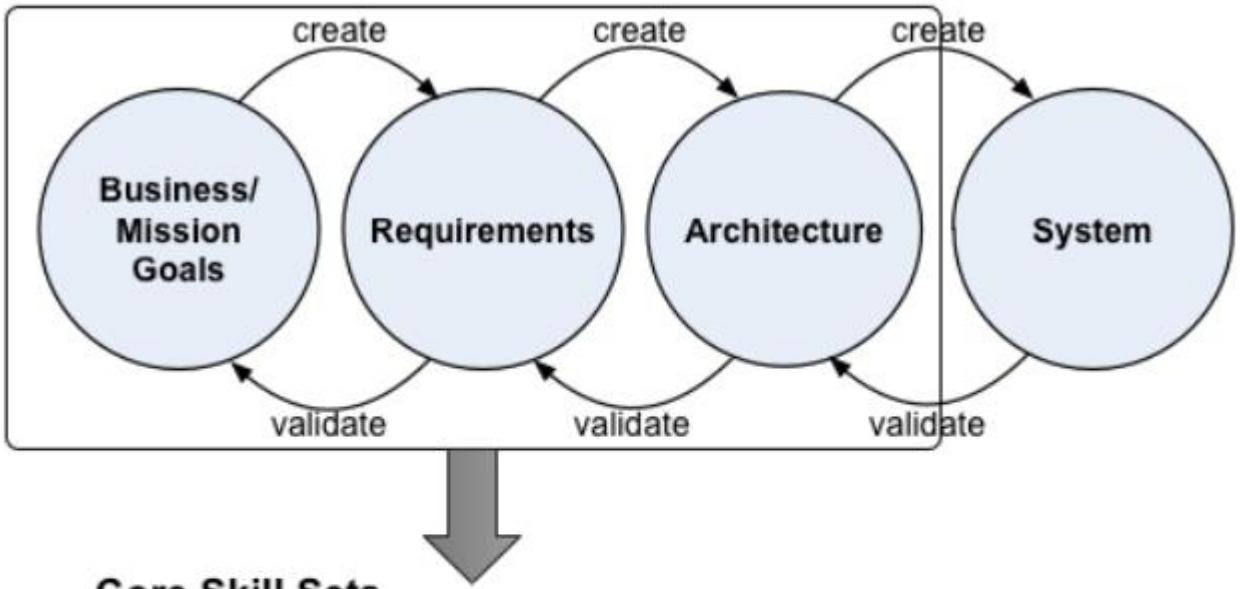
Vistas de Arquitetura - complemento

Ilídio Oliveira

v2023-05-09

universidade de aveiro
departamento de eletrónica,
telecomunicações e informática
deti

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

O que é a arquitetura de software?

A Arquitectura é um Conjunto de Estruturas (de Software)

Partes/elementos relevantes ligados por alguma relação.

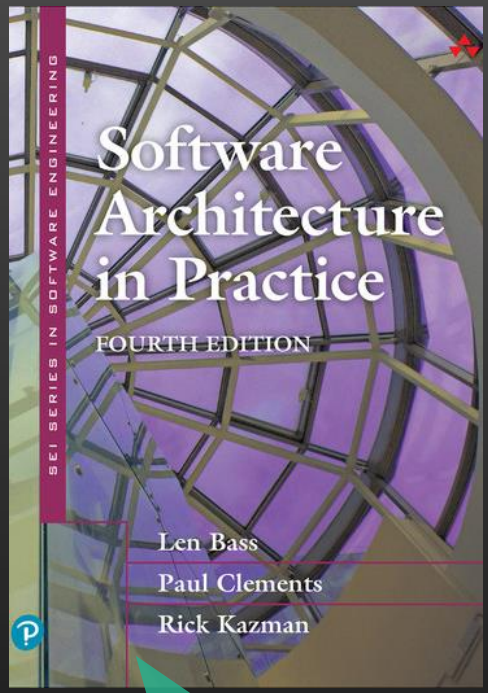
Arquitectura é uma Abstracção

Omite intencionalmente certas informações sobre os elementos que não são úteis para o raciocínio sobre o sistema

Detalhes privados dos elementos (têm a ver exclusivamente com a implementação interna) não são de arquitectura

Arquitectura Inclui Comportamento

O comportamento dos elementos engloba a forma como interagem uns com os outros e com o ambiente.



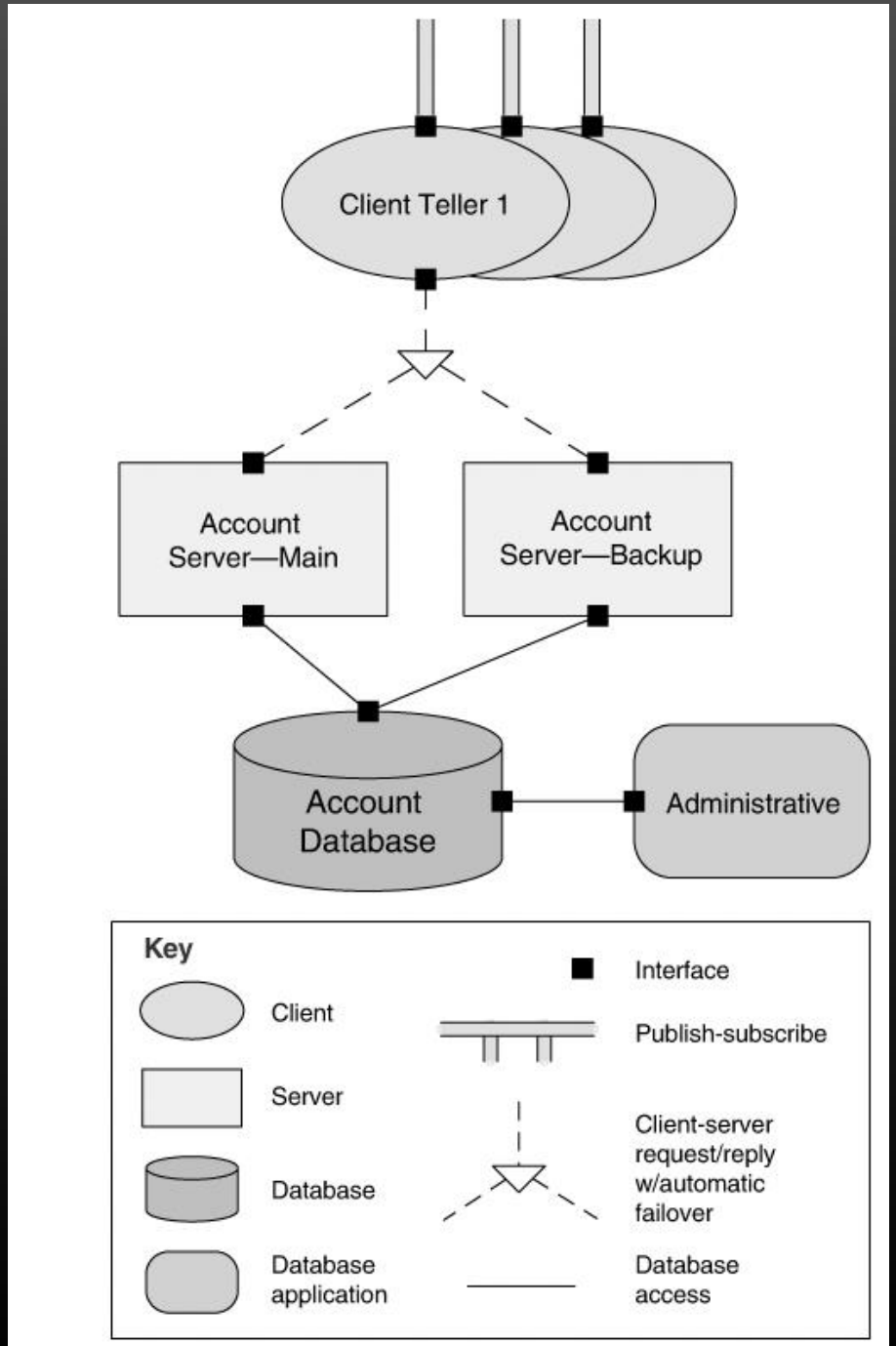
The software architecture of a system is the set of structures needed to reason about the system. These structures comprise software elements, relations among them, and properties of both.

Architecture is about reasoning-enabling structures.

