

41951- ANÁLISE DE SISTEMAS

Arquitetura do software e a UML

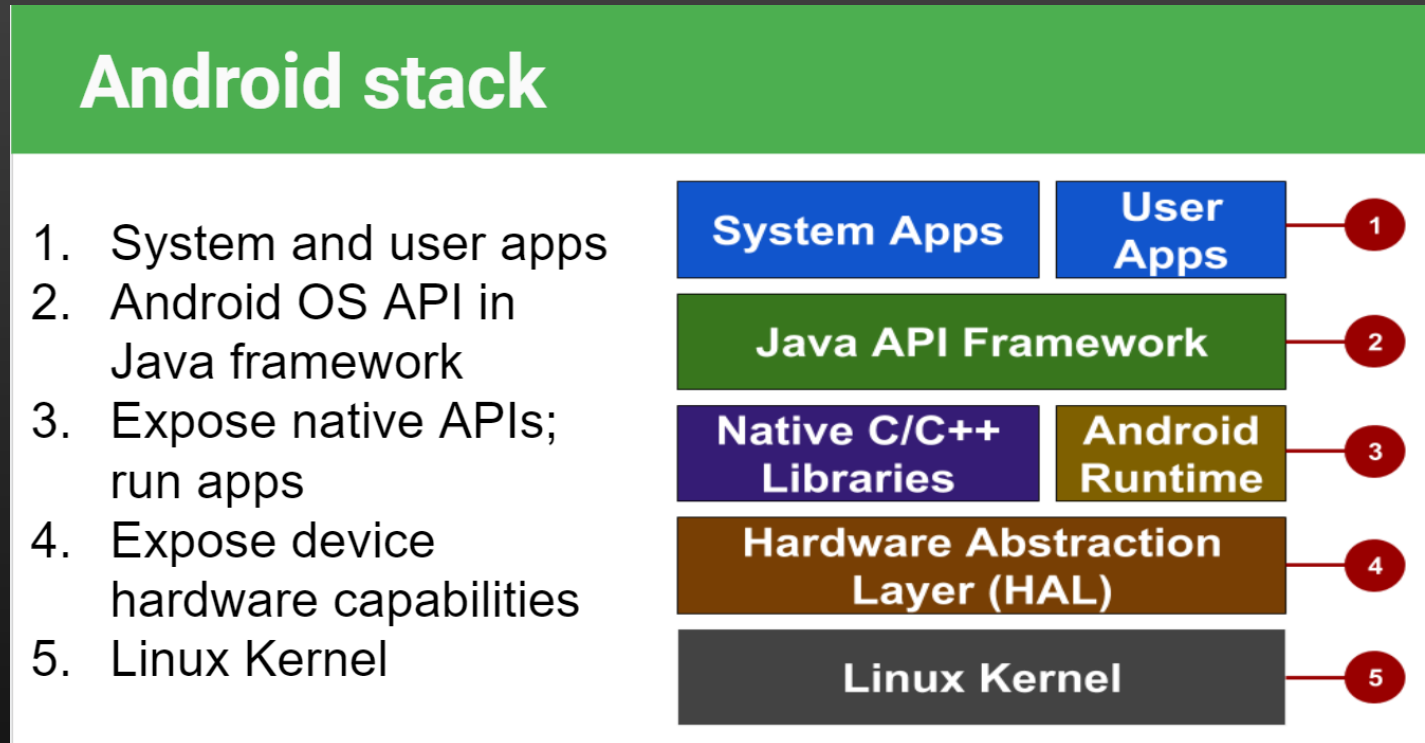
- D. de pacotes, components e de instalação

Ilídio Oliveira | v2022/03/29

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)

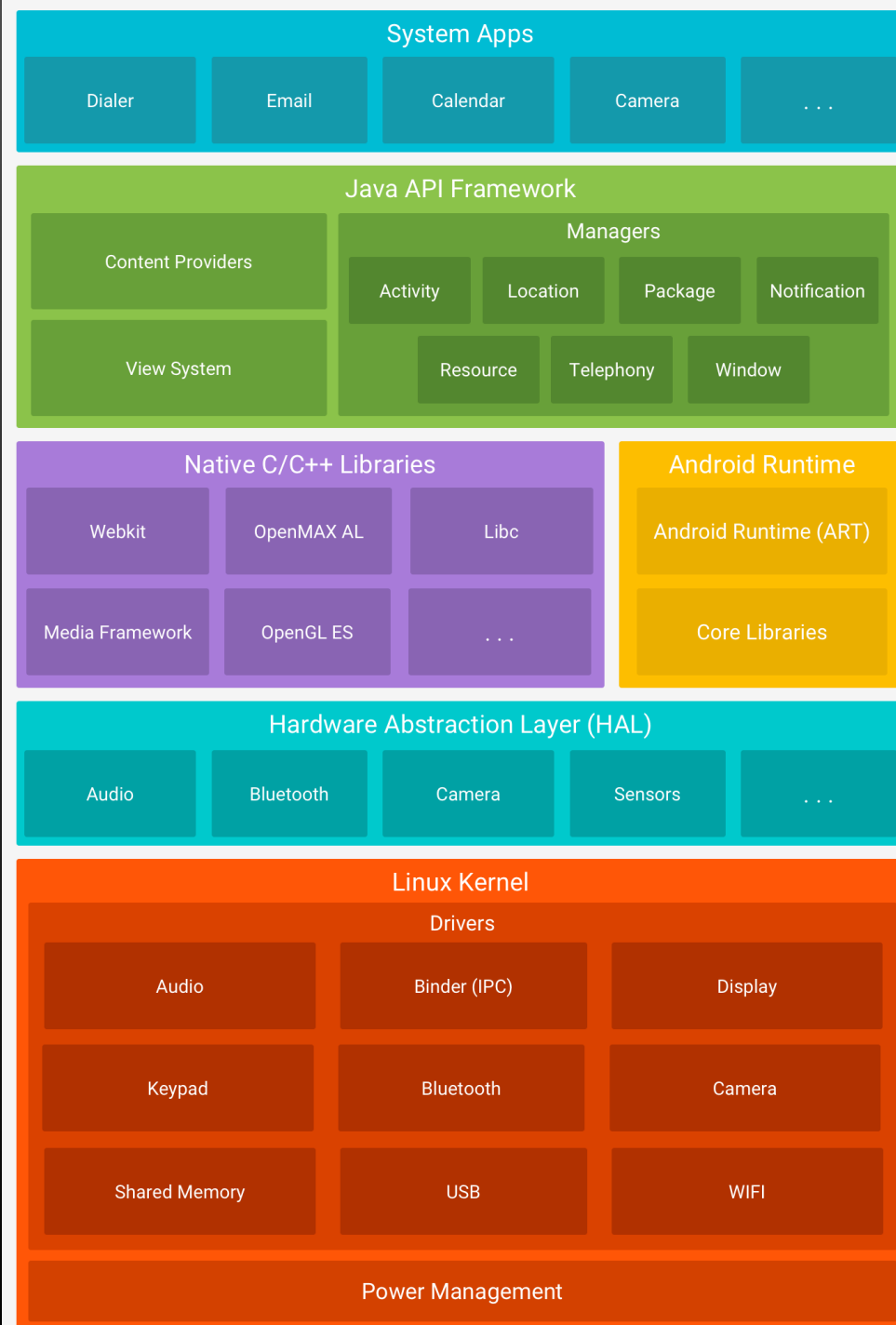


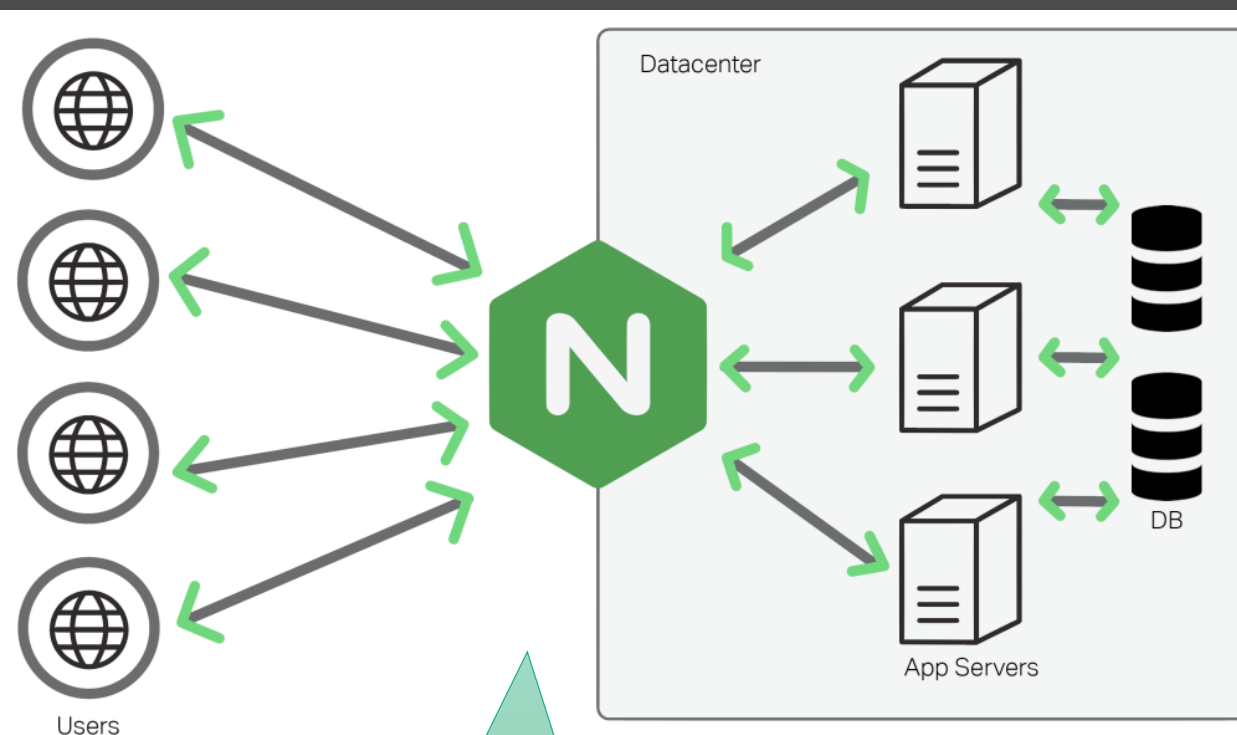
Um exemplo de arquitetura: a organização do sistema Android.

Android *stack*

Módulos em cada camada
(partições).

