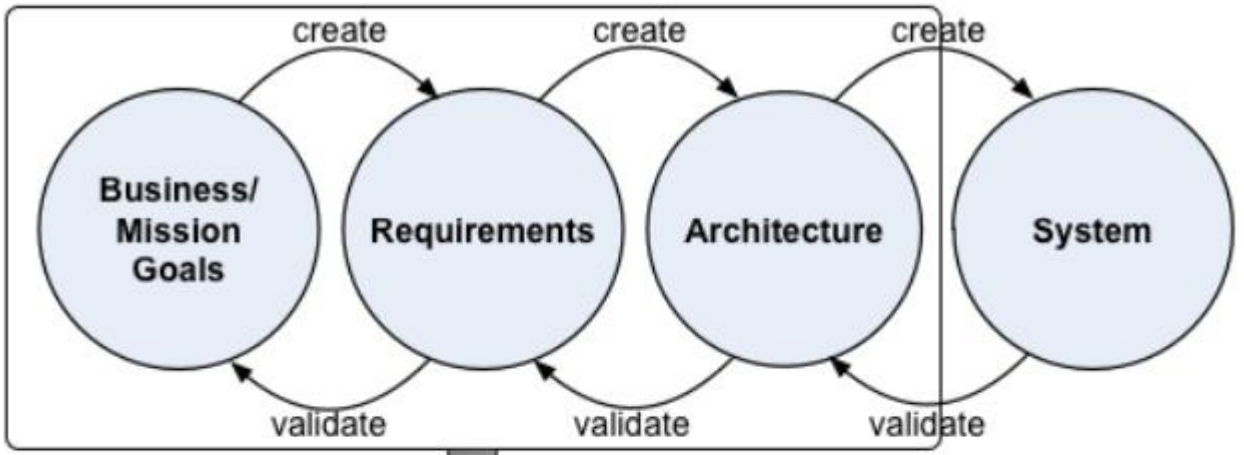
The software architecture of a program or computing system is a depiction of the system that aids in understanding how the system will behave.

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.



work/display.cfm?customel_datapageid_4050=21328

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

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?

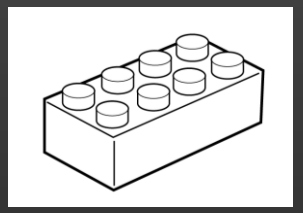
Compromissos e decisões!

O que compõe uma “arquitetura”?

Elementos

Abstrações/blocos usados na construção do sistema

Os aspetos privados de um elemento são o assunto do desenho e implementação

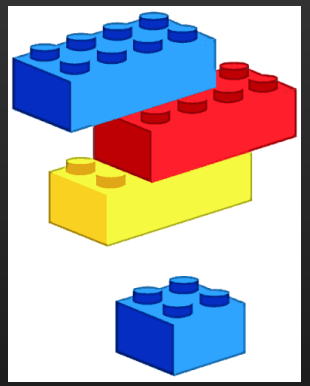


Interações de elementos

interagir através de interfaces públicas

Relacionamentos normalmente são concretizados com interfaces

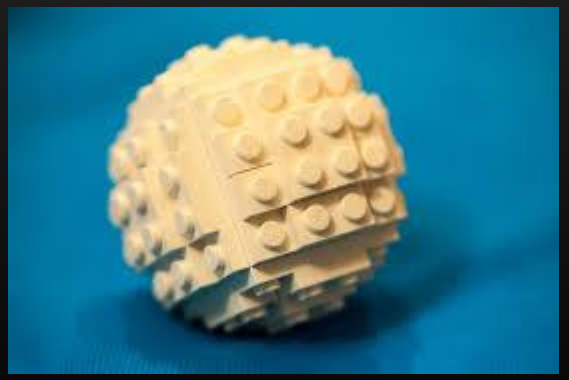
Arquitetura trata as interfaces públicas



Estruturas

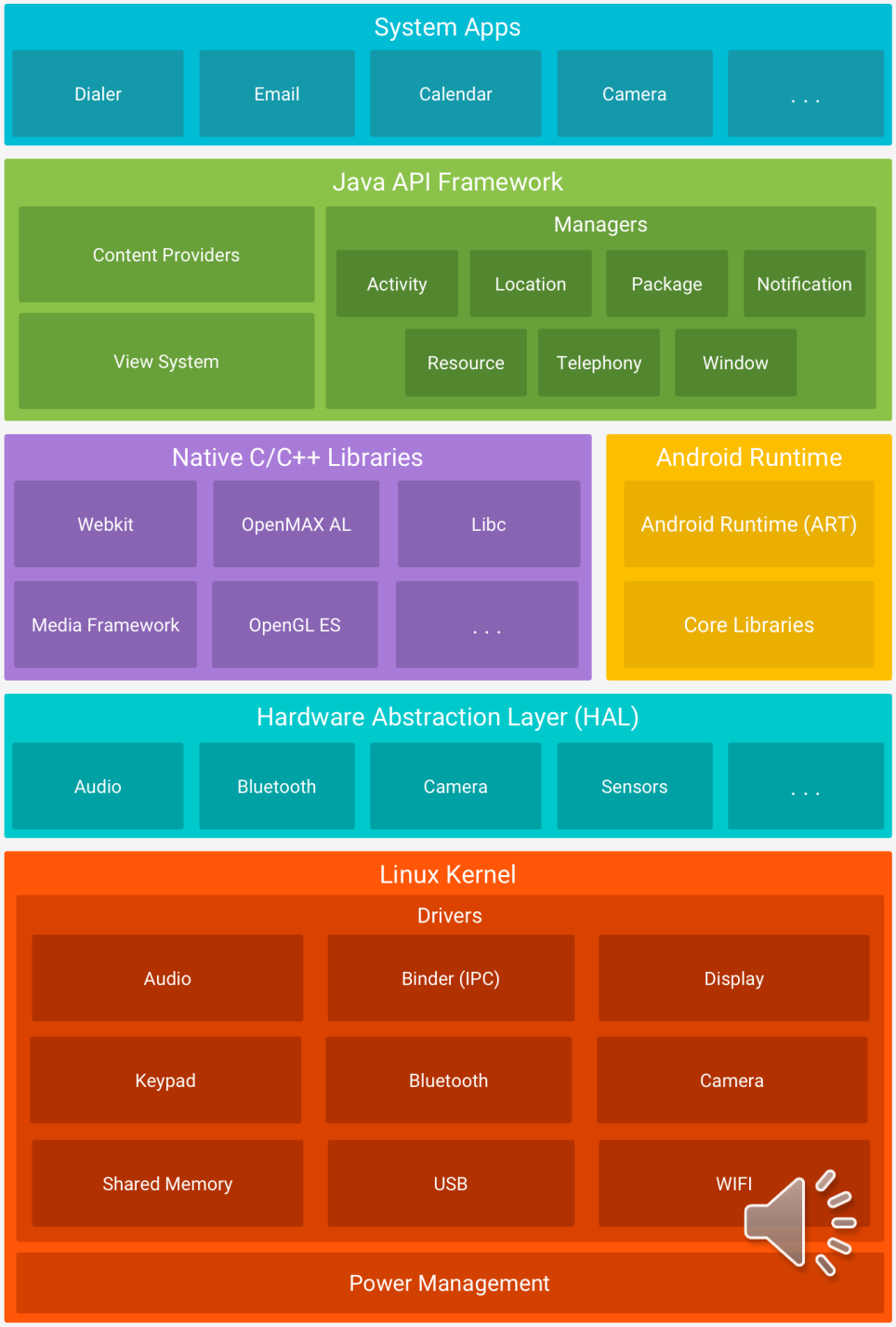
Defina a partição estática e atribuição de funcionalidade

