

41951- ANÁLISE DE SISTEMAS

UML Use case diagrams

Ilídio Oliveira

v2022-03-08

Learning objectives for this lecture

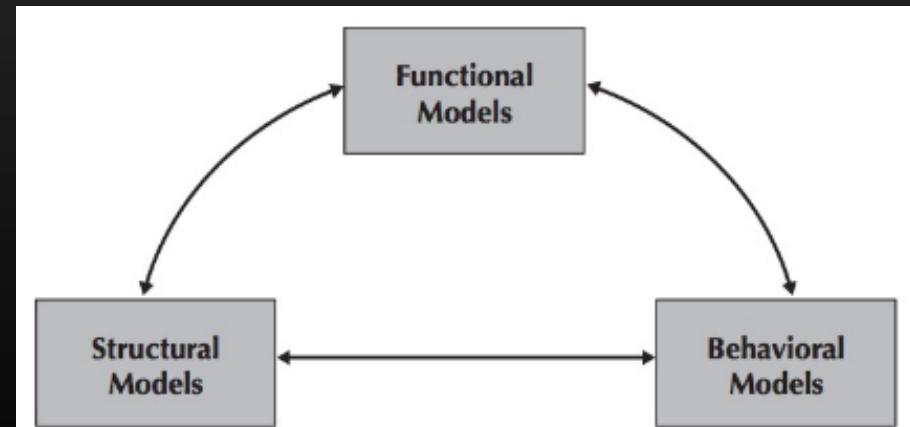
- Explain the process used to discover use cases
- Read and create Use Case diagrams
- Explain the components of a use case description and the associated templates
- Create functional models of business processes using use-case diagrams and use case descriptions.

3 complementary views on a system

Functional. What the system does? The external behavior of the system (black-box).

Structural. What are the parts ("things") the system is made off?

Behavior. How does the system do the operations? View of the system interactions along time.



Functional models

Develop use-cases from the requirements

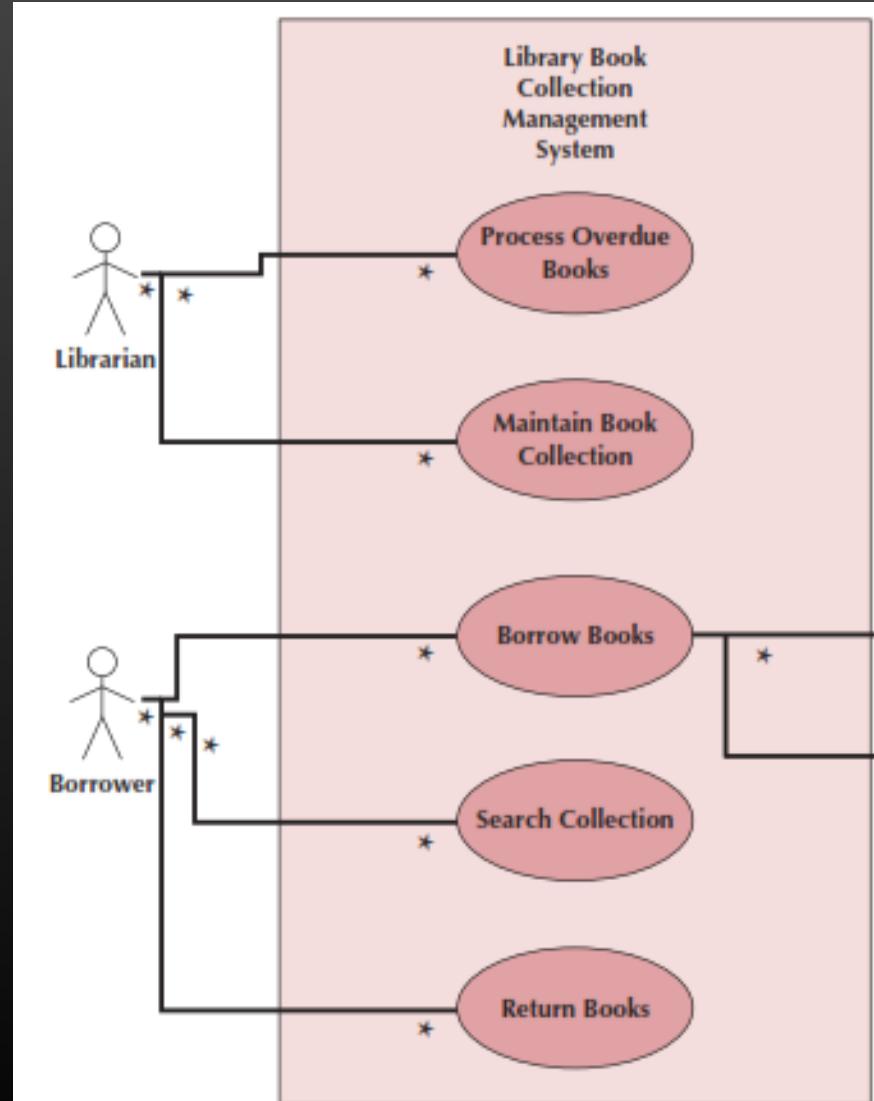
Use-case: how a business system interacts with its environment

Includes a diagram and a description to depict the discrete activities that the users perform

Develop activity diagrams from the use-cases

These model the business processes or how a business operates

Used to illustrate the movement of objects (data) between activities

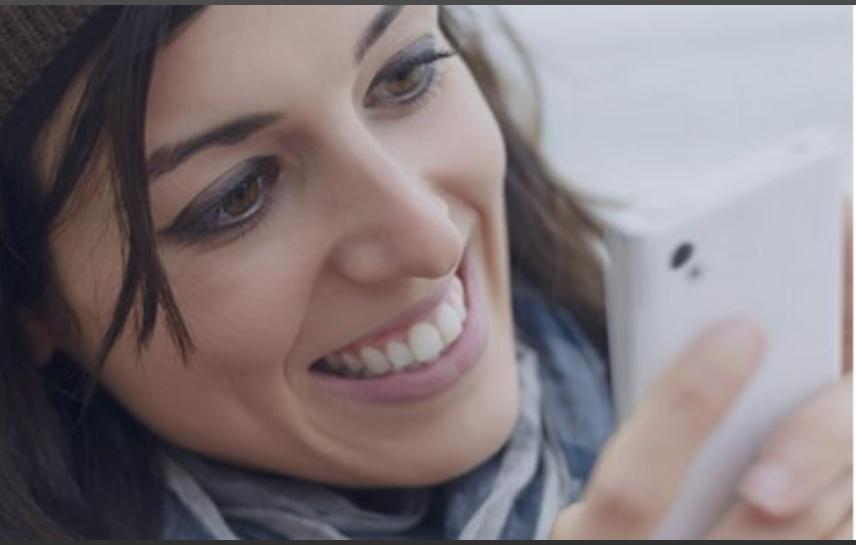


Which main requirements elicitation approaches exist?

What is the goal the user wants to achieve? vs.

What capability should the system process?





- Consulta de Saldos e Movimentos de Contas e Cartões de Crédito;
- Consulta de Posição Integrada;
- Transferências para beneficiários, contas BPI ou contas de outros Bancos (zona SEPA);
- Pagamentos de Serviços, Estado e Telemóveis;
- Criação e gestão de beneficiários de transferências e de pagamentos predefinidos;
- Constituição, reforço e mobilização de contas poupança objetivo;
- Cartões: pedido de alteração de Limites de Crédito, alteração de opção de pagamento e pagamento de Saldo ou Reforço;
- Consulta de catálogo e aquisição de Produtos Prestígio;
- Acesso a contactos, localização e serviços de Balcões, Centros de Investimento e Centros de Empresas;
- Login com código de 4 dígitos ou com impressão digital.

 Contas Cartões Crédito Poupança e Investimento Imóveis Seguros À sua Medida

Eu quero...



Ser cliente
da Caixa



Comprar
uma casa



Comprar
um Carro



Viajar



Preparar o Futuro
dos meus Filhos



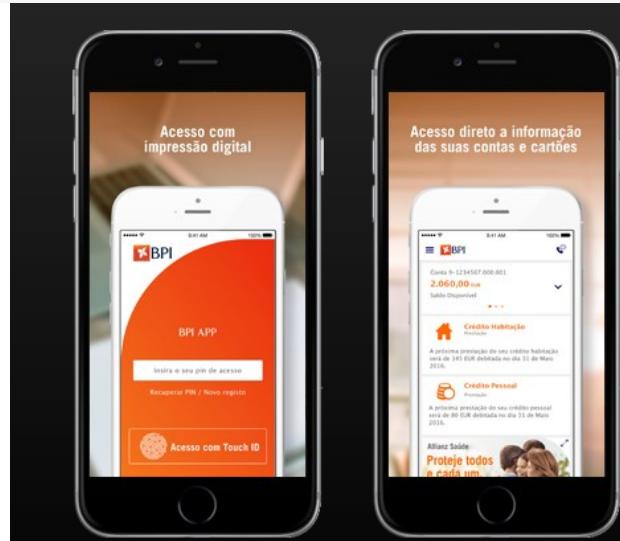
Poupar para
o Futuro



Preparar a
minha Reforma



Proteger a minha
Família

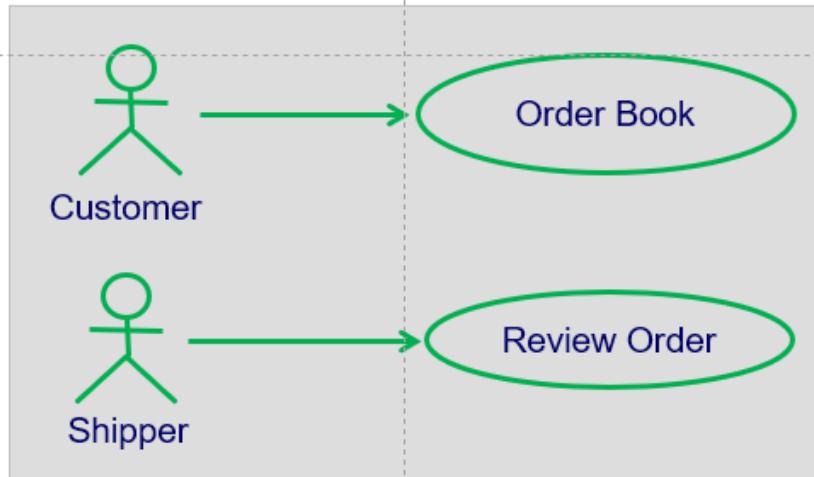


OpenUP recommended practices

Use Case Driven Development

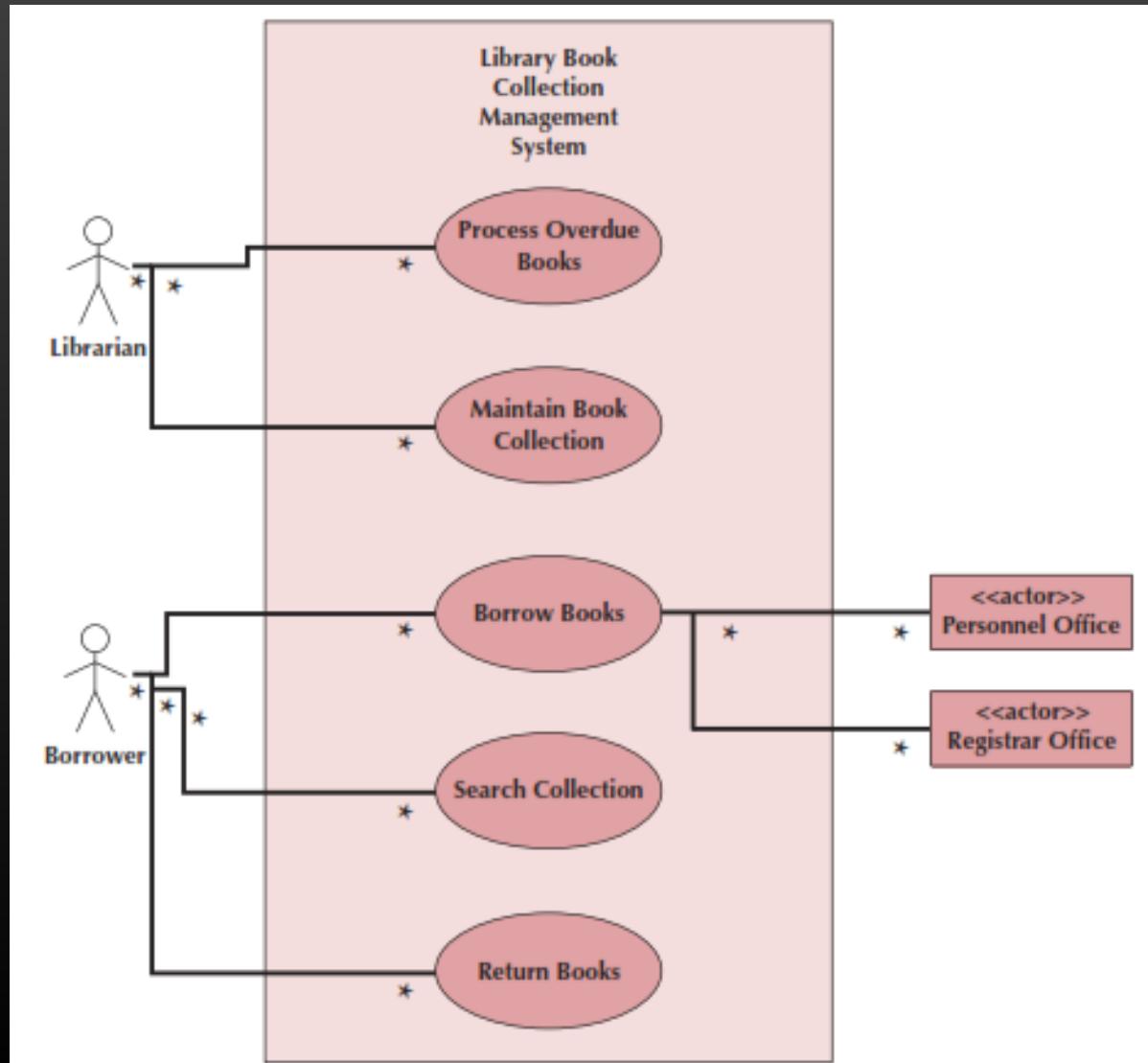


- This practice describes how to capture requirements with a combination of use cases and system-wide requirements, and then drive development and testing from those use cases.



Example Use-Case

Library Book Collection Management - System Use Case Diagram



Credit: Dennis et al, "Systems Analysis and Design: An Object Oriented Approach with UML", 5th ed Oliveira

Who (the actor) does what (interaction) on the system, with a goal in mind (motivation)

UC is a flow of actions that produces a result with values to a particular actor (including the variations in the flow, related to the same goal)

The required yet sufficient amount of activity (interaction) that produces a result of interest for an actor

UC provides a context to a related set of requirements.
(Favors a more coherent division of the system)

Use-cases discovery (in-class...)

→ ...



<https://www.menti.com/9ysw7pk2ge>



The use case model includes an overview (diagrams) + description of scenarios

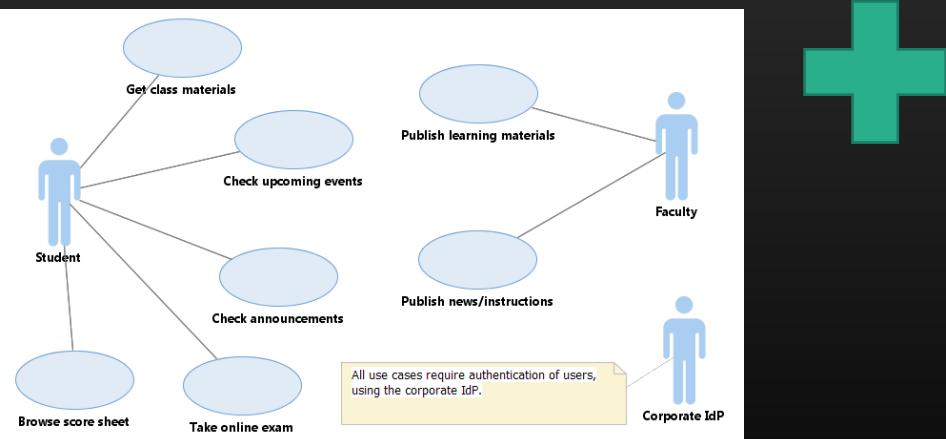
Use cases documentation (use cases modeling)

Visual overview

Use-cases Diagram (UML)

Description of scenarios

Structured narratives (text)
Optional: activity models.



ID and Name:	UC-4 Request a Chemical
Created By:	Lori
Date Created:	8/22/13
Primary Actor:	Requester
Secondary Actors:	Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.
Trigger:	Requester indicates that he wants to request a chemical.
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.
Normal Flow:	4.0 Request a Chemical from the Chemical Stockroom 1. Requester specifies the desired chemical. 2. System lists containers of the desired chemical that are in the chemical stockroom, if any. 3. System gives Requester the option to View Container History for any container. 4. Requester selects a specific container or asks to place a vendor order (see 4.1). 5. Requester enters other information to complete the request. 6. System stores the request and notifies the Chemical Stockroom.
Alternative Flows:	4.1 Request a Chemical from a Vendor 1. Requester searches vendor catalogs for the chemical (see 4.1.E1). 2. System displays a list of vendors for the chemical with available container sizes, grades, and prices. 3. Requester selects a vendor, container size, grade, and number of containers. 4. Requester enters other information to complete the request. 5. System stores the request and notifies the Buyer.
Exceptions:	4.1.E1 Chemical Is Not Commercially Available 1. System displays message: No vendors for that chemical. 2. System asks Requester if he wants to request another chemical (3a) or to exit (4a). 3a. Requester asks to request another chemical. 3b. System starts normal flow over. 4a. Requester asks to exit. 4b. System terminates use case.
Priority:	High
Frequency of Use:	Approximately 5 times per week by each chemist, 200 times per week by chemical

Types of Use Cases

Purpose	Amount of information	
	Overview/High-level	Detail
Essential	High-level overview of issues essential to understanding required functionality	Detailed description of issues essential to understanding required functionality
Real	High-level overview of a specific set of steps performed on the real system once implemented	Detailed description of a specific set of steps performed on the real system once implemented

Use case model

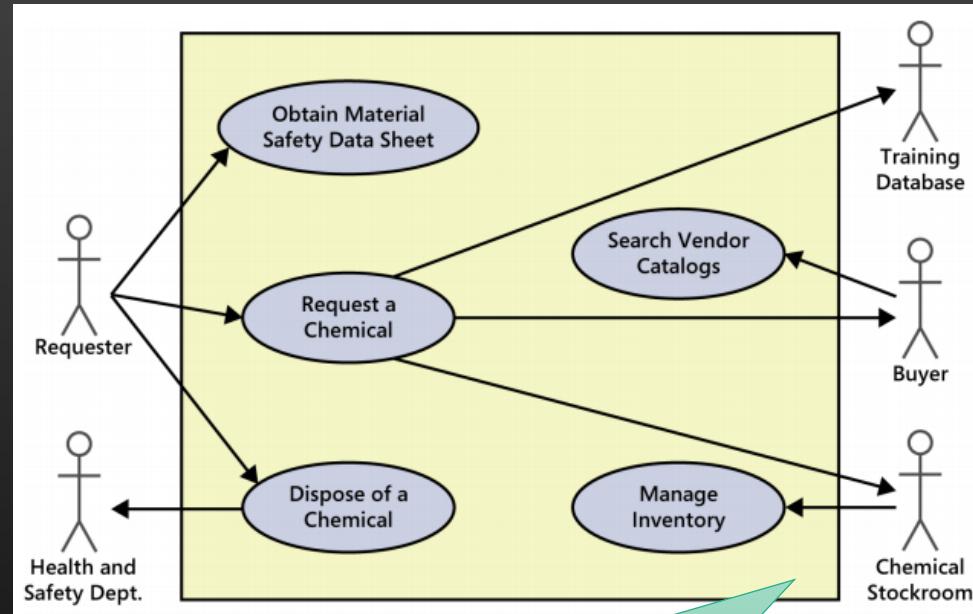
The **users (or systems)** who interact with the system with some objective in mind are modeled as **actors**.

The ways/episodes in which the system will be used to achieve these goals are modeled as **use cases**.

A **use case diagram** is a model of the useful ways to use a system. This allows you to quickly grasp the scope of the system – what is included and what is not – and give the team a global view of what the system will do.

The **intentional high-level perspective** offers a birds-eye view, without losing ourselves in the details of the internal parts of the system.

→ A context/instrument to **discuss and discover** the system requirements!



Note: we will not use “arrows” in use-case/actor associations

Elements of the use case model

Actor

Any entity (the role of someone, another system,...) external to the system under specification, which interacts with it

Scenario

A particular situation/history of use of the system, i.e., a possible path in the execution of a case of use

E.g.: payment of the purchase with cash or card; failure in payment due to lack of card authorization;...

Use case

Set of scenarios related to the same goal
An episode of using + variants.

Associations

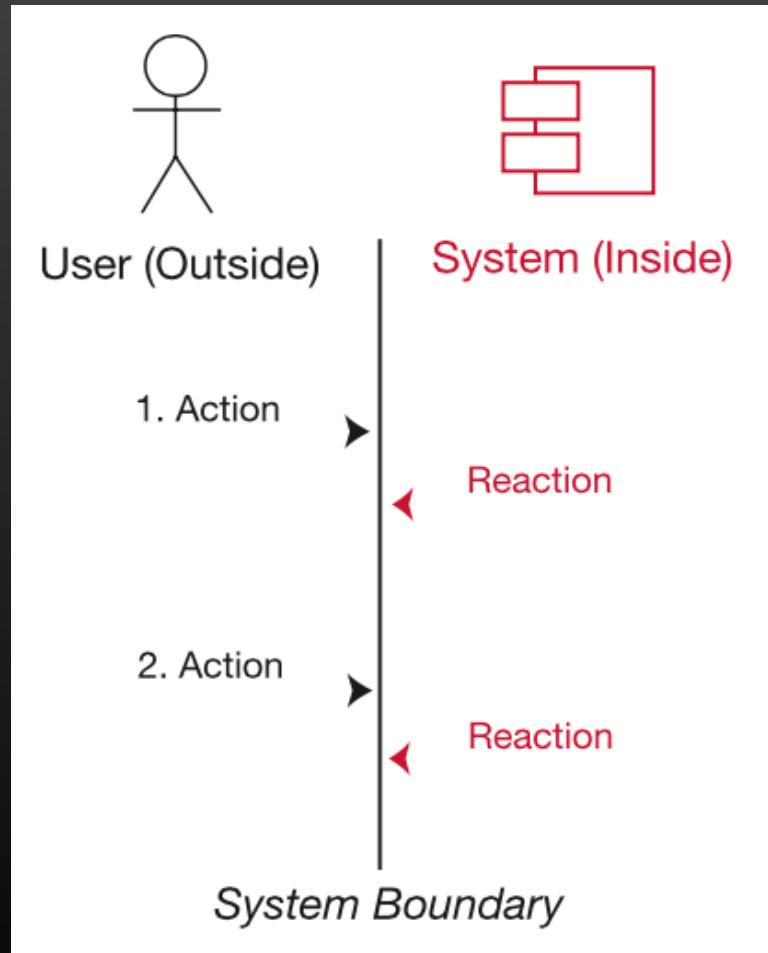
Relationships of interest between actors/CaU, CaU/CaU, actors/actors
Most relevant:
Actor *participates in* a use case.

**Use cases hold/encapsulate
interaction scenarios**

The use case captures a dialog between the actor(s) and the system

CaU: Pay at checkout

1. Customer arrives at POS checkout with goods to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price is calculated from a set of price rules.
Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any)



The Use Case has several flows

The typical flow

The “normal script” for the actor/system collaboration

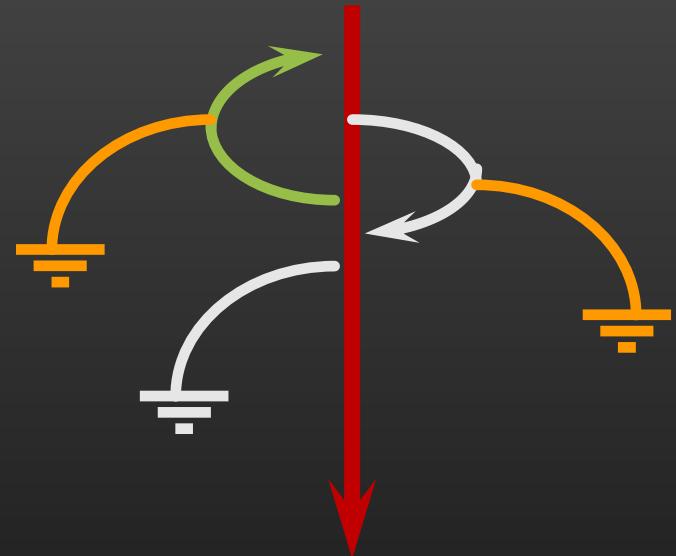
Several alternative flows

Variations due to options from the users

Uncommon/special cases

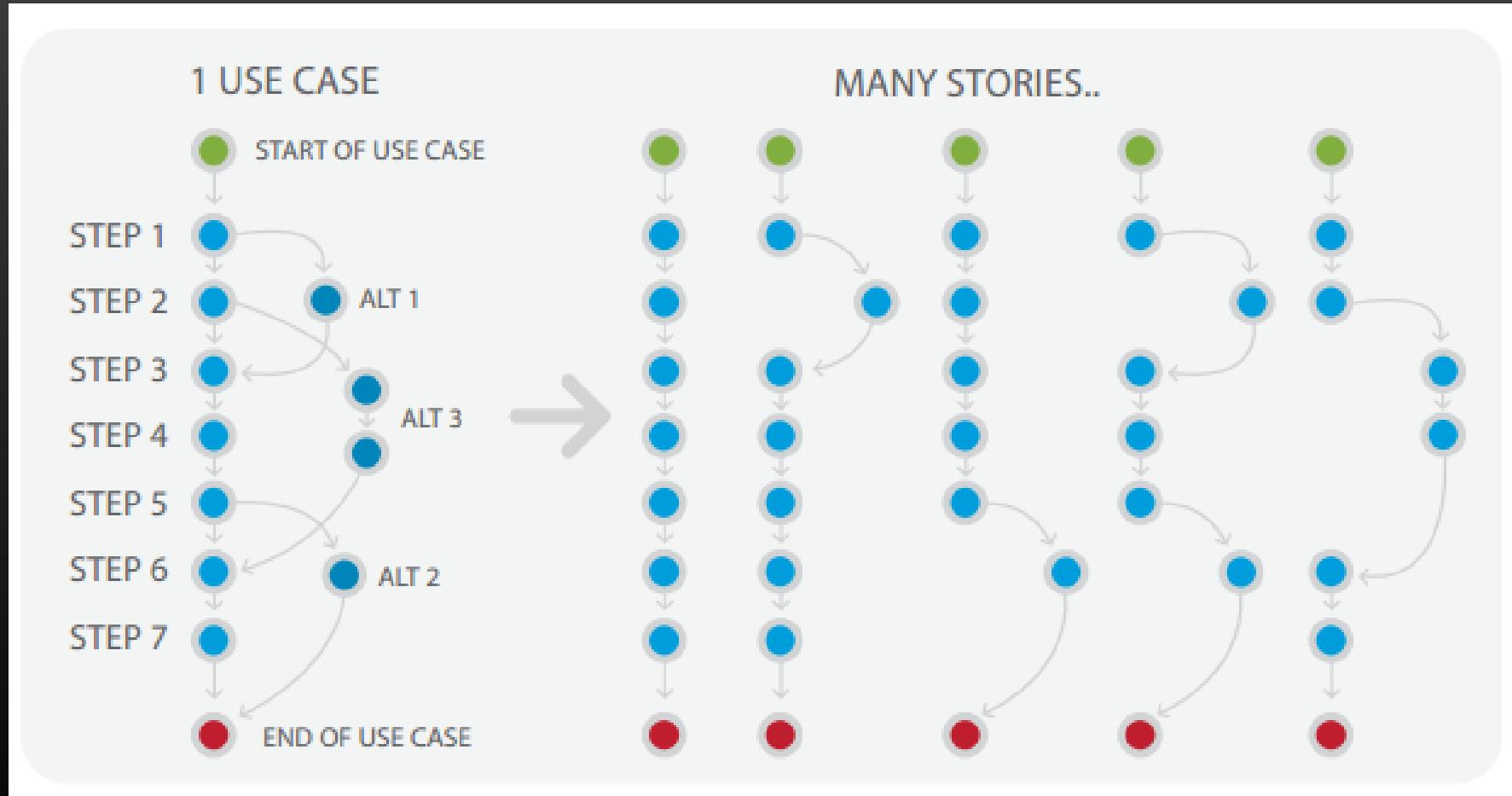
Exception conditions and errors (and what should be done)

e.g.: how to enter the product description (in the sale scenario) when the bar code can not be read?