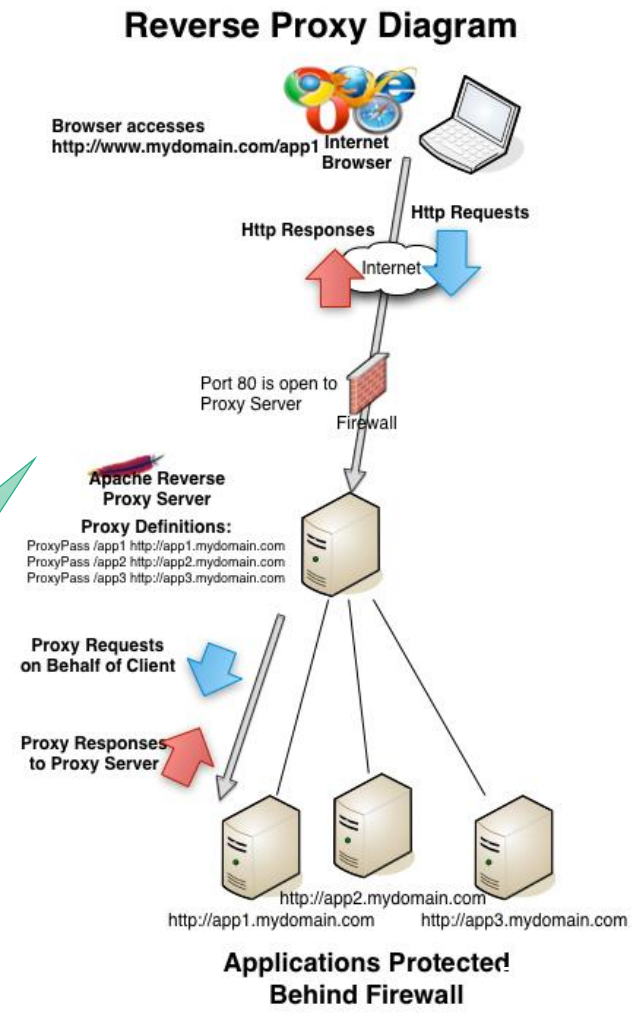
I Oliveira





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

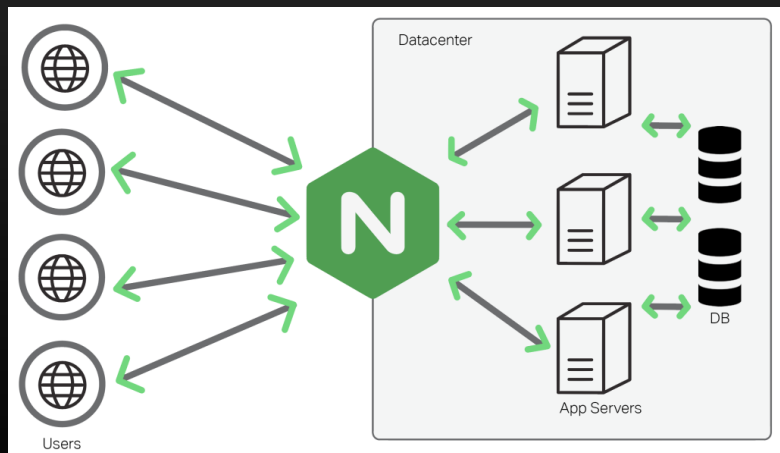
O que é que as “ilustrações” anteriores têm em comum?

- Explicar a organização de uma solução, em termos dos seus “grandes peças” (*high-level*), representando uma distribuição de responsabilidades
- 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)

I Oliveira



O que trata a arquitetura do software?

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, and the relationships among them.

Relationships

Related Elements

- [Architecture Notebook](#)
- [Design](#)
- [Executable Architecture](#)
- [How to adopt the Evolutionary Architecture](#)

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

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

Arquitetura de Software

Uma arquitetura é o conjunto de decisões importantes sobre a organização de um sistema de software

Plano (decisões) com a estratégia para a implementação.

...a seleção dos elementos estruturais e das suas interfaces pelas quais o sistema é composto, juntamente com o seu comportamento especificado nas colaborações entre esses elementos, a composição desses elementos estruturais e comportamentais em subsistemas progressivamente maiores, e o estilo arquitetónico que orienta esta organização [BRJ99]

→ As grandes linhas sobre organização e as interações do sistema



Who Needs an Architect?

Martin Fowler

<https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf>

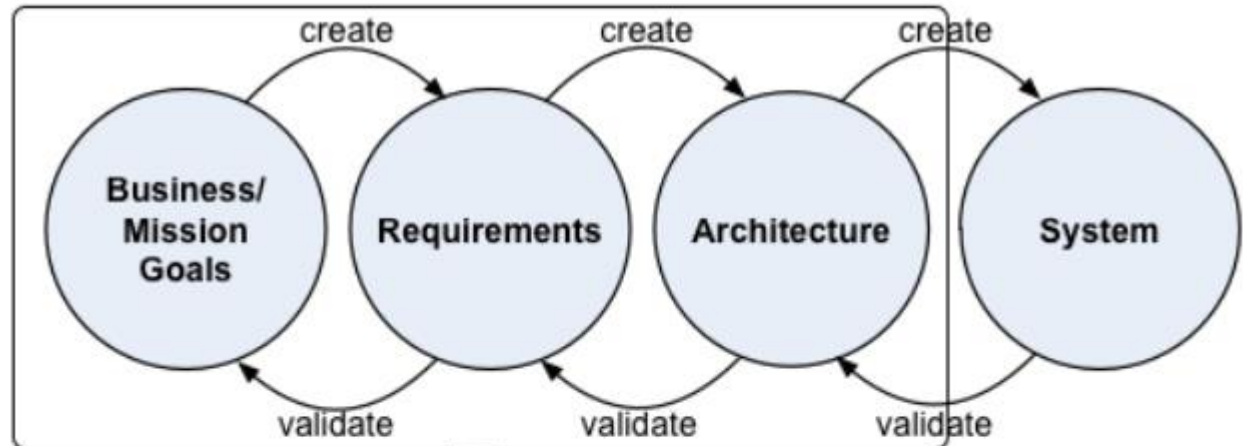
Outra maneira de colocar a questão...

In the words of Ralph Jordan:

So, a better definition would be "In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.' This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers."

There is another style of definition of architecture which is something like "architecture is the set of design decisions that must be made early in a project." I complain about that one, too, saying that architecture is the decisions that you wish you could get right early in a project, but that you are not necessarily more likely to get them right than any other.

Papel do arquiteto (de software)



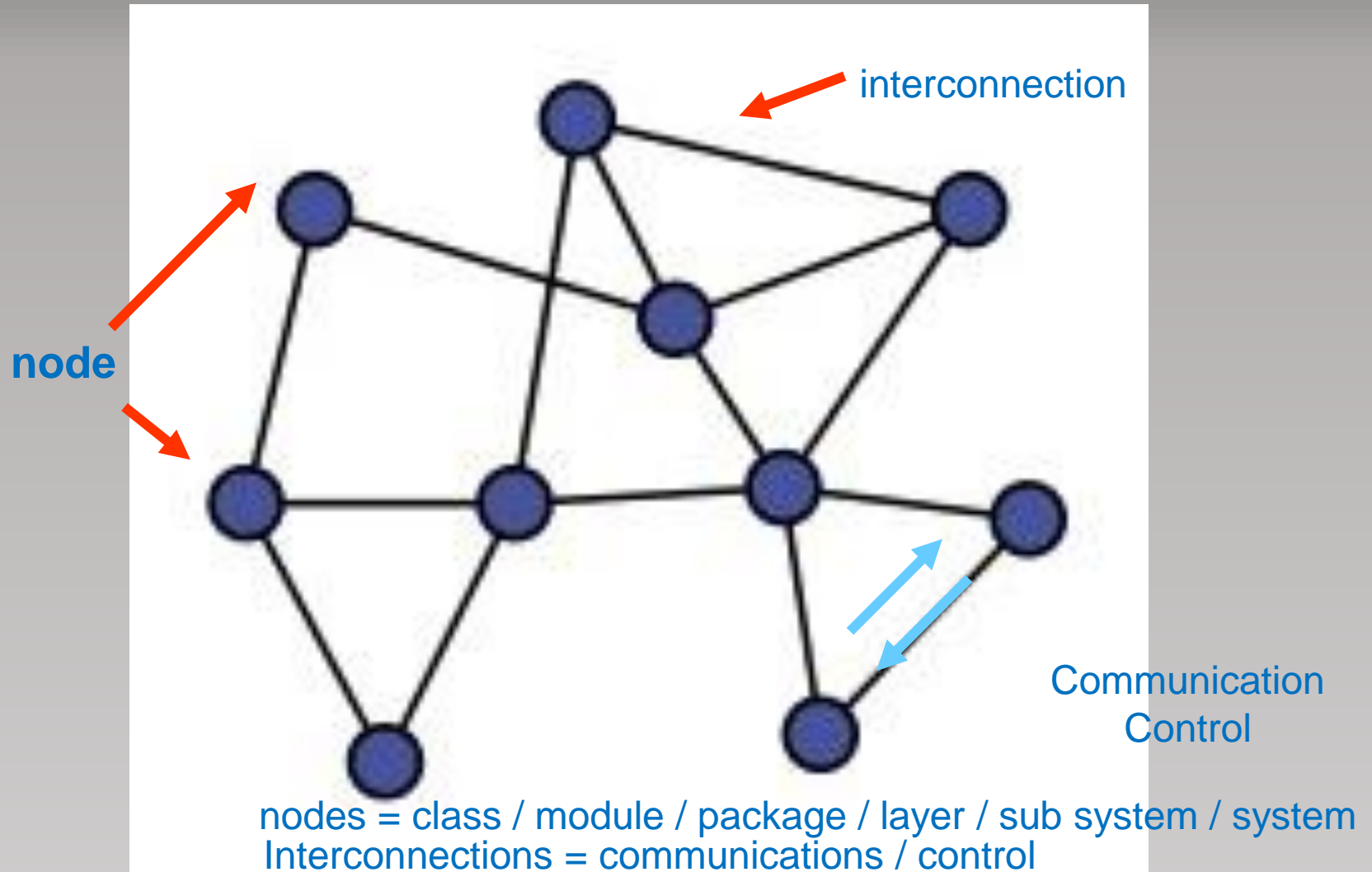
Core Skill Sets

- 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>

Elementos de uma arquitetura

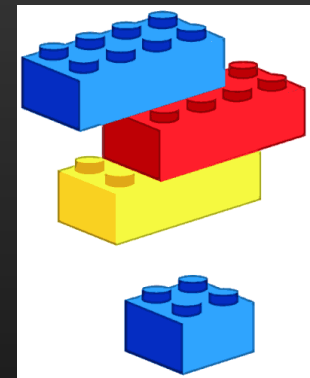
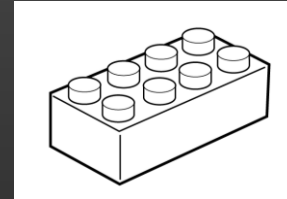


O que compõe uma “arquitetura”?