Suporta a atribuição de comportamentos



http://www.columbia.edu/cu/gsapp/BT/BSI/SUSPENSI/suspensi.html

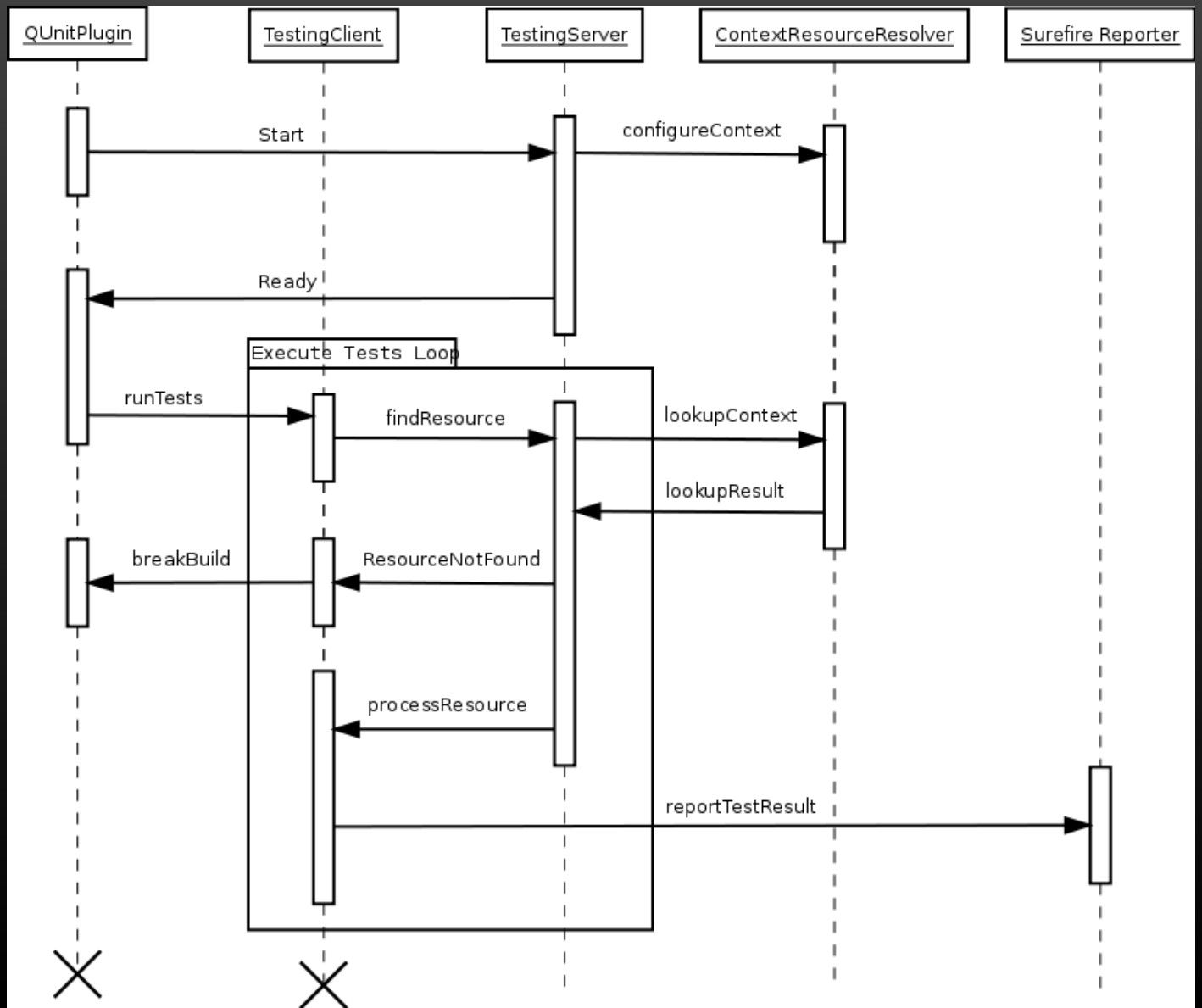
Vista estrutural (as partes constituntes)



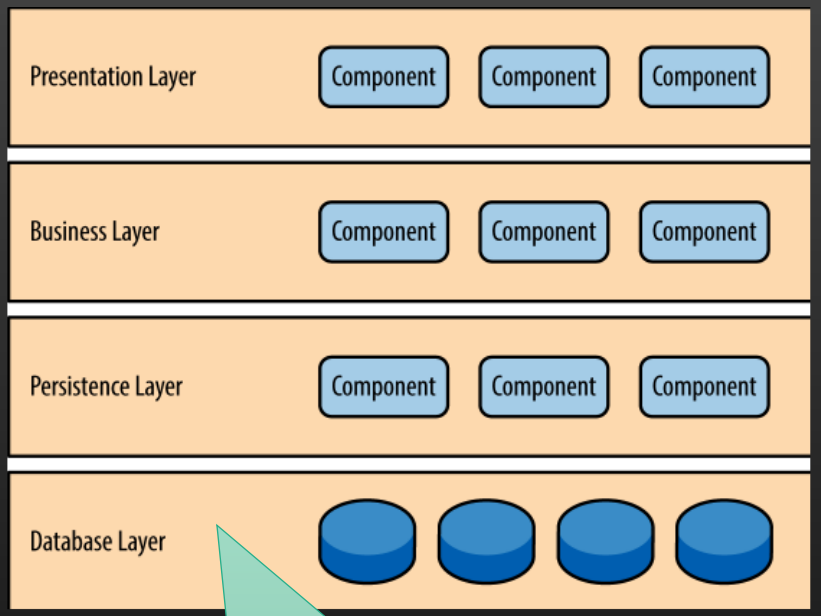
Android platform architecture

I Oliveira

Vista dinâmica (interação). Notar o nível de abstração (subsistema, não objetos)



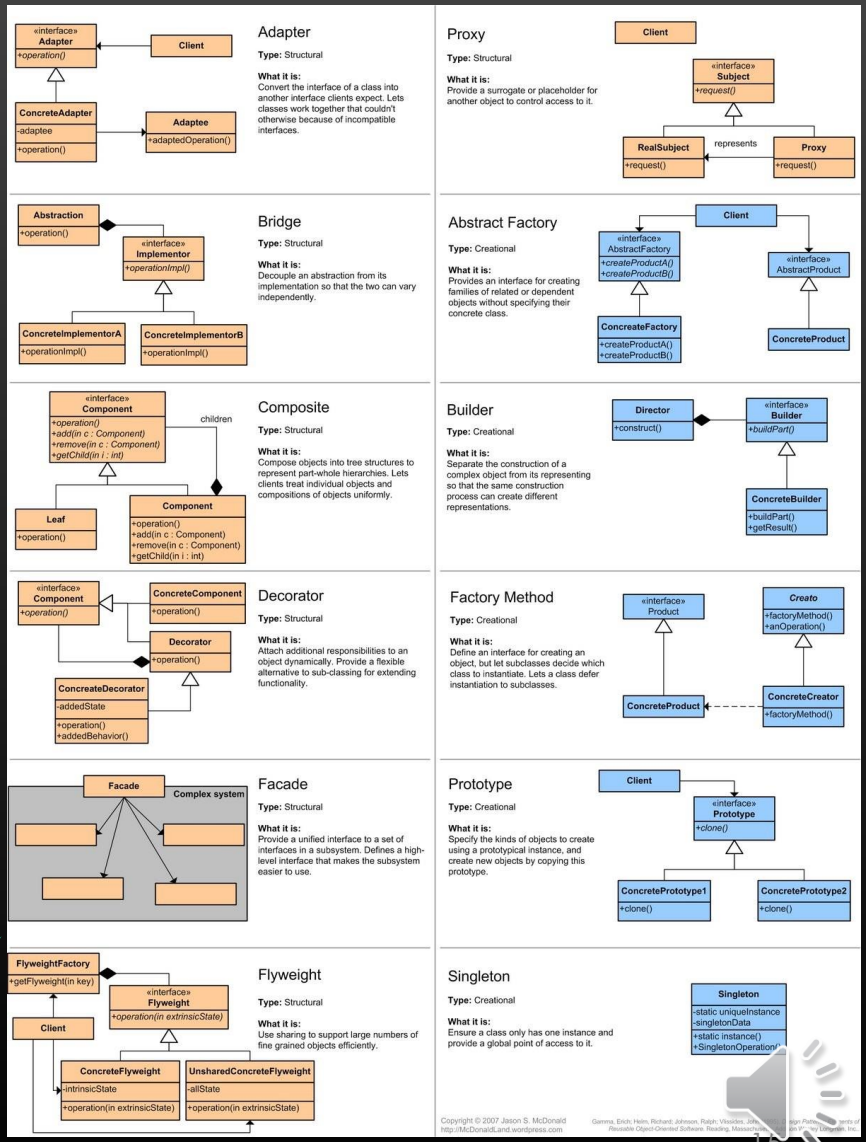
Arquitetura, componentes, classes



Como é que um sistema de software está organizado em grandes módulos. Os "grupos de funcionalidade" podem ser vistos como componentes.