Três tipos de estruturas #1

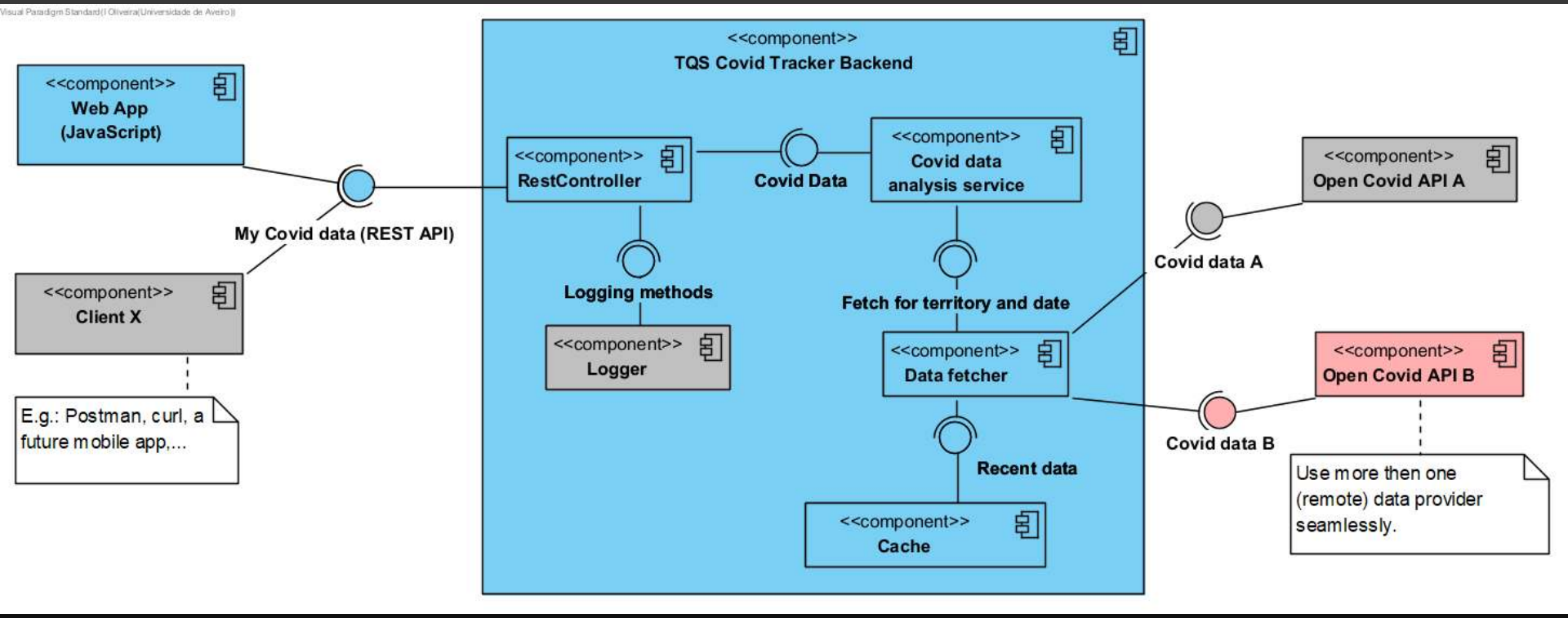
1/ As estruturas de componentes e conectores (*Component-and-connector* - C&C) focam-se na forma como os elementos interagem uns com os outros em tempo de execução para realizar as funções do sistema.

Descrevem como o sistema é estruturado como um conjunto de elementos que têm comportamento em tempo de execução (componentes) e interacções (conectores).



Caso de exemplo: componentes-conetores (C&C)

Service structure: as unidades aqui são serviços (em runtime) que interoperam através de um mecanismo de coordenação.

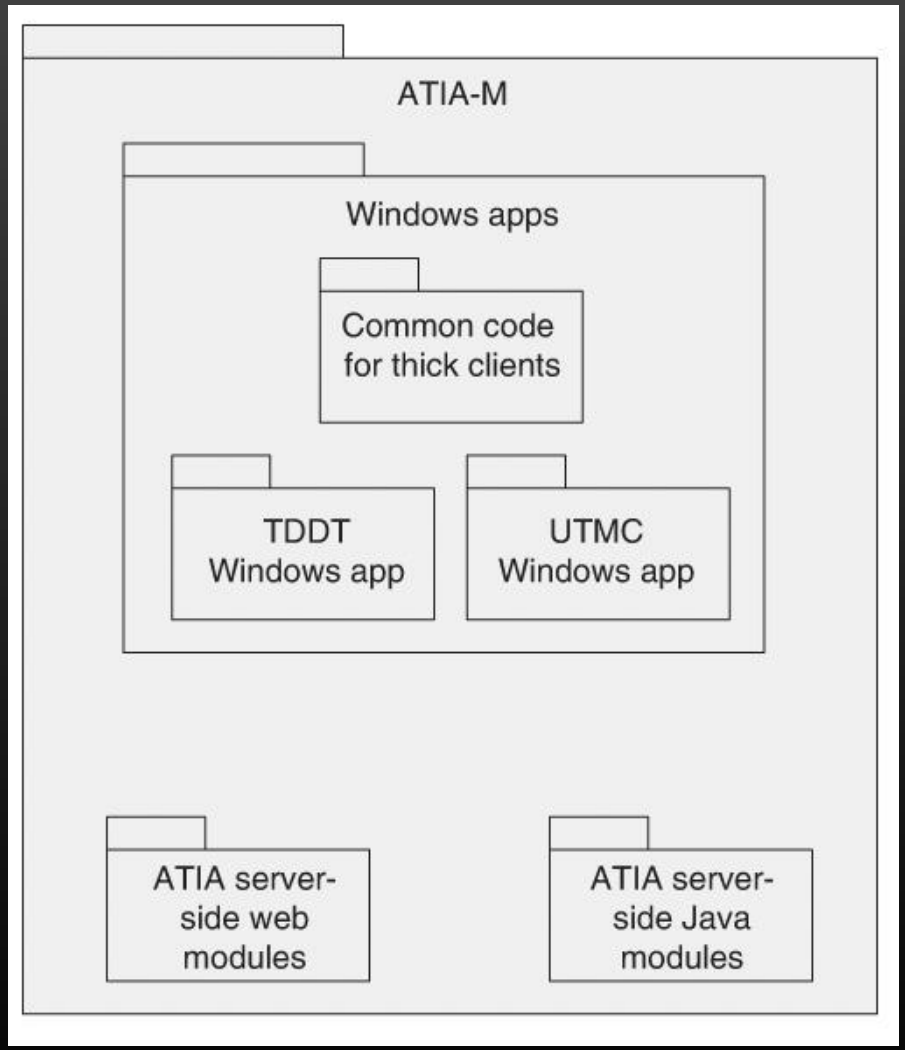


Tipos de estruturas #2

2/ As estruturas do tipo módulo dividem os sistemas em unidades de implementação. Os módulos mostram como um sistema pode ser dividido como um conjunto de unidades de código ou de dados, que devem ser implementadas ou adquiridas.

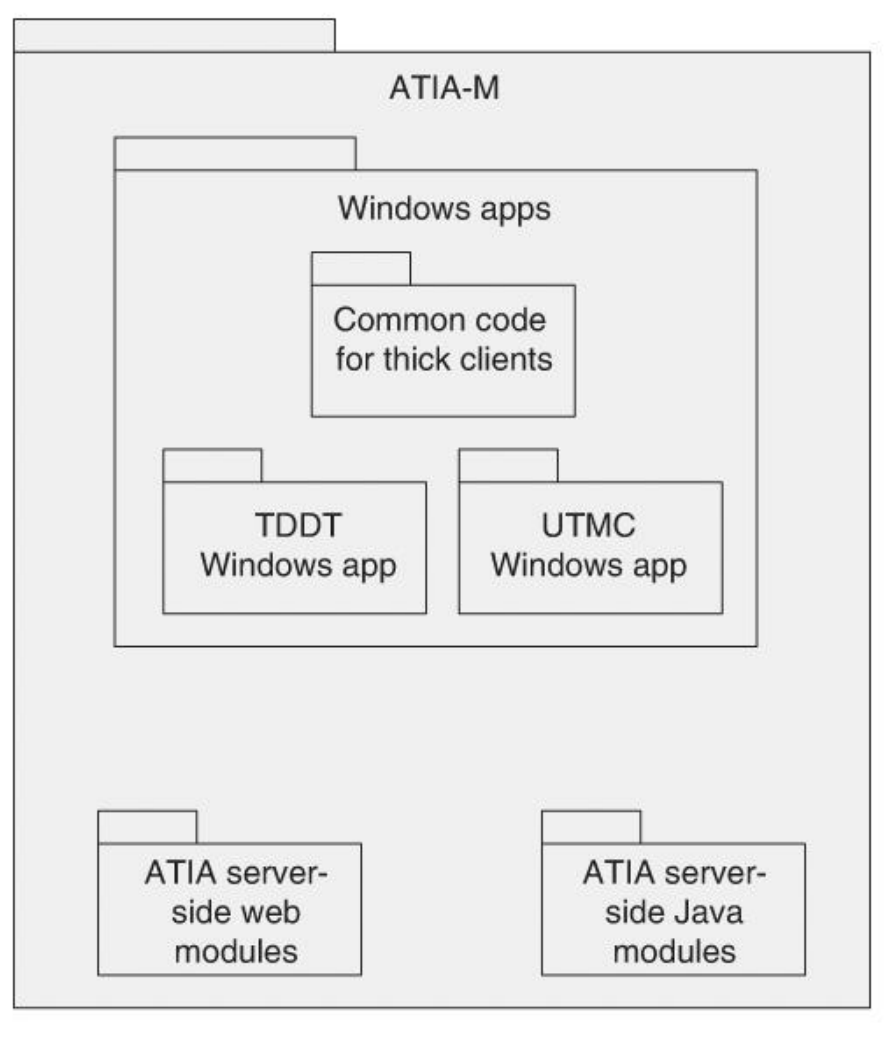
Os módulos são usados para atribuir trabalho às equipas.

Elementos do tipo módulo podem ser classes, pacotes, camadas, ou meramente divisões de funcionalidade, todas elas refletindo unidades de implementação.