Elementos

Abstrações/blocos usados na construção do sistema

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



Interações de elementos

interagir através de interfaces públicas

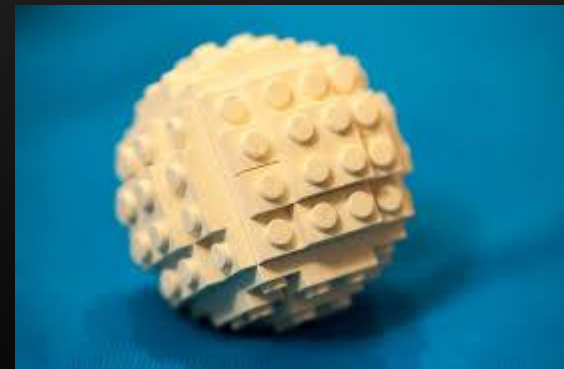
Relacionamentos normalmente são concretizados com interfaces

Arquitetura trata as interfaces públicas

Estruturas

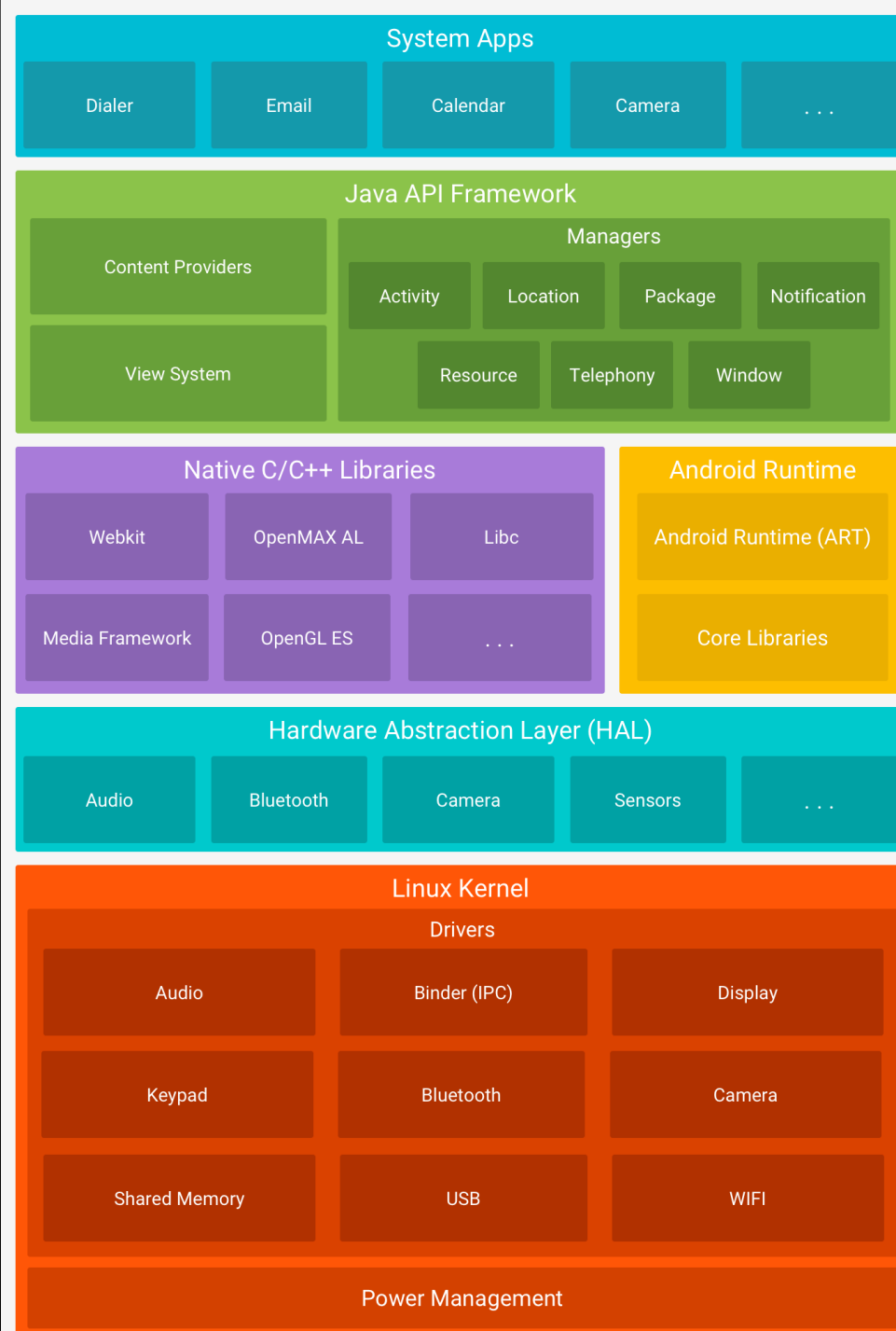
Defina a partição estática e atribuição de funcionalidade

Suporta a atribuição de comportamentos



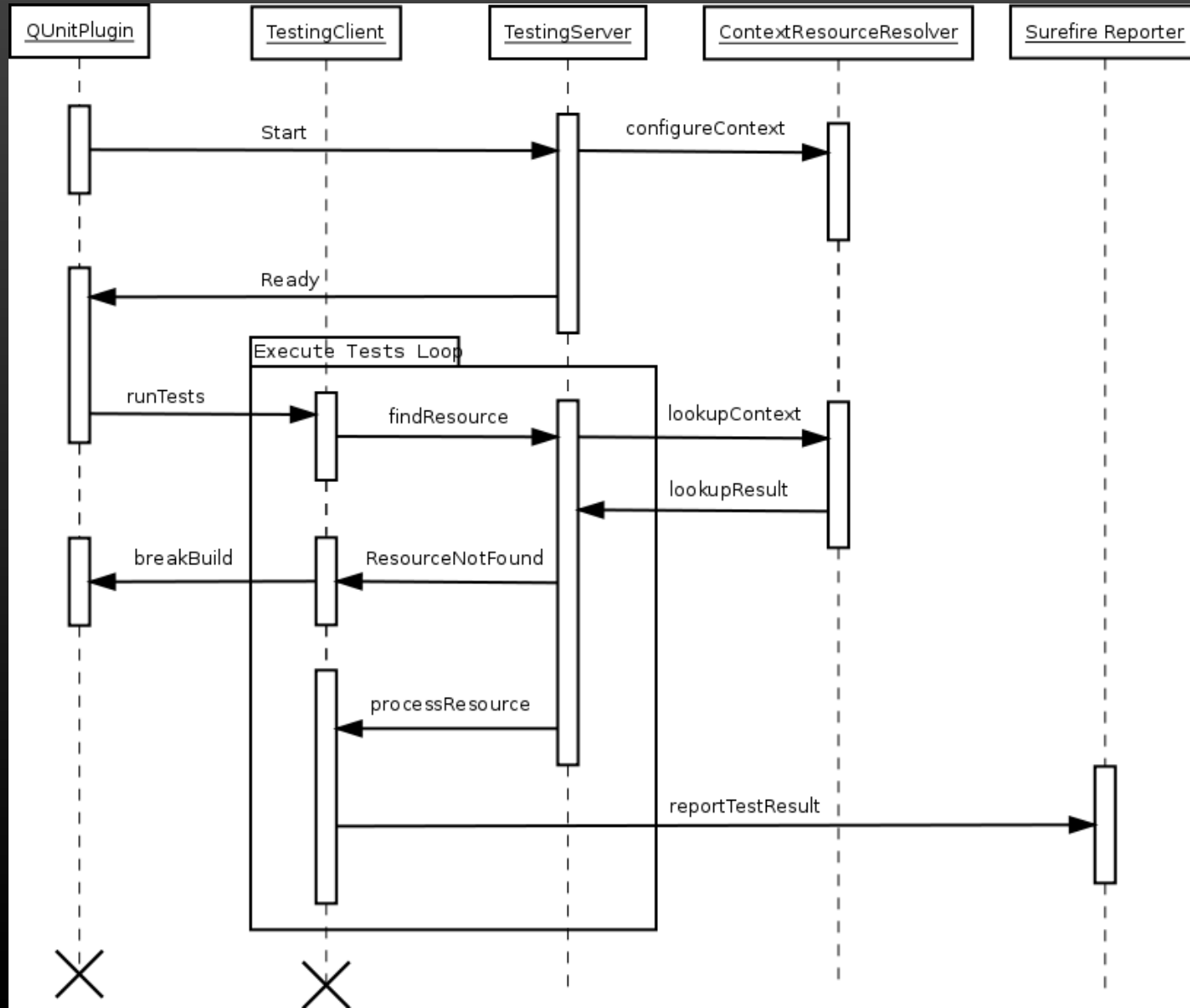
<http://www.columbia.edu/cu/gsap/bt/bsi/suspensi/suspensi.html>

Vista estrutural (as partes constituintes)

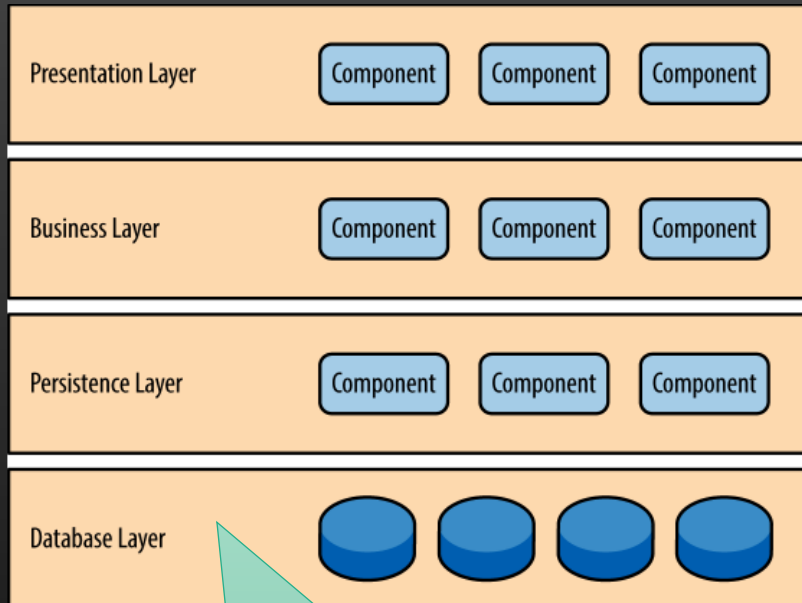


Android platform architecture

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



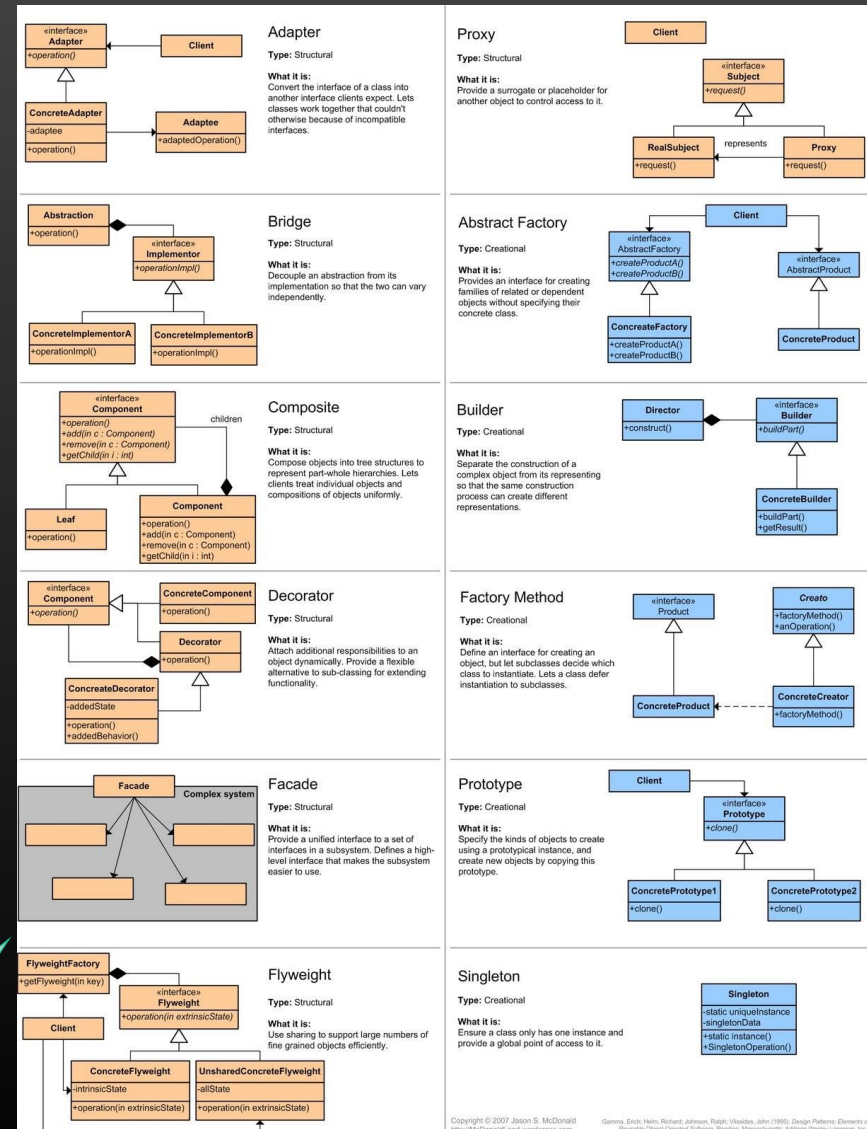
Arquitetura, componentes, classes



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



A arquitetura é consequência dos requisitos e estabelece compromissos

Decisões de arquitetura

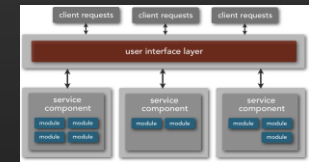
A decisão de usar uma aplicação web como camada de interação da solução