One use case has different flows

Still, all flows are related to the same goal in-mind.

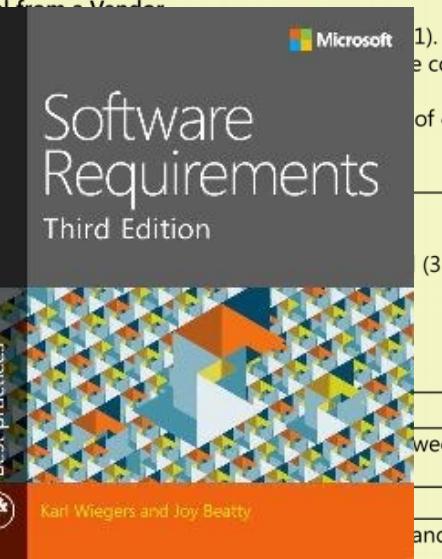


Essential elements of a use case specification

- ✓ A unique identifier and a succinct name that states the user goal
- ✓ A brief textual description that describes the purpose of the use case
- ✓ A trigger condition that initiates execution of the use case
- ✓ Zero or more preconditions that must be satisfied before the use case can begin
- ✓ One or more postconditions that describe the state of the system after the use case is successfully completed
- ✓ A numbered list of steps that shows the sequence of interactions between the actor and the system – a dialog – that leads from the preconditions to the postconditions

→ Credit: Wiegers 2013

ID and Name:	UC-4 Request a Chemical
Created By:	Lori
Date Created:	8/22/13
Primary Actor:	Requester
Secondary Actors:	Buyer, Chemical Stockroom, T
Description:	The Requester specifies the desired chemical to request by entering number or by importing its structure from a chemical drawing tool. the Requester a container of the chemical from the chemical stock order one from a vendor.
Trigger:	Requester indicates that he wants to request a chemical.
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.
Normal Flow:	4.0 Request a Chemical from the Chemical Stockroom 1. Requester specifies the desired chemical. 2. System lists containers of the desired chemical that are in the che 3. System gives Requester the option to View Container History for 4. Requester selects a specific container or asks to place a vendor or 5. Requester enters other information to complete the request. 6. System stores the request and notifies the Chemical Stockroom.
Alternative Flows:	4.1 Request a Chemical from a Vendor 1. Requester search 2. System displays a and prices. 3. Requester selects 4. Requester enters 5. System stores the
Exceptions:	4.1.E1 Chemical Is I 1. System displays n 2. System asks Requ 3a. Requester asks t 3b. System starts no 4a. Requester asks t 4b. System terminat
Priority:	High
Frequency of Use:	Approximately 5 tim stockroom staff
Business Rules:	BR-28, BR-31
Other Information:	The system must be any of the supports



"Request a Chemical" use case specification

ID and Name:	UC-4 Request a Chemical	
Created By:	Lori	Date Created: 8/22/13
Primary Actor:	Requester	Secondary Actors: Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.	
Trigger:	Requester indicates that he wants to request a chemical.	
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.	
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.	
Normal Flow:	4.0 Request a Chemical from the Chemical Stockroom <ol style="list-style-type: none">1. Requester specifies the desired chemical.2. System lists containers of the desired chemical that are in the chemical stockroom, if any.3. System gives Requester the option to View Container History for any container.4. Requester selects a specific container or asks to place a vendor order (see 4.1).5. Requester enters other information to complete the request.6. System stores the request and notifies the Chemical Stockroom.	
Alternative Flows:	4.1 Request a Chemical from a Vendor <ol style="list-style-type: none">1. Requester searches vendor catalogs for the chemical (see 4.1.E1).2. System displays a list of vendors for the chemical with available container sizes, grades, and prices.3. Requester selects a vendor, container size, grade, and number of containers.4. Requester enters other information to complete the request.5. System stores the request and notifies the Buyer.	
Exceptions:	4.1.E1 Chemical Is Not Commercially Available <ol style="list-style-type: none">1. System displays message: No vendors for that chemical.2. System asks Requester if he wants to request another chemical (3a) or to exit (4a).	

The use case details describe an interaction

HOW TO WRITE A USE CASE: THE THREE MAGIC QUESTIONS

Well, OK, this whole chapter describes how to write a use case. But when writing use cases, you need to keep asking the following three fundamental questions:¹

1. What happens?

(This gets your “sunny-day scenario” started.)

2. And then what happens?

(Keep asking this question until your “sunny-day scenario” is complete.)

3. What else might happen?

(Keep asking this one until you’ve identified all the “rainy-day scenarios” you can think of, and described the related behavior.)

BASIC COURSE:

The Customer clicks the Write Review button for the book currently being viewed, and the system shows the Write Review screen. The Customer types in a Book Review, gives it a Book Rating out of five stars, and clicks the Send button. The system ensures that the Book Review isn't too long or short, and that the Book Rating is within one and five stars. The system then displays a confirmation screen, and the review is sent to a Moderator, ready to be added.

ALTERNATE COURSES:

User not logged in: The user is first taken to the Login screen and then to the Write Review screen once he is logged in.

The user enters a review that is too long (text > 1MB): The system rejects the review and responds with a message explaining why the review was rejected.

The review is too short (< 10 characters): The system rejects the review.

Use case:	Brief description:
Create new assignment	<p>The Teaching Staff creates a new Activity of type Assignment, directly inserting it in the page layout. The assignment must define a title and a time period for submissions and can be configured to work with individual or group submissions. The assignment is listed in the student view and on the specified date (or immediately, if none is given) accepts submissions from registered students.</p>
Use case: <u>Add new assignment</u> Brief description: <p>The Faculty creates assignments for students, directly inserting it in the course page. The assignment defines a time period for submissions and can be configured to work with individual or group submissions. The assignment is listed in the student view and on the specified date (or immediately, if none is given) accepts submissions from students.</p> Basic flow: <ol style="list-style-type: none"> 1. Log-in using corporate IdP. 2. Select desired course. 3. Turn editing mode on. 4. Add Assignment activity in the page layout. 5. Configure Assignment activity. 6. Commit changes. Alternative flows: <p>Step 1: IdP unavailable.</p> <p>Step 4/5: Instead of a new, empty assignment, the user may reuse an existing one.</p> Open issues: <p>Step 3/4. The course is closed. Are changes allowed to past courses?</p> <p>Step 5. The browser does not accept the rich text editor. Default to plain text?</p>	

Use Case Writing Guidelines

1. Write in the form of subject-verb-direct object
2. Make sure it is clear who the initiator of the step is
3. Write from independent observer's perspective
4. Write at about the same level of abstraction
5. Ensure the use case has a sensible set of steps
6. Apply the KISS principle liberally.
7. Write repeating instructions after the set of steps to be repeated

How to discover the Use Cases?

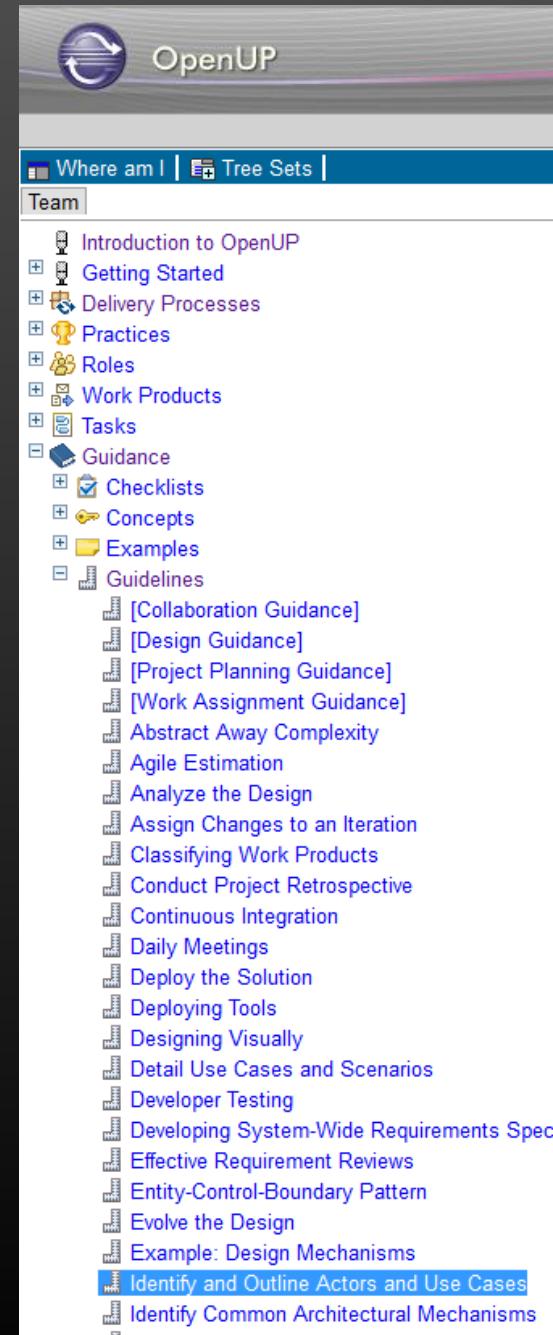
Identify the system boundary

Identify the actors who somehow interact with the system

For each actor, identify the objectives/motivations to use the system

Define CaU that satisfies the objectives of the actors

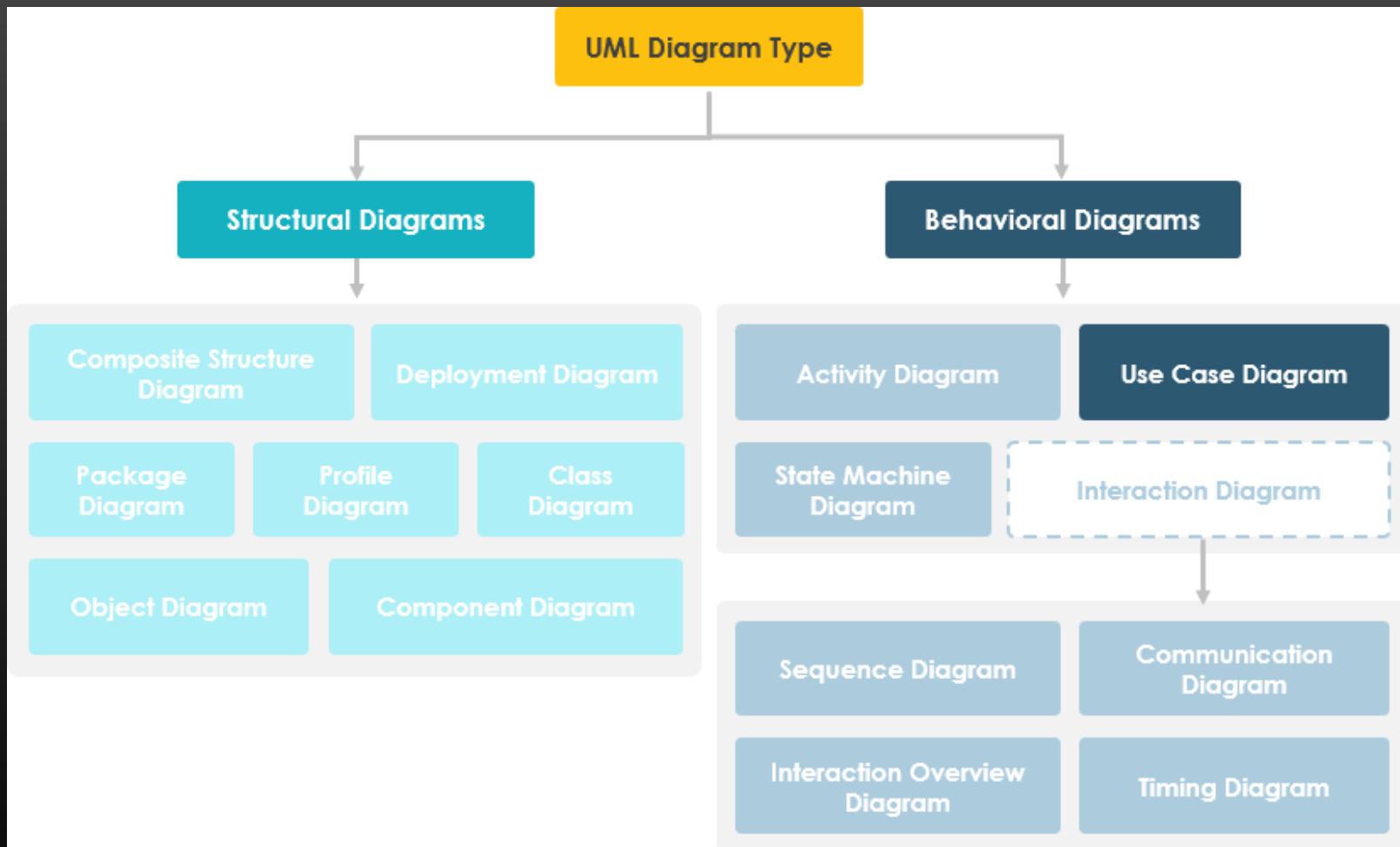
Give names that reflect the motivation of the actor



The screenshot shows the OpenUP software interface. At the top, there is a logo and the text "OpenUP". Below the logo is a blue header bar with the links "Where am I" and "Tree Sets". The main area has a title "Team" with a small icon. To the right of the title is a navigation tree. The tree starts with "Introduction to OpenUP" and branches into several categories: Getting Started, Delivery Processes, Practices, Roles, Work Products, Tasks, Guidance, Checklists, Concepts, Examples, and Guidelines. Under "Guidelines", there are many sub-items listed vertically, including: [Collaboration Guidance], [Design Guidance], [Project Planning Guidance], [Work Assignment Guidance], Abstract Away Complexity, Agile Estimation, Analyze the Design, Assign Changes to an Iteration, Classifying Work Products, Conduct Project Retrospective, Continuous Integration, Daily Meetings, Deploy the Solution, Deploying Tools, Designing Visually, Detail Use Cases and Scenarios, Developer Testing, Developing System-Wide Requirements Spec, Effective Requirement Reviews, Entity-Control-Boundary Pattern, Evolve the Design, Example: Design Mechanisms, Identify and Outline Actors and Use Cases (which is highlighted in blue), and Identify Common Architectural Mechanisms.

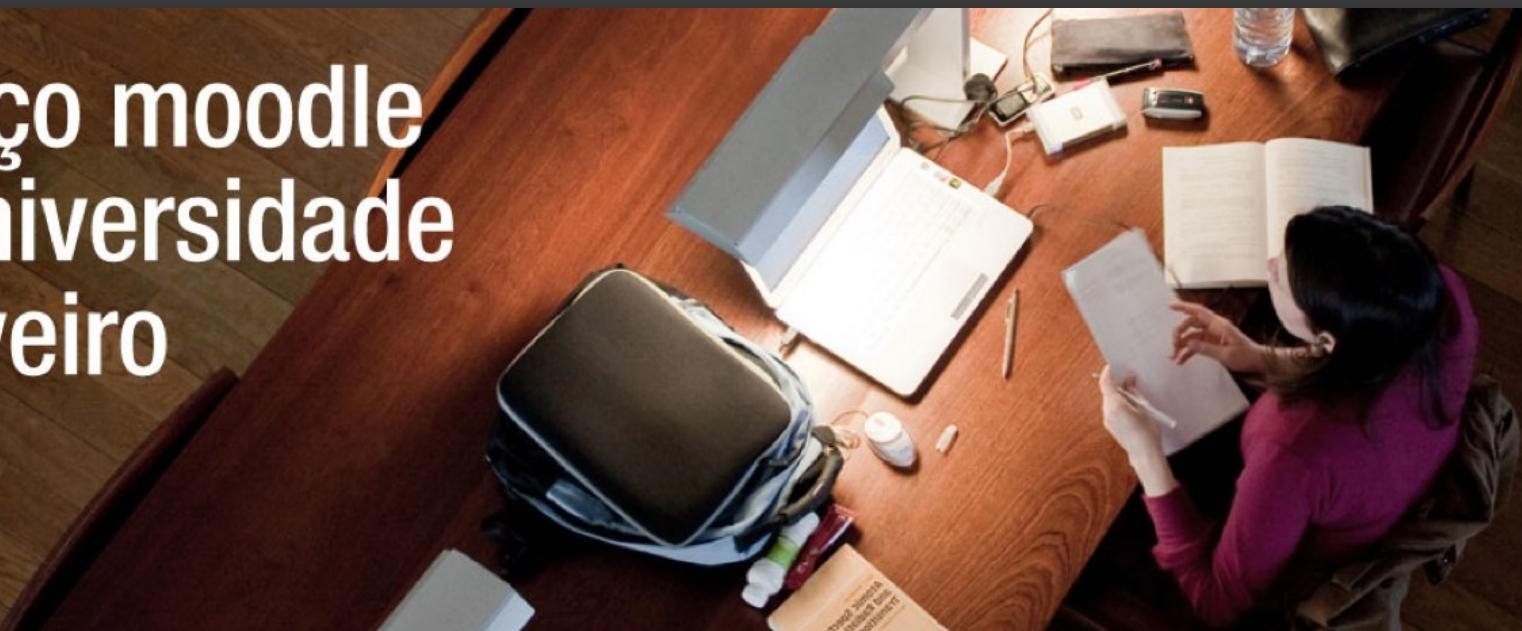
Guideline: Identify and Outline Actors and Use Cases

UML support



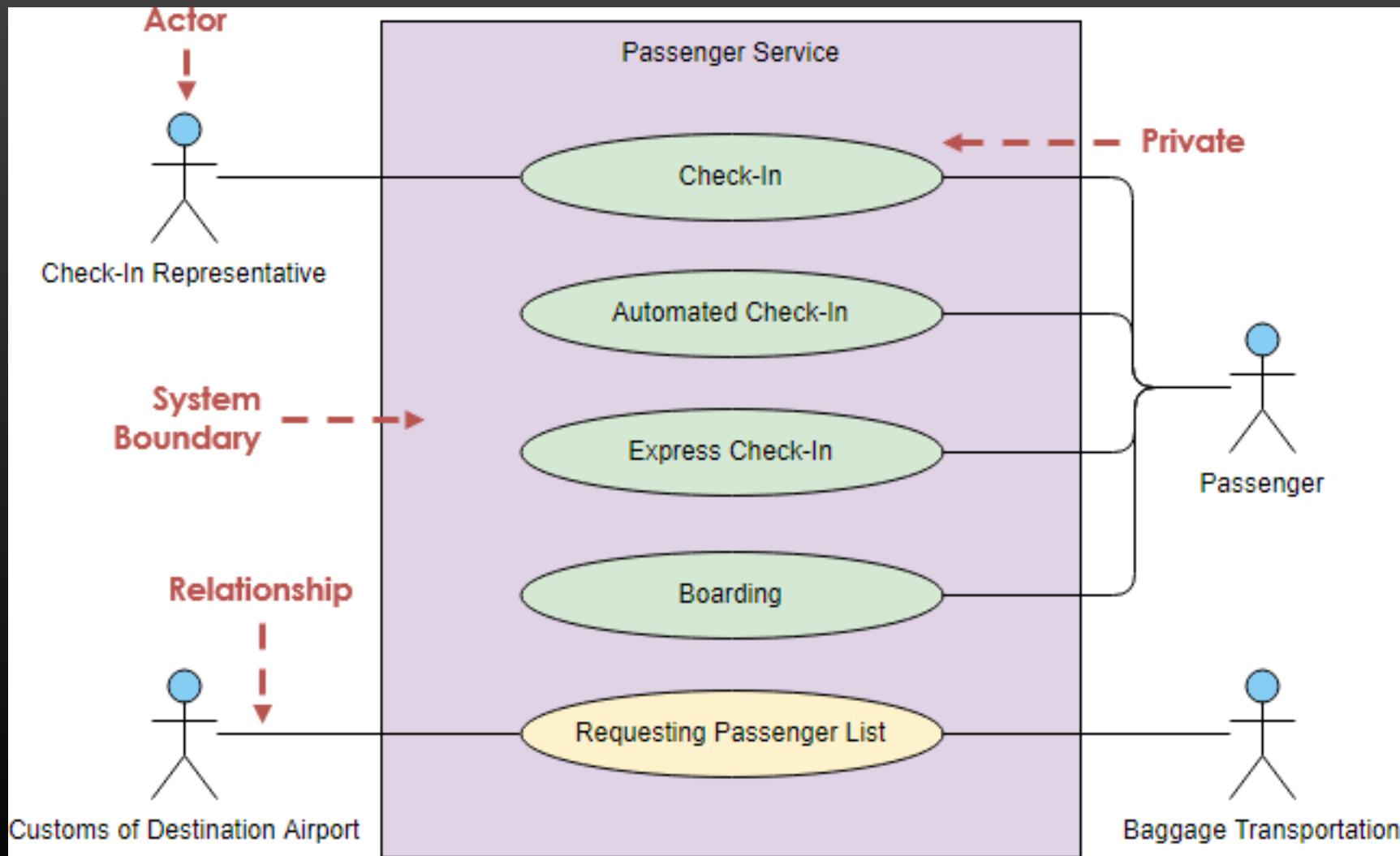
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

espaço moodle da universidade de aveiro



See also: [tutorial on use-case diagrams](#)

Elements of the UC diagram



Reusing behavior with **include**

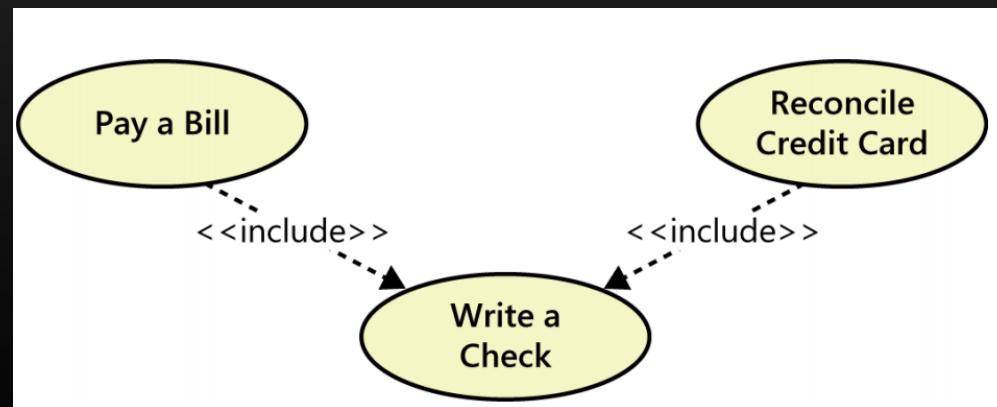
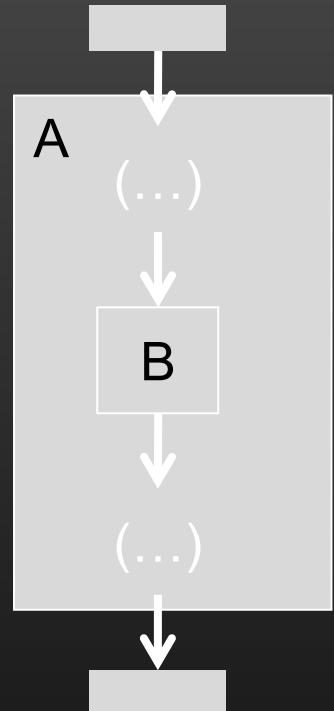
A-story includes the B-story

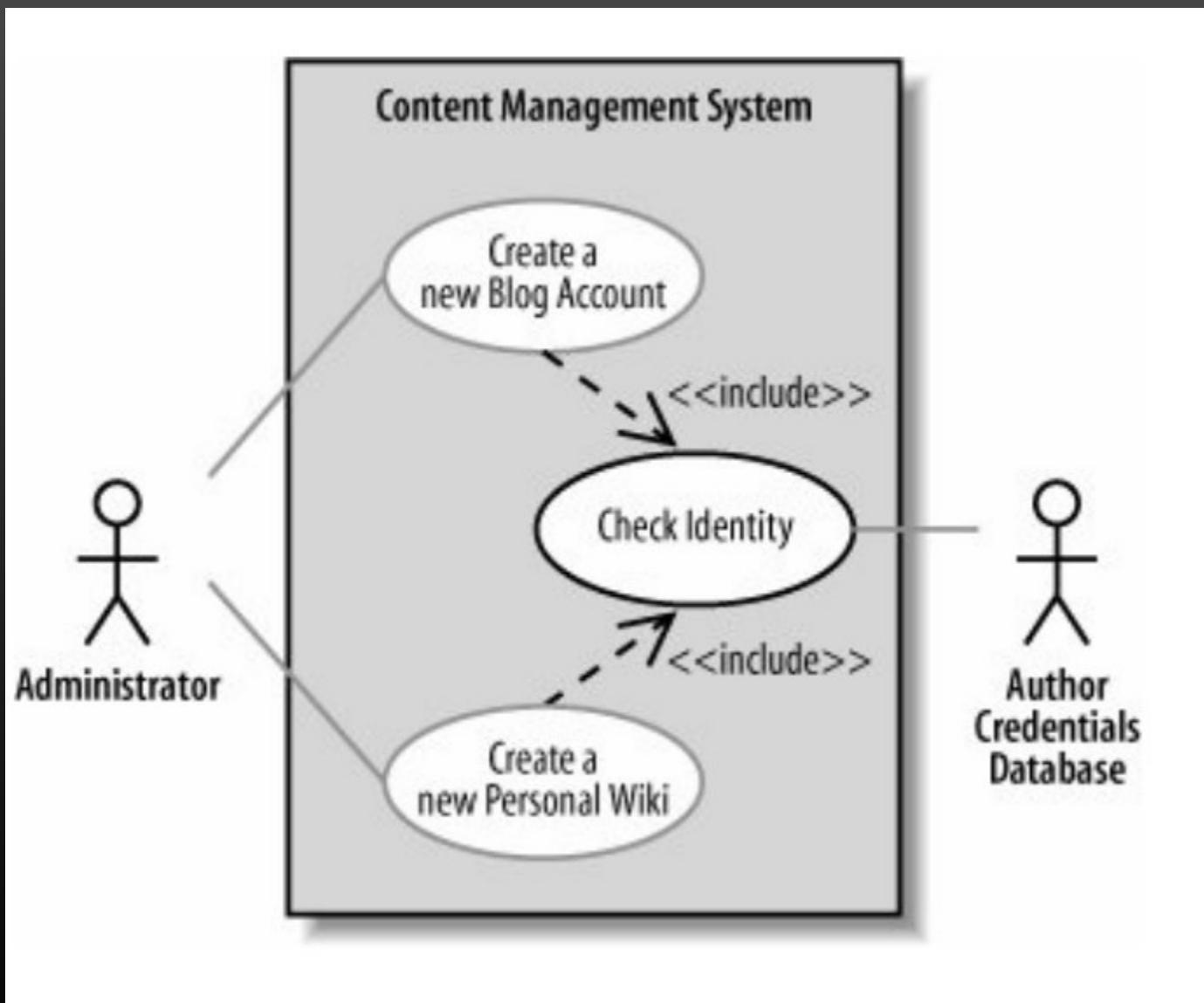
The behavior in A always includes the behavior modeled in B