Exemplo de estrutura: módulos

Decomposição (relação "is-a-submodule-of")



Tipos de estruturas #3

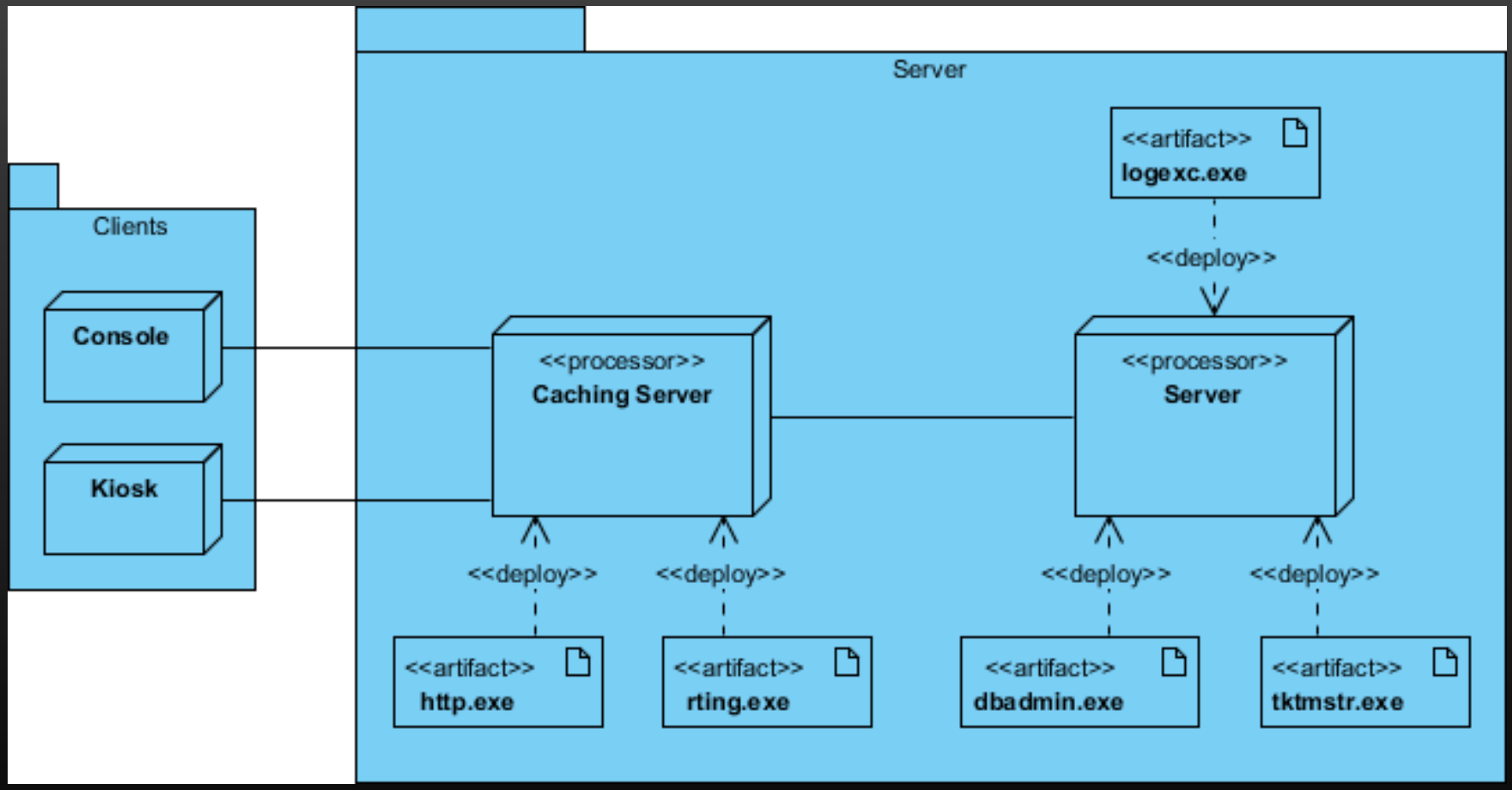
3/ As estruturas de alocação *(Allocation structures)* estabelecem o mapeamento das estruturas de software nas estruturas de não-software do sistema, tais como os seus ambientes de desenvolvimento, teste, e execução.

As estruturas de alocação respondem a questões como as seguintes:

- Em que processador(es) cada elemento de software é executado?

- Em que directórios ou ficheiros é cada elemento armazenado durante o desenvolvimento, teste e construção do sistema?

Vista de alocação na UML (d. instalação/*deployment*)



Requisitos com impacto nas decisões de arquitetura

As decisões de arquitetura são conduzidas pelos requisitos

Para o arquitecto, nem todos os requisitos são idênticos.

Alguns têm um impacto muito mais profundo na arquitectura do que outros.

Um requisito de arquitectura significativo é um requisito que terá um efeito profundo na arquitectura - ou seja, a arquitectura pode ser substancialmente diferente conforme a presença/ausência de tal requisito.

Requisitos significativos para a arquitetura

Architecturally significant requirements: requirements that play an important role in determining the architecture of the system. Such requirements require special attention. Not all requirements have equal significance with regards to the architecture.

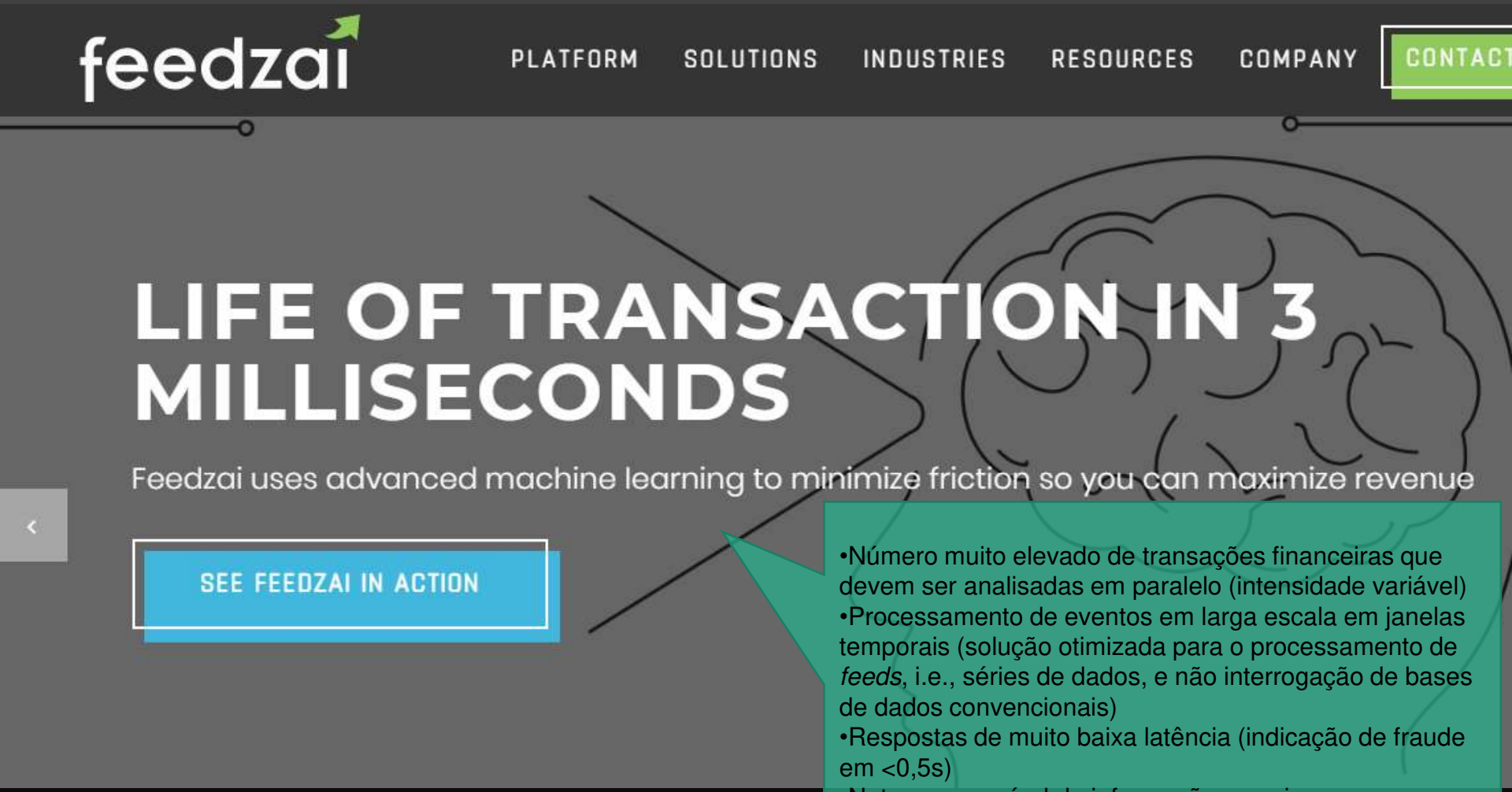
Typically, these are requirements that are technically challenging, technically constraining, or central to the system's purpose

The following are good examples of Architecturally Significant Requirements:

- The system must stream video media worldwide, with a delay to start playing <5 seg for 90% of accesses.
- The mobile application should be available both for Android and iOS with similar capabilities.
- The system must deploy on Microsoft Windows XP and Linux.
- The system must encrypt all network traffic.
- The ATM system must dispense cash on demand to validated account holders with sufficient cleared funds.

https://sweet.ua.pt/ico/OpenUp/OpenUP_v1514/core.tech.common.extend_supp/guidances/concepts/arch_significant_requirements_1EE5D757.html?nodeId=7dddf10c

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

Os requisitos conduzem a arquitetura

E.g.: desempenho como atributo de qualidade

Figure 9.1 gives an example concrete performance scenario: *Five hundred users initiate 2,000 requests in a 30-second interval, under normal operations. The system processes all of the requests with an average latency of two seconds.*