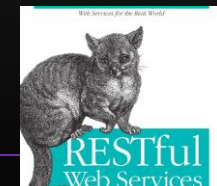
A decisão de usar Java Server Faces para o “web framework” de desenvolvimento



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



PLATFORM

SOLUTIONS

INDUSTRIES

RESOURCES

COMPANY

CONTACT

LIFE OF TRANSACTION IN 3 MILLISECONDS

Feedzai uses advanced machine learning to minimize friction so you can maximize revenue

SEE FEEDZAI IN ACTION

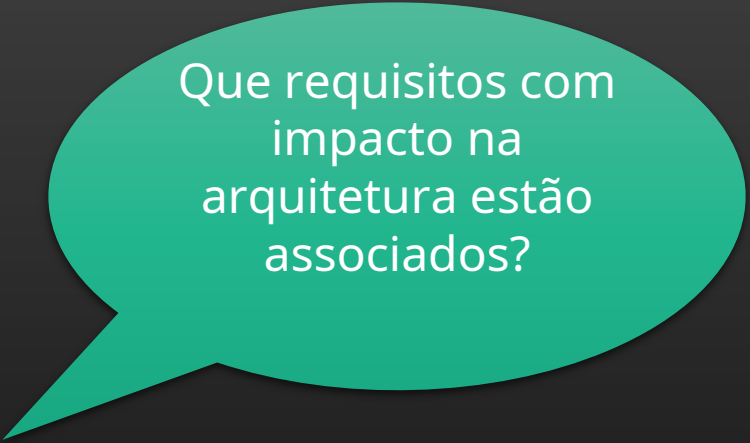
- Número muito elevado de transações financeiras que devem ser analisadas em paralelo (intensidade variável)
- Processamento de eventos em larga escala em janelas temporais (solução otimizada para o processamento de *feeds*, i.e., séries de dados, e não interrogação de bases de dados convencionais)
- 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

<https://feedzai.com>

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



Que requisitos com impacto na arquitetura estão associados?

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)

Vistas de arquitetura na UML

OpenUP practice: evolutionary architecture

Practices > Technical Practices > Evolutionary Architecture

Practice: Evolutionary Architecture



Analyze the major technical concerns that affect the system, and capture the architectural decisions to ensure that those decisions are assessed and communicated.

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

Relationships

Content References

- How to adopt the Evolutionary Architecture
- 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

- [\[Technical Design\]](#)

A arquitetura do sistema aborda diferentes perspectivas de análise

① Arquitetura lógica do software

Organização geral dos blocos de software

Independente da tecnologia de implementação

② Arquitetura de componentes do software

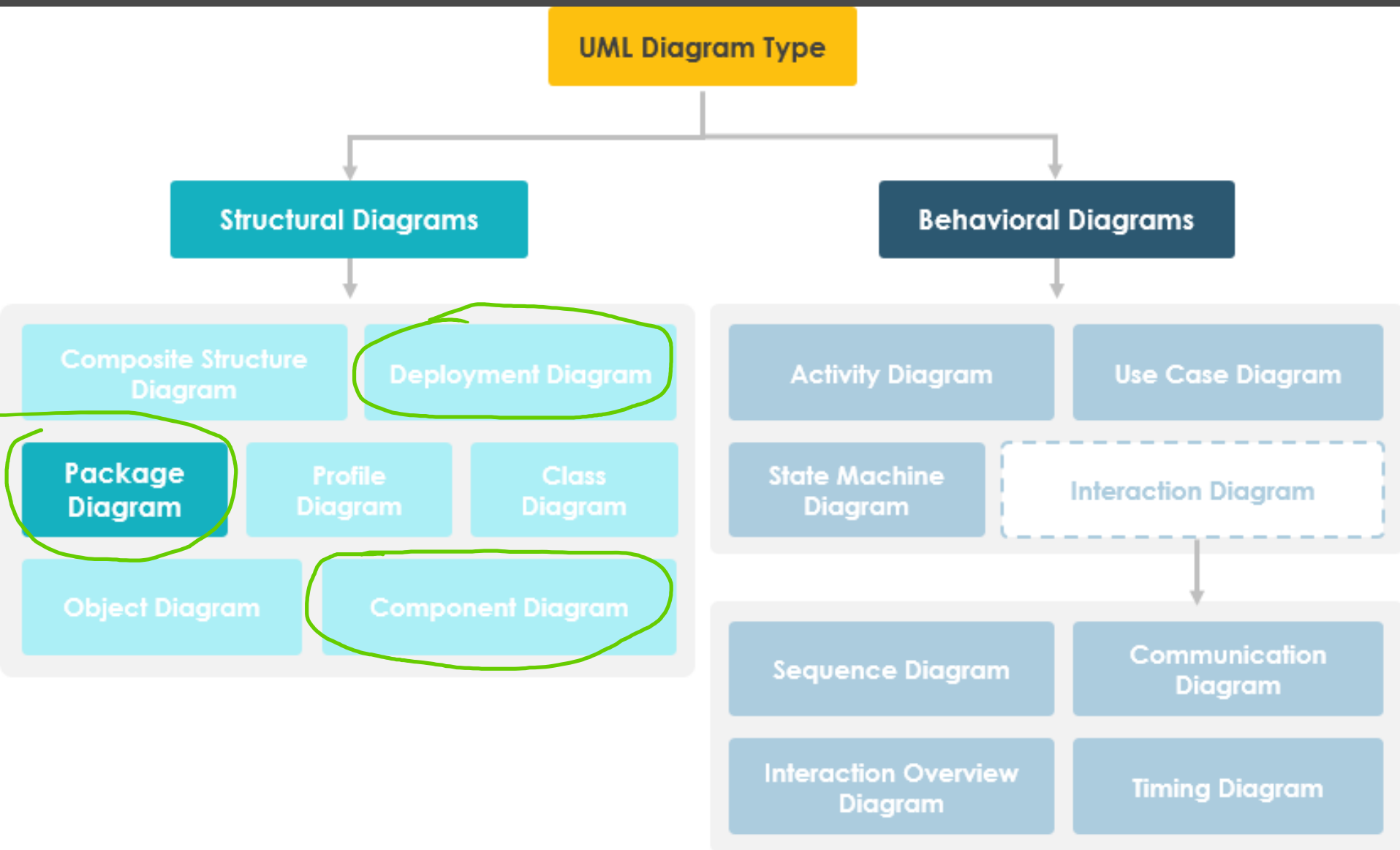
Peças construídas com uma tecnologia concreta

Construção “modular”

- E.g.: existem pré-feitos?

③ Arquitetura de instalação

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



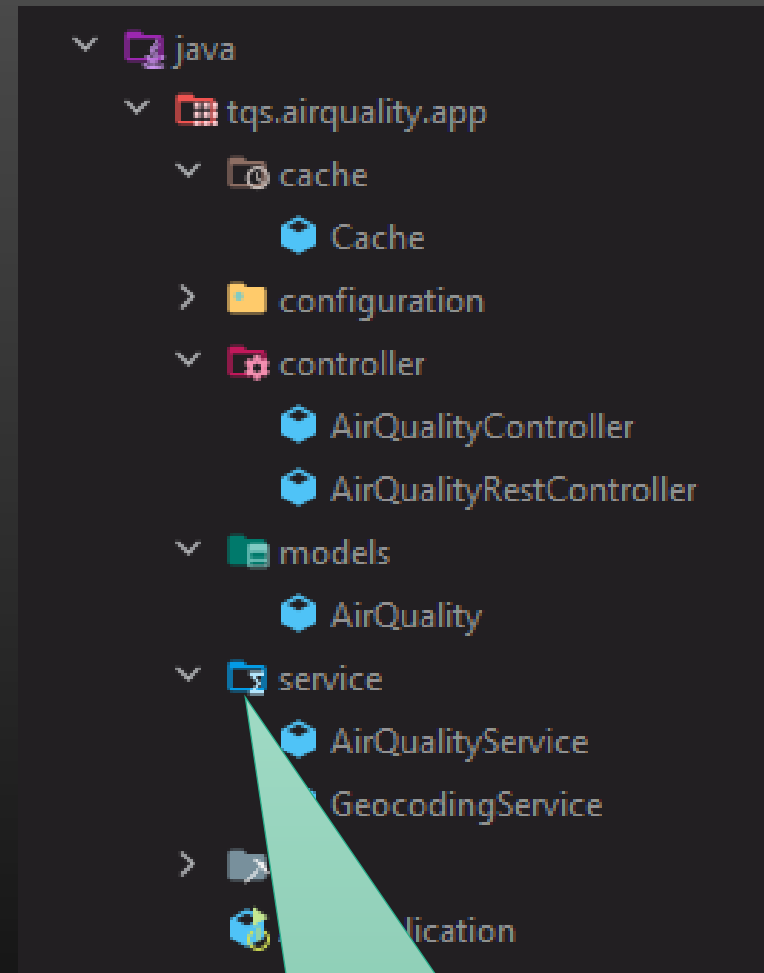
Arquitetura lógica

Organização geral da solução em blocos (*packages*)

Os *packages* podem representar agrupamentos muito diferentes.

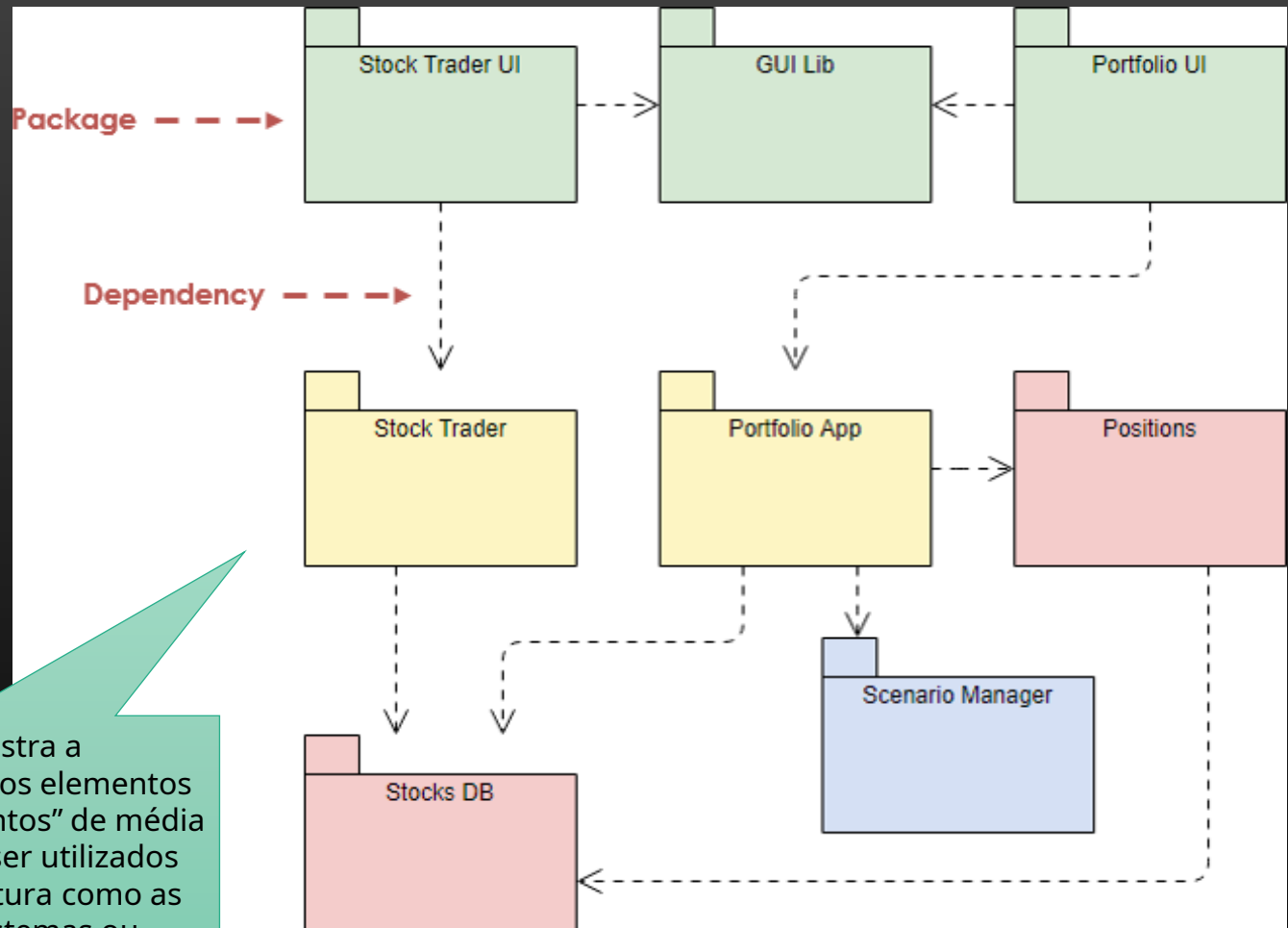
E.g.:

- *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.

Elementos do diagrama de pacotes

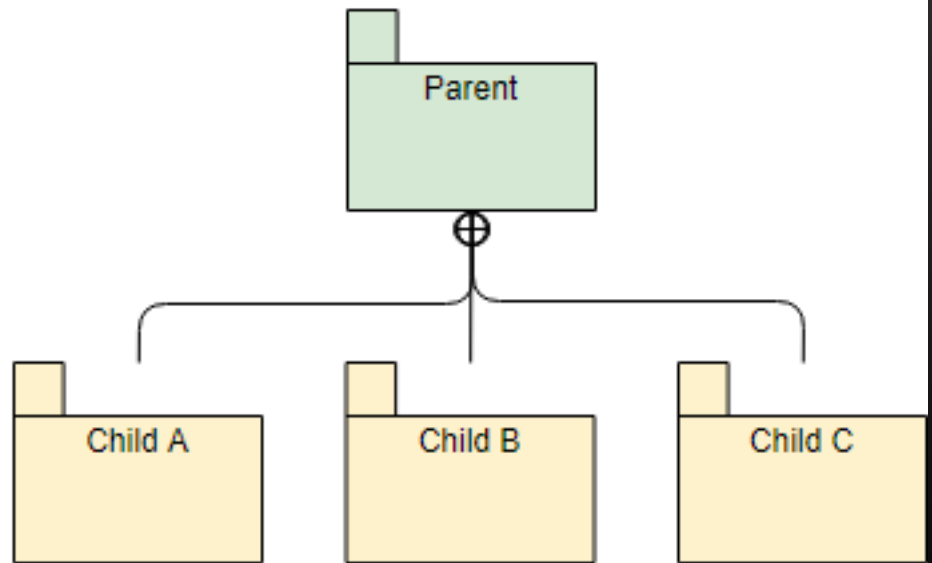
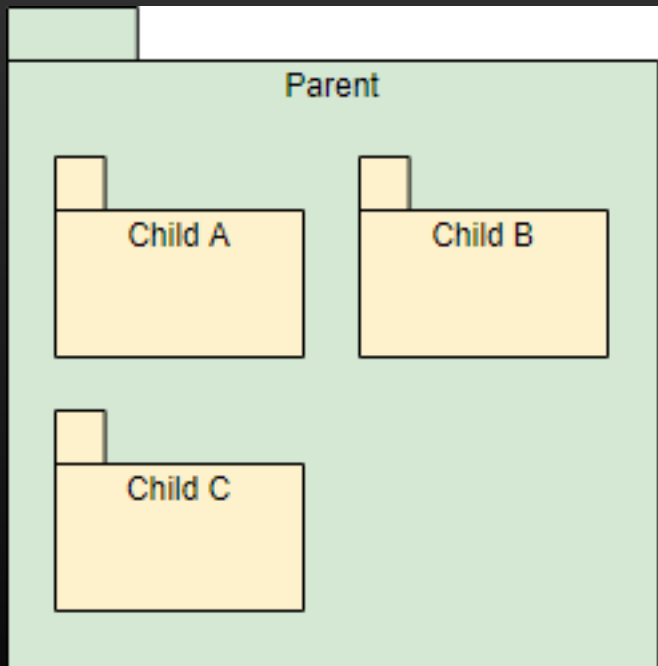


O diagrama de pacotes mostra a disposição e organização dos elementos do modelo em "agrupamentos" de média a larga escala que podem ser utilizados para mostrar tanto a estrutura como as dependências entre sub-sistemas ou módulos.

Comunicar a arquitetura lógica com pacotes (*packages*)

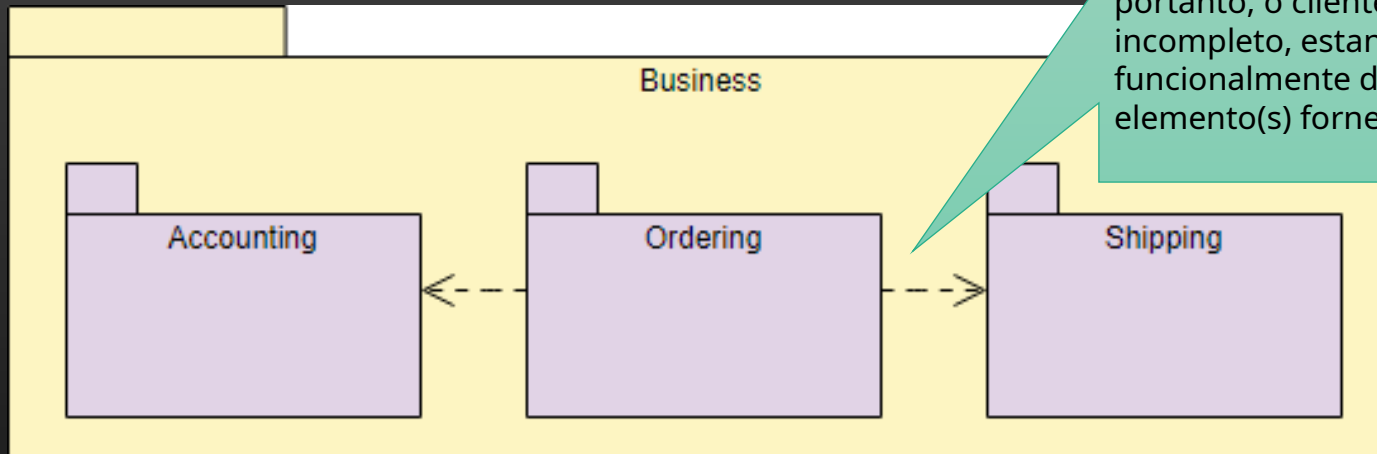
“Contido em”

- duas representações alternativas

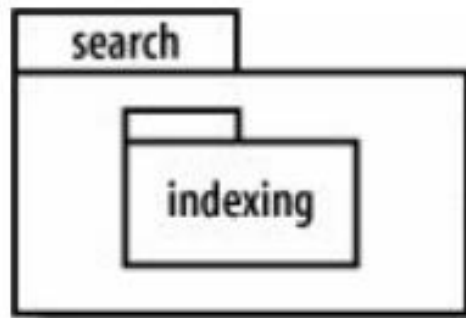


Associações entre pacotes

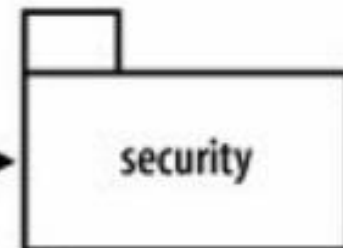
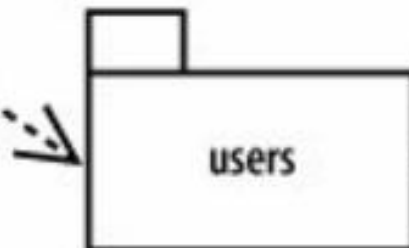
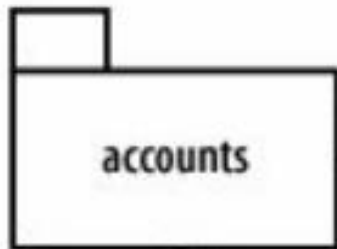
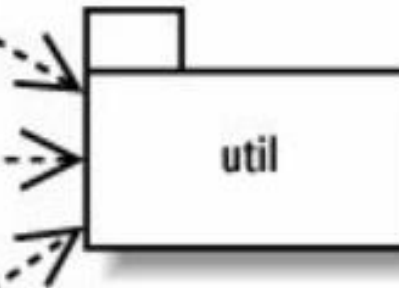
Dependência → transmite a ideia que partes de um elemento (pacote) precisam de (usar) partes de outro.
A dependência é designada como uma relação fornecedor - cliente, em que o fornecedor proporciona algo ao cliente e, portanto, o cliente é, em certo sentido, incompleto, estando semanticamente ou funcionalmente dependente do(s) elemento(s) fornecedor(es)



um pacote importa a funcionalidade de outro pacote



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



A arquitetura do sistema aborda diferentes perspectivas de análise

❶ Arquitetura lógica do software

Organização geral dos blocos de software

Independente da tecnologia de implementação

❷ Arquitetura 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,...)