Implementação interna de cada componente pode usar certos arranjos de classes (padrões frequentes).

I Oliveira

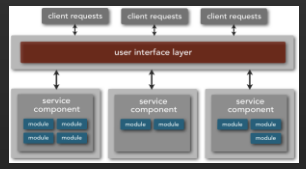


Decisões de arquitetura

A decisão de usar uma aplicação web como interface da aplicação

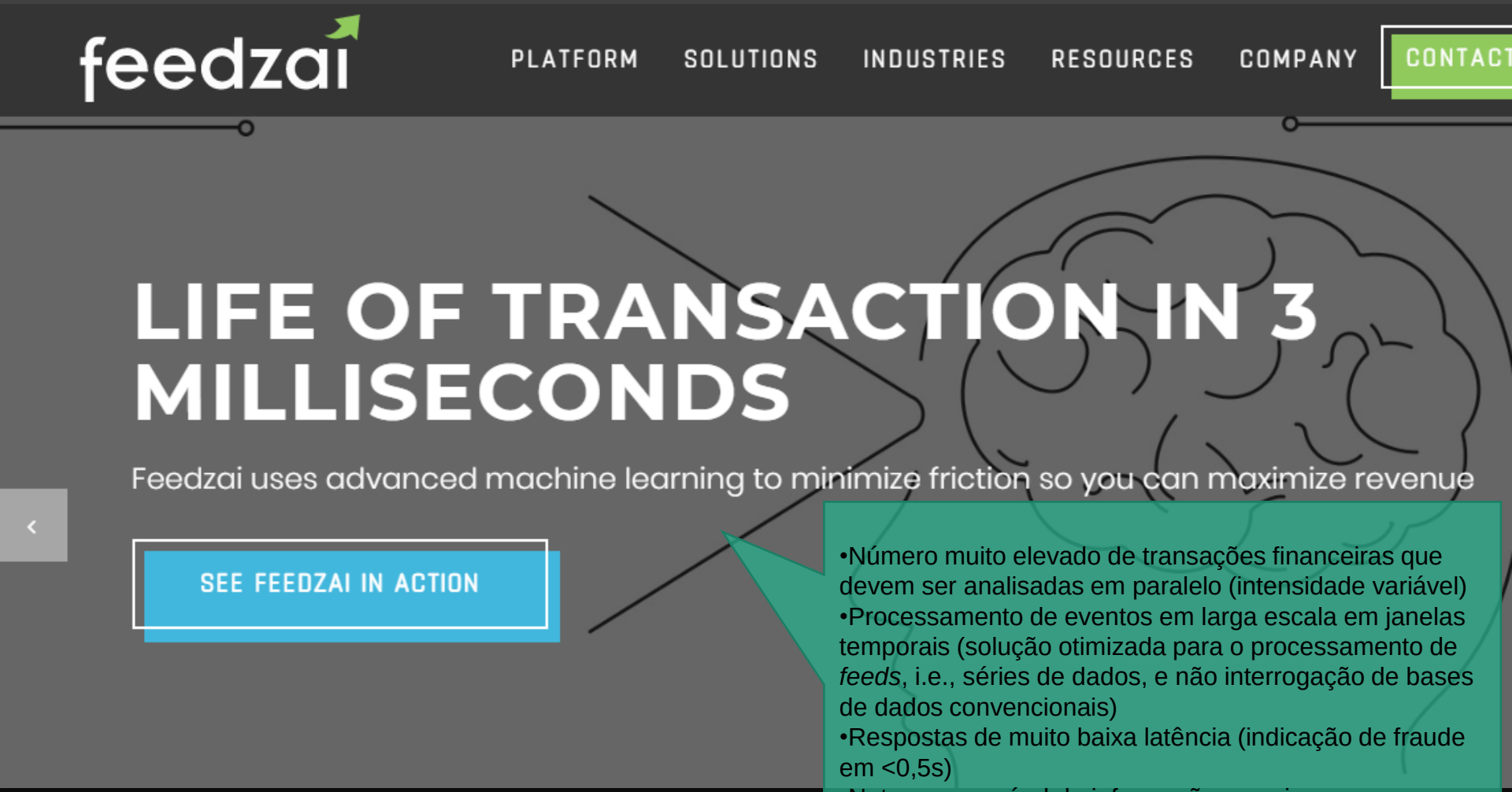
A decisão de usar Java Server Faces para o "web framework" de desenvolvimento

A decisão de distribuir componentes por vários nós para aumentar a capacidade de escalar.



A decisão de usar o modelo REST para organizar a integração com sistemas externos.

Exemplo de um sistema complexo: Feedzai



https://feedzai.com

- Número muito elevado de transações financeiras que devem ser analisadas em paralelo (intensidade variável)
- Processamento de eventos em larga escala em janelas temporais (solução otimizada para o processamento de *feeds*, i.e., séries de dados, e não interrogação de bases de dados convencionais)
- Respostas de muito baixa latência (indicação de fraude em <0,5s)
- Natureza sensível da informação: canais seguros e invioláveis.
- Há clientes que preferem a solução na cloud e outros que querem usar apenas instalações no seu datacenter.

Precisamos de pensar na arquitetura de sistemas complexos

Alguns exemplos de sistemas complexos:

a) **Wikipedia**

b) **Multi-player online RPG**

c) **Amazon web store**

d) **Twitter**

e) **Netflix**

Em que tipo de requisitos de arquitetura fazem pensar?

Exemplos

Wikipedia

Enorme quantidade de documentos de texto

Gerir milhões de documentos: armazenamento, recuperação

Pesquisar de conteúdo em documentos

Como criar e editar documentos de forma distribuída? Direitos de acesso dos utilizador?

Multi-player online RPG

Grande quantidade de jogadores interagindo entre si (por exemplo, milhares de utilizadores)

Como distribuir a carga? Como otimizar a latência?

Há utilizadores banidos?

Integrar faturação (em compras de jogos), etc?

Como prevenir hackers/ batoteiros?

Exemplos

Amazon web store

Lidar com picos de utilização (por exemplo: *black friday*)

Sistema de recomendação de produtos (AI)

Quais os utilizadores que têm interesses semelhantes aos de outros utilizadores?

O que deve ser rastreado? Cliques? Compras? Comentários?

Há problemas de privacidade?

Twitter

Grande número de utilizadores, enorme quantidade de eventos, interações complexas

Integrações complexas: redes, redes sociais, etc.

Sistemas de entrega fiáveis. Comprovativo de entrega?

Basear-se em protocolos Web

Exemplos

Netflix

Rede de distribuição de conteúdos em larga escala (CDN): equilíbrio de carga, réplicas,...

Utilização interativa, conteúdos multimédia (latência muito baixa)

Proteção de direitos (DRM)

Arquitetura no openUP

Elaboration: saber como construir, construindo uma parte

Importante mitigar riscos técnicos

Definir a arquitetura do Sistema, implementar uma parte (→ arquitetura executável)

Validar a arquitetura

Construir "esqueletos" para os componentes principais e começar a sua integração.

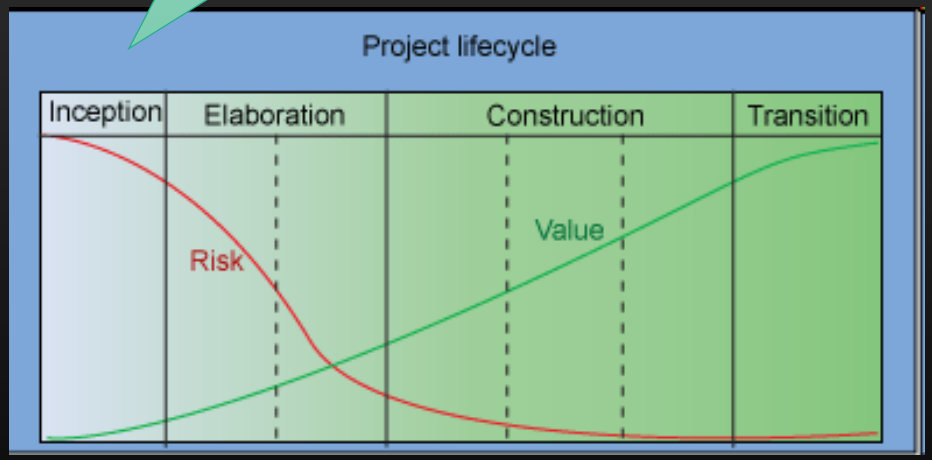
Identificar dependências de sistemas e componentes externos e especificar como serão integrados.