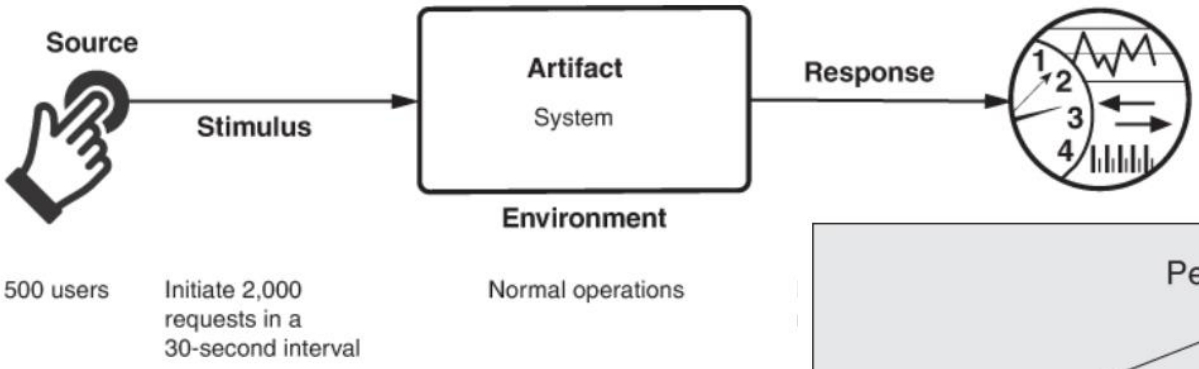


Figure 9.1 Sample performance scenario

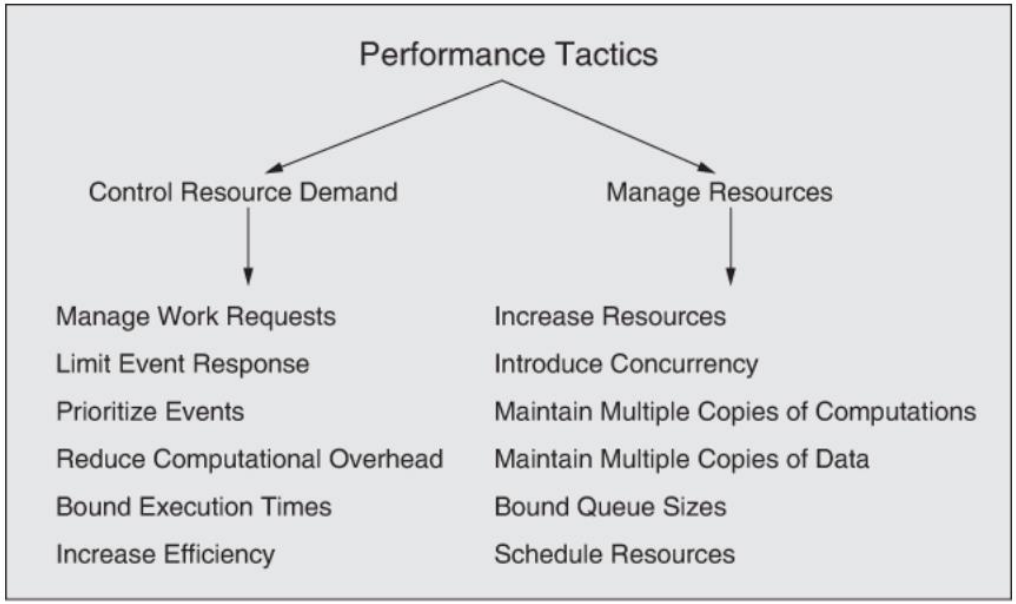


Figure 9.3 Performance tactics

Os requisitos conduzem a arquitetura

E.g.: disponibilidade como atributo de qualidade

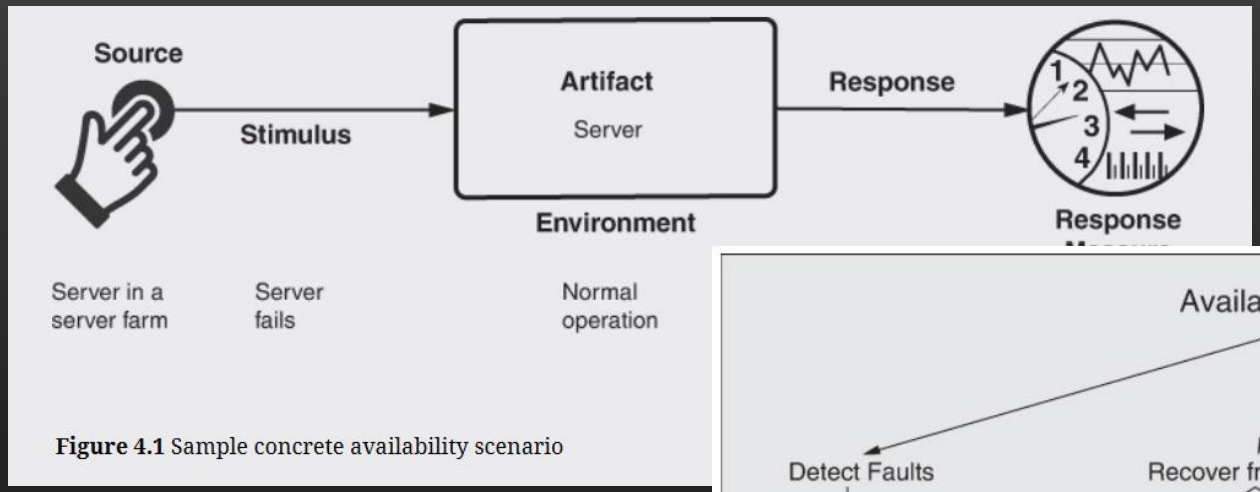
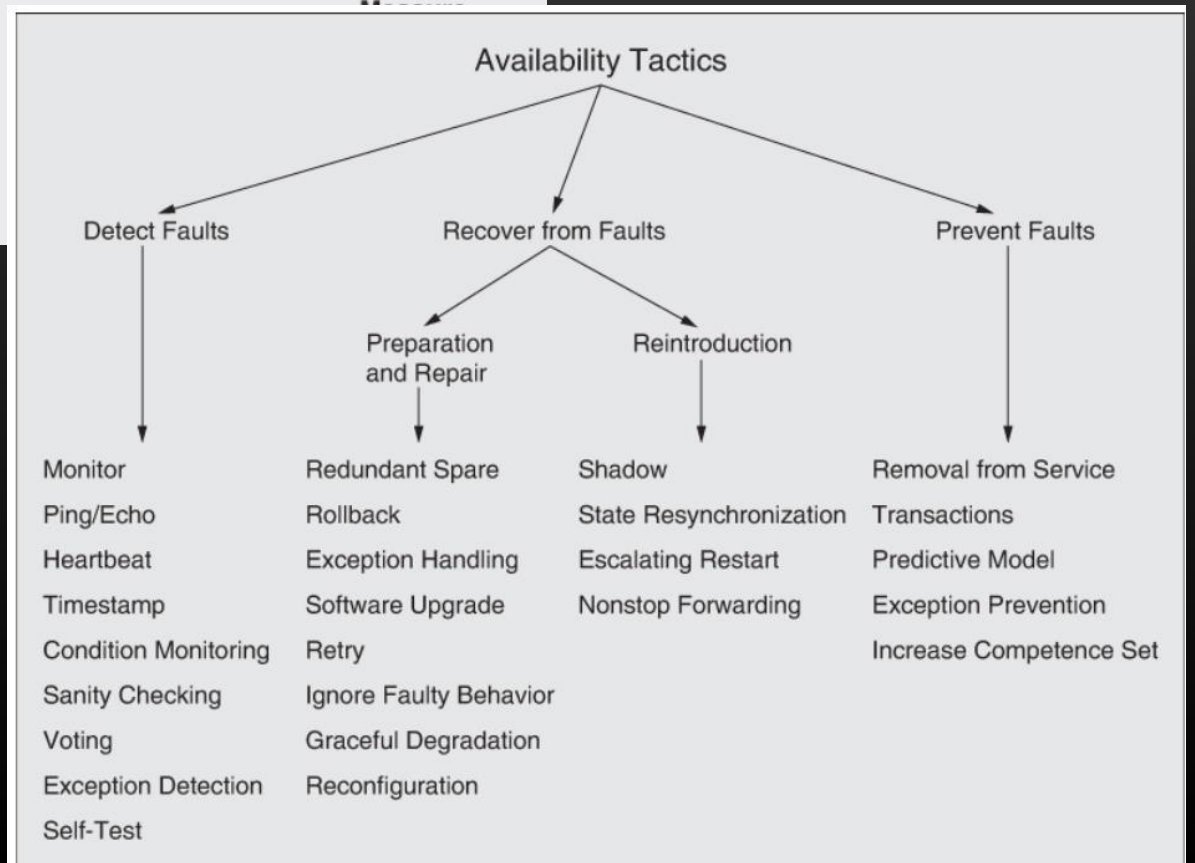
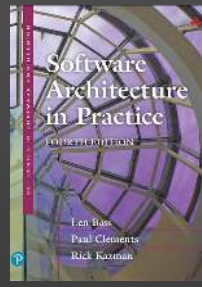


Figure 4.1 Sample concrete availability scenario



A arquitectura de software é importante! Razões técnicas e não técnicas



Uma arquitectura irá limitar ou potenciar os atributos de qualidade de um sistema.

As decisões tomadas numa arquitectura permitemr pensar e gerir a mudança à medida que o sistema evolui.

A análise de uma arquitectura permite uma visão antecipada das qualidades de um sistema.

Uma arquitectura documentada promove a comunicação entre as pessoas envolvidas (stakeholders).

A arquitectura é percursora das primeiras (e mais fundamentais) decisões de projecto, e as mais difíceis de alterar.

Uma arquitectura pode fornecer a base para o desenvolvimento incremental.

Uma arquitectura é o artefacto chave que permite ao arquitecto e ao gestor do projecto argumentarem sobre custos e prazos.