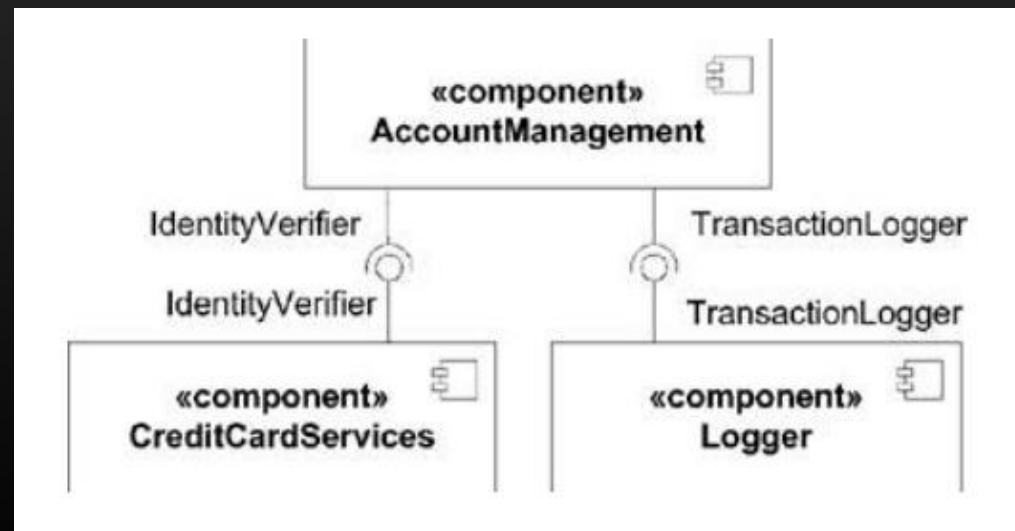
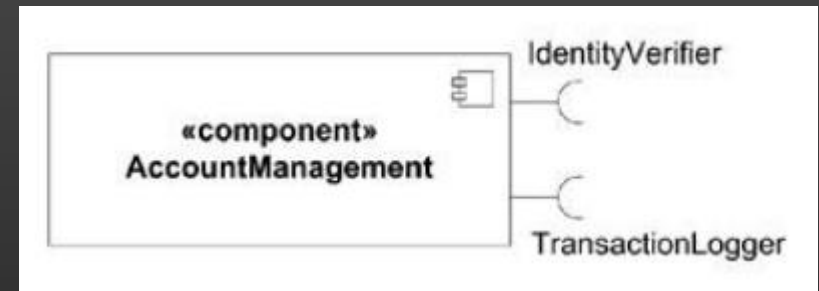
Componente


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







Customer Service
Kundenservice
Service Consommateurs
Servicio Al Consumidor
LEGO.com/service or dial

00800 5346 5555 :     

1-800-422-5346 :  

2x 300424	2x 4550348	2x 302423	2x 4567338	2x 4211525	2x 4504382	1x 4180536
1x 6022159	6x 302324	1x 4211043	2x 4646861	2x 4160857	2x 379426	1x 4180508
2x 4504381	2x 307024	2x 4210631	2x 3003944	2x 4161326	1x 302226	2x 4632571
	2x 306924	3x 6000606	2x 4244368	2x 245001	4x 614126	1x 4277932
			1x 6001197	1x 371001	1x 6021504	
			1x 4217722	1x 366601	1x 303426	
				1x 4508408		

6037713 / 6037714

LEGO.com

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 *application servers*)

The logo for MVNREPOSITORY, featuring the text "MVNREPOSITORY" in a bold, blue, sans-serif font. The "MVN" part is slightly larger and more prominent than the "REPOSITORY" part.

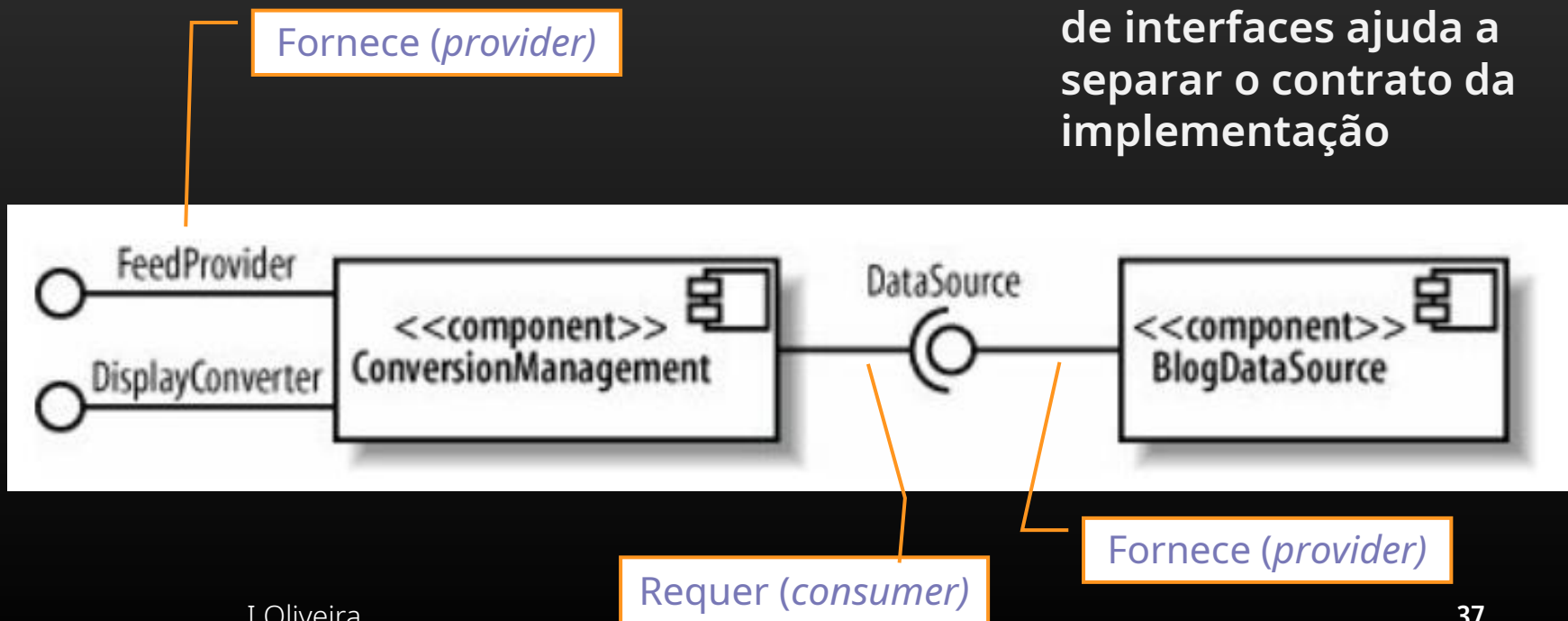
<https://mvnrepository.com>

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

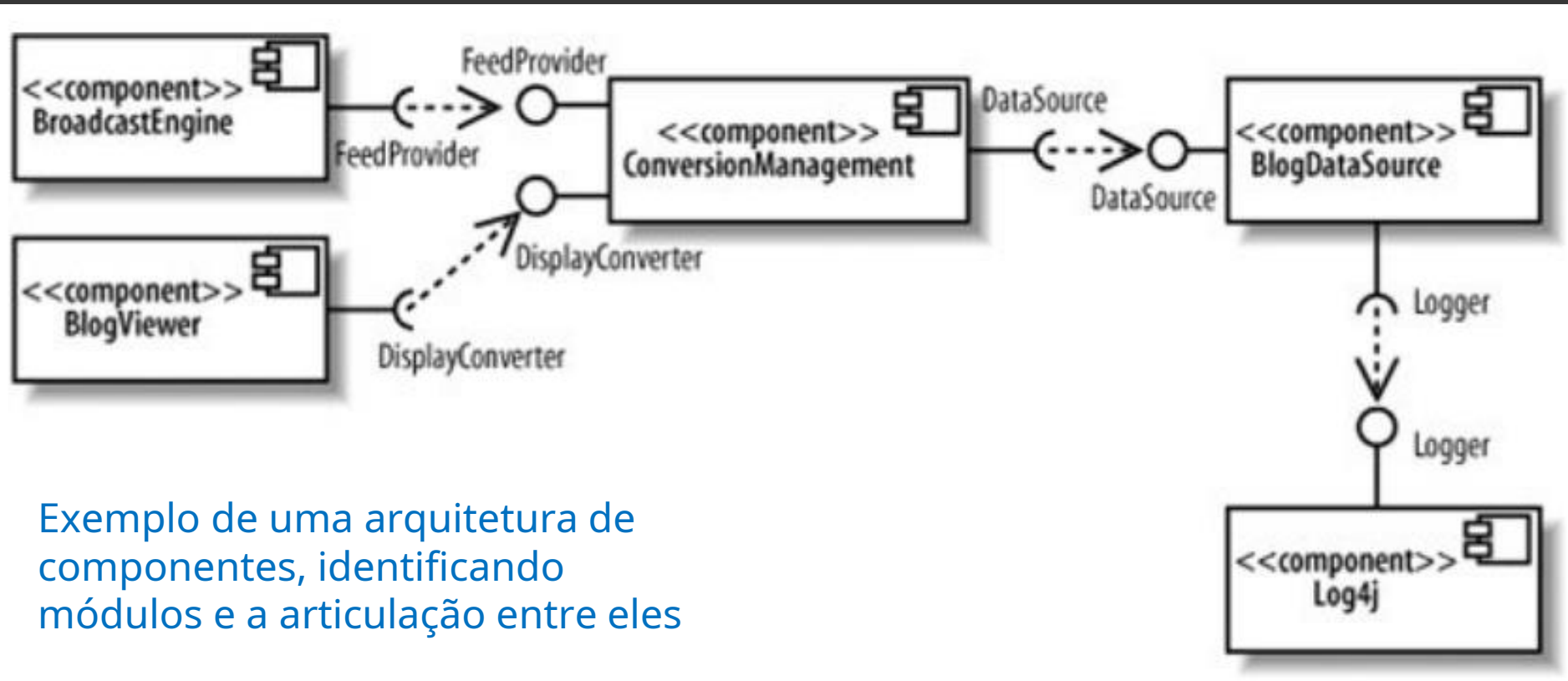
Solução modular com componentes

Com os componentes, pretende-se arquiteturas com baixo “coupling”