~10% do código será implementado

Basear a arquitetura nos casos de utilização nucleares

20% dos casos de uso determinam 80% da arquitetura

Controlar os riscos técnicos, aprofundando os requisitos e desenvolvendo a arquitetura / plano técnico. Comprovar a arquitetura implementando uma parte.



Credit: Per Kroll (IBM)

OpenUP practice: evolutionary architecture

Practices > Technical Practices > Evolutionary Architecture

Practice: Evolutionary Architecture

Analyze the major technical concerns that affect t…tion, and capture th…tural decisions to ensure that those decisions are assessed and comm…

Analisar as principais preocupações técnicas; recolher decisões de arquitetura; avaliar, documentar e comunicar decisões.

Relationships

Content References

- How to adopt the Evolutionary Arch…
- Key Concepts
  - Architectural Mechanism
  - Architectural Views and Viewpoints
  - Software Architecture
- Architecture Notebook
- Envision the Architecture
- Refine the Architecture
- Guidance
  - Guidelines
    - Abstract Away Complexity
    - Modeling the Architecture
    - Software Reuse

Inputs

- [Technical Design]

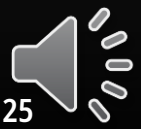
Porque é que é “evolutiva”?

No OpenUP:

- Analisar as principais questões técnicas que afetam a solução
- Documentar decisões de arquitetura para garantir que foram avaliadas e comunicadas
- Implementar e testar capacidades-chave como forma de lidar com os desafios de arquitetura
- Evoluir ao longo do tempo, a par com o trabalho de implementação “normal”

Ideia de fundo:

a escolha da arquitetura comporta riscos que devem ser controlados cedo, experimentando as capacidades-chave.



Vistas de arquitetura na UML

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,...)

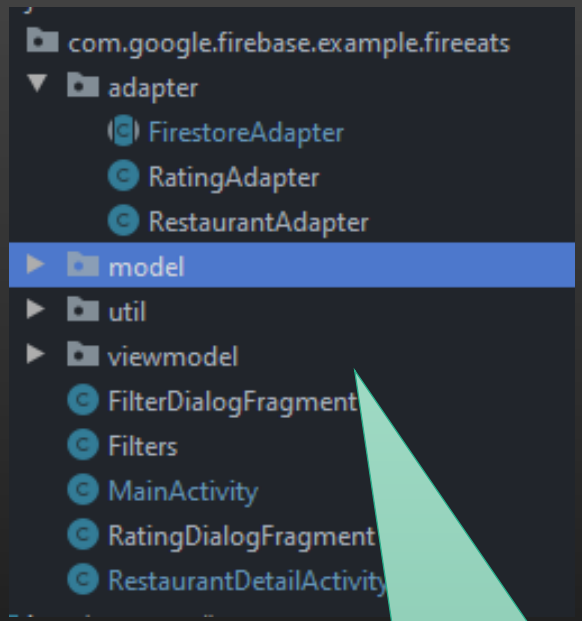
Arquitetura lógica

Organização geral da solução em blocos (*packages)*

Os *packages* podem representar agrupamentos muito diferentes.

E.g.:

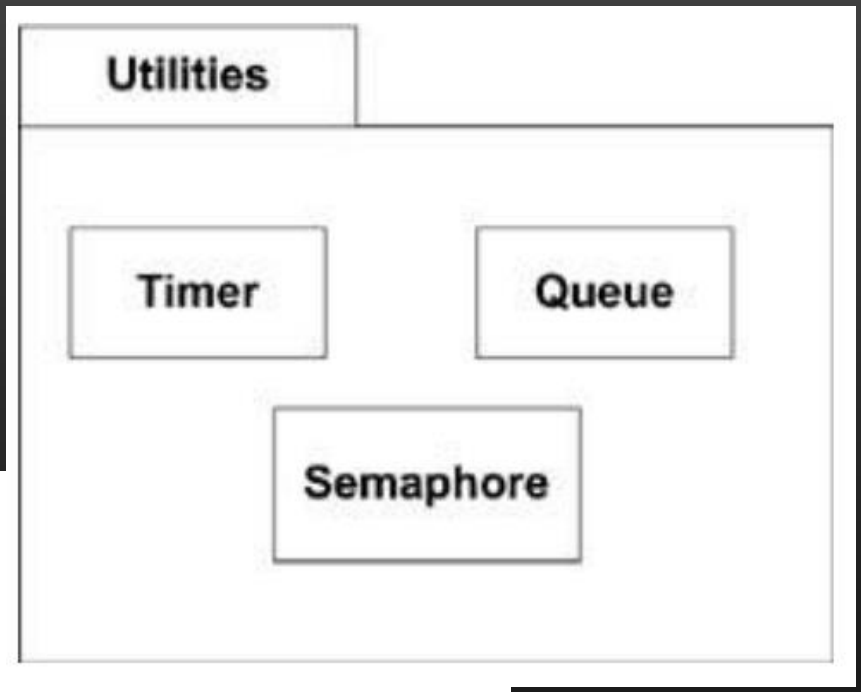
- *Packages* num programa em Java
- *Packages* num modelo UML
- Subsistemas/divisões do sistema sob especificação

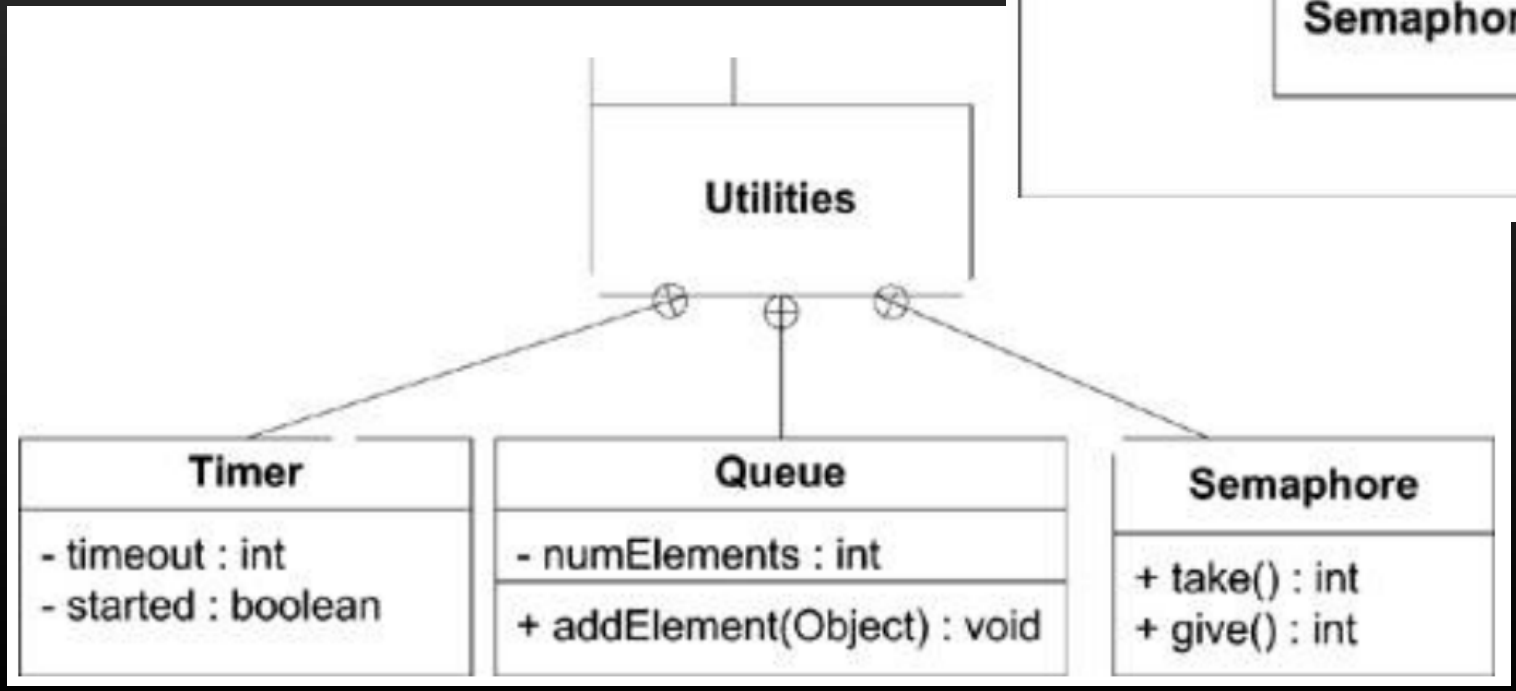


A ideia de package é usada em Java para formar grupos de entidades relacionadas.
Os *packages* podem ser hierárquicos.