Can help with “factoring out” common behavior

Include is the stereotype of the dependency relationship

- Not a verb here

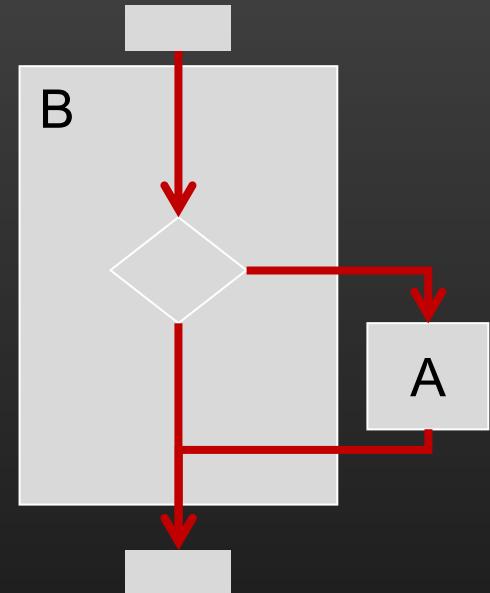




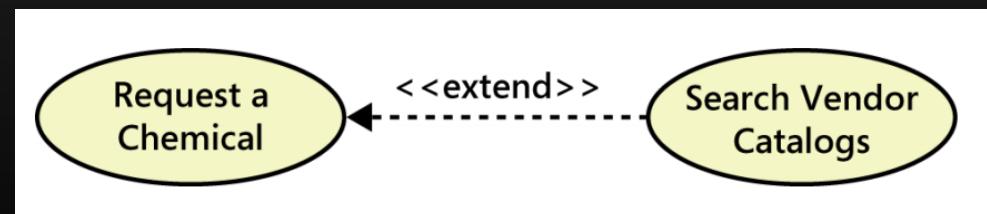
Optional behavior activation with **extend**

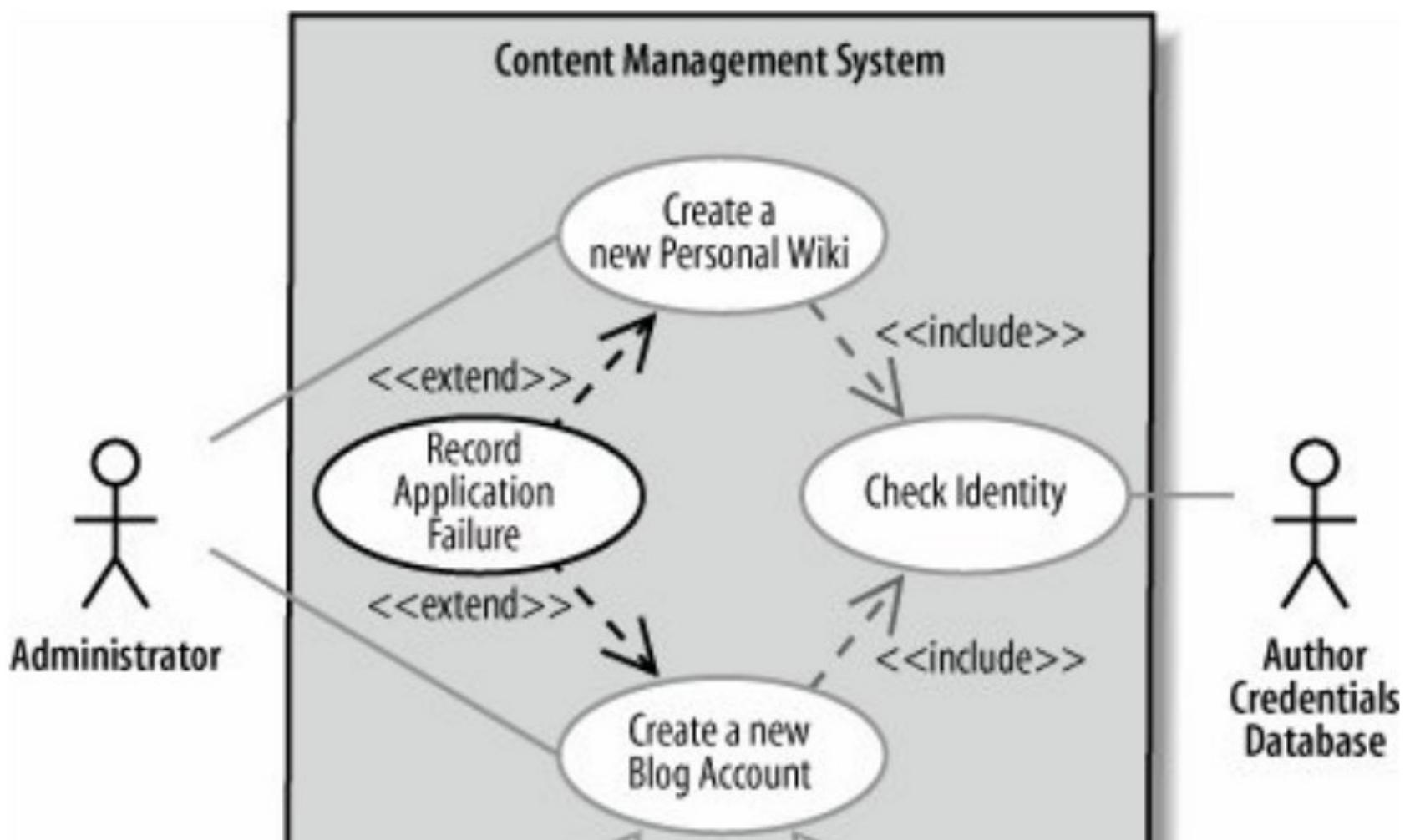
A-story may extend B-story

The behavior of B can incorporate the behavior of A, depending on the verification of an "extension condition" (extension point)

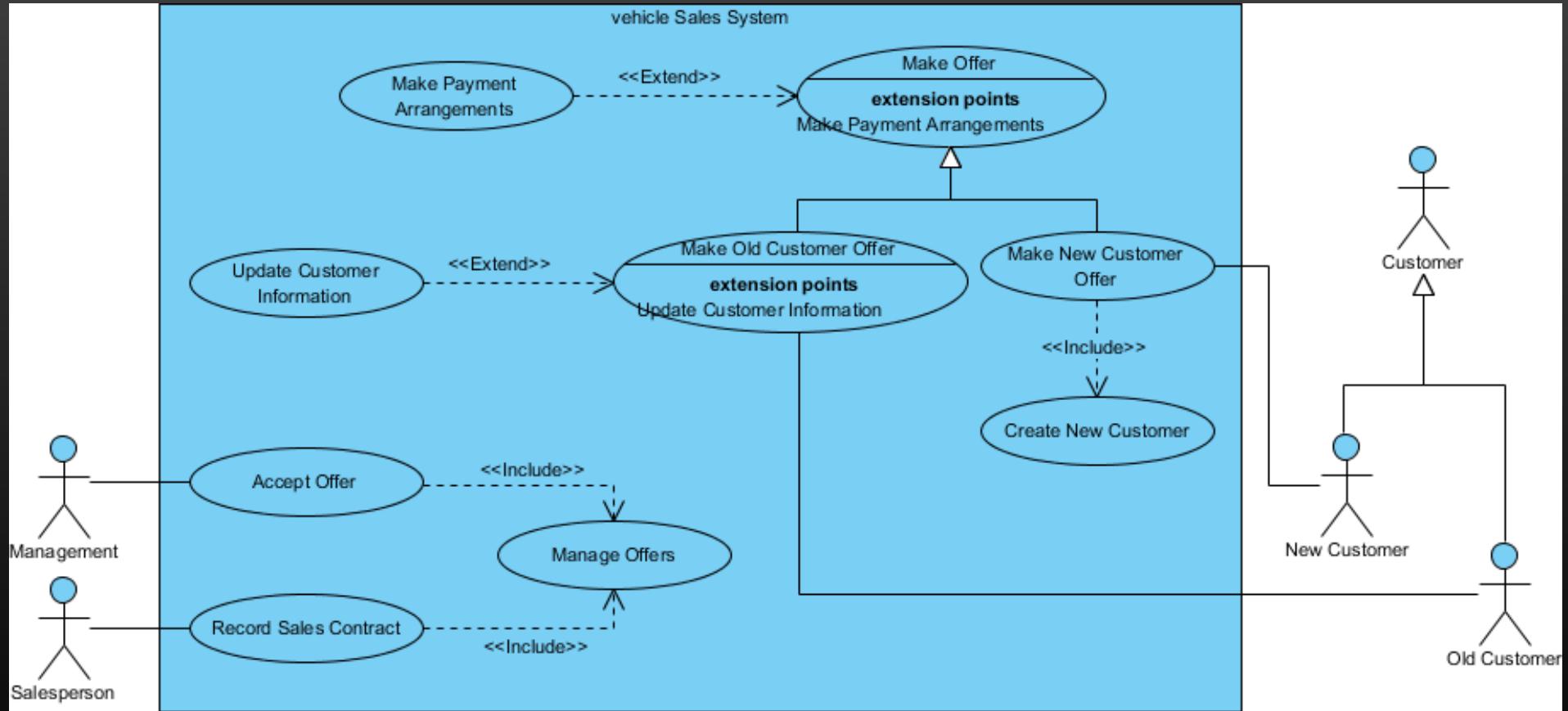


Unlike the include relationship,
extend models
optional/conditional behavior





Sample (full syntax)



The use cases may be organized in coherent functional groups → using packages

general

- + Add to Wish List
- + Cancel Order
- + Edit Shopping Cart
- + Login
- + Logout
- + Open an Account
- + Return a book
- + View Order History
- + Where's My Stuff?

admin

- + Customer Service
- + Seller
- + Shipping Clerk
- + Webmaster
- + Add Books to Catalog
- + Add Editorial Review
- + Add External Books to Catalog
- + Dispatch Order
- + Moderate Customer Reviews
- + Monitor Stock Levels
- + Order Books from Publisher
- + Process Refund
- + Remove Books from Catalog
- + Remove External Books from Catalog
- + Respond to Enquiry
- + Unlock Locked Account

shopping

- + Customer
- + Add Item to Shopping Cart
- + Checkout
- + Edit Shopping Cart
- + Enter Address
- + Pay by Card
- + Pay by Check
- + Pay by Purchase Order
- + Remove Item From Shopping Cart

searching

- + Advanced Search
- + Search by Author
- + Search by Category

Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 4• <u>How to write effective use cases</u>, VisualParadigm documentation• Jacobson, I., Spence, I., & Kerr, B. (2016). <u>Use-case 2.0</u>. Communications of the ACM, 59(5), 61–69.	

41951- ANÁLISE DE SISTEMAS

UML Activity diagram

Ilídio Oliveira | v2022/03/08

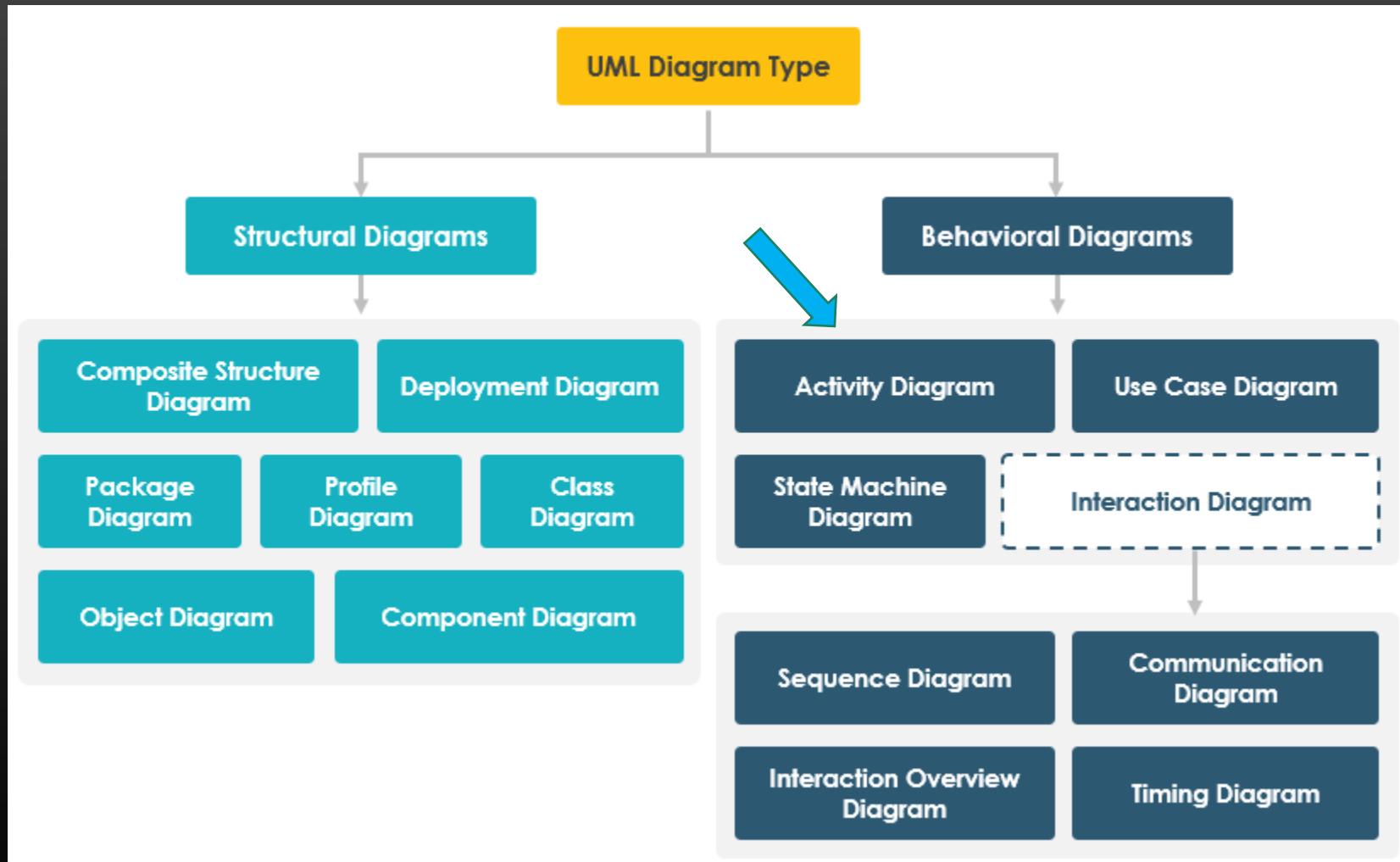
Learning objectives for this lecture

Read and create Activity Diagrams

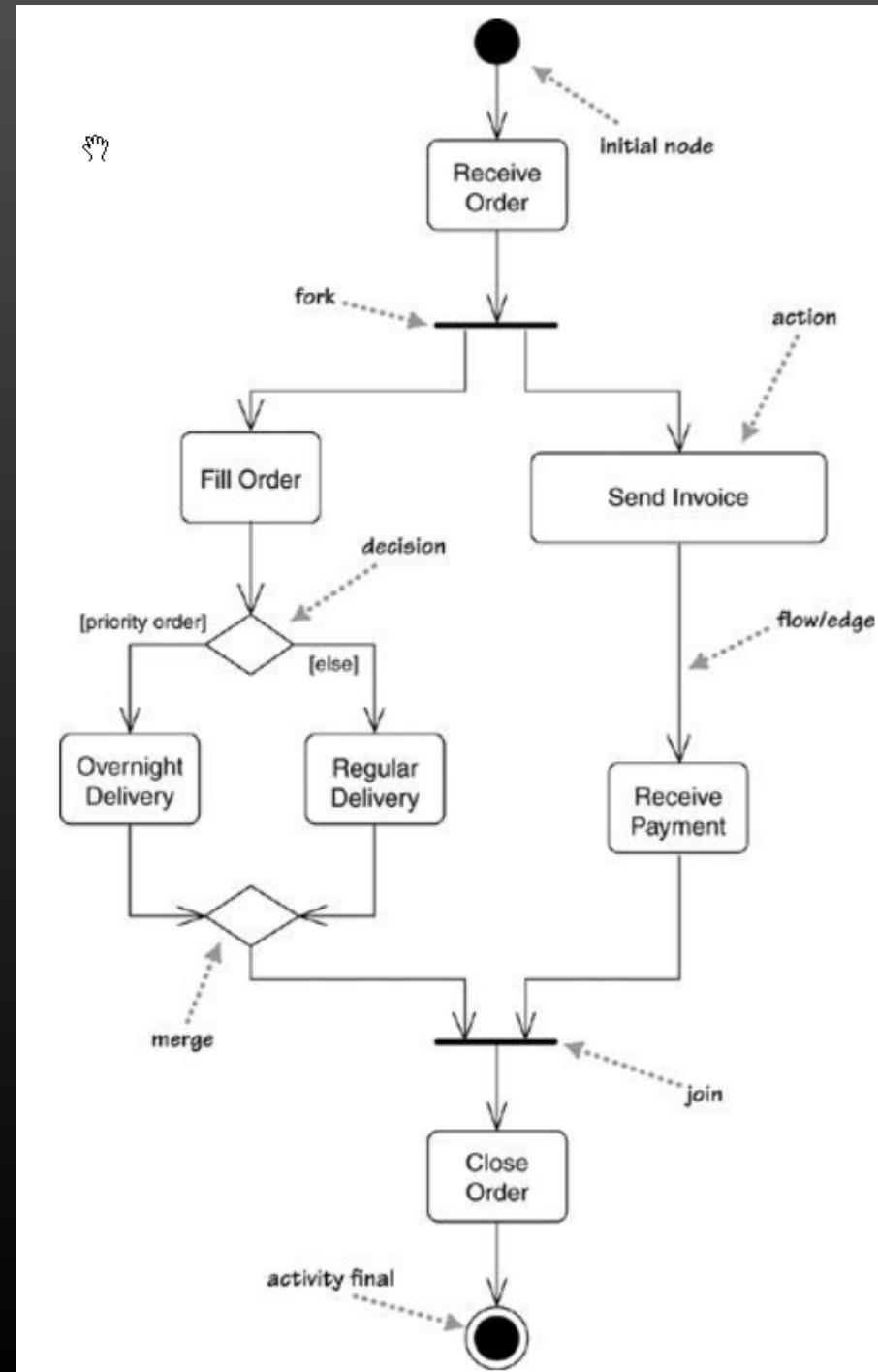
Distinguish structured activities from actions; control flows from data flows

Identify when to use Activity modeling

Diagramas da UML 2.x



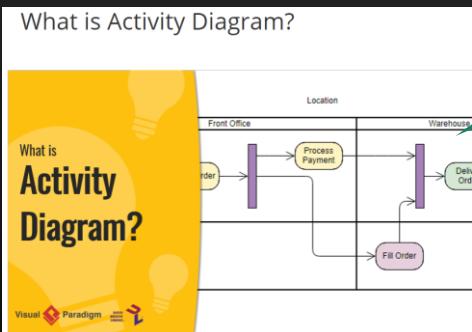
Elementos do diagrama de atividades



Os diagramas de atividade mostram o fluxo de ações (e de dados)

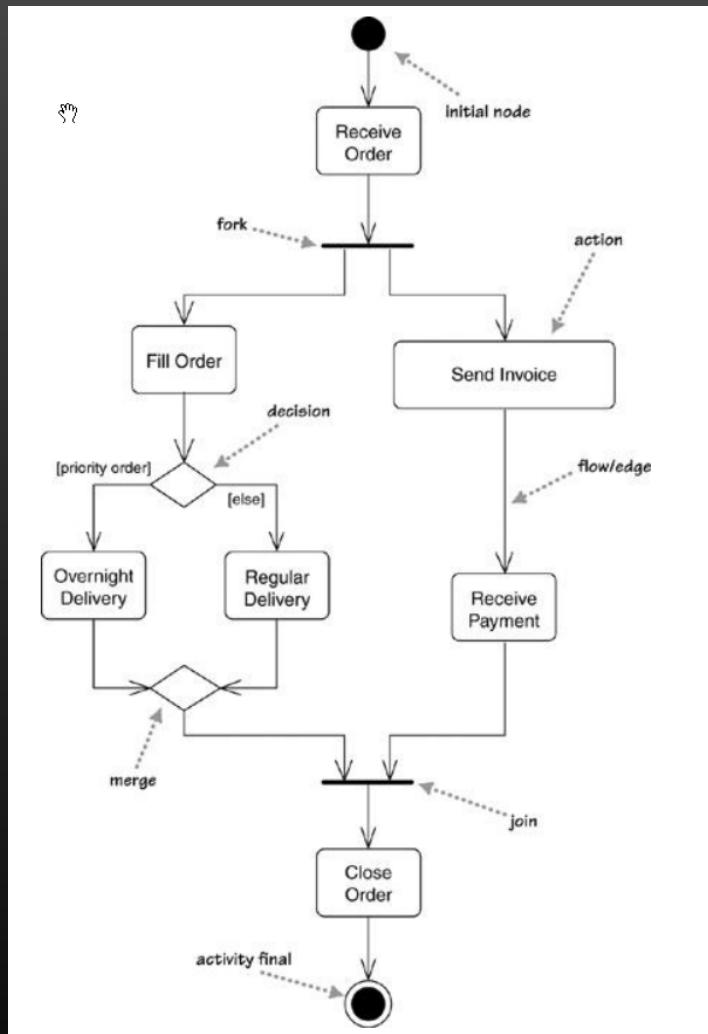
Quando aplicar?

- Modelar fluxos de trabalho/processos de negócio
- Modelar processos computacionais: e.g. descrever um algoritmo complexo

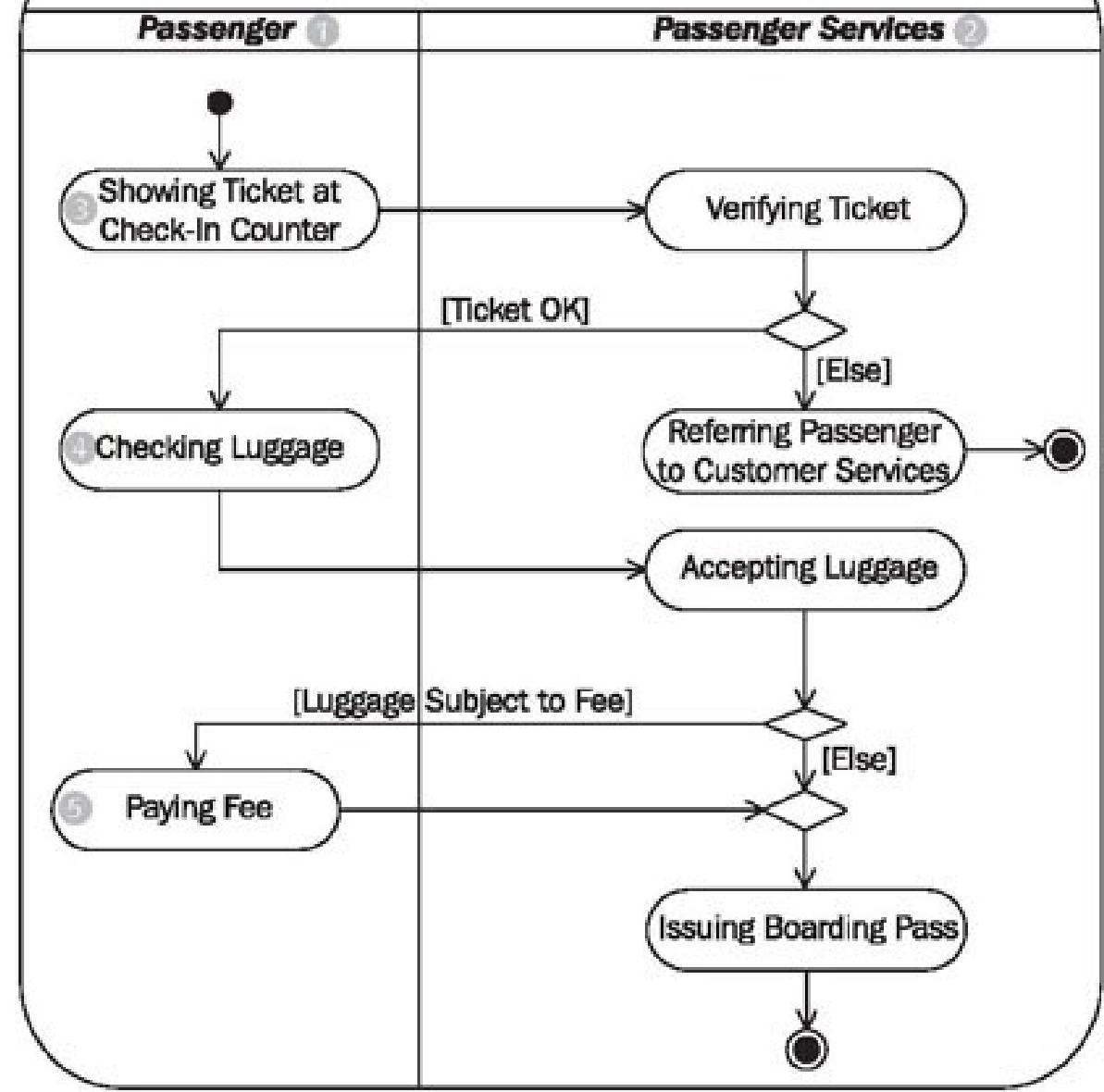


Ver explicação tutorial aqui!