Uma arquitectura pode ser criada como um modelo transferível e reutilizável que forma o cerne de uma linha de produtos.

O desenvolvimento baseado na arquitectura centra a atenção na conjugação de componentes, em vez de simplesmente na sua criação.

Ao restringir alternativas de desenho, a arquitectura canaliza a criatividade dos programadores de forma produtiva.

Uma arquitectura pode servir de base para a formação de um novo membro da equipa.

Estilos de arquitetura (introd.)

Estilos de arquitetura

Apesar da enorme diversidade dos produtos de software, é possível falar-se em certos "padrões de arquitetura" nas soluções empresariais

Um "padrão de arquitetura" reflete uma abordagem-tipo, i.e., uma maneira de organizar a solução que se provou adequada para certos tipos de projetos.

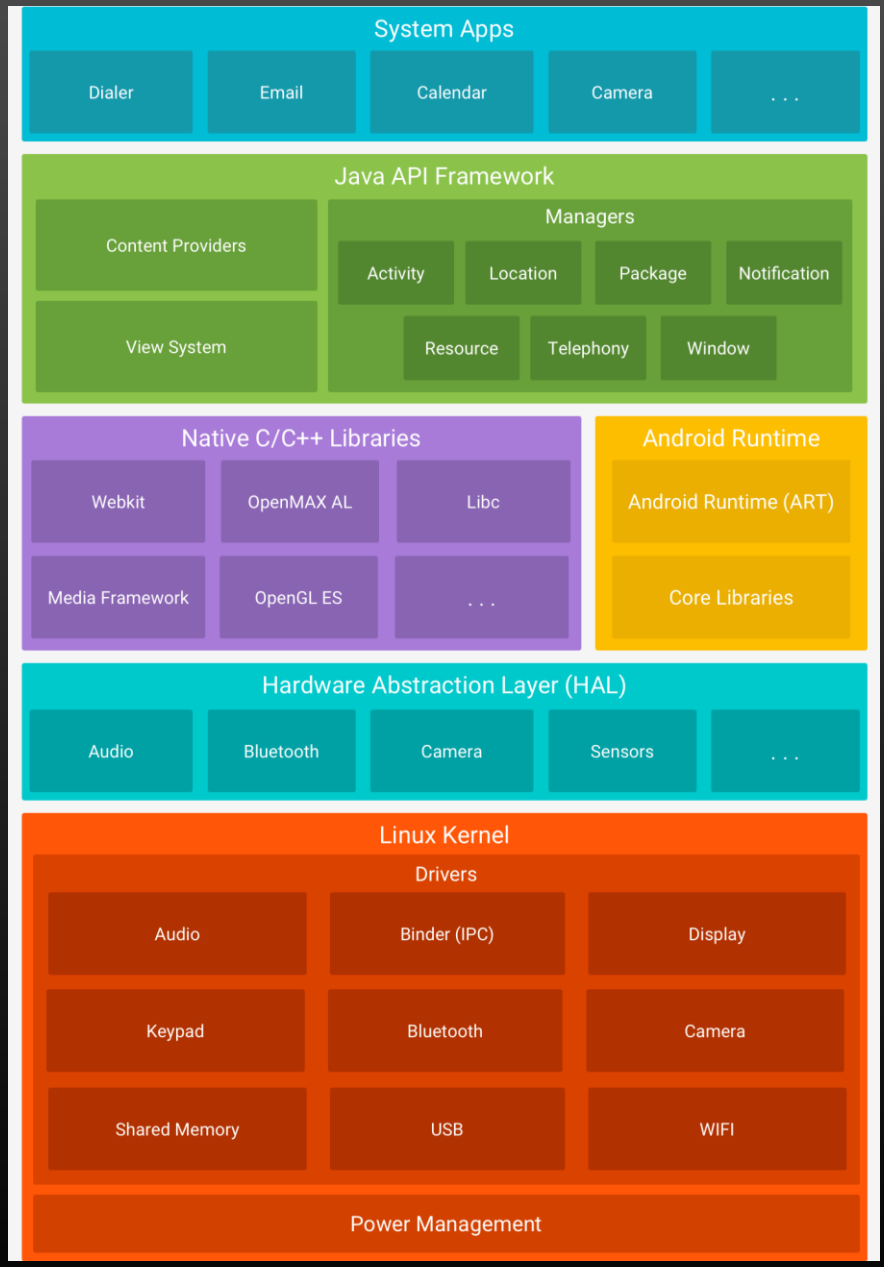
Exemplos:

- Microkernel Architecture
- Layered Architecture
- Event-Driven Architecture
- Service-oriented Architecture
- Microservices Architecture
- ...

Qual é o “padrão”?

Uma arquitetura por camadas

- Camadas “debaixo” fornecem serviços às camadas “de cima”
- Cada camada comunica apenas com as camadas adjacentes (consome serviços da de baixo, oferece serviços à de cima)
- Há geralmente um crescendo no nível de abstração: camadas abaixo mais próximas do hardware/armazenamento; camada superior voltada para o utilizador.
- Cada camada pode ser organizada em módulos, mostrando componentes no mesmo nível de abstração



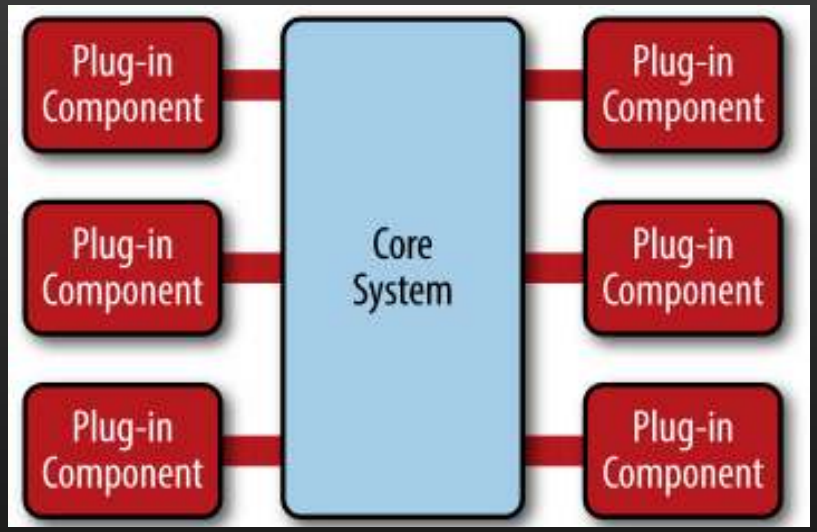
https://developer.android.com/guide/platform

Arquitetura *Microkernel*

Núcleo da aplicação (estável) **+** ***plugins*** (dinâmico)

Novas funcionalidades da aplicação são adicionadas como plug-ins

Vantagens: extensibilidade, separação de assuntos e isolamento.



Processamento de eventos (*event-driven)*

O padrão de “arquitetura por eventos” é um estilo de arquitetura assíncrono e distribuído, usado em aplicações que precisam de ser muito escaláveis (e.g.: processar milhares de leituras de sensores por segundo)

A arquitetura por eventos é composta por componentes de processamento de eventos altamente dissociados (*decoupled*) e especializados (dedicados a um tipo de processamento) que recebem e processam os eventos.

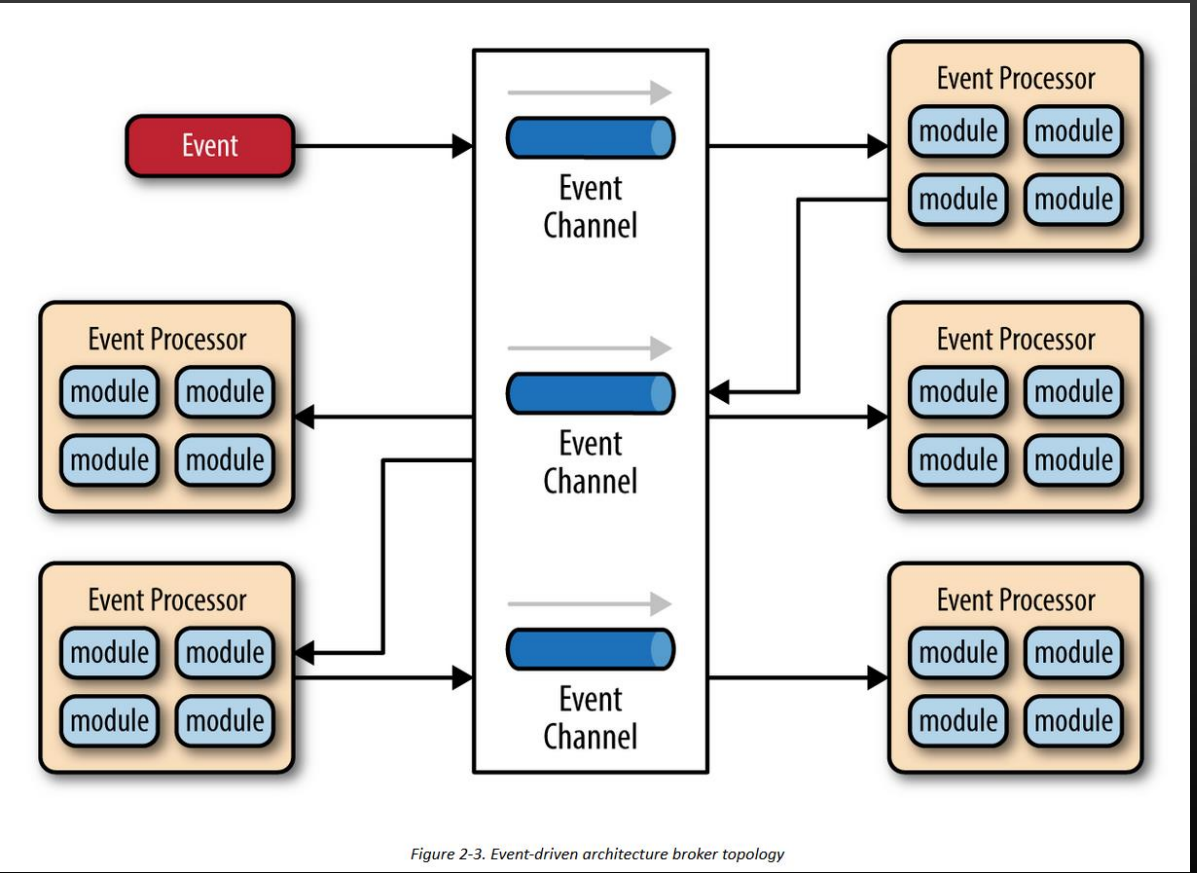


Figure 2-3. Event-driven architecture broker topology

Um exemplo da indústria: uma solução de M2M (*machine-to-machine*)