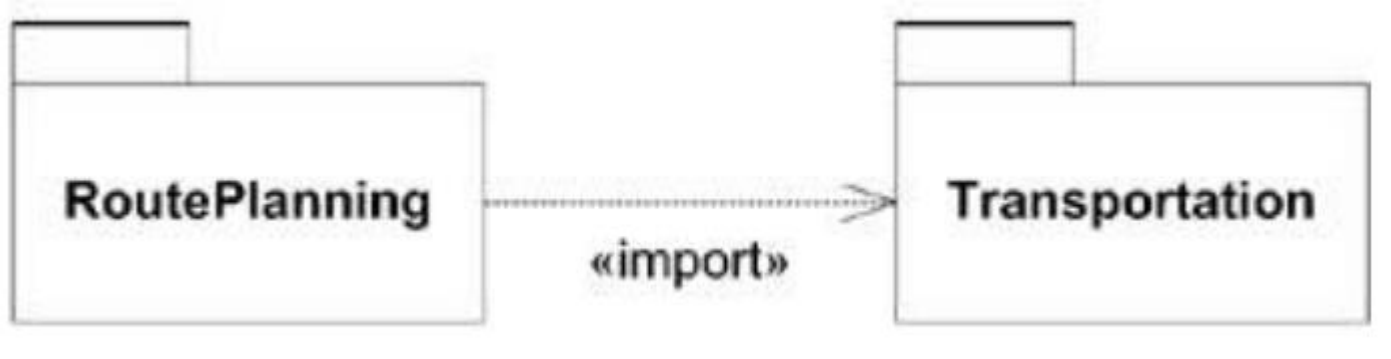
Comunicar a arquitetura lógica com pacotes (*packages)*

Classes contidas num pacote
- duas representações alternativas



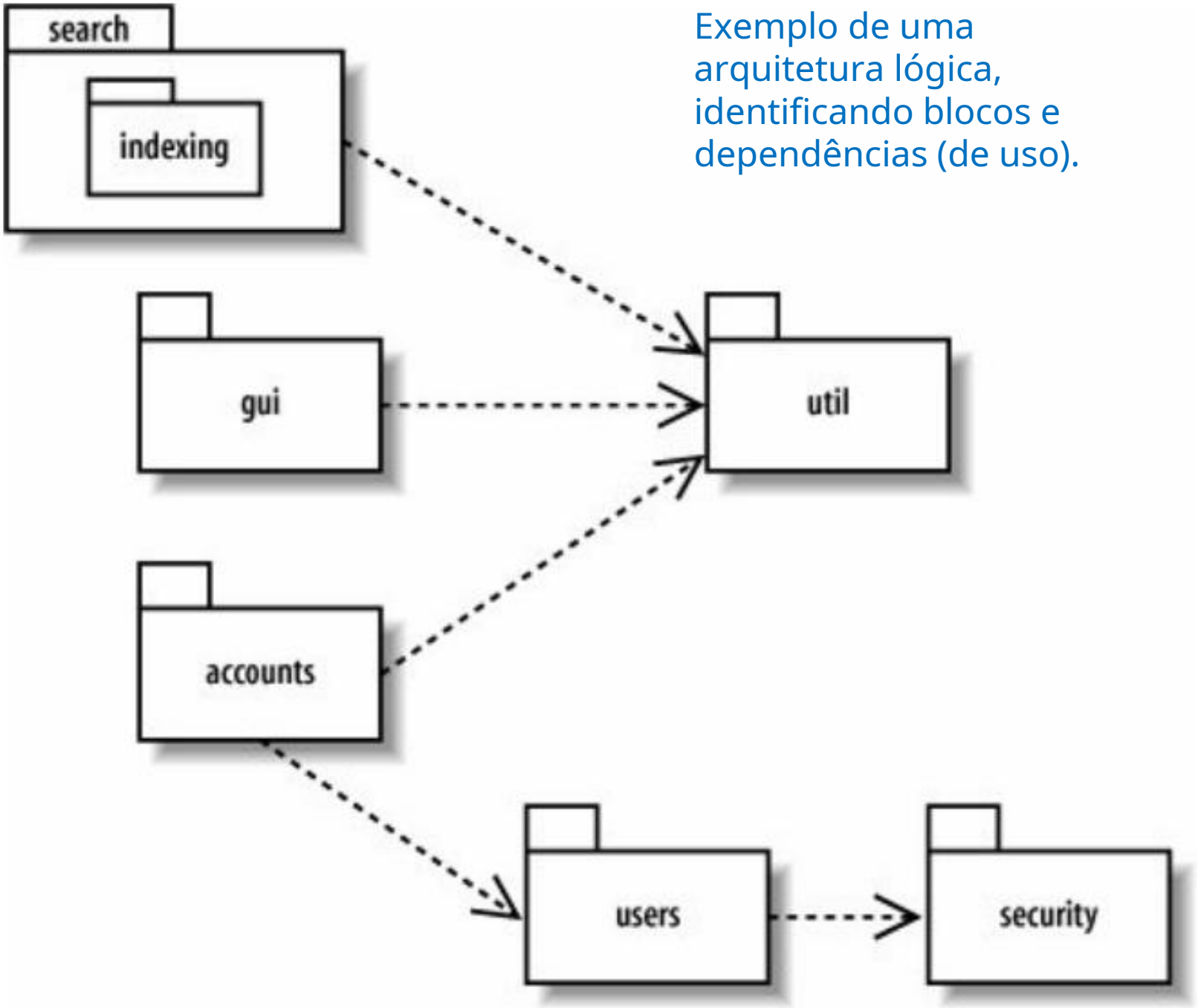


Associações entre pacotes



Dependência do tipo “import”

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



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,...)

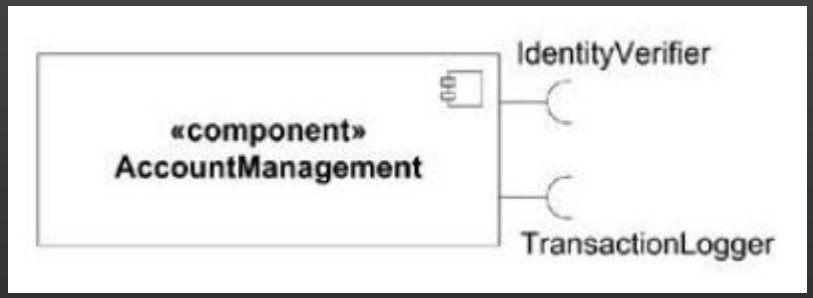
Componente

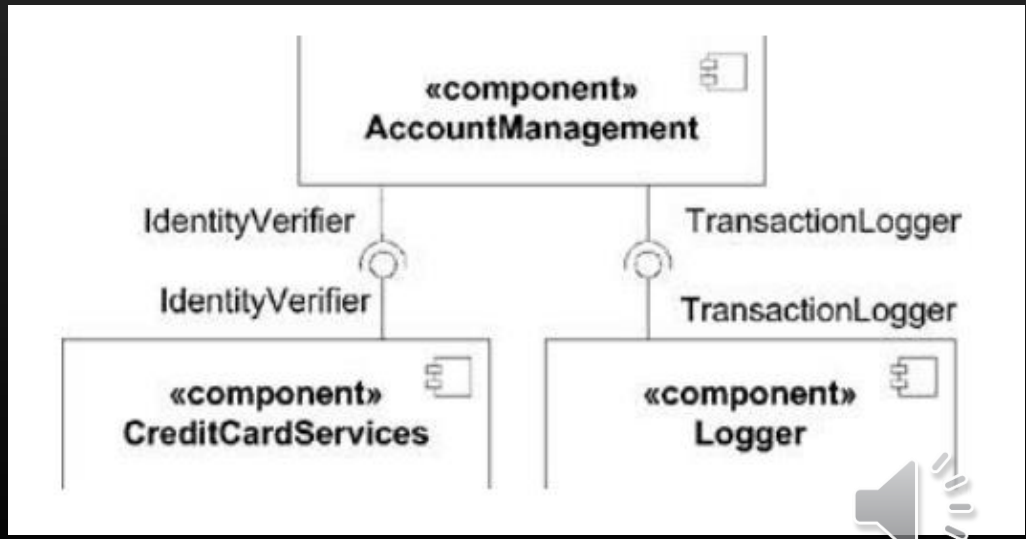
É normal dividir sistemas complexos em subsistemas mais geríveis