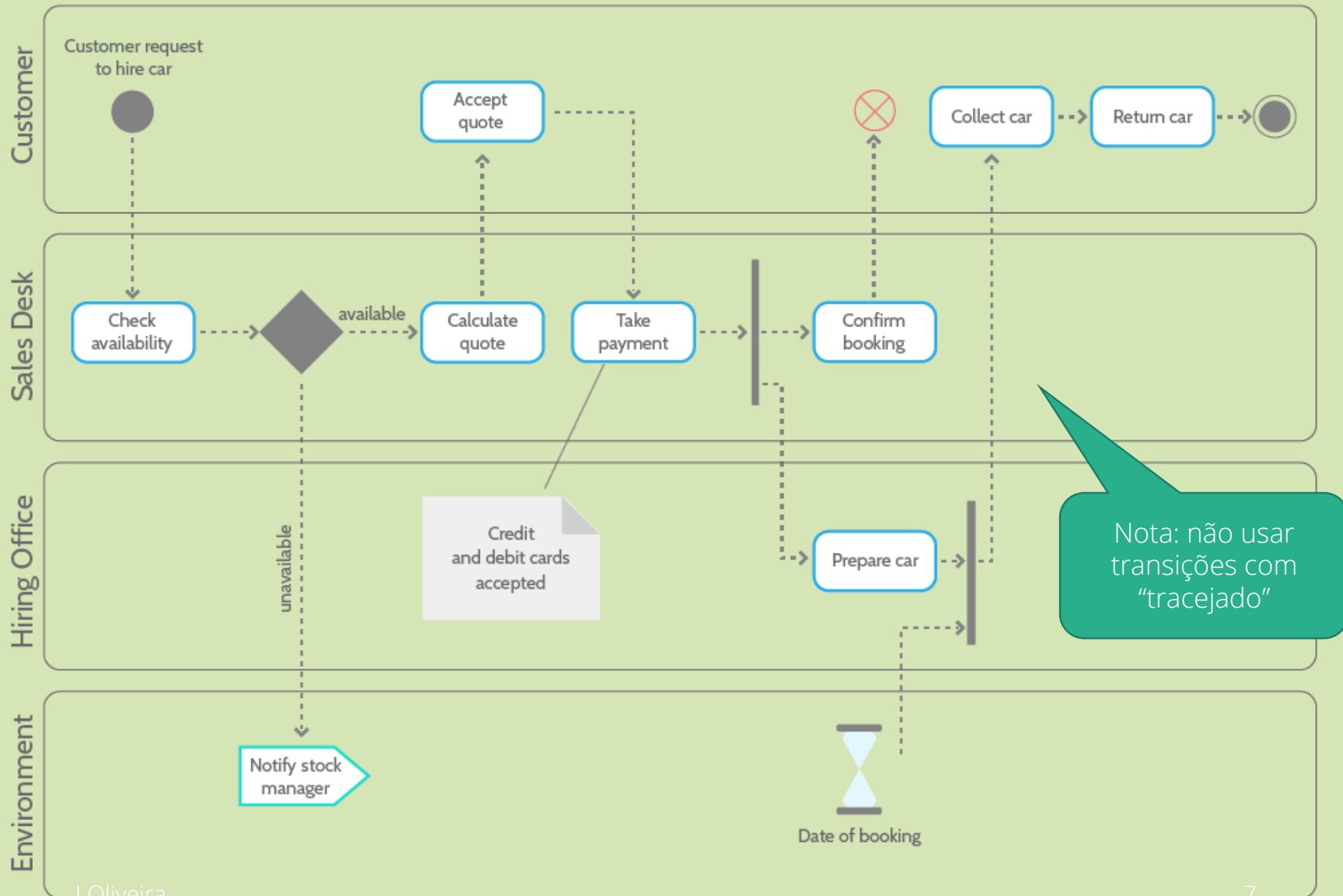
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

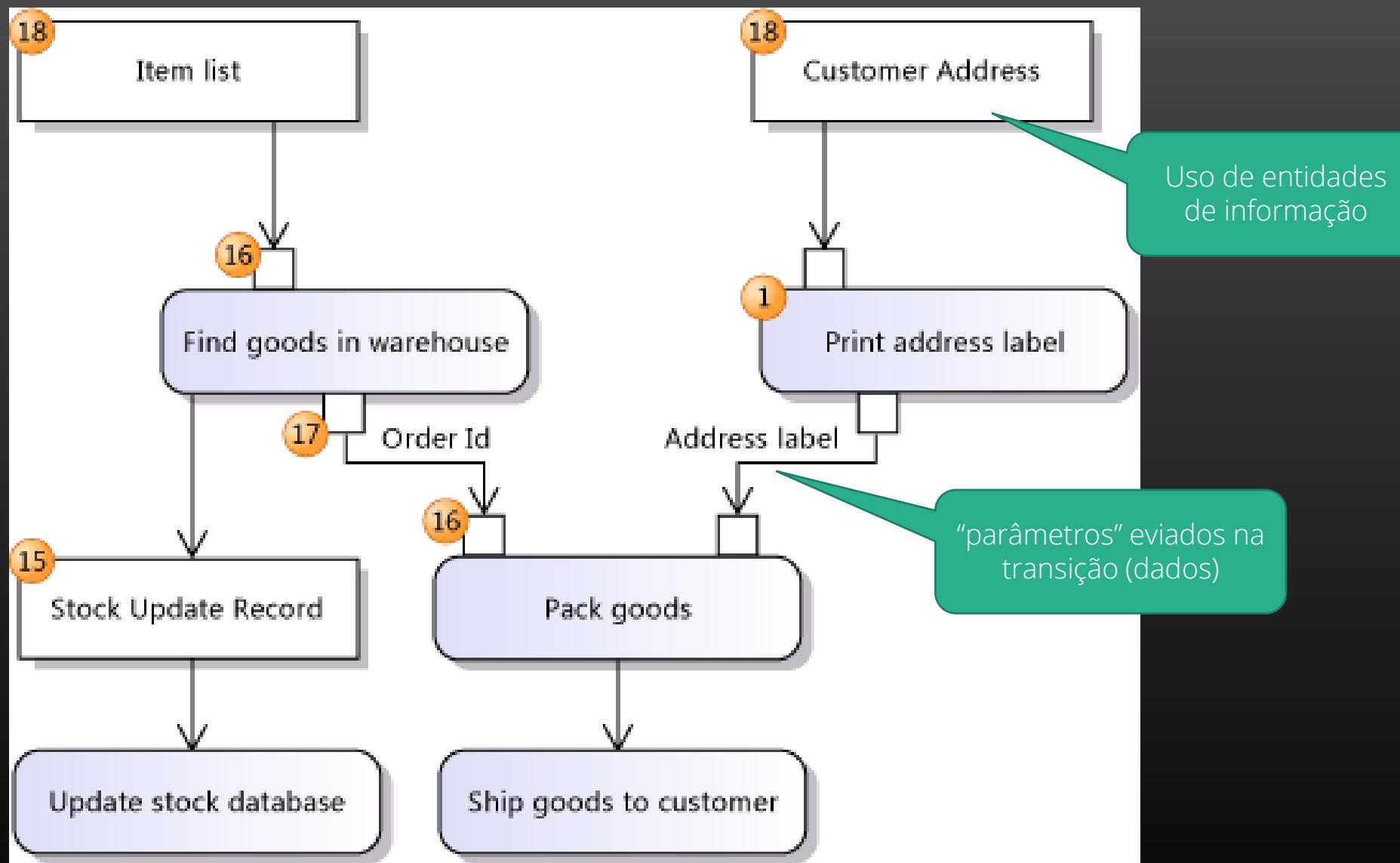


Passenger checks in



Car Hire





Definido na UML.

Diagram	Description	Advantages	Disadvantages
System flowchart	Earliest form for depicting sequencing of activities.	<ul style="list-style-type: none">■ Intuitive. Each type of input and output is clearly marked with its own symbol.■ Includes logic symbols.	<ul style="list-style-type: none">■ Not compliant with UML.■ Can be hard to learn (many symbols).
Activity diagram	UML tool for describing logic. Used to describe entire system, a use case, or an activity within a use case. Has two versions: <ul style="list-style-type: none">■ Activity diagram without partitions (swimlanes): Does not show who does what.■ Activity diagram with partitions: shows who does what.	<ul style="list-style-type: none">■ Part of UML standard.■ Can handle many situations in one diagram.■ Simple diagramming conventions.■ Encourages thinking about opportunities for parallel activities (more than one activity going on at the same time).	<ul style="list-style-type: none">■ Ability to handle many situations can lead to a diagram that is too complex to follow.
Business process diagram (BPD)	Business process modeling notation (BPMN) tool for describing workflow	<ul style="list-style-type: none">■ Part of BMN standard, managed by the OMG■ Rich symbol set can model complex and subtle workflow requirements better than activity diagrams.	<ul style="list-style-type: none">■ Not UML-compliant■ Difficult to understand without prior training

Os diagramas de atividade mostram o fluxo de ações (e de dados)

Quando aplicar?

- Modelar fluxos de trabalho/processos de negócio
- Descrever um algoritmo complexo
- Descrever a sequência de interações entre atores e o sistema sob especificação, num caso de utilização

Pode ser usado para descrever os processos organizacionais existentes / novos

- Neutro em relação à programação
- Bom a captar papéis
- Pode captar o fluxo de dados também

Ter presente os destinatários é ↑ importante na modelação de processos

"Seriously, you need about 20% of the UML to do 80% of the kind of design you might want to do in a project – agile or not – but use the UML with a very light touch: use the notation to reason about a system, to communicate your intent to others...and then throw away most of your diagrams." – Grady Booch

So it is important to keep things as simple as possible if you want to model actionable diagrams that can be internalized by your team and evaluated accurately by managers for effective process optimization.

Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 4	<ul style="list-style-type: none">• Vídeo tutorial com uma explicação dos Diagramas de Atividades• Visual Paradigm tutorials: what is the Activity Diagram• MSDN, “Developing models for software design”

41951- ANÁLISE DE SISTEMAS

Visual Modelling with UML

Ilídio Oliveira | v2022/03/08

Learning objectives for this lecture

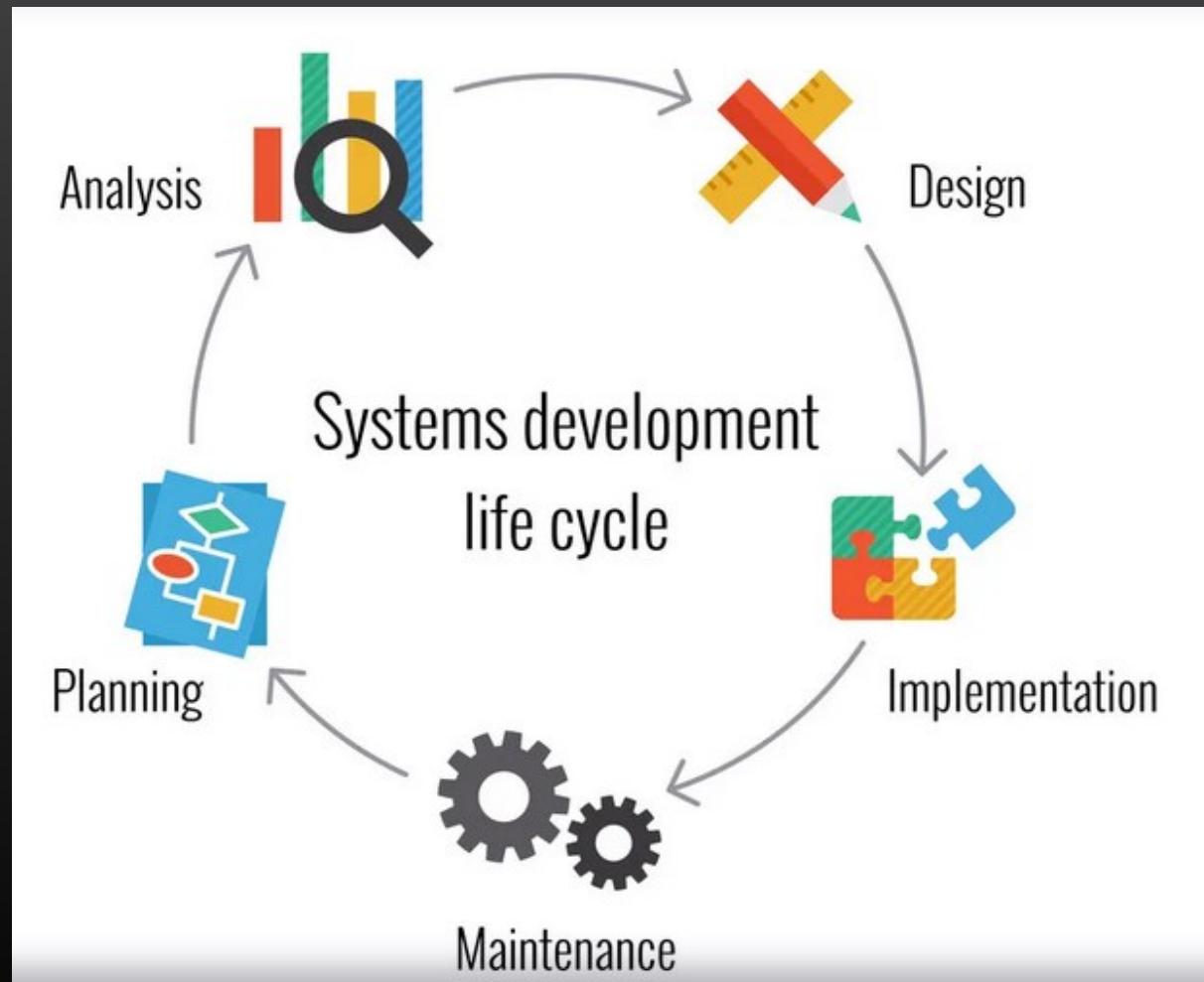
Justify the use of models in systems engineering

Enumerate advantages of visual models

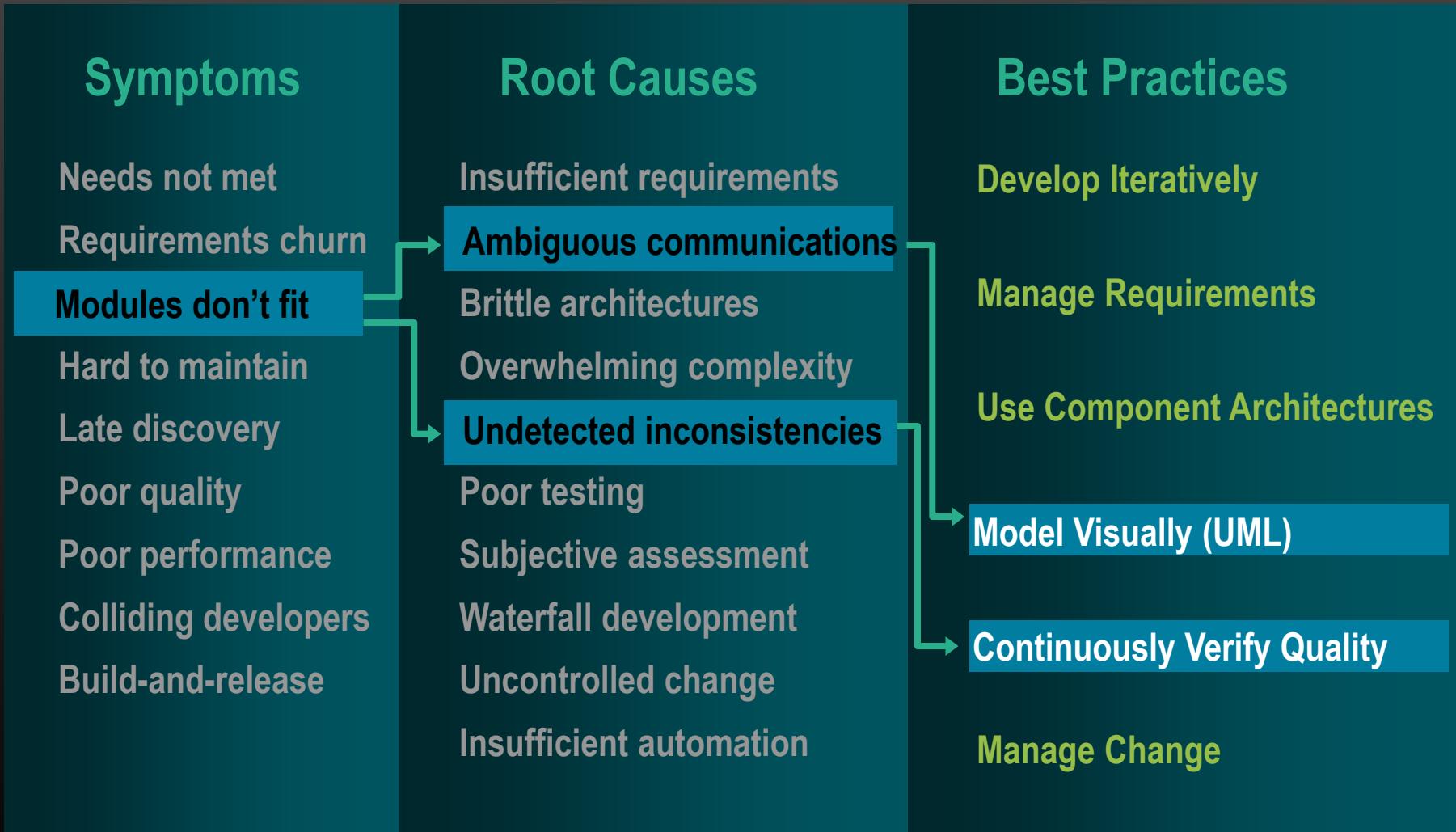
Explain the organization of the UML

Identify the main diagrams in UML and their modeling viewpoint

Systems development lifecycle (SDLC)



Problems and solutions in the SDLC (a Rational Unified Process perspective)



Modeling

UML as a visual specification language

Usamos modelos visuais para captar partes do mundo/realidade

D Trumpet Version

Allegro Assai
from
Brandenburg Concerto #2

J. S. Bach
arranged by Mark Adler

Trumpet

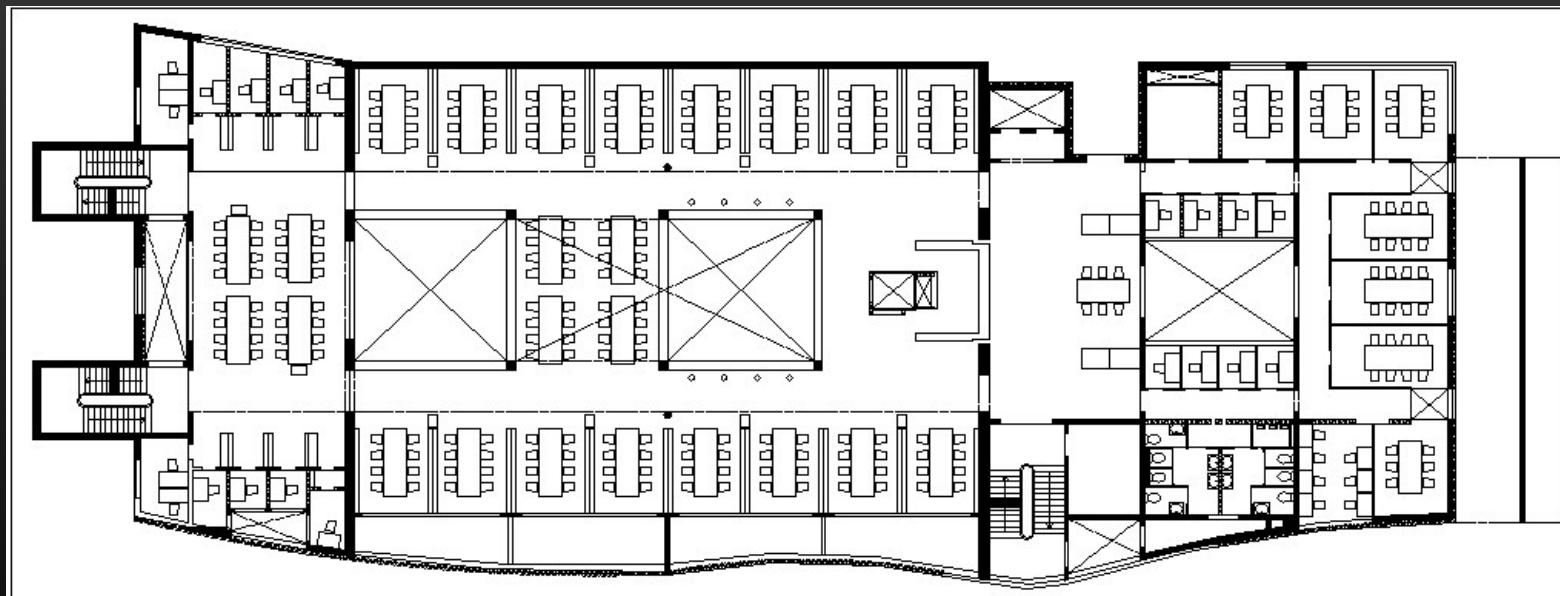
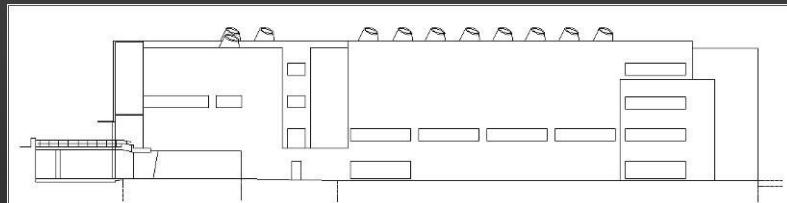
Organ



- Uma linguagem comum (escrever, ler)
- Especificações visuais são mais inteligíveis
- Compor: aplicar talento e disciplinas técnicas
- Orquestra: a prova que os modelos funcionam!



Um modelo é uma simplificação da realidade



Os modelos ajudam a gerir a complexidade

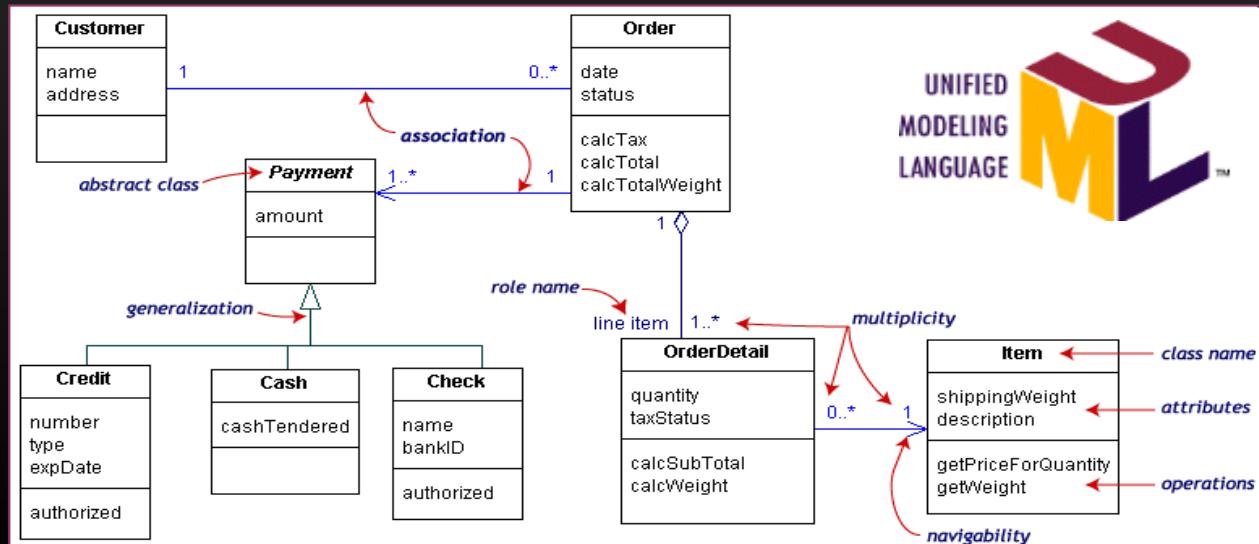
4 razões para usar modelos (G. Booch):

Ajudar a **visualizar** um sistema (*high-level*)

Especificar/documentar a **estrutura e o comportamento** do sistema (antes de implementar)

Serve como **referência** para orientar construção (“planta”)

Documentar as decisões (de desenho) que foram feitas



Modelação visual no desenvolvimento

UML 2: Unified Modeling Language

Linguagem de modelação normalizada

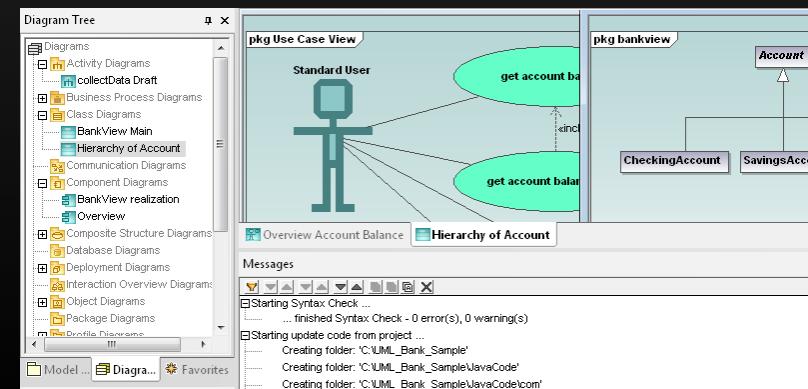
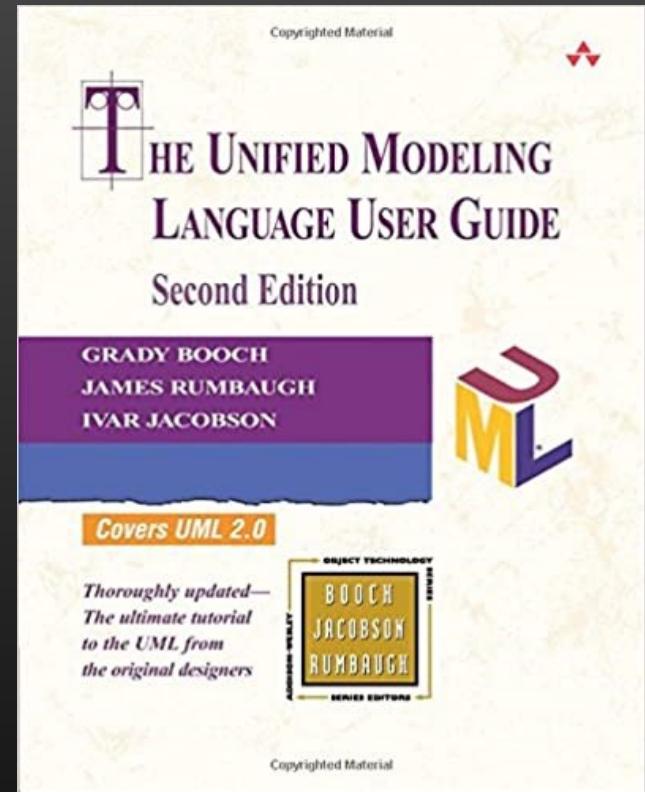
Benefícios

Promover a comunicação mais clara e sucinta

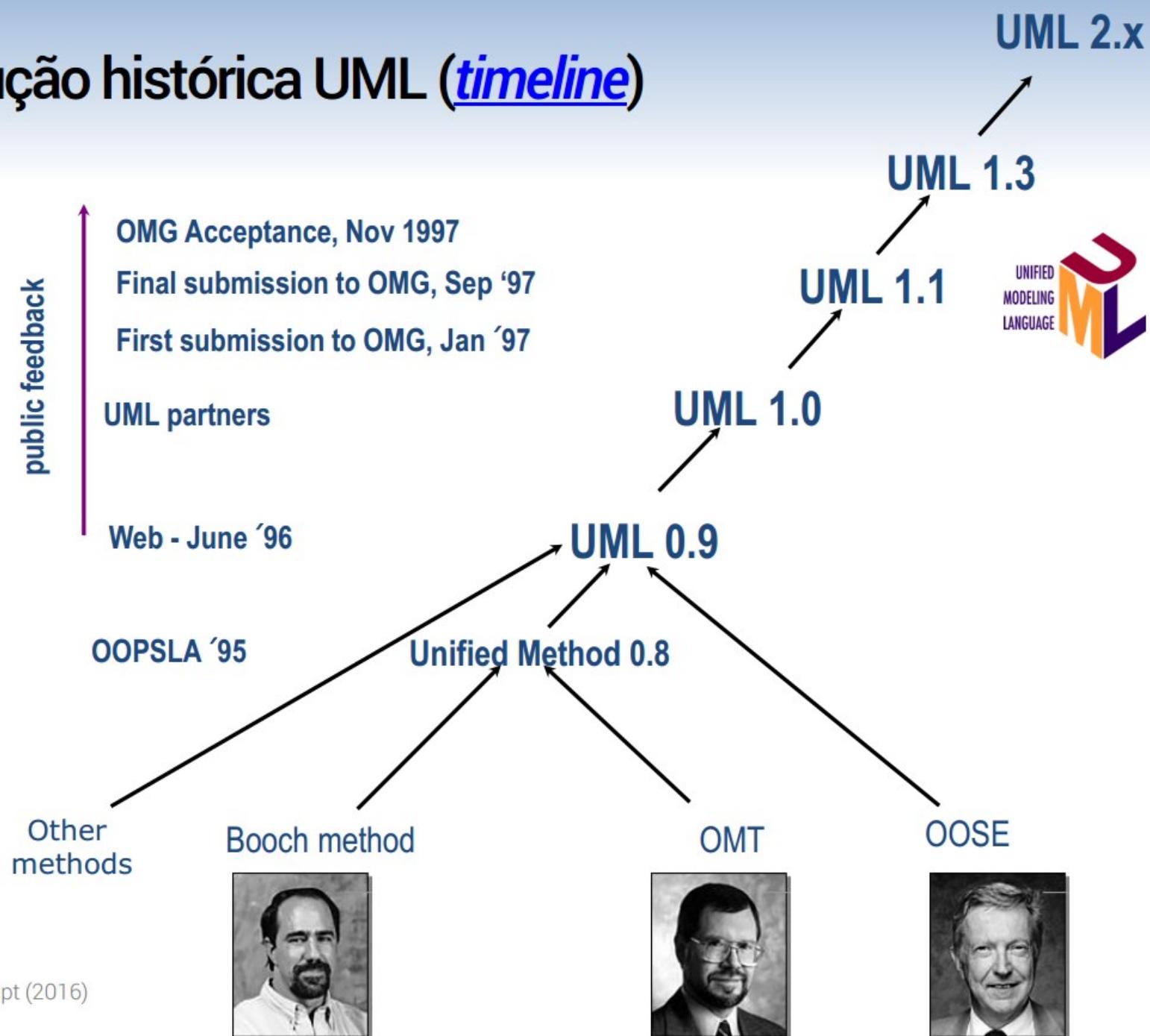
Manter o desenho (planeamento) e a implementação (construção) coerentes

Mostrar ou esconder diferentes níveis de detalhe, conforme apropriado

Pode suportar, em parte, processos de construção automática (gerar a solução a partir do modelo)



Evolução histórica UML (*timeline*)



“Three Amigos”

I. Jacobson: thinking in use-cases...

G. Booch: thinking in OO development process...

J. Rumbauch: thinking in classes modeling...



The UML effort started officially in October 1994, with the version 0.8 draft being released in October 1995. The Three Amigos, as they are sometimes referred to (Booch, Rumbauch, and Jacobson) had successfully unified semantics and notation, ultimately meaning that users could focus on their own work and worry less about the specifics of a given method.