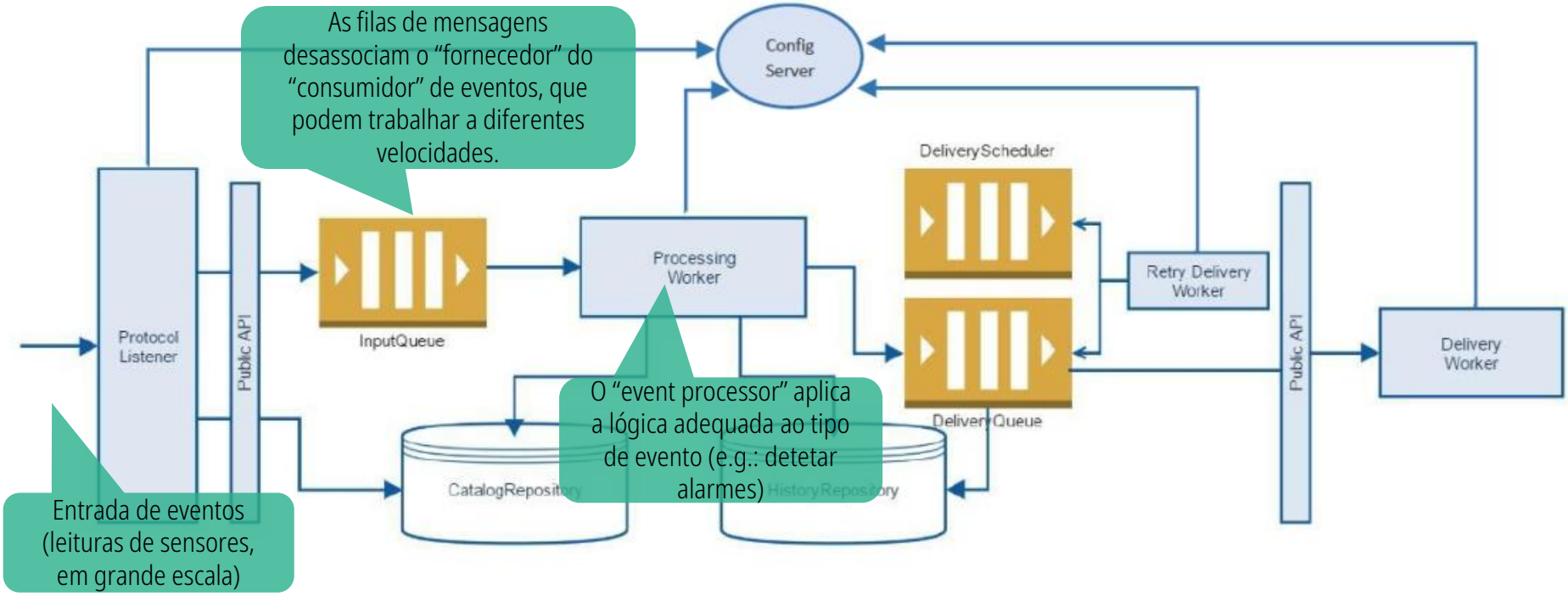


Figura 5.2: Arquitetura e componentes da SmartIOT.

Mais info: https://repositorio-aberto.up.pt/handle/10216/109650

Arquitetura por camadas

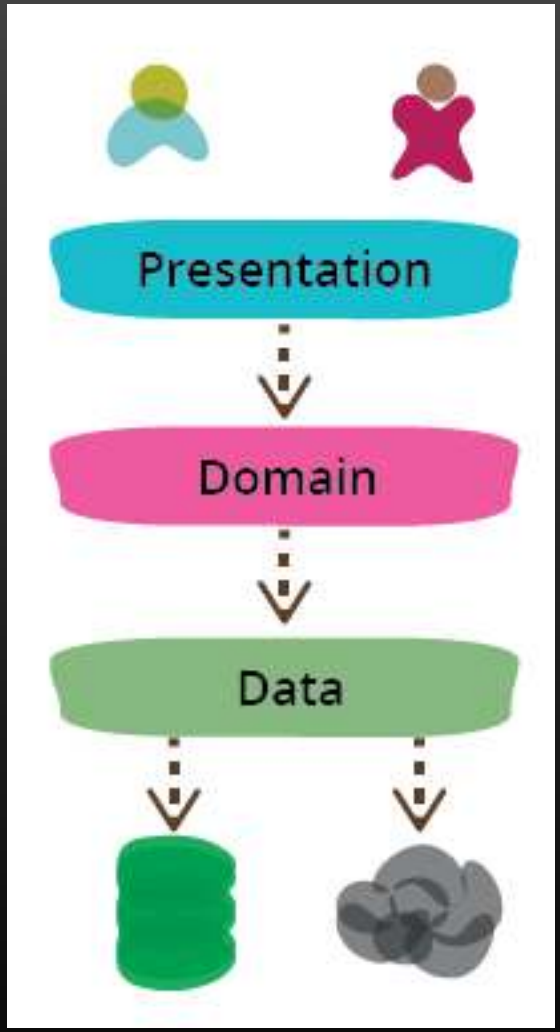
Divisão modular da solução de software em camadas/níveis de abstração.

As camadas são sobrepostas

Cada camada tem uma especialização

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

Não se pode saltar camadas: os componentes, em cada camada, “falam” com as camadas adjacentes.



https://martinfowler.com/bliki/PresentationDomainDataLayering.html

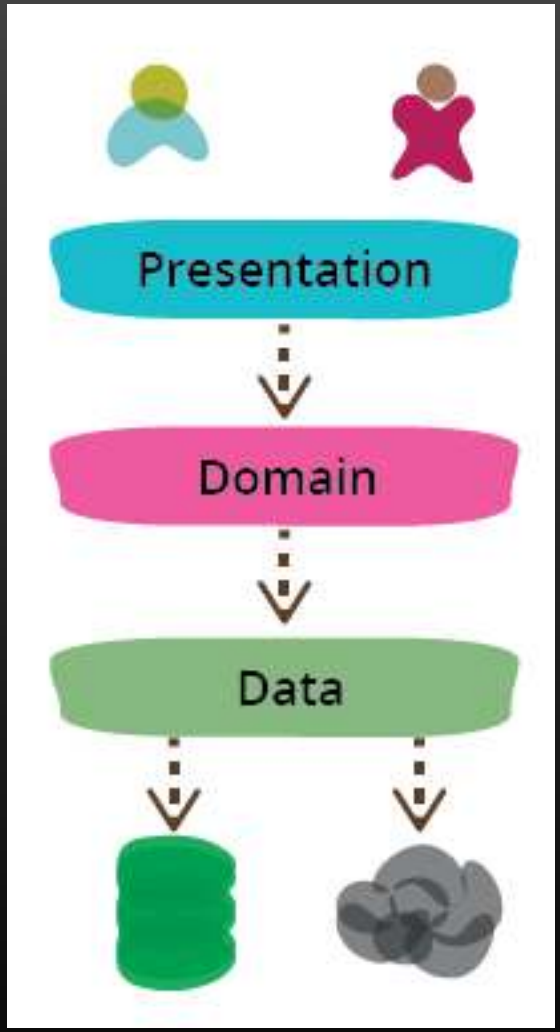
A arquitetura de 3 camadas

Uma das formas mais comuns de modularizar um programa orientado para a gestão de informação é separá-lo em três camadas principais:

1) apresentação (*user interface*, UI),
2) lógica de domínio (a.k.a. *business logic*)
3) acesso a dados.