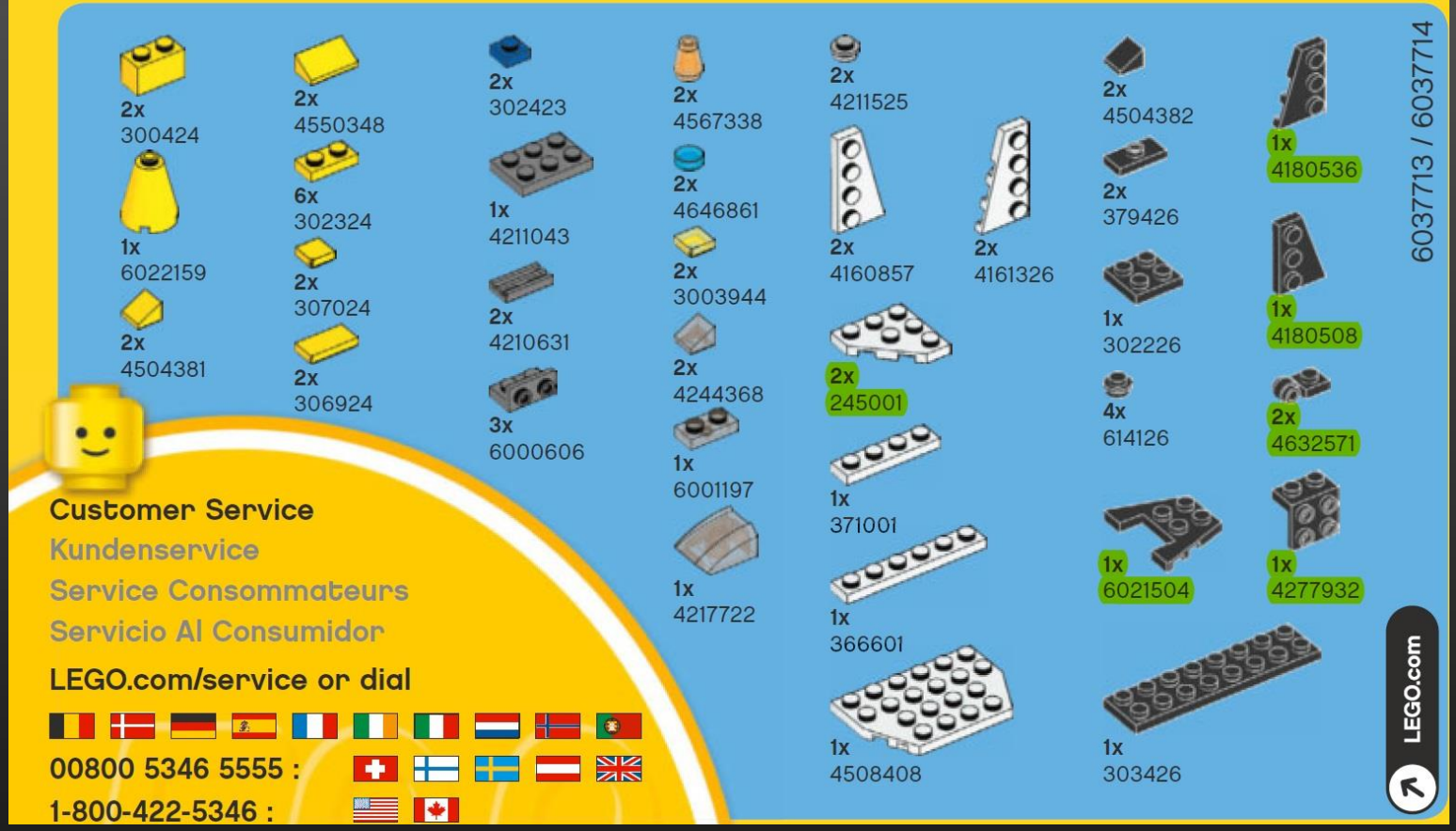
O componente é uma peça substituível, reusável de um sistema maior, cujos detalhes de implementação são abstraídos

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 (da estrutura interna)

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*)

https://mvnrepository.com

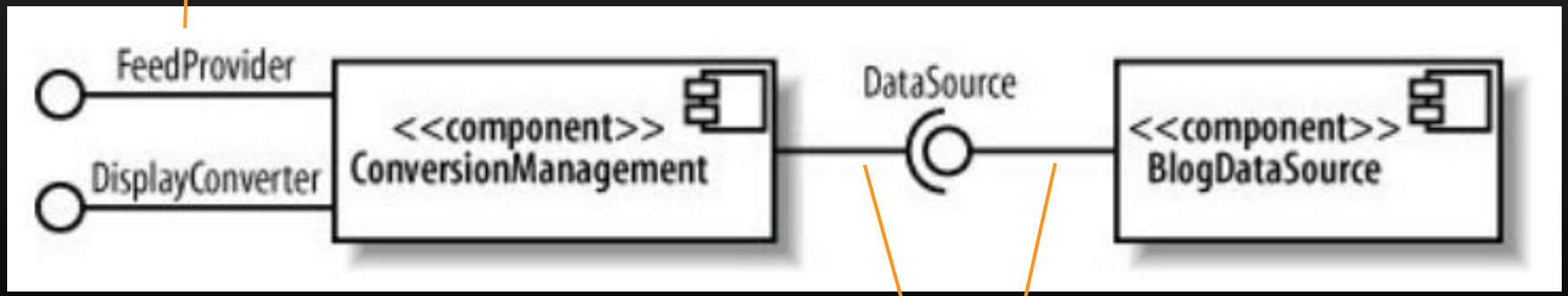
Uma “loja” de componentes, encapsulados, reutilizáveis e substituíveis.