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

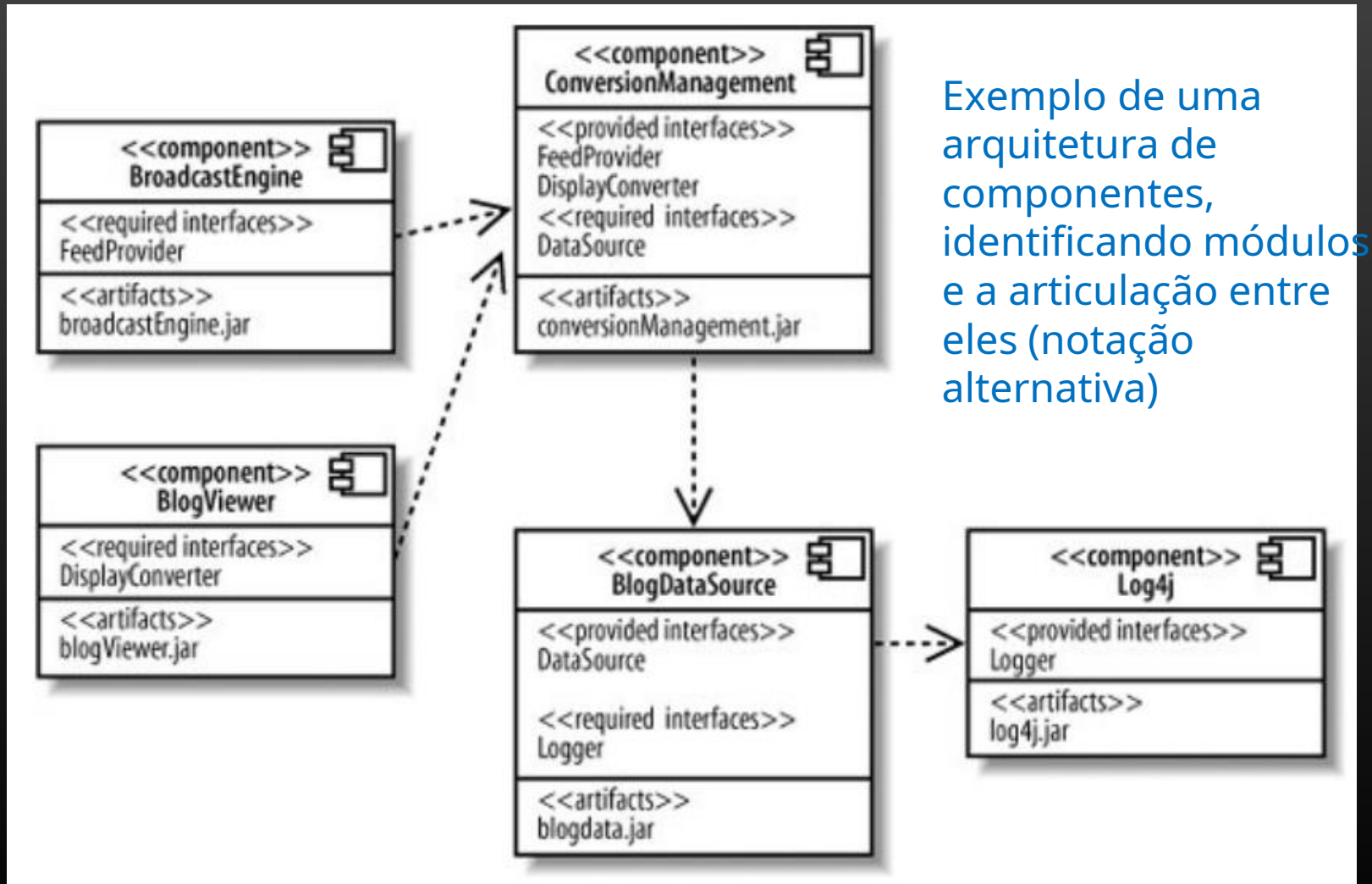


O mesmo modelo, notação ligeiramente diferente 1

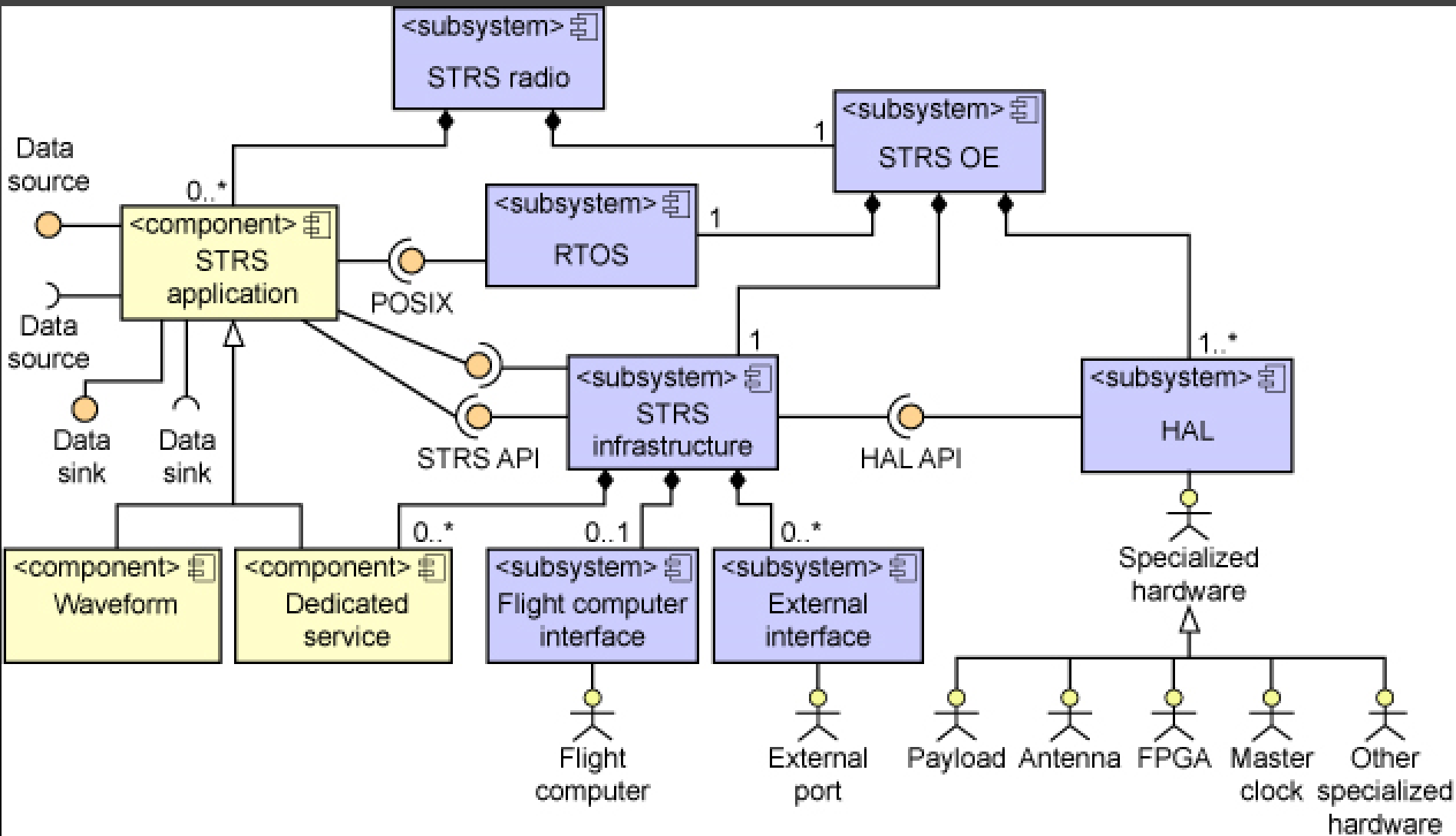


Exemplo de uma arquitetura de componentes, identificando módulos e a articulação entre eles

O mesmo modelo, notação ligeiramente diferente 2

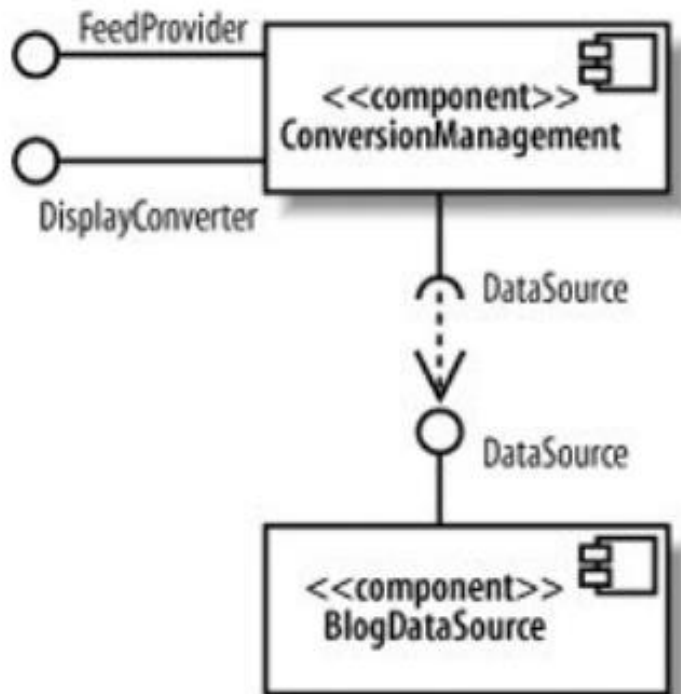


An example from NASA: Radio System

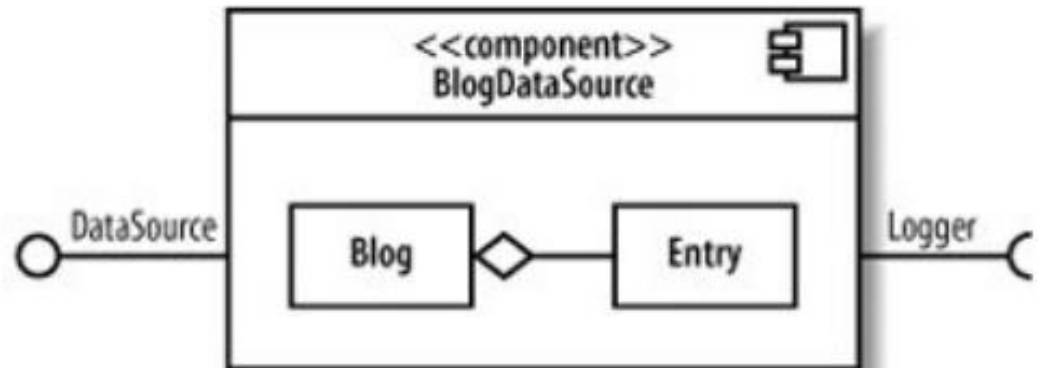


Notação “caixa fechada” / “caixa aberta”

Example Black-Box Component View



Example White-Box Component View



A arquitetura do sistema aborda diferentes perspectivas de análise

❶ Arquitetura lógica do software

Organização geral dos blocos de software

Independente da tecnologia de implementação

❷ Arquitetura 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

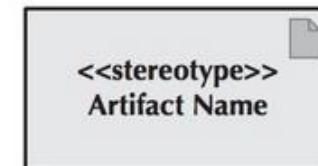
E.g.: executáveis, bibliotecas, configurações, scripts

A node:

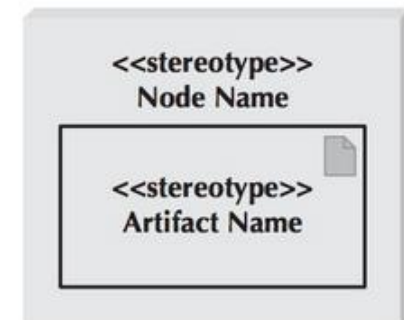
- Is a computational resource, e.g., a client computer, server, separate network, or individual network device.
- Is labeled by its name.
- May contain a stereotype to specifically label the type of node being represented, e.g., device, client workstation, application server, mobile device, etc.

**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.

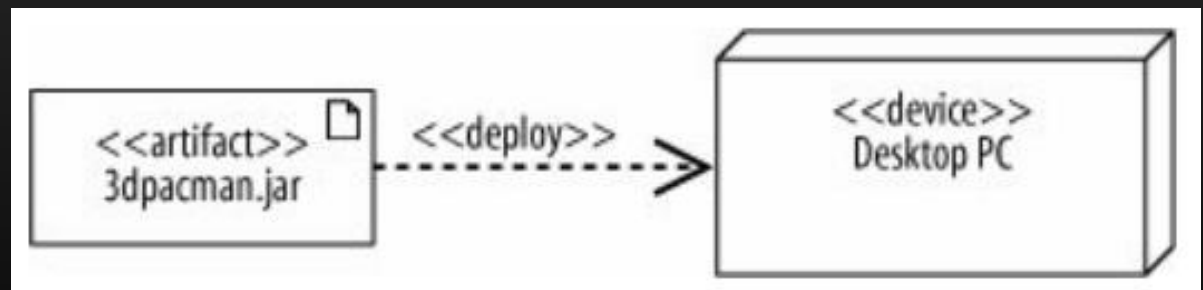
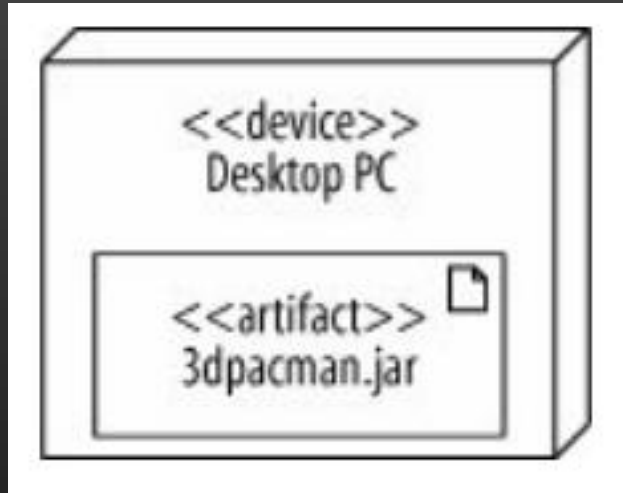
**A communication path:**

- 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).