UML é uma especificação do OMG

[ABOUT US](#)[RESOURCE HUB](#) ▾[OMG SPECIFICATIONS](#) ▾[PROGRAMS](#) ▾[MEMBERSHIP](#) ▾[MEMBERS AREA](#) ▾

ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION VERSION 2.5.1

2.5.1 • UML • SPECIFICATIONS

UML®**Unified Modeling Language**

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

Title: Unified Modeling Language

Acronym: UML®

Version: 2.5.1

Document Status: formal ⓘ

Publication Date: December 2017

Categories: [Modeling](#) [Software Engineering](#)



Specification

Também reconhecida como um standard internacional ISO

The screenshot shows the ISO website interface. At the top, there's a dark header bar with the ISO logo, navigation links for Standards, About us, Standards Development, News, Store, and a search bar labeled "Search ISO". Below this is a blue navigation bar with links for Standards catalogue, Online collections, and Graphical symbols. The main content area shows the breadcrumb path: ISO Store > Store > Standards catalogue > By TC > JTC 1 Information technology > SC 7. The title "ISO/IEC 19505-1:2012" is prominently displayed, followed by the subtitle "Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 1: Infrastructure". Below the title is a section titled "Abstract" with a "Preview ISO/IEC 19505-1:2012" button. A detailed description of the standard's purpose follows. To the right, there's a "FORMAT" dropdown menu set to "PDF" and a "LANGUAGE" dropdown menu set to "English".

ISO/IEC 19505-1:2012[®]

Information technology -- Object Management Group Unified Modeling Language (OMG UML) -- Part 1: Infrastructure

Abstract

Preview ISO/IEC 19505-1:2012

ISO/IEC 19505-1:2012 defines the Unified Modeling Language (UML), revision 2. The objective of UML is to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes.

FORMAT ?

PDF

PAPER

LANGUAGE

English

English



Aplicações principais da UML

Análise e desenho de sistemas de software

Estrutura e comportamento de sistemas baseados em software

- Elementos do modelo representam entidades do mundo do software

Especialmente adequada para o desenvolvimento por objetos (*object-oriented*)

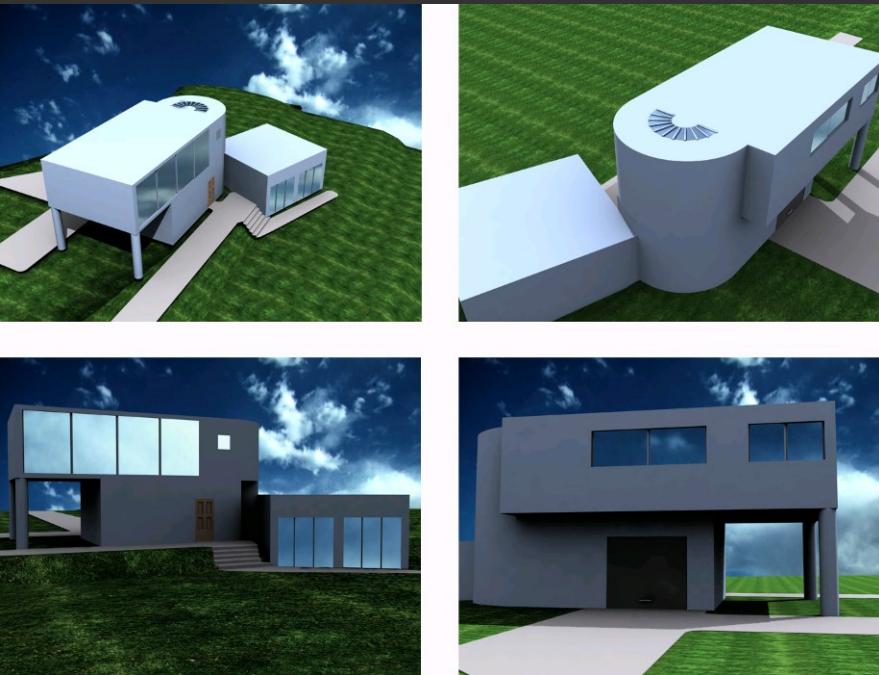
Domínio do problema (processos de trabalho,...)

Especificar ou documentar o domínio de aplicação/negócio

- Elementos do modelo representam entidades do negócio

Não implica ou assume uma implementação em software

Não há uma vista única, mas várias e complementares



Para que serve o sistema?

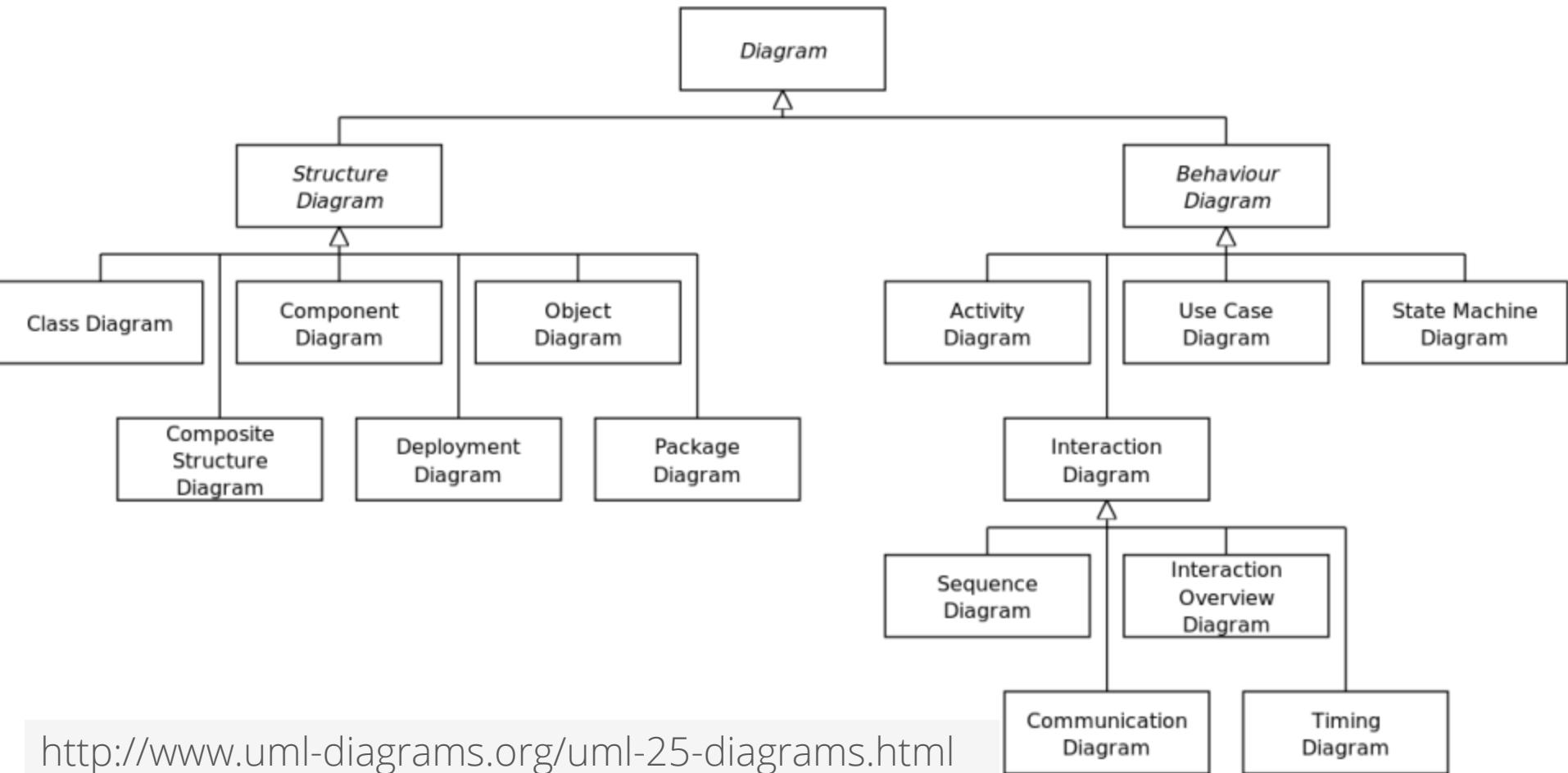
Quais são as estruturas de informação?

Decomposição funcional de atividades complexas

Visualizar a organização do software em partes e as suas interações

Etc.

Diagramas da UML 2.x

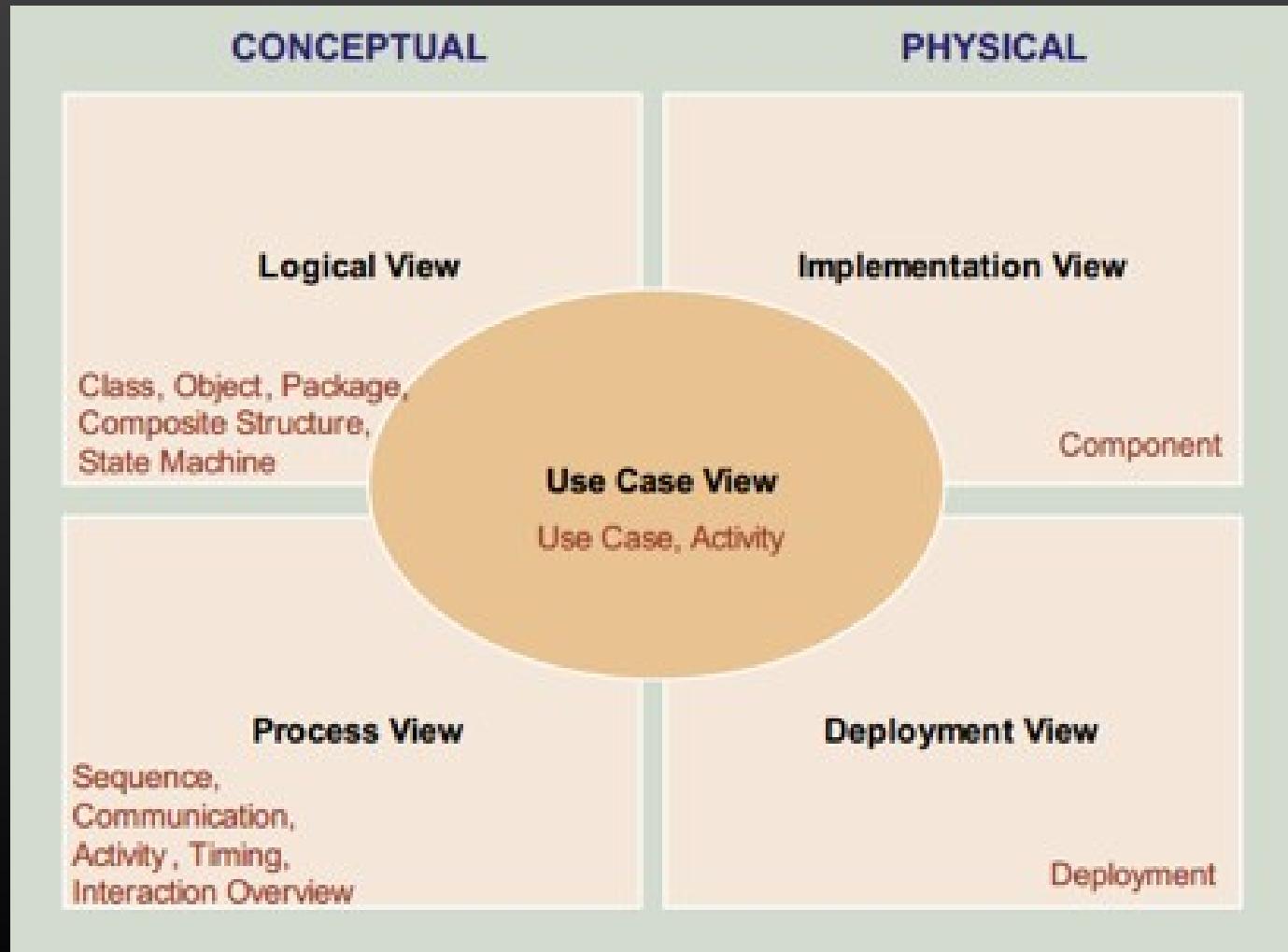


<http://www.uml-diagrams.org/uml-25-diagrams.html>

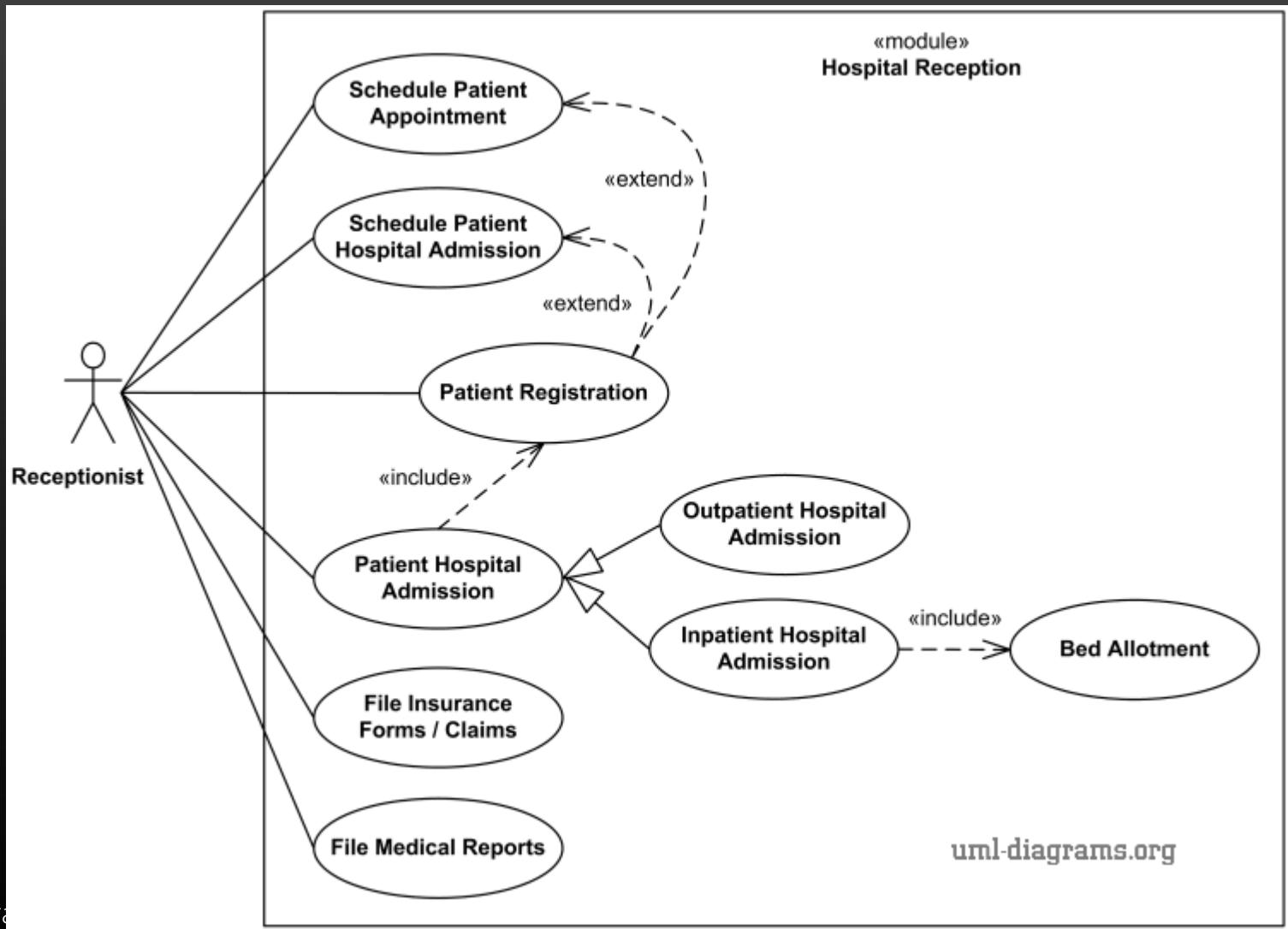
APLICAÇÃO DA UML AO LONGO DO PROCESSO DE DESENVOLVIMENTO

ILÍDIO OLIVEIRA ico@ua.pt
v2017-06-02

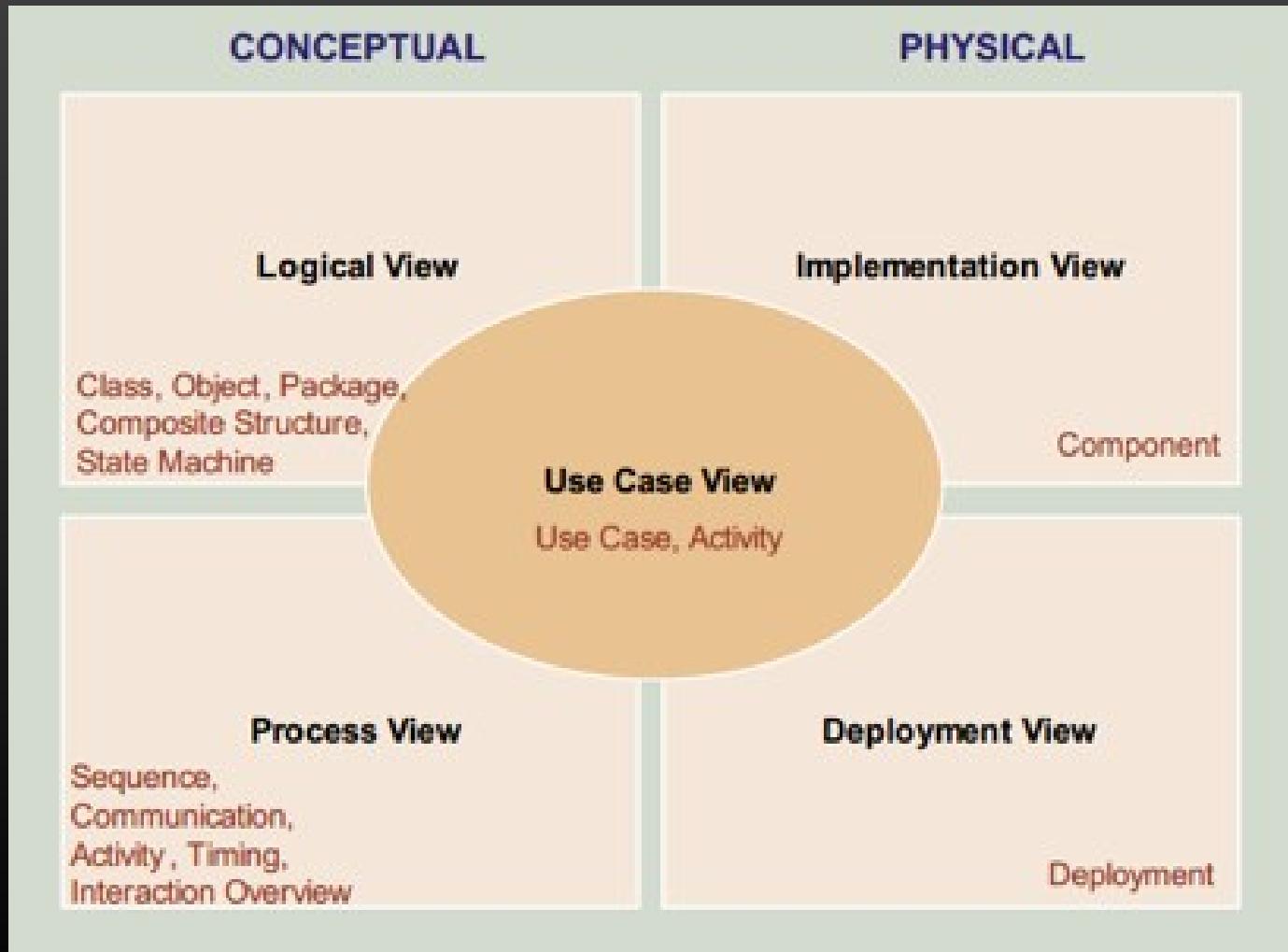
Diversos diagramas para abranger diferentes perspetivas de análise



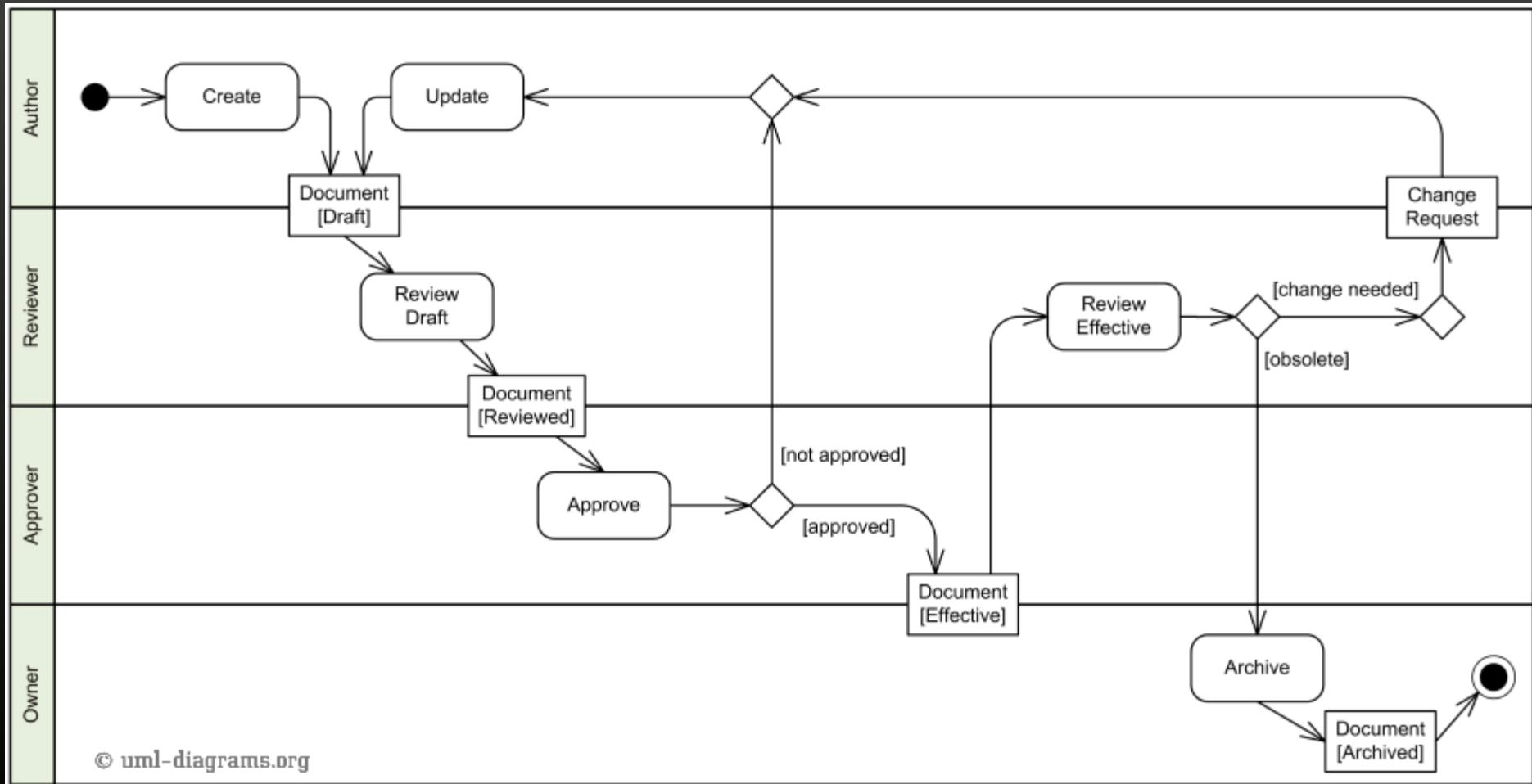
CaU do Sistema: organizar a funcionalidade do sistema em episódios de utilização



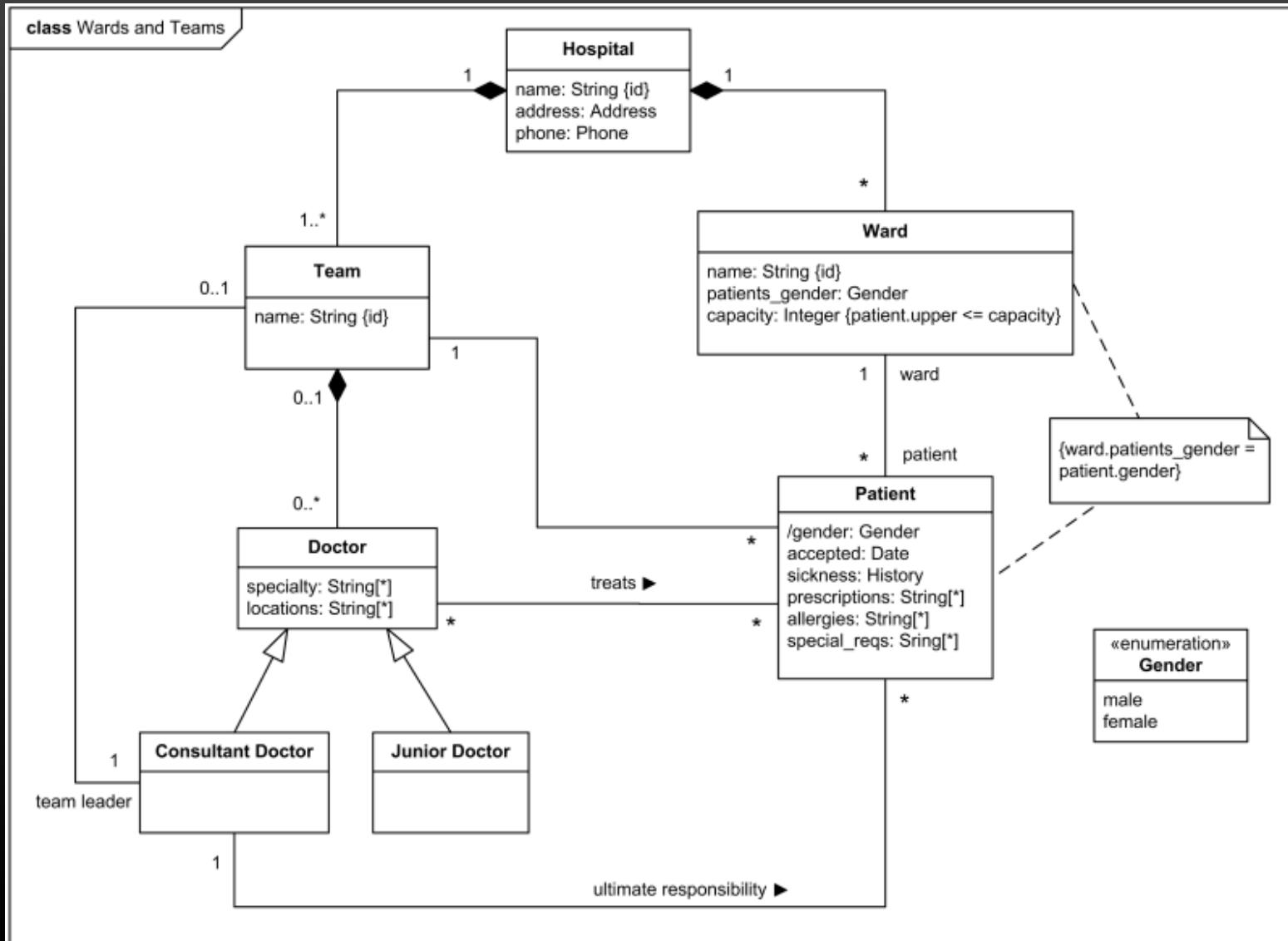
Diversos diagramas para abranger diferentes perspetivas de análise