Figure

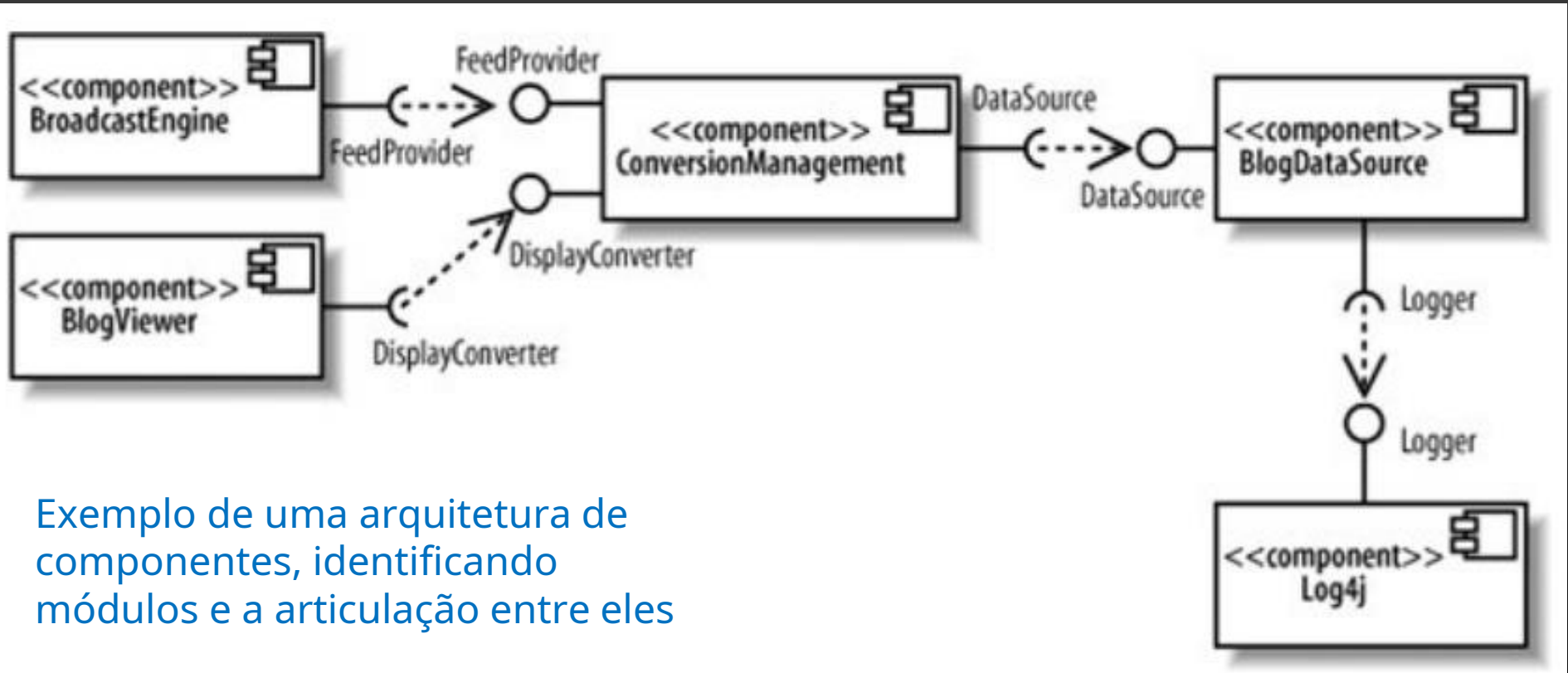
Solução modular com componentes

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

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

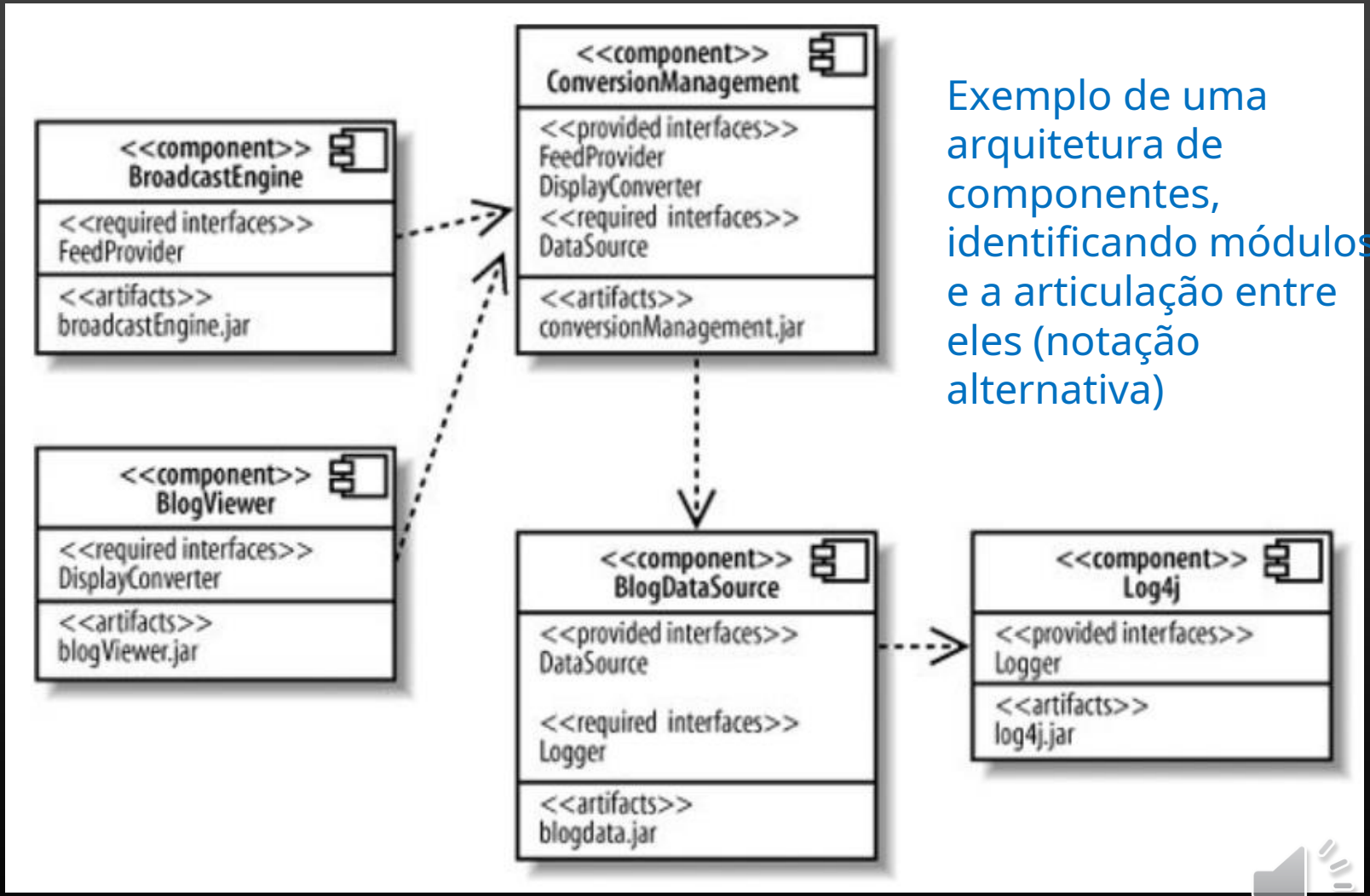


O mesmo modelo, notação ligeiramente diferente 1



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

O mesmo modelo, notação ligeiramente diferente 2



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

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,...)

Diagramas de instalação da UML

Nós (*node*)

Um equipamento de hardware

Ambiente de execução

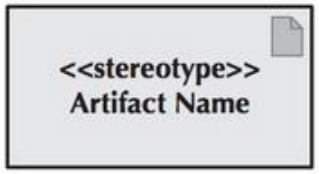
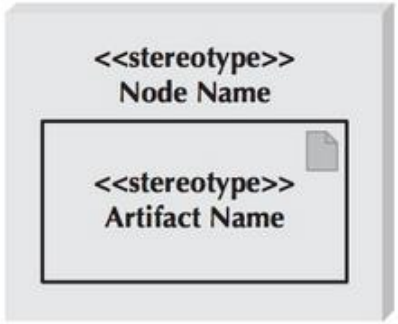
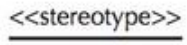
Um ambiente externo à solução que proporciona o contexto necessário à sua execução

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

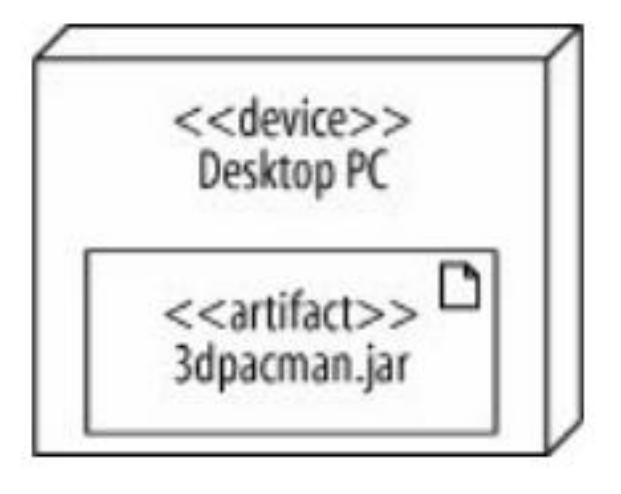
Artefactos (*artifact)*

Ficheiros concretos que são executados ou utilizados pela solução

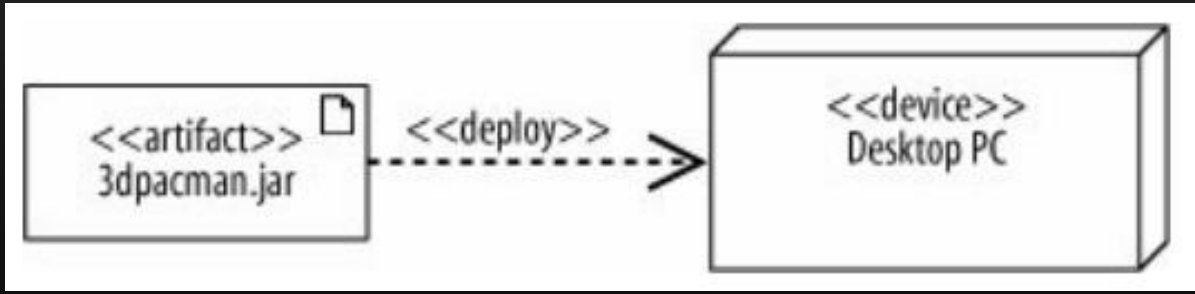
E.g.: executáveis, bibliotecas, configurações, scripts