<<stereotype>>

Os artefactos são executados em nós

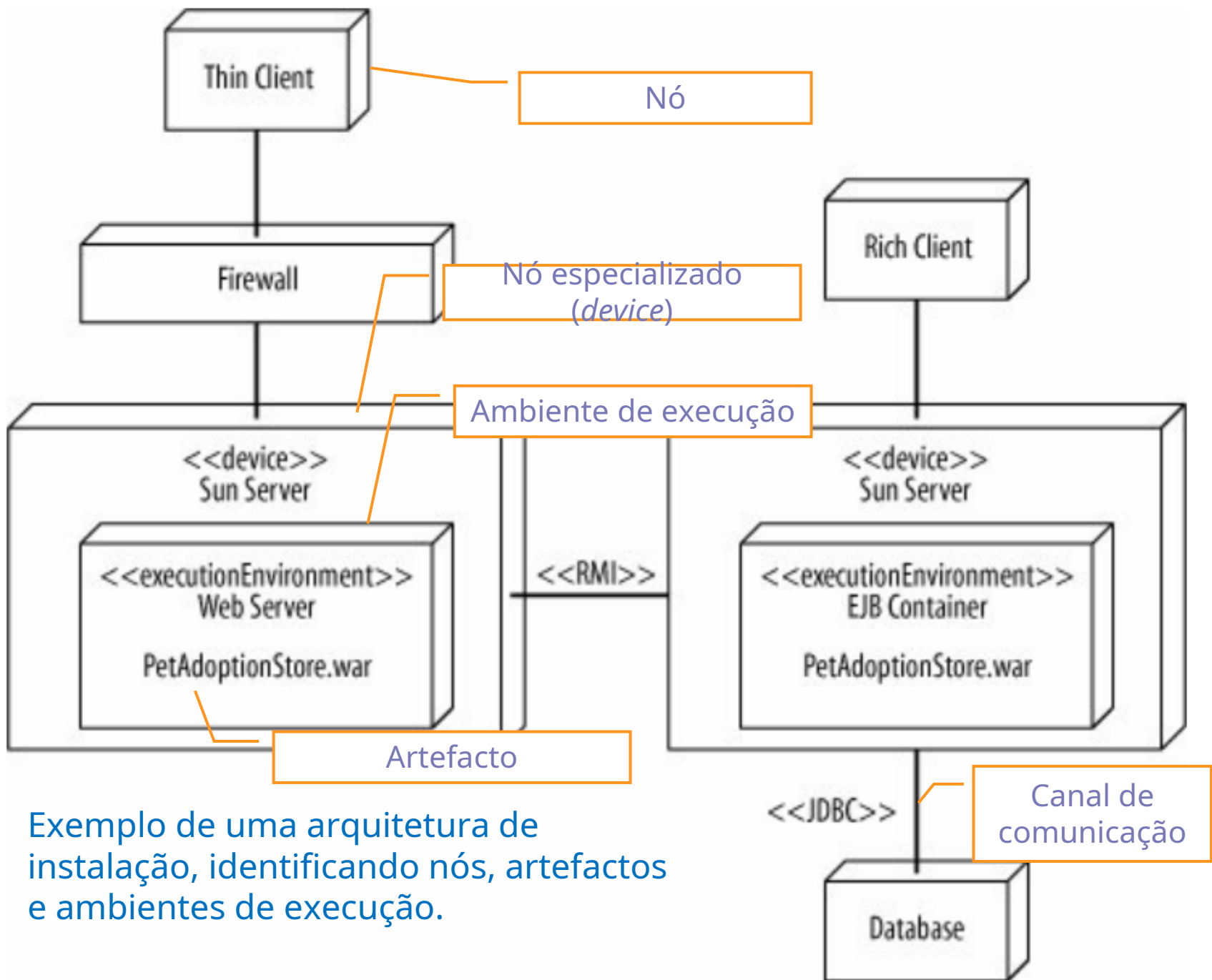
Notações alternativas.



Rastreabilidade até aos componentes



A relação “manifest” permite associar componentes a artefactos.



Exemplos (diagrama de instalação)

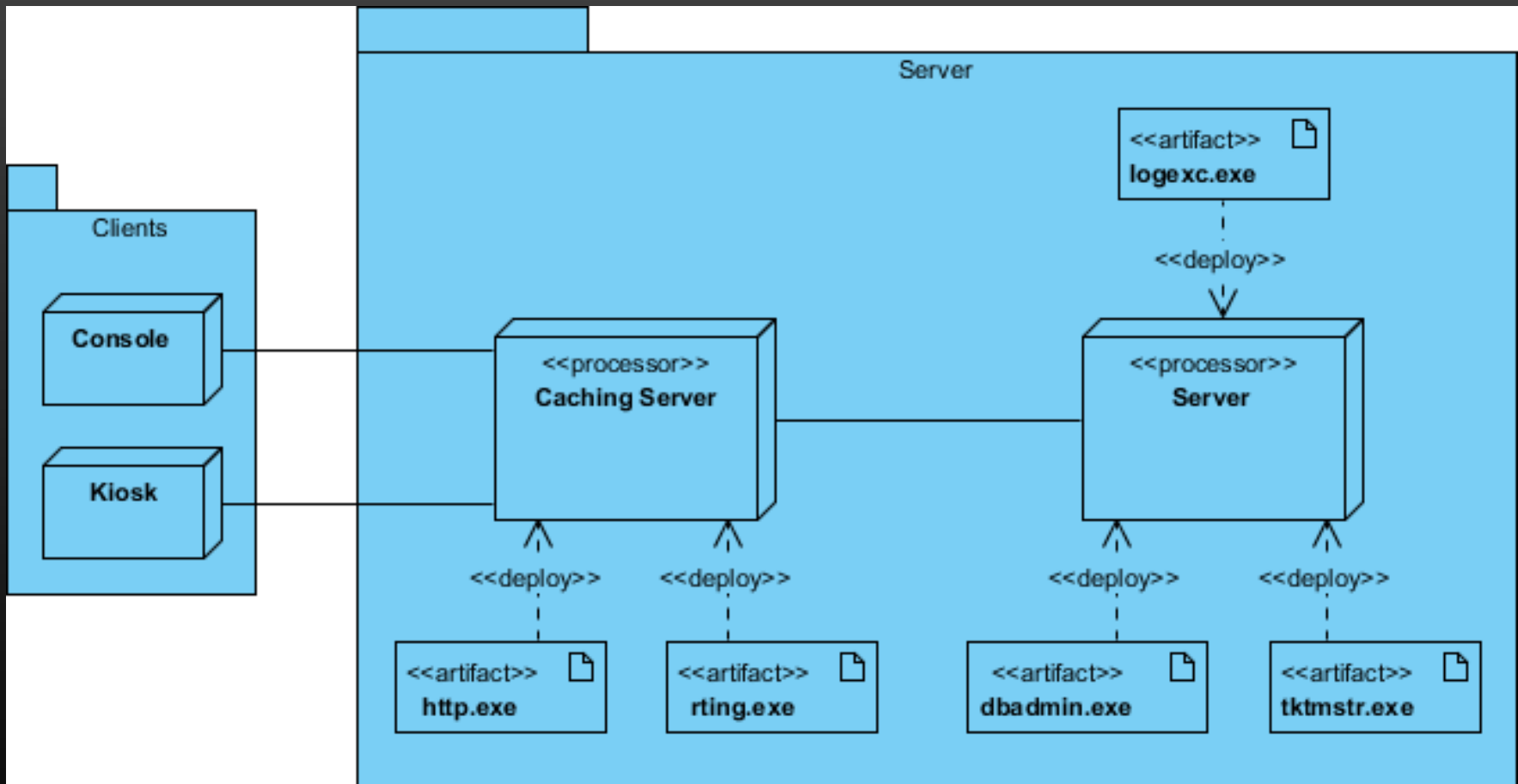
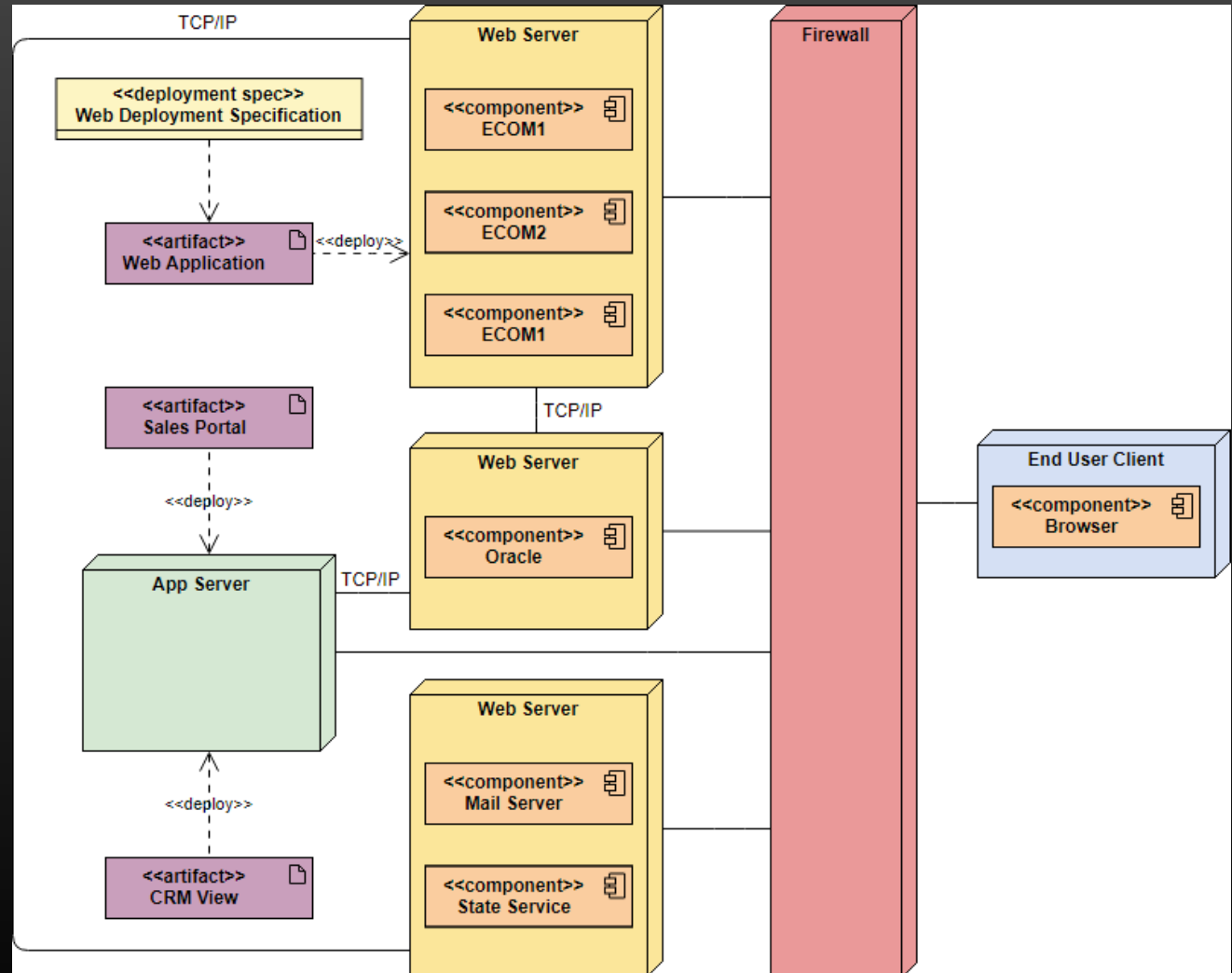


Diagrama de instalação



References

Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 7 & 11	<p>Visual Paradigm tutorials on UML Diagrams:</p> <ul style="list-style-type: none">• https://online.visual-paradigm.com/diagrams/tutorials/