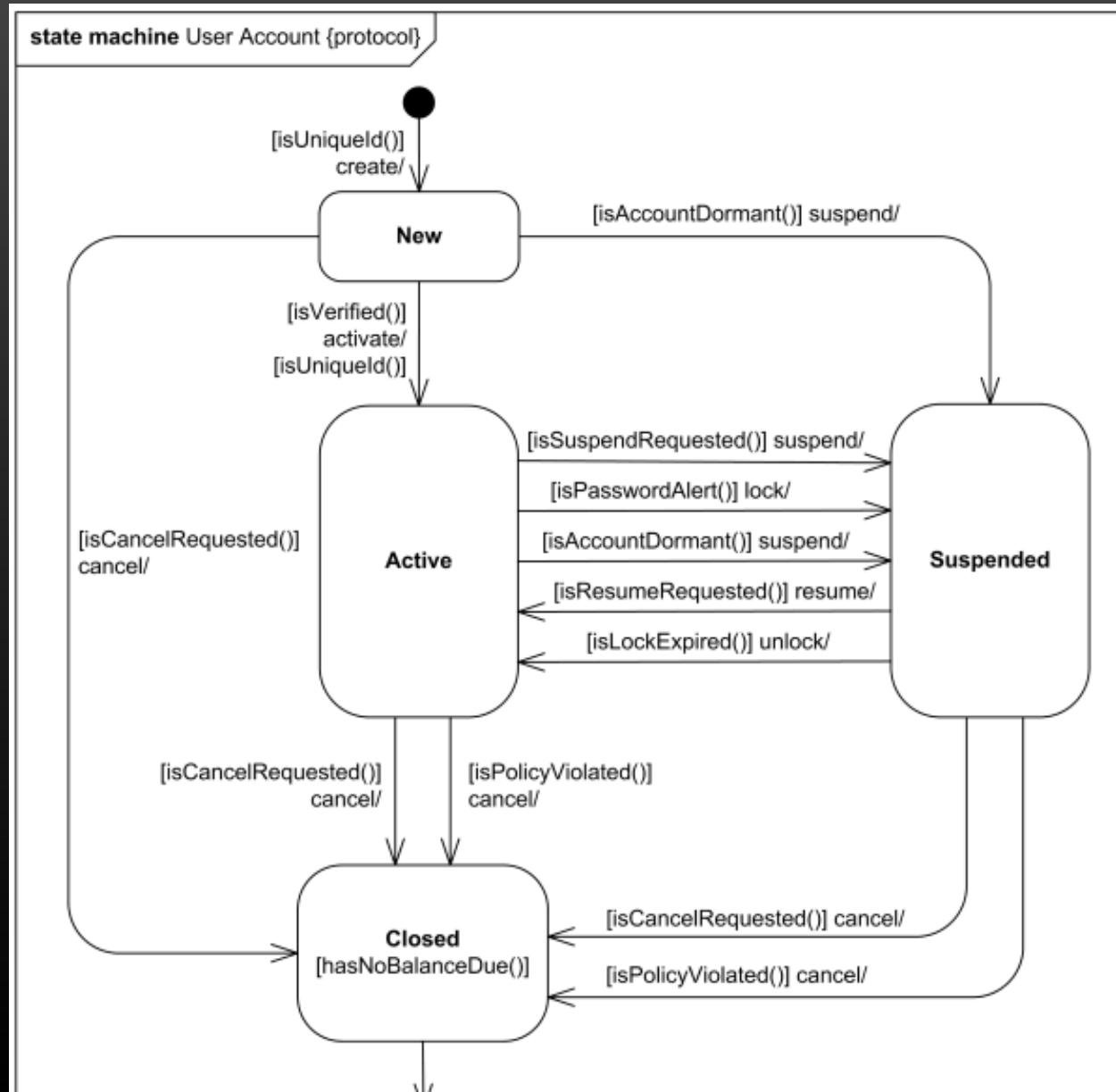
Diagramas de atividades para explicar procedimentos do domínio



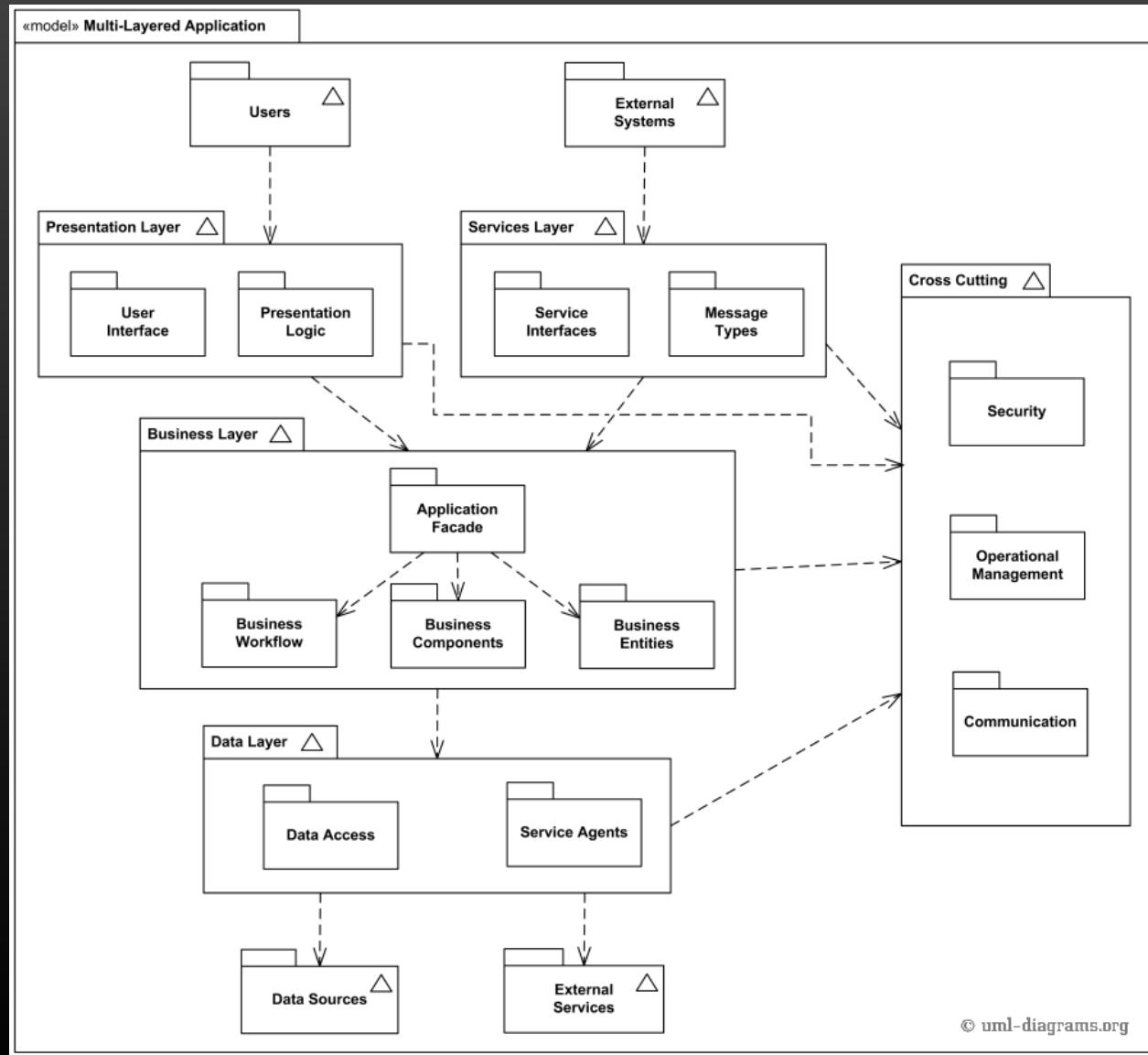
Classes para representar os conceitos da área do problema (modelo do domínio)



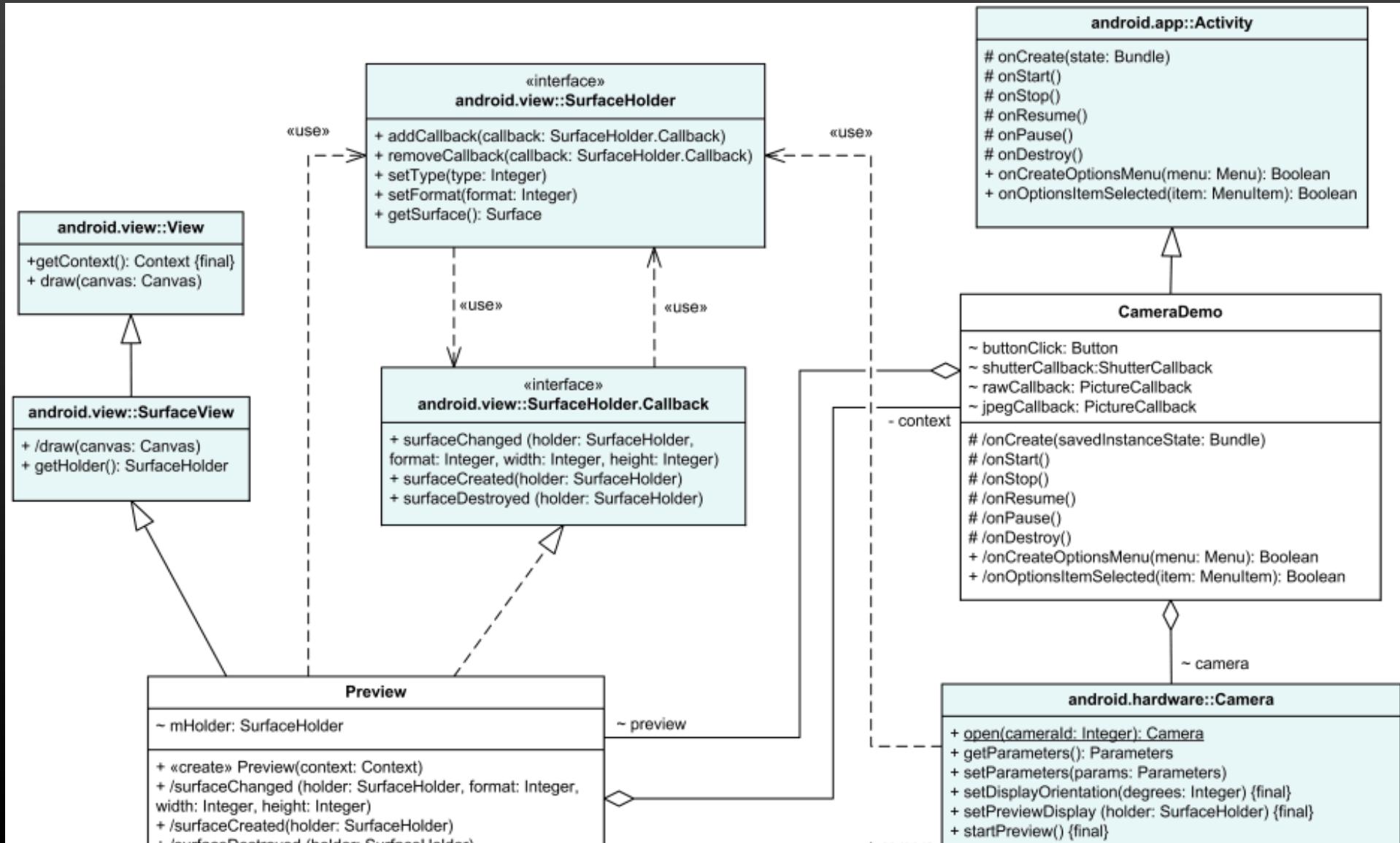
Máquina de estados de entidades/objetos



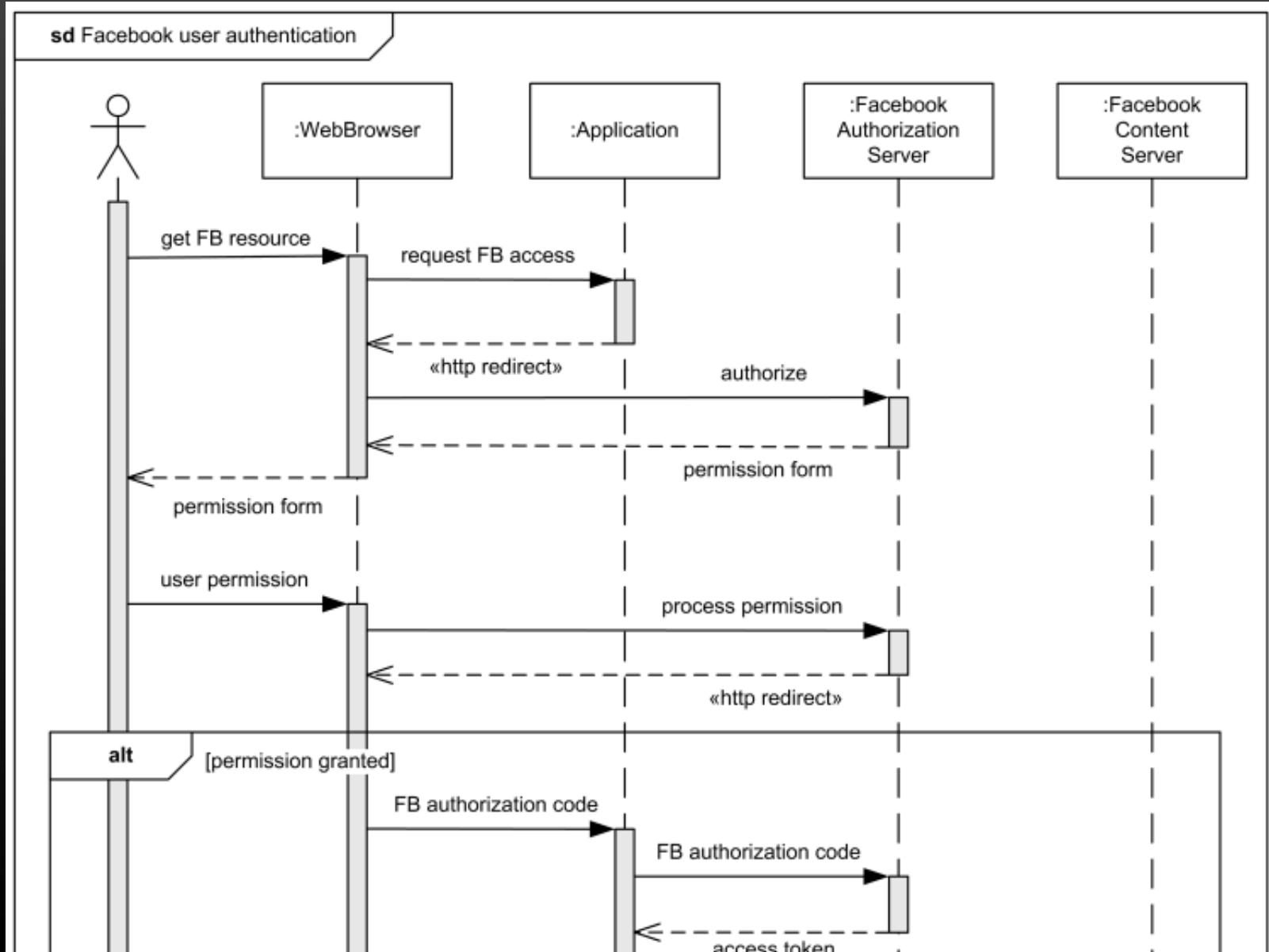
Visualizar a arquitetura lógica com D. Pacotes



Classes para visualizar objetos de um linguagem de programação

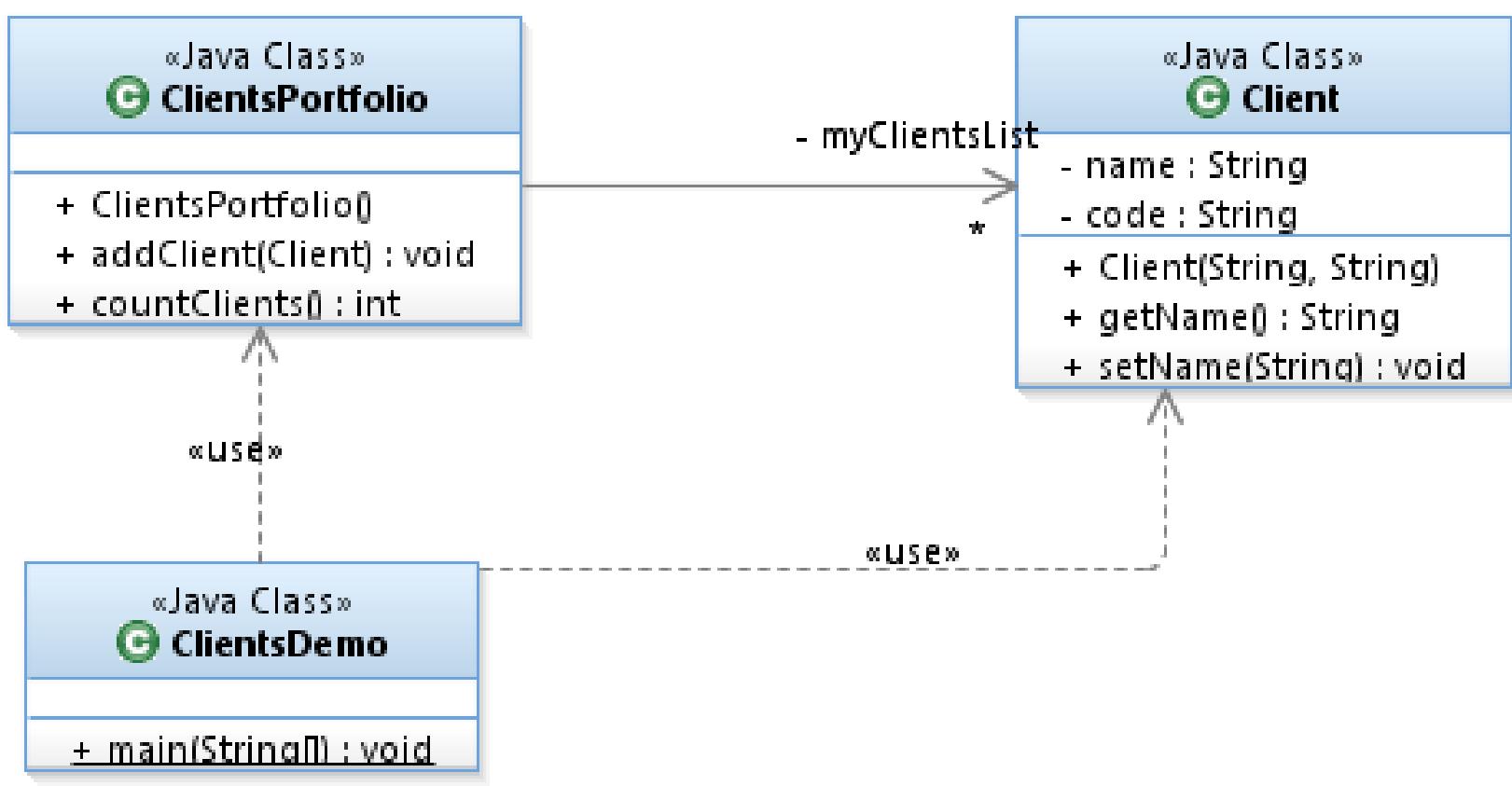


Interações entre componentes do software

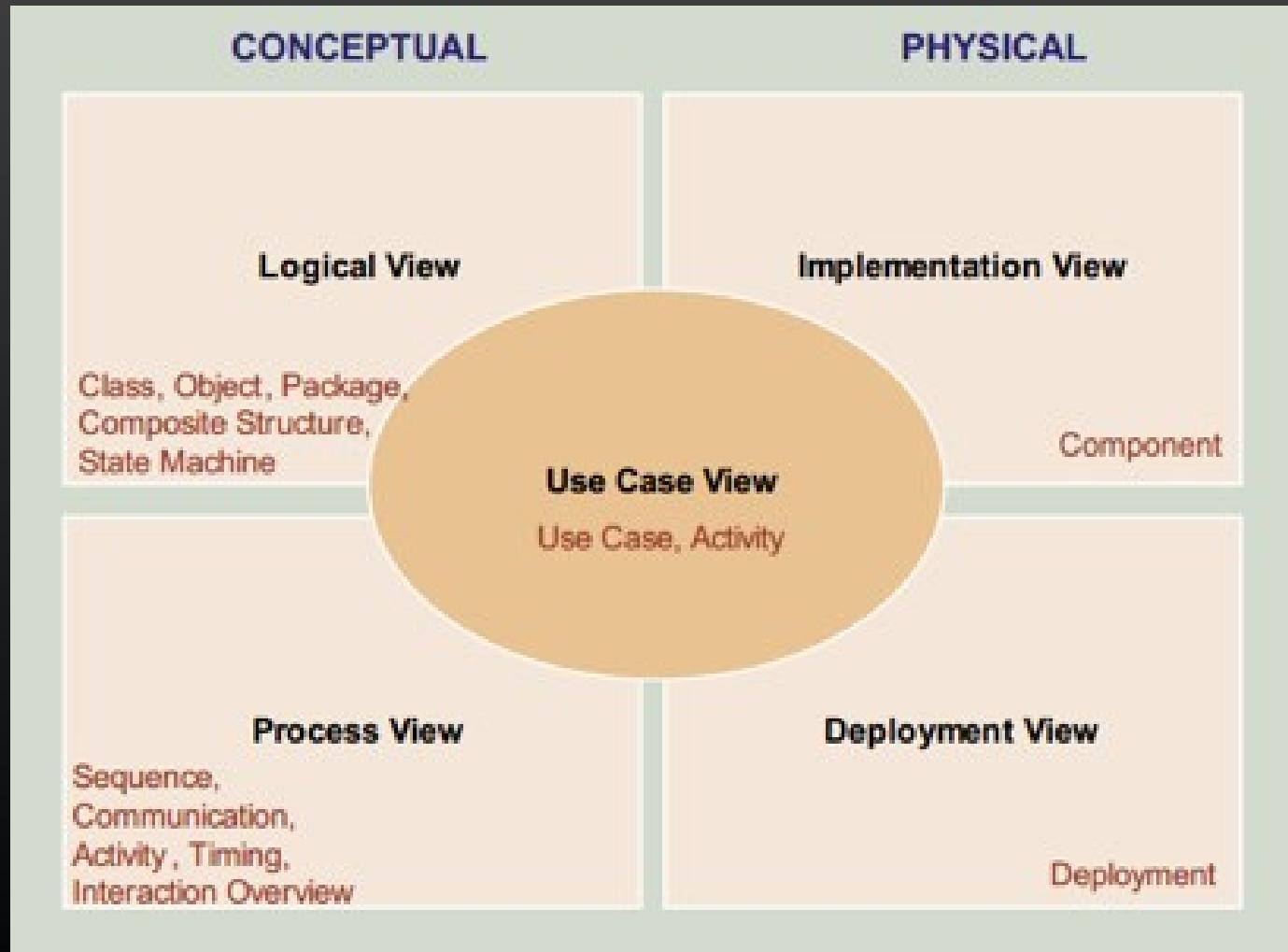


Objetos em código

```
public class ClientsPortfolio {  
    private ArrayList<Client> myClientsList;  
  
    public ClientsPortfolio() {  
        myClientsList = new ArrayList<>();  
    }  
    public void addClient(Client newClient) {  
        this.myClientsList.add(newClient);  
    }  
    public int countClients() {  
        return this.myClientsList.size();  
    }  
}
```



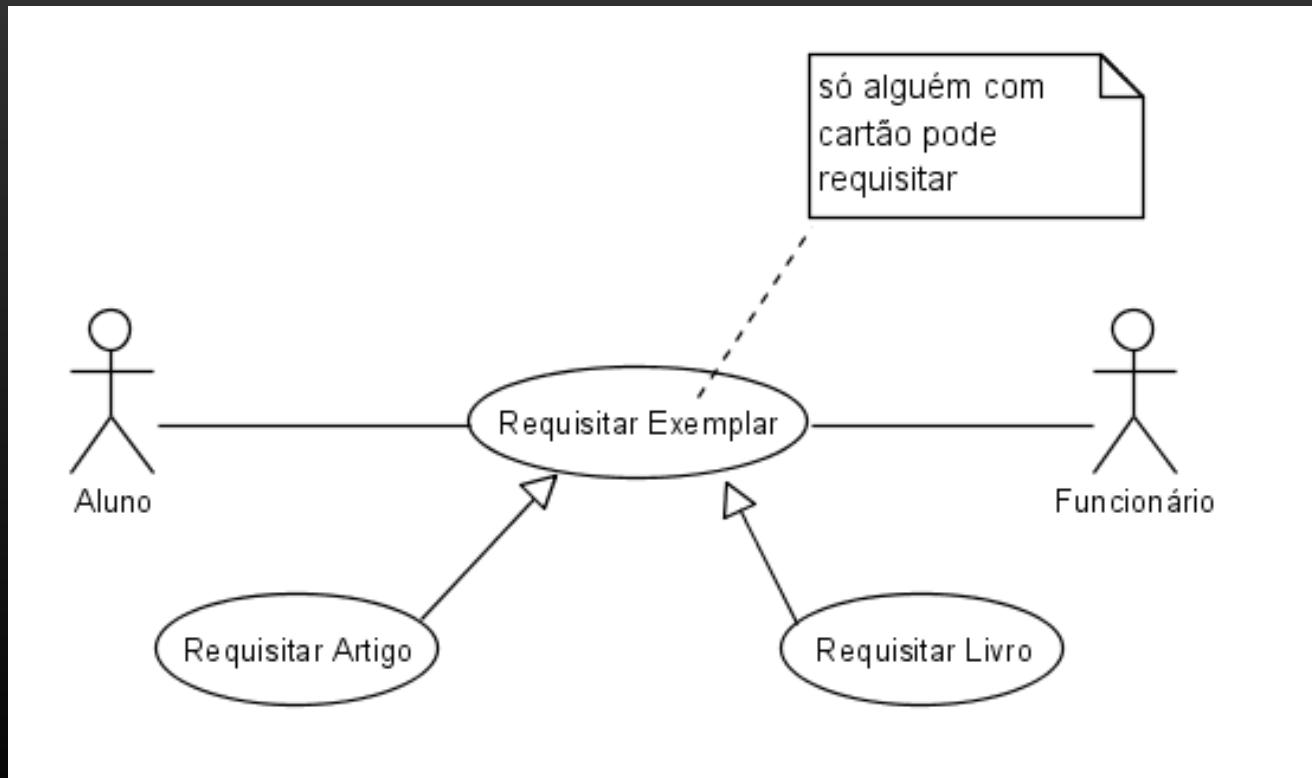
Diversos diagramas para abranger diferentes perspetivas de análise