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

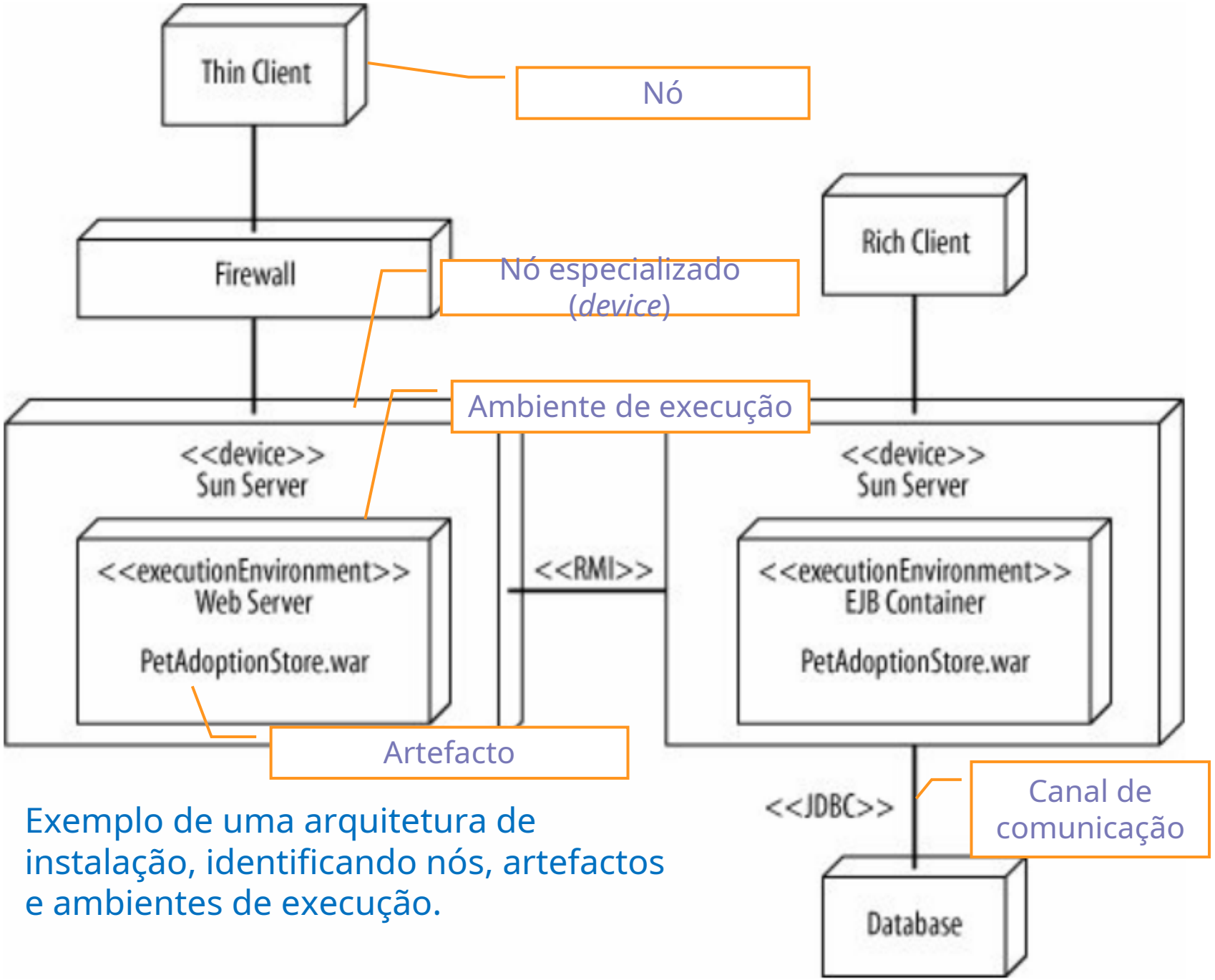


Notações alternativas.



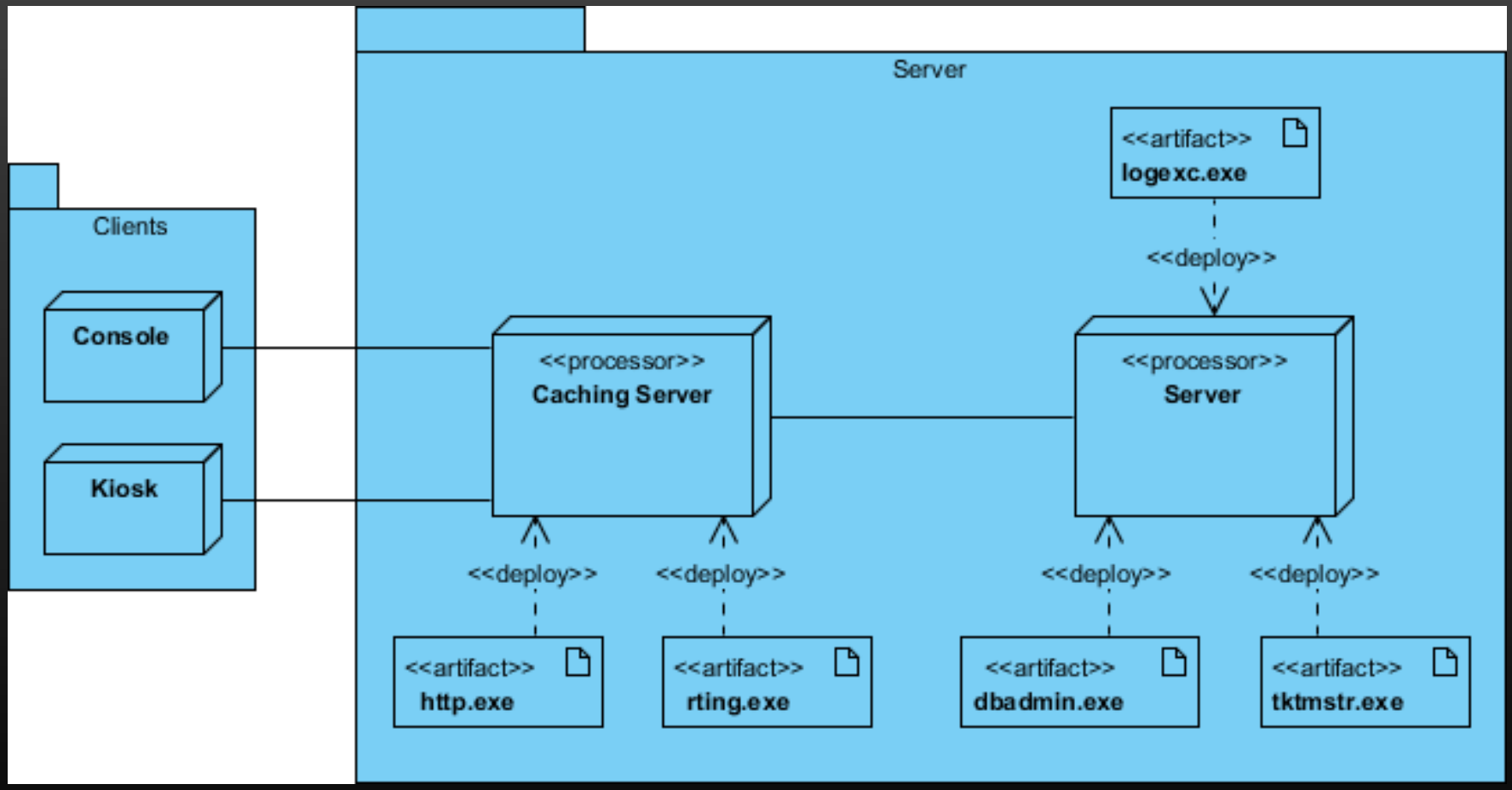
Rastreabilidade até aos componentes

A relação “manifest” permite associar componentes a artefactos.



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

Exemplos (diagrama de instalação)



References

Core readings	Suggested readings
• [Dennis15] – Chap. 7 & 11	