Desta forma, é normal organizar uma aplicação web em três camadas:

- Uma camada web que sabe lidar com pedidos HTTP e preparar as páginas HTML,
- uma camada com a lógica de negócio, que contém regras (algoritmos), validações e cálculos,
- e uma camada de acesso de dados que assegura como gerir os dados de forma persistente, numa base de dados ou com integração de serviços remotos.



https://martinfowler.com/bliki/PresentationDomainDataLayering.html

Camadas e partições (modularização)

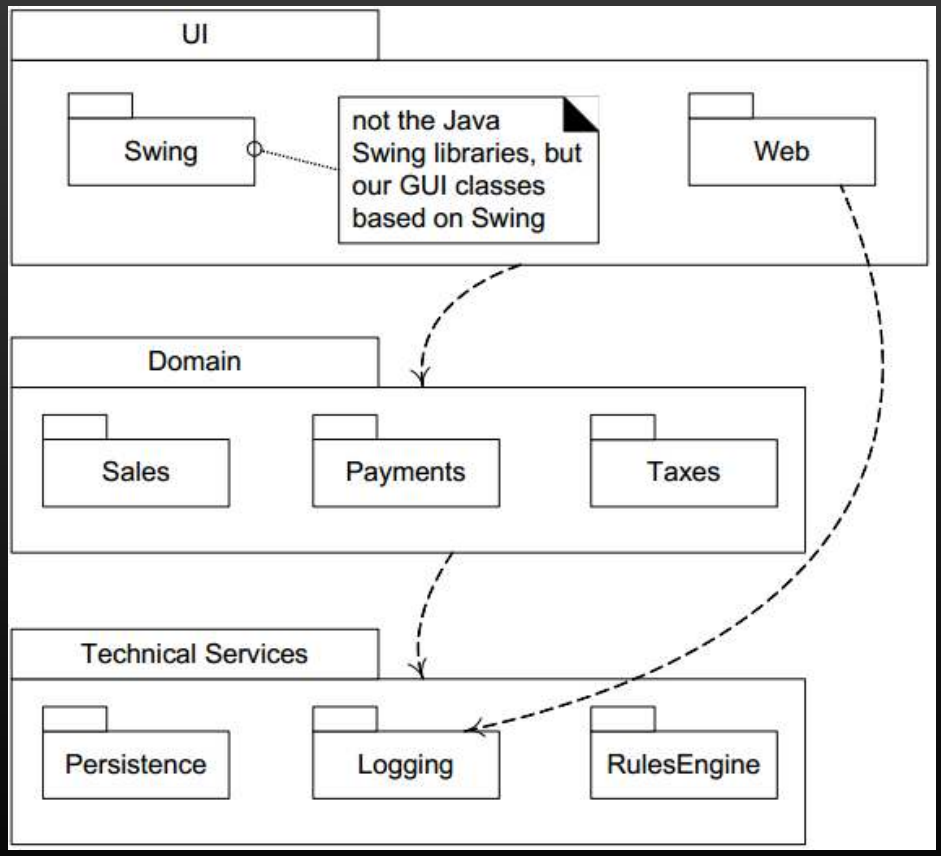
Camadas verticais:

- divisão por níveis de abstração

Partições horizontais:

- Módulos dentro de uma camada

E.g.: a lógica do domínio está dividida em grandes módulos funcionais especializados, agrupando as programação relativa às Vendas, aos Pagamentos e à Fiscalidade.



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)

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)

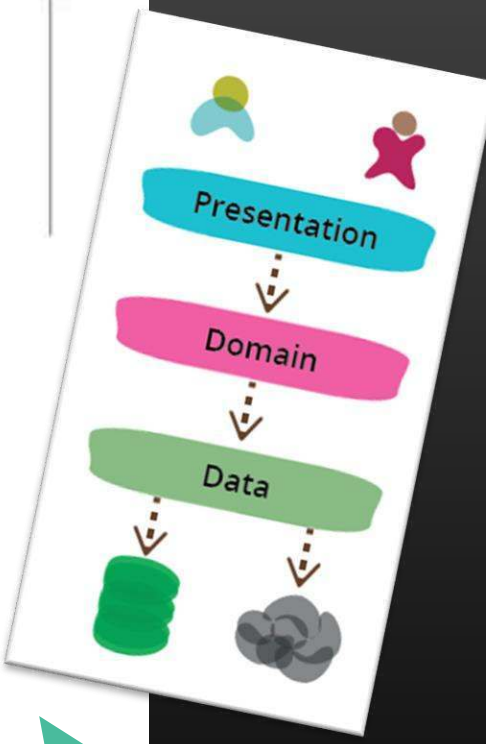
low-level technical services, utilities, and frameworks
data structures, threads, math, file, DB, and network I/O

Foundation
(AKA Core Services, Base Services, Low-level Technical Services/Infrastructure)

width implies range of applicability

more app specific

dependency



Algumas arquiteturas baseiam-se no modelo três camadas que ampliam para mostrar uma maior especialização de responsabilidades

[Larman'04]

Voltando ao início: 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.

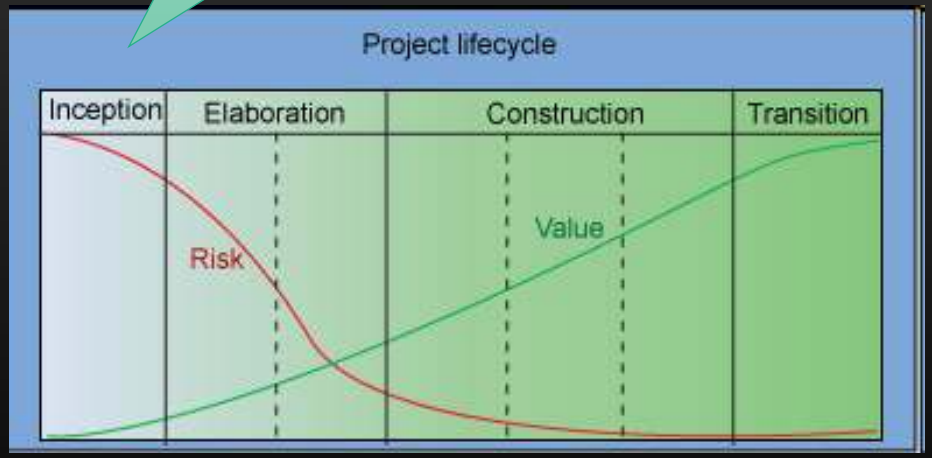
Identificar dependências de sistemas e componentes externos e especificar como serão integrados.

~10% do código será implementado

Basear a arquitetura nos casos de utilização nucleares

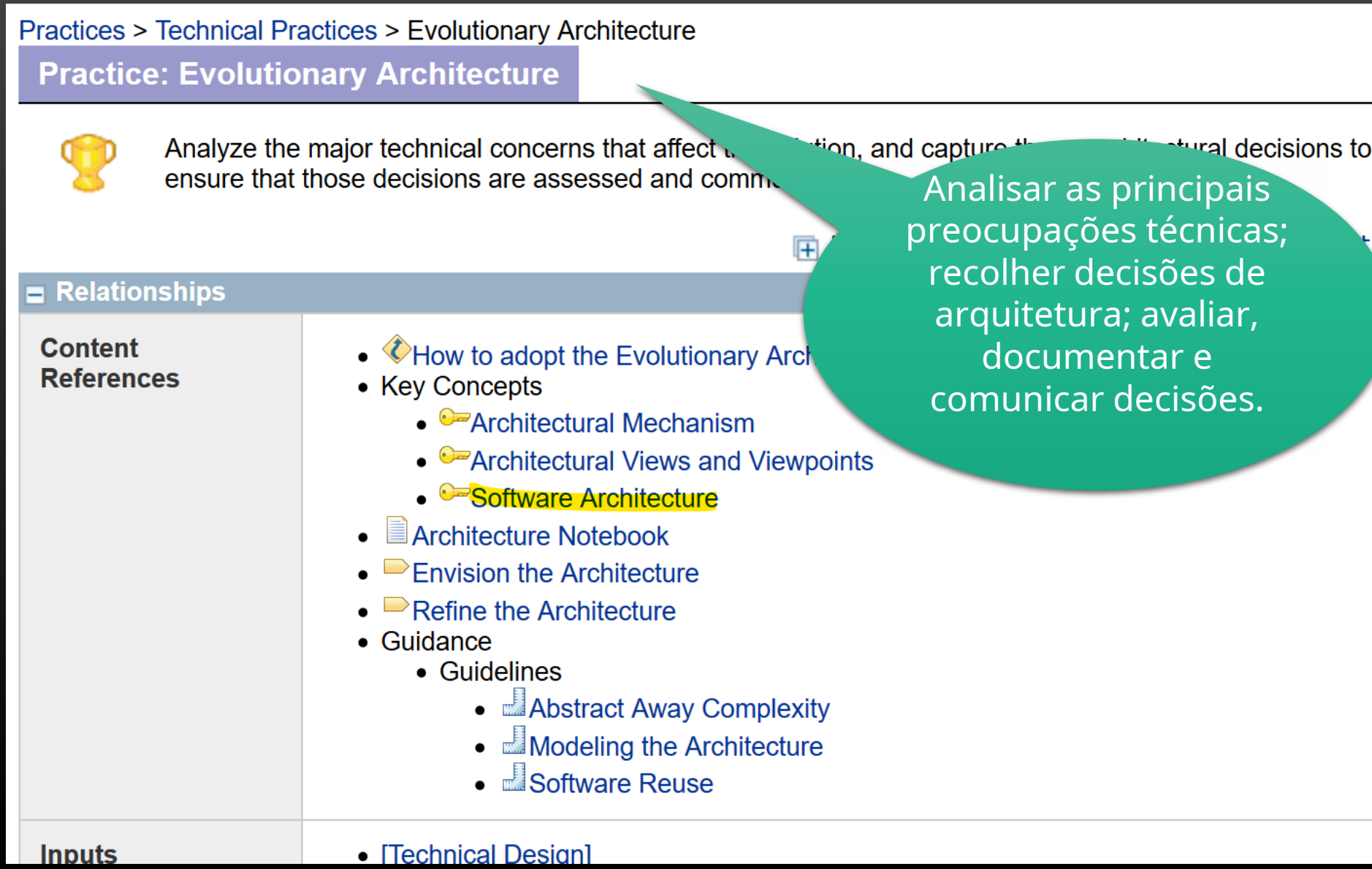
20% dos casos de uso determinam 80% da arquitetura

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



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

Porque é que é “evolutiva”?