Elementos comuns

Anotações

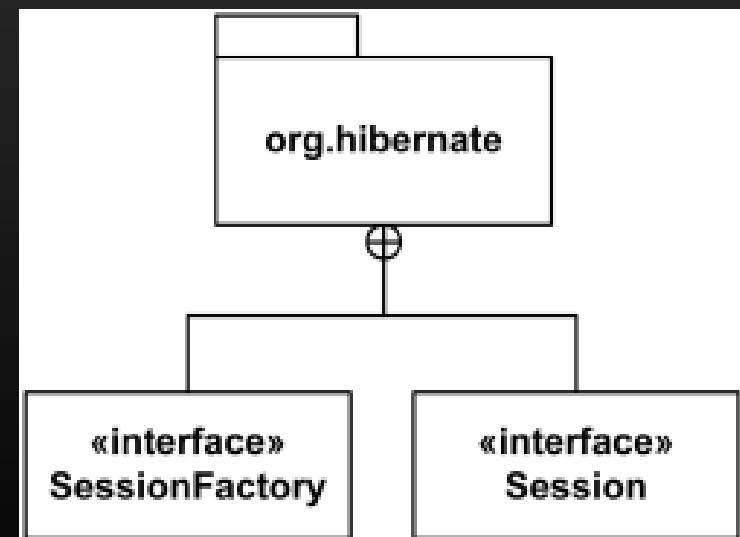
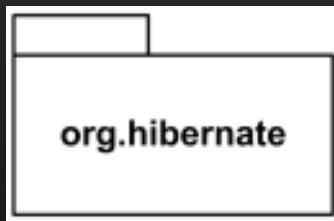
Um comentário que pode ser usado para anotar qualquer elemento



Pacotes

um mecanismo para dividir um modelo em partes

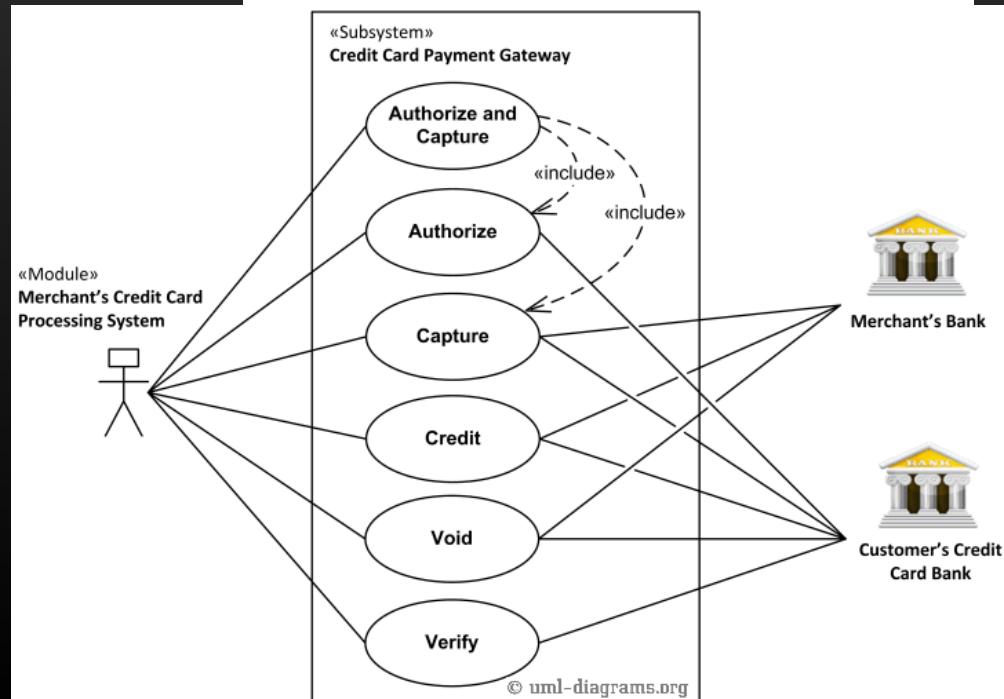
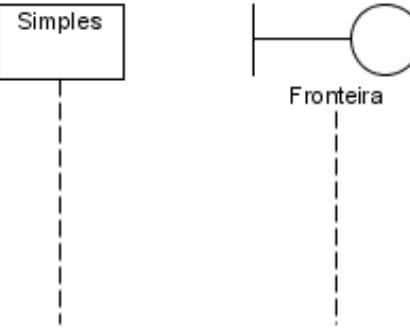
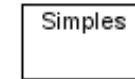
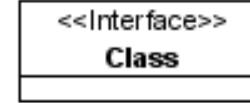
serve como mecanismo genérico para fazer agrupamentos



Estereotipo (*stereotype*)

uma especialização da semântica de um elemento do modelação

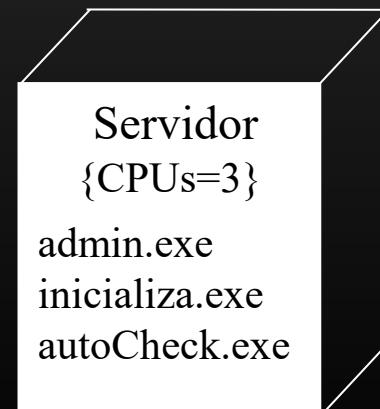
marcada com «...» ou com a alteração da decoração



Valores etiquetados (*tagged values*)

Estender elementos do modelo com uma linguagem “computável” (pares atributo/valor)

```
«Computer»  
{Vendor = "Acer",  
CPU = "AMD Phenom X4",  
Memory = "4 GB DDR2"}  
Aspire X1300
```

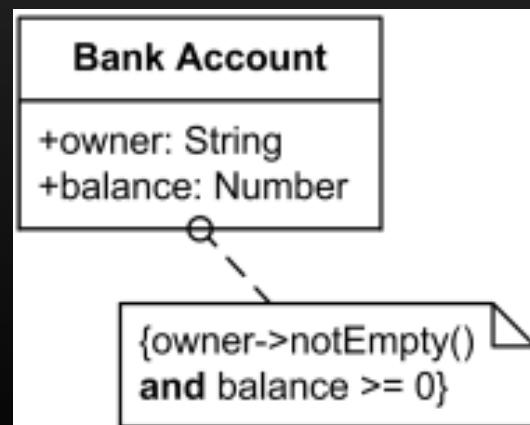
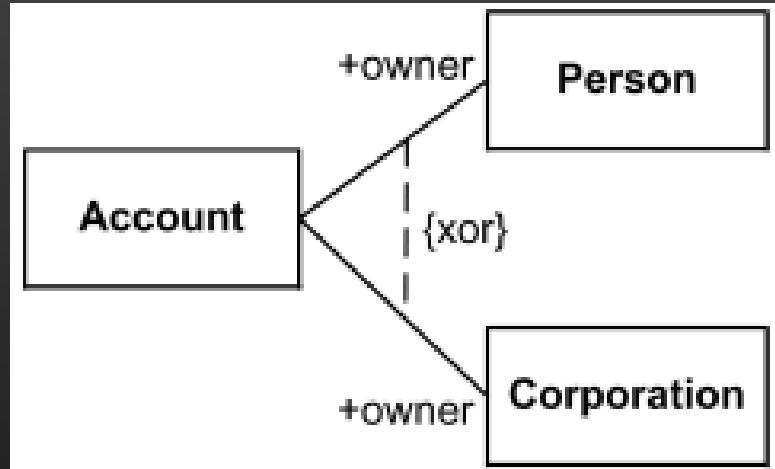
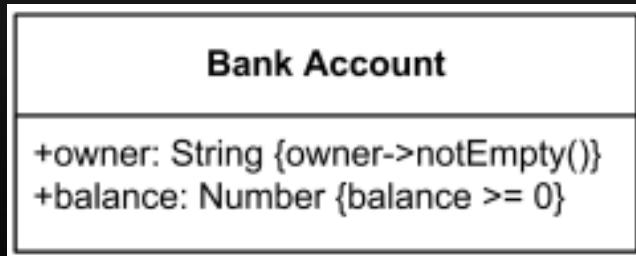


Restrições

Adicionar regras ao modelo ou condicionar a sua interpretação

condição ou restrição relacionada com um ou mais elementos

Linguagem própria para declarar restrições (OCL)



Readings & references

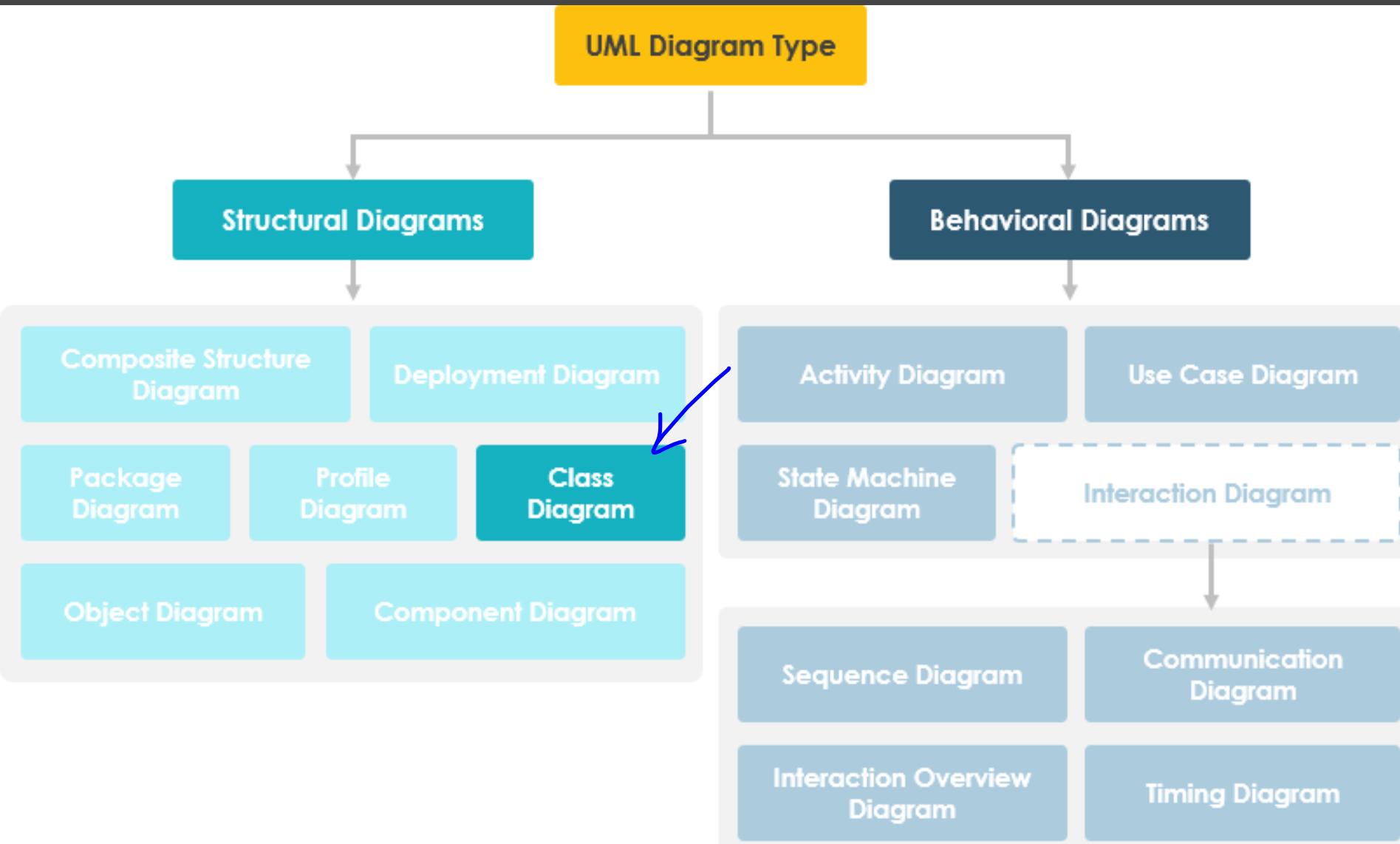
Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 1	[LAR'12] Larman, C. (2012). <i>Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development</i> . Pearson Education. → chap. 10, chap. 15.

41951- ANÁLISE DE SISTEMAS

Modelação com classes – cont. Diagrama de classes da UML

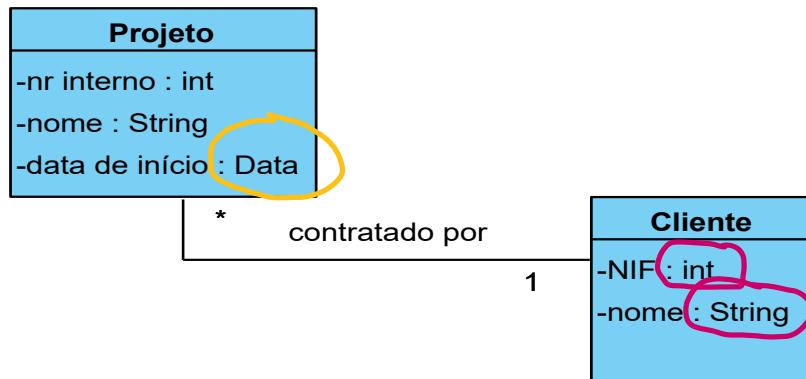
Ilídio Oliveira

v2022-03-15

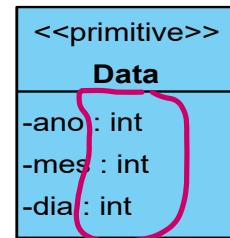


Tipos de dados

Visual Paradigm Standard(I Oliveira(Universidade de Aveiro))



Um tipo de dados auxiliar. Não é um conceito do domínio.

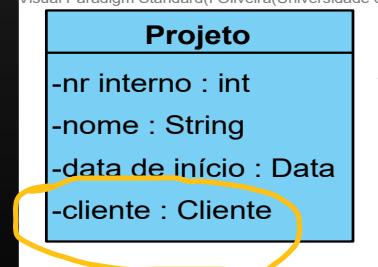


Um modelo completo especifica os tipos de dados dos atributos.

Podemos usar os tipos “básicos” ou criar tipos adicionais

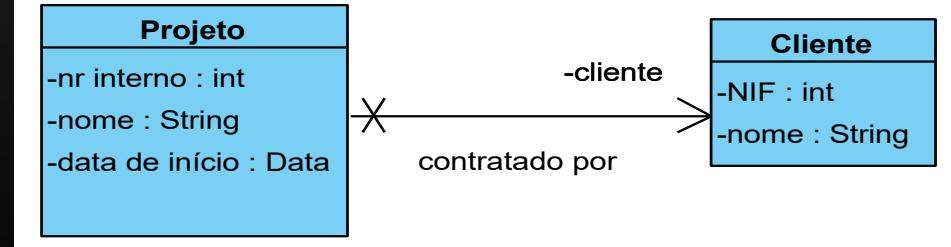
Na verdade, qualquer classe pode ser usada para indicar um tipo de dados de um atributo (básicos vs objetos)

Visual Paradigm Standard(I Oliveira(Universidade de Aveiro))



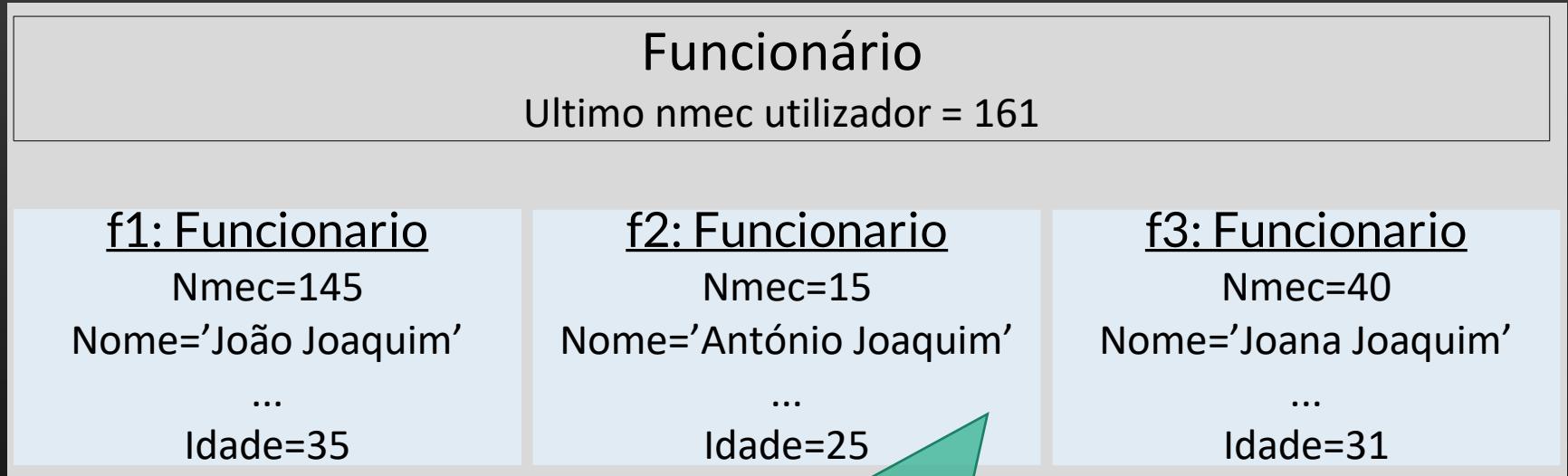
Um tipo de dados que é uma classe do domínio implica uma associação.
Vamos preferir mostrar como associação.

Visual Paradigm Standard(I Oliveira(Universidade de Aveiro))



Atributos da classe (*static*)

Atributos de classe são partilhados por todas as instâncias

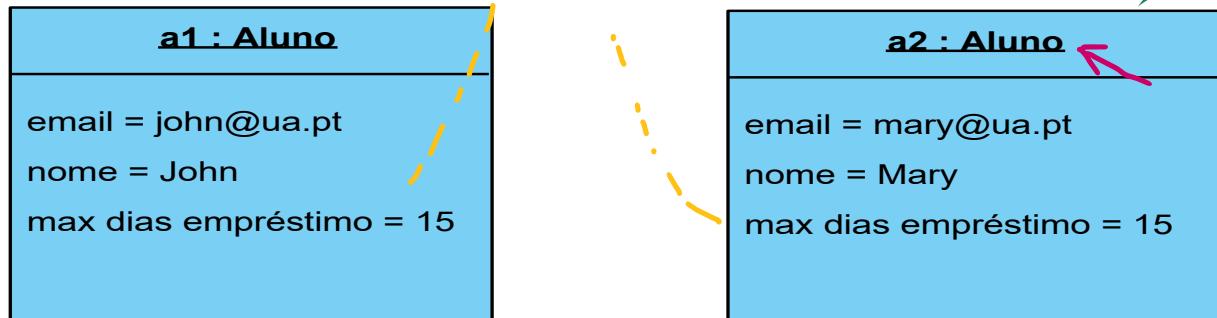
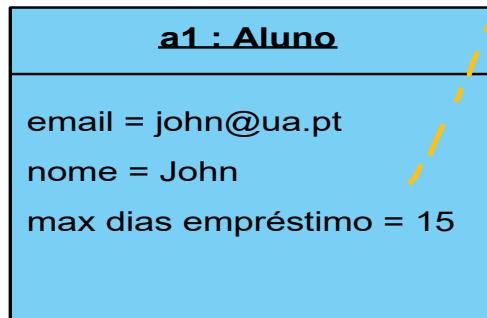
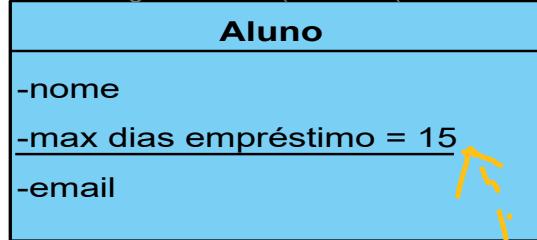


Os objetos f1, f2, f3 têm estado interno,
independente entre si.

Podem existir atributos que afetem o conjunto
dos objetos, i.e., atributos cujo valor é partilhado
por todos: "Ultimo nmec utilizado".
Por isso, o âmbito deste atributo deve ser o
classificador (e não a instância)

Atributos da classe (*static*)

Visual Paradigm Standard(I Oliveira(Universidade de Aveiro))



Classe com um atributo *static*
(valor partilhado por todas as instâncias)

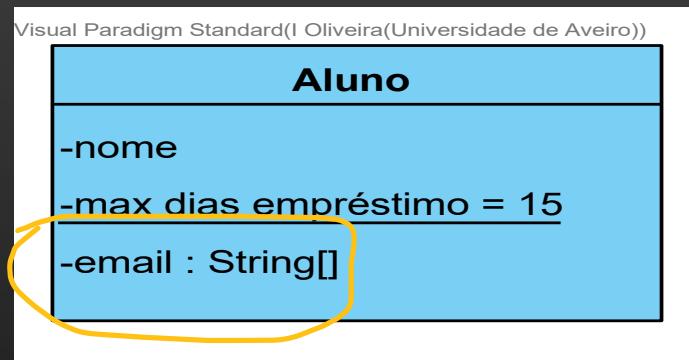
Alguns objetos que instanciam a classe. E.g.: o objeto chamado "a2" do tipo "Aluno"

} instâncias

Atributos multi-valor

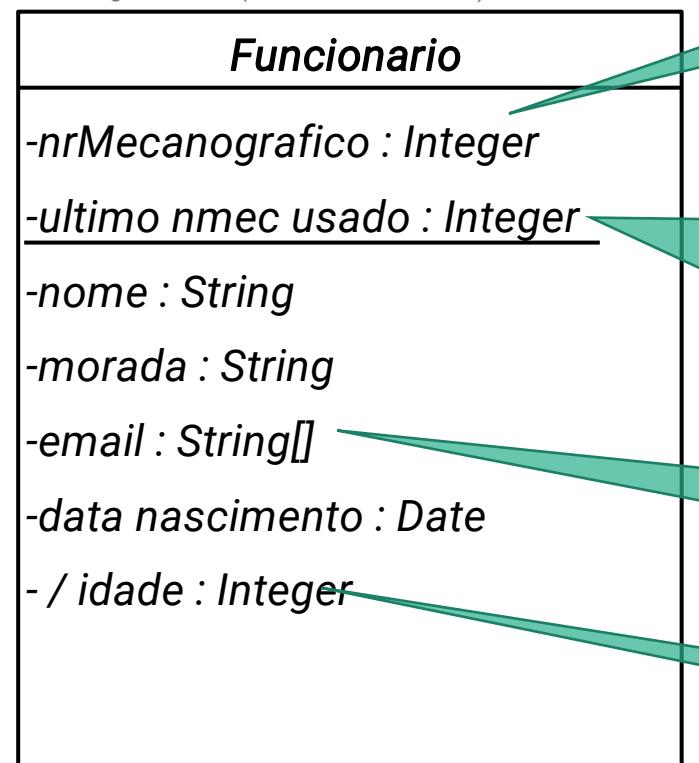
Conceito de "vetor"

Exemplo: uma instância anónima, da classe Aluno, com os atributos (*slots*) definidos. Email é um atributo multi-valor, nesta instância, tem associada uma lista de 3 Strings



Caraterização dos atributos

Visual Paradigm Standard(Universidade de Aveiro)



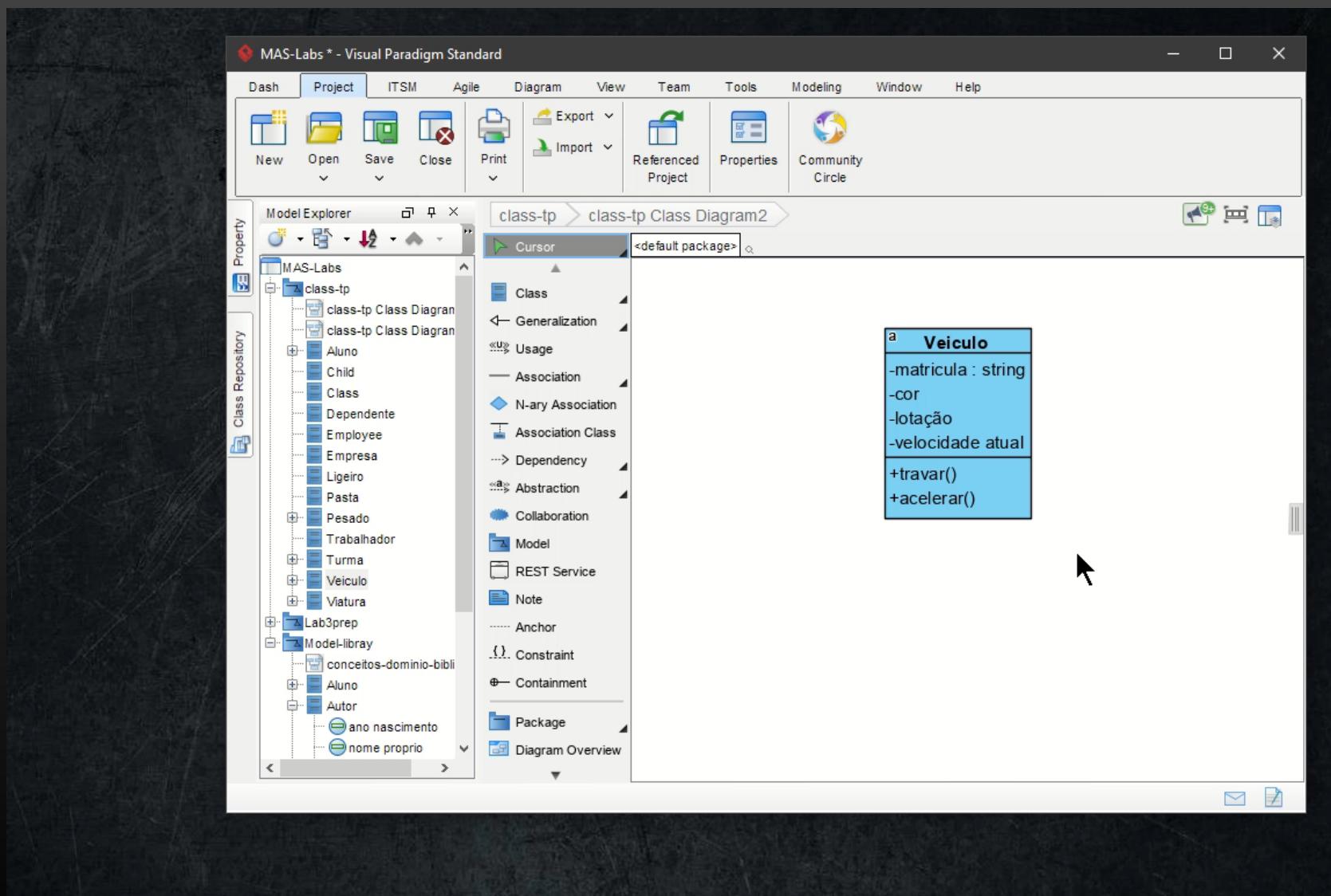
Tipo de dados (do atributo)

Atributo definido na classe, comum a todas as instâncias (static)

Multiplicidade (valores)

Atributo derivado (podia ser deduzido do restante estado)