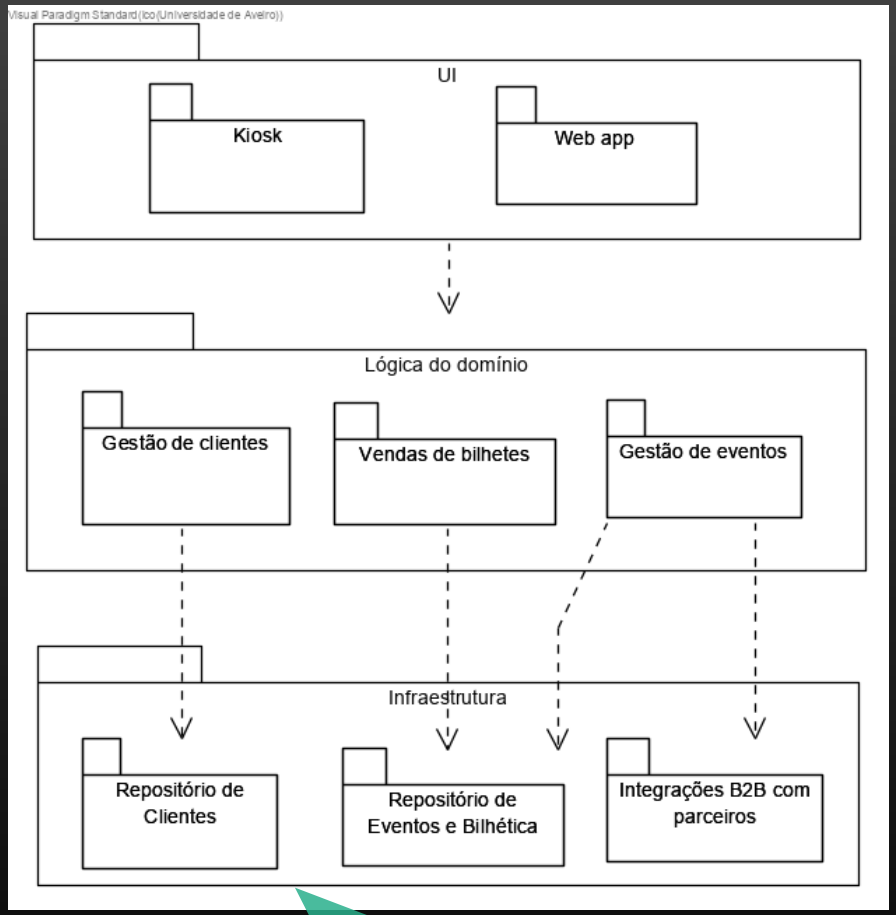
No OpenUP:

- 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.

Considerar o uso de arquiteturas lógicas nos projetos de grupo.



Opção I: vista puramente lógica, sem elementos de implementação. Destaca a modularização esperada do sistema.

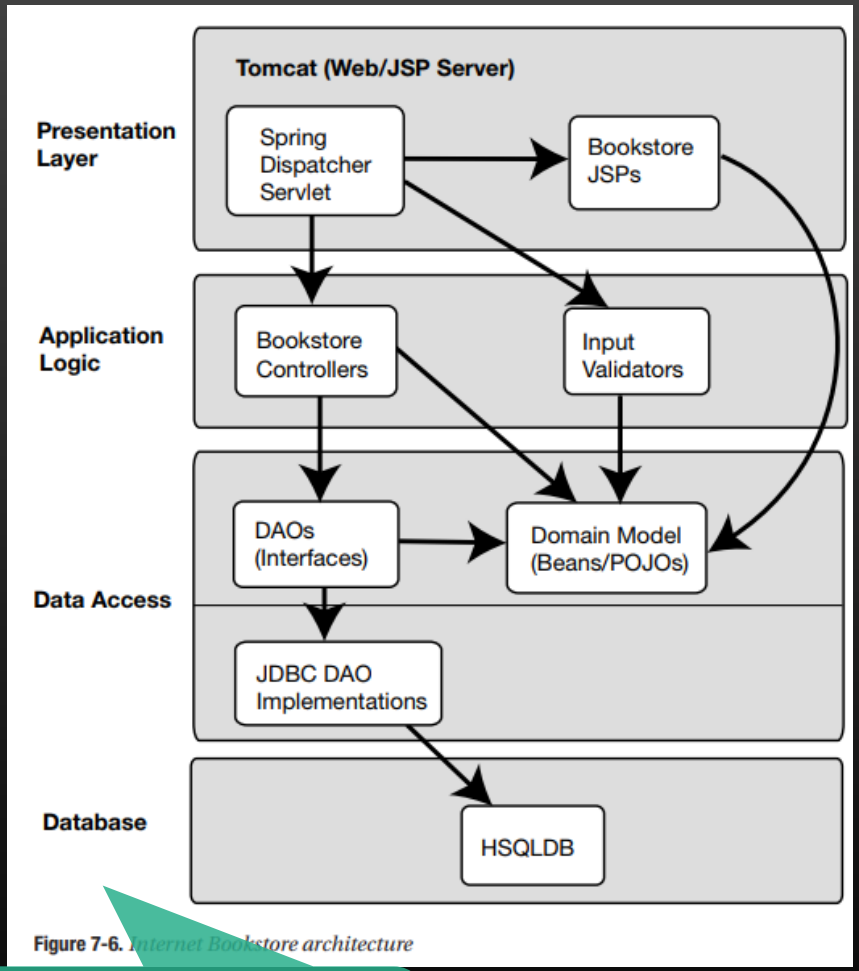
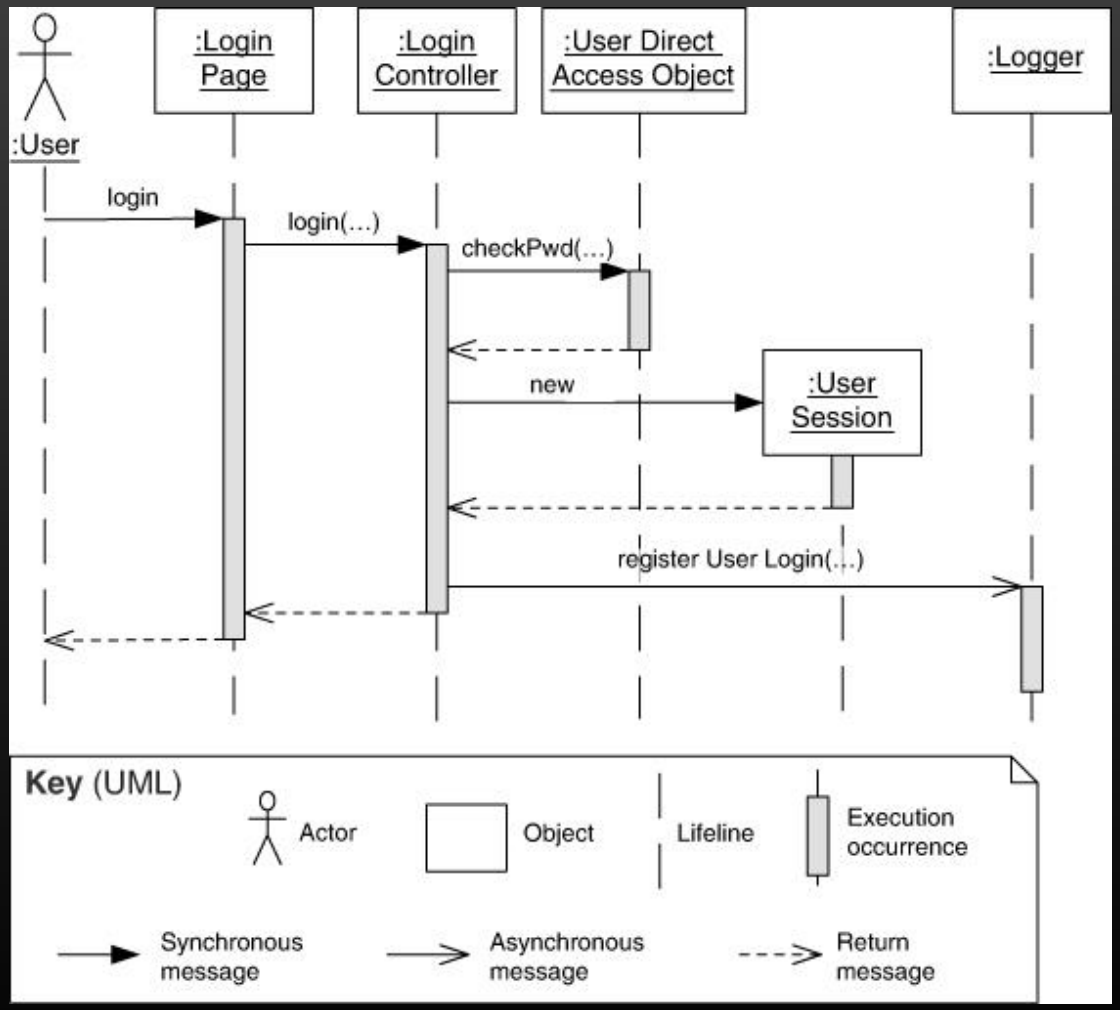


Figure 7-6. *Internet Bookstore architecture*

Opção II: *stack* de módulos, mostra os componentes da aplicação tendo em conta os elementos da implementação. Explica a forma como as tecnologias/ambientes/bibliotecas serão usadas na construção da "full stack".

Vista comportamental (d. sequência)



Referências

Core readings	Suggested readings
• [Larman04] – Chap. 13	• Bass, Clements, Kazman, "Software Architecture in Practice", 4th ed. • Fowler, "Software Architecture Guide"