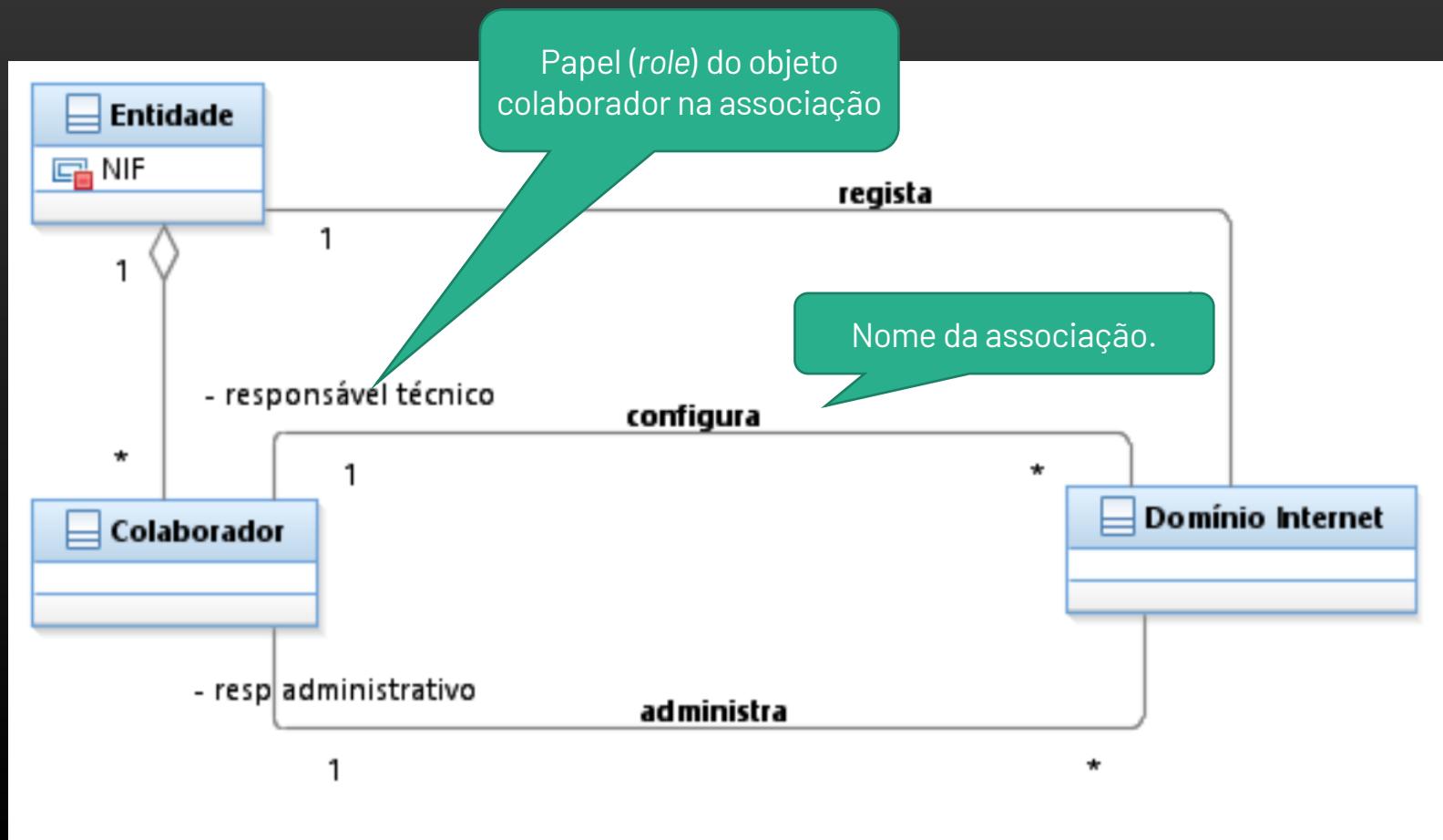
Caracterização dos atributos no Visual Paradigm



Papéis ≠ nome da associação

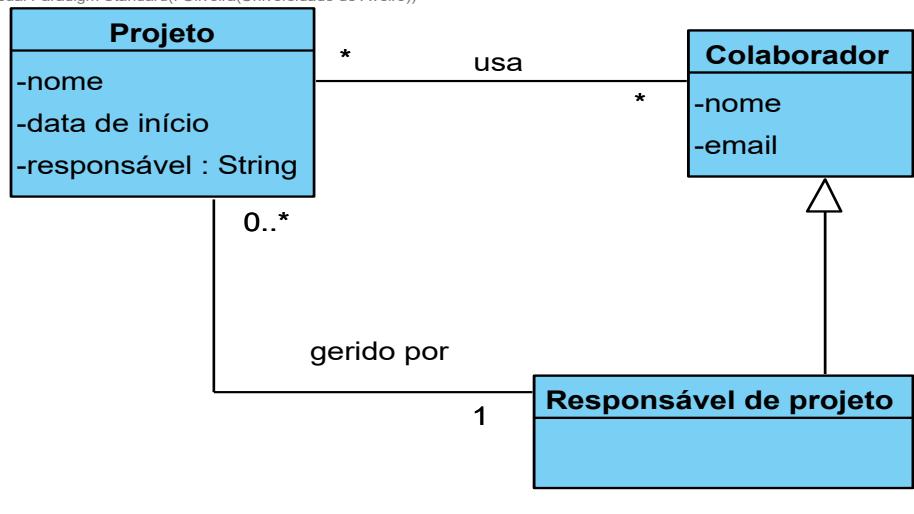
O **Nome** deve ajudar a ler a associação, como uma frase.

O **Papel** carateriza a forma como instâncias de um tipo participam na associação.



Opções de modelação: papéis

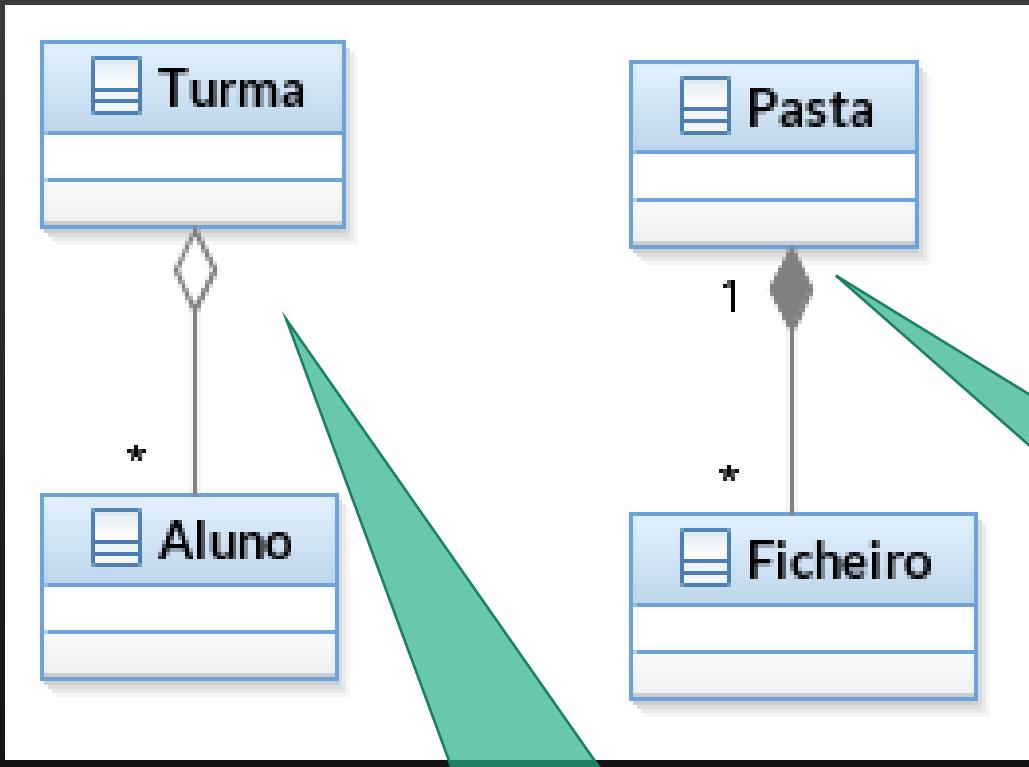
Visual Paradigm Standard(I Oliveira(Universidade de Aveiro))



Visual Paradigm Standard(I Oliveira(Universidade de Aveiro))



Agregação vs composição



Agregação
(a Turma agrupa objectos Aluno, mas o mesmo aluno pode ser parte de várias turmas → Aluno é partilhado)

Agregação: A comporta partes B de forma não exclusiva

As instâncias de B são partilháveis

Composição: A comporta partes B de forma exclusiva

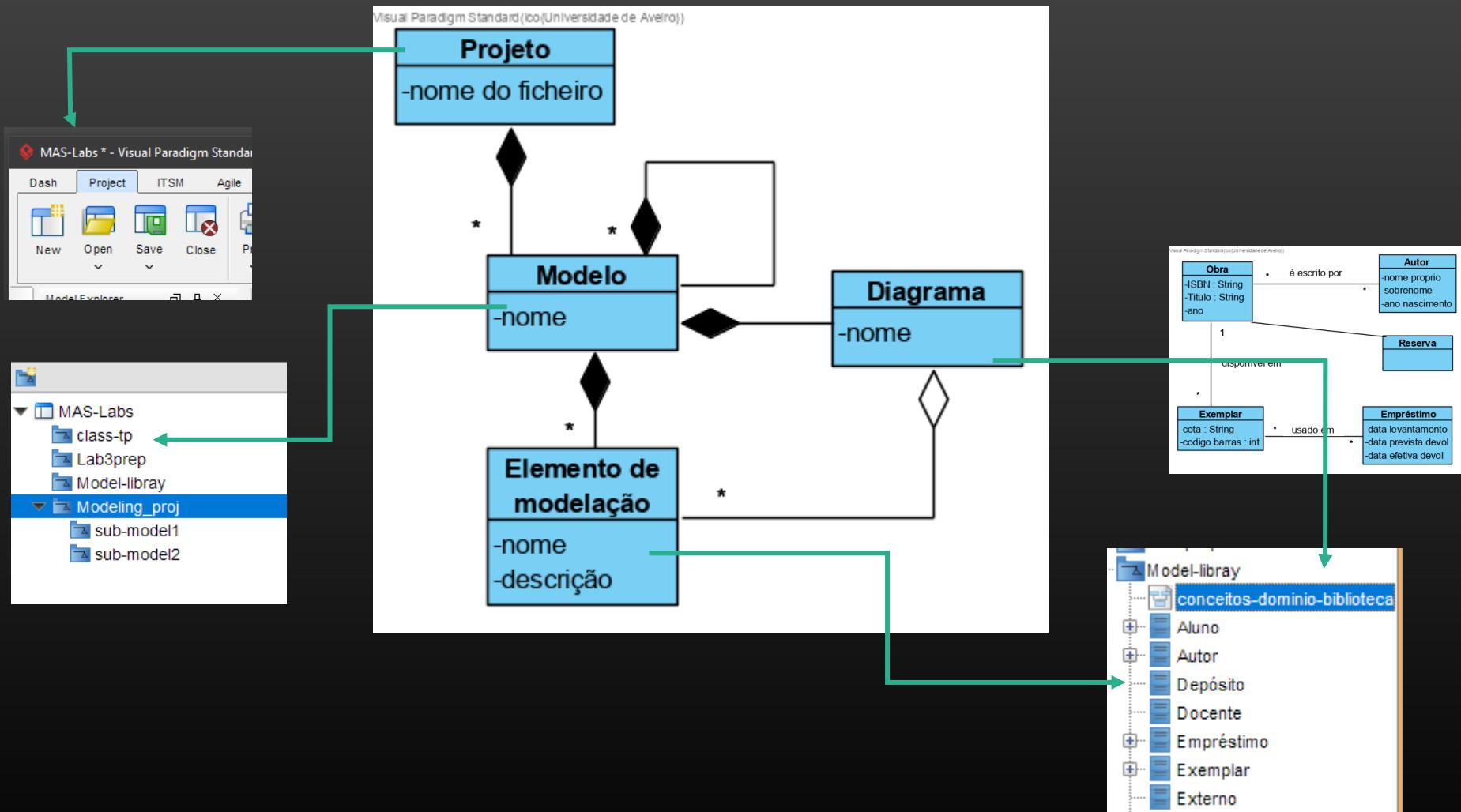
Os objetos B são partes constituintes de A

Instâncias de B não são partilháveis

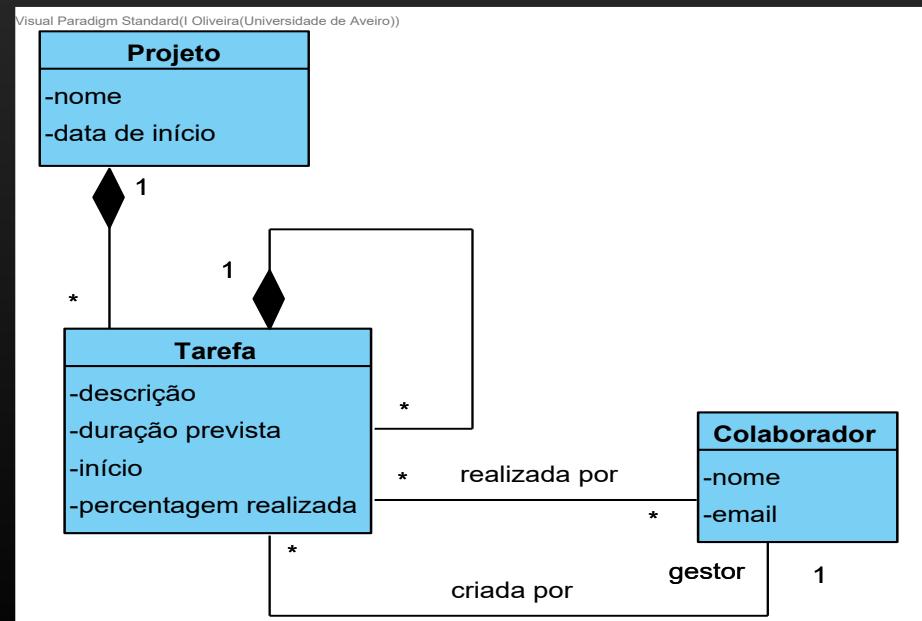
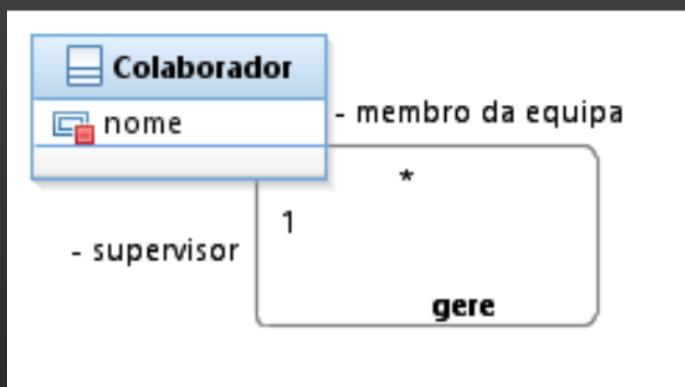
Composição

(a Pasta comporta vários Ficheiros; um ficheiro nunca faz parte de várias pastas → exclusivo de uma Pasta)

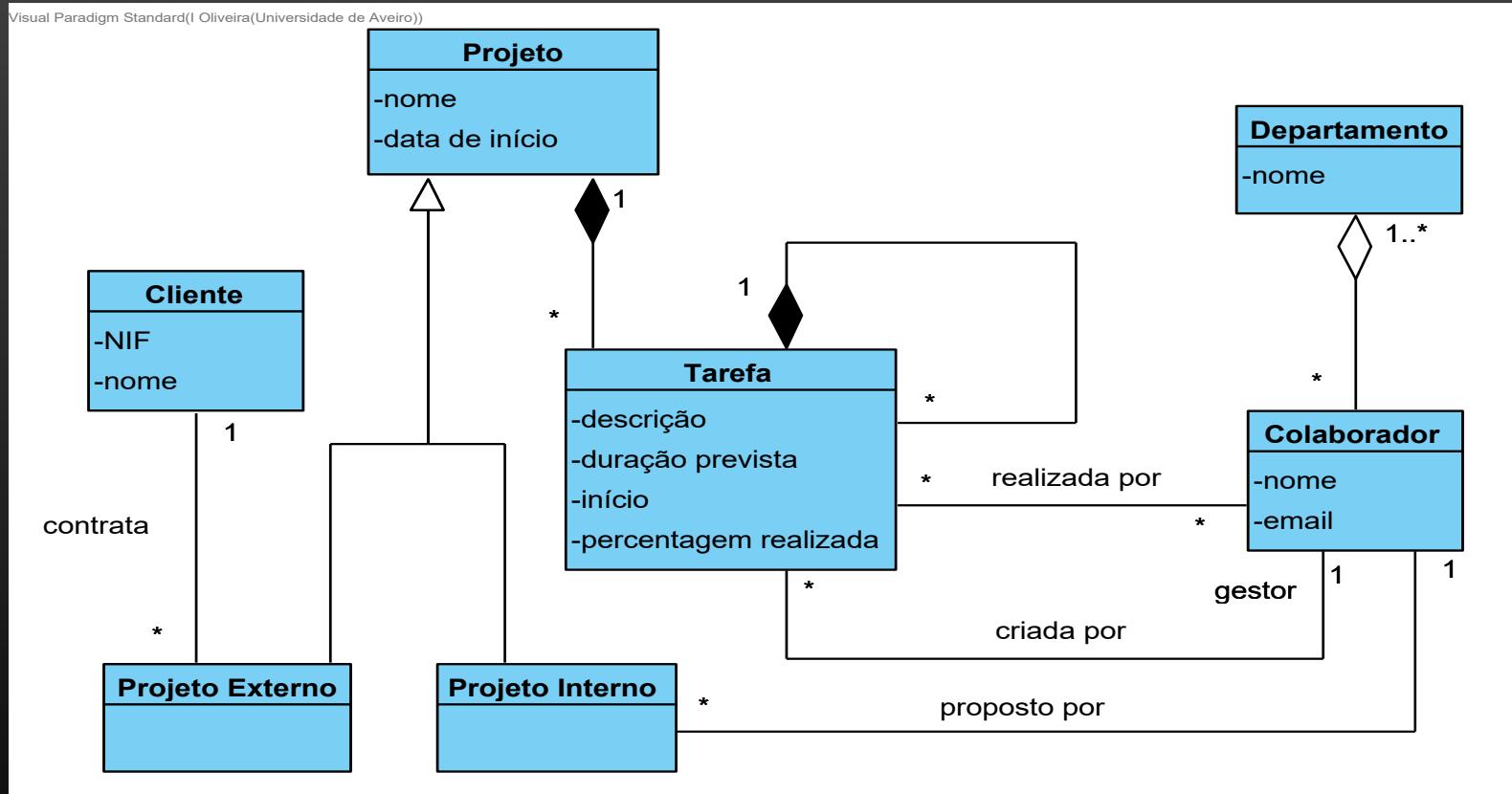
Exemplo: estrutura de um projeto de modelação no VP



Associações reflexivas relacionam instâncias da mesma classe

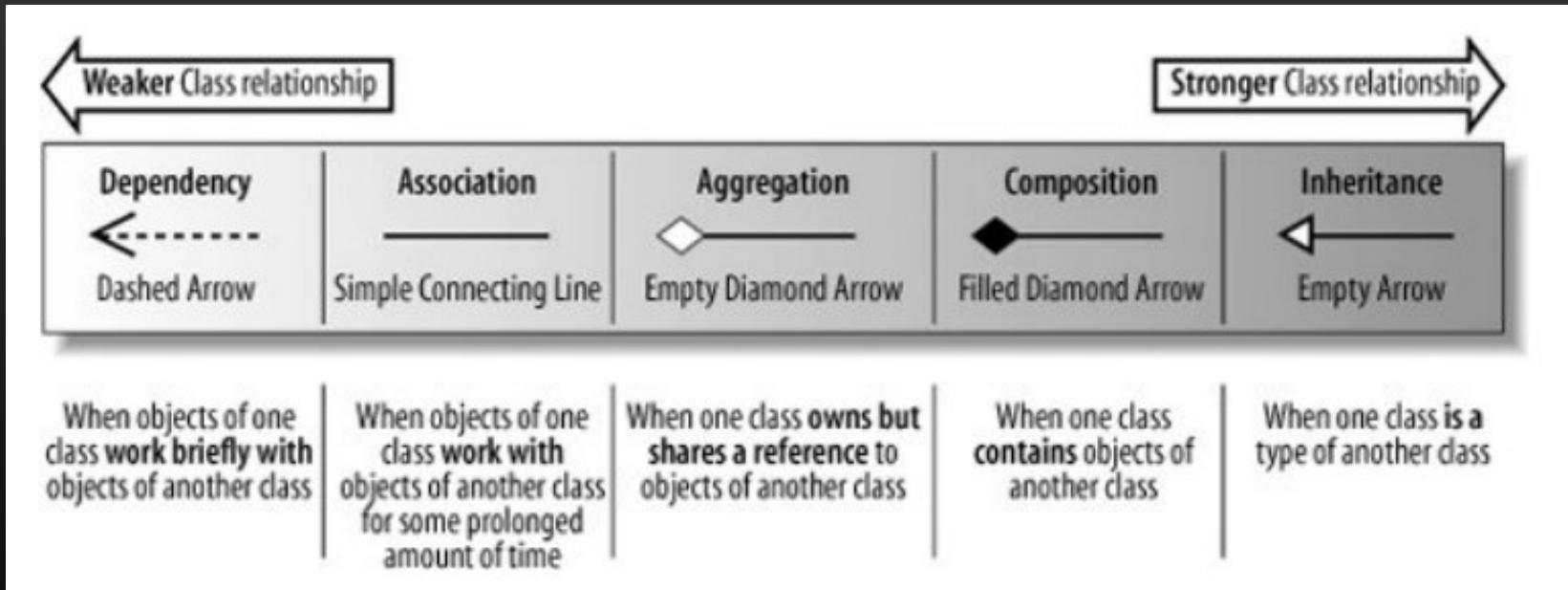


“Força” da associação entre os conceitos (*coupling*)

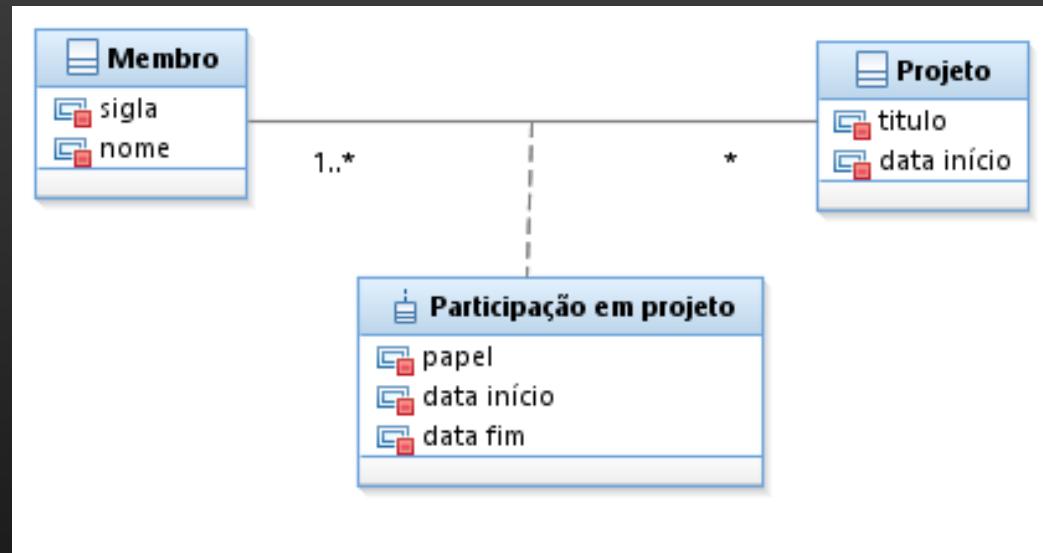


“Força” da ligação estrutural entre classes

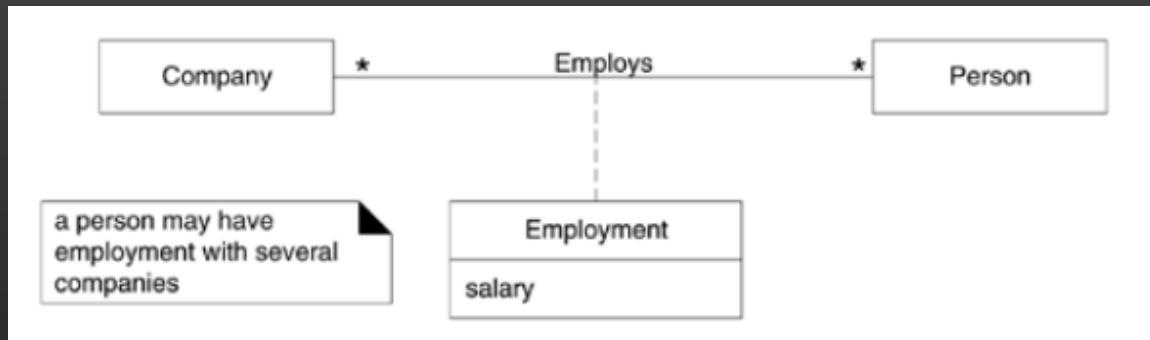
Cinco tipos diferentes de relacionamento entre classes, com diferentes níveis de ligação



Classes-associação captam a informação que descreve o relacionamento

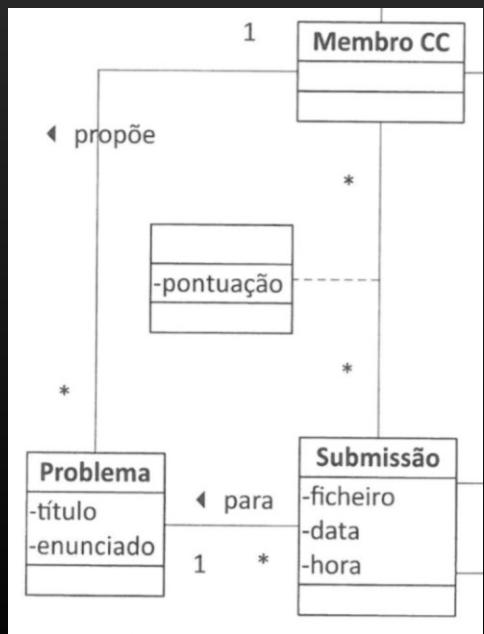


Indicações para o uso de uma classe-associação no modelo do domínio



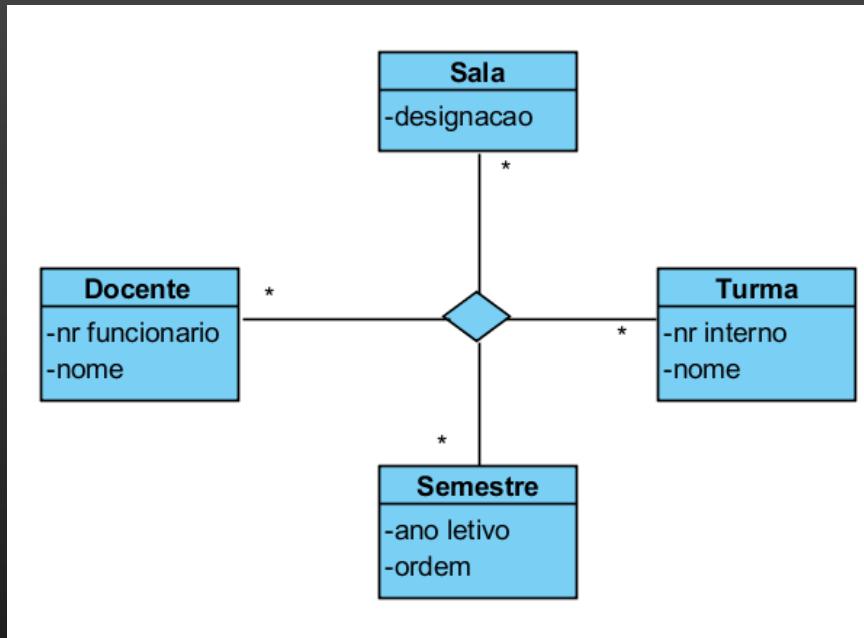
Um atributo está subordinado à ocorrência uma associação.

As instâncias da classe-associação têm um tempo de vida dependente da associação



Há uma relação de N:M entre dois conceitos e é necessária informação para caracterizar a própria associação

Associações N-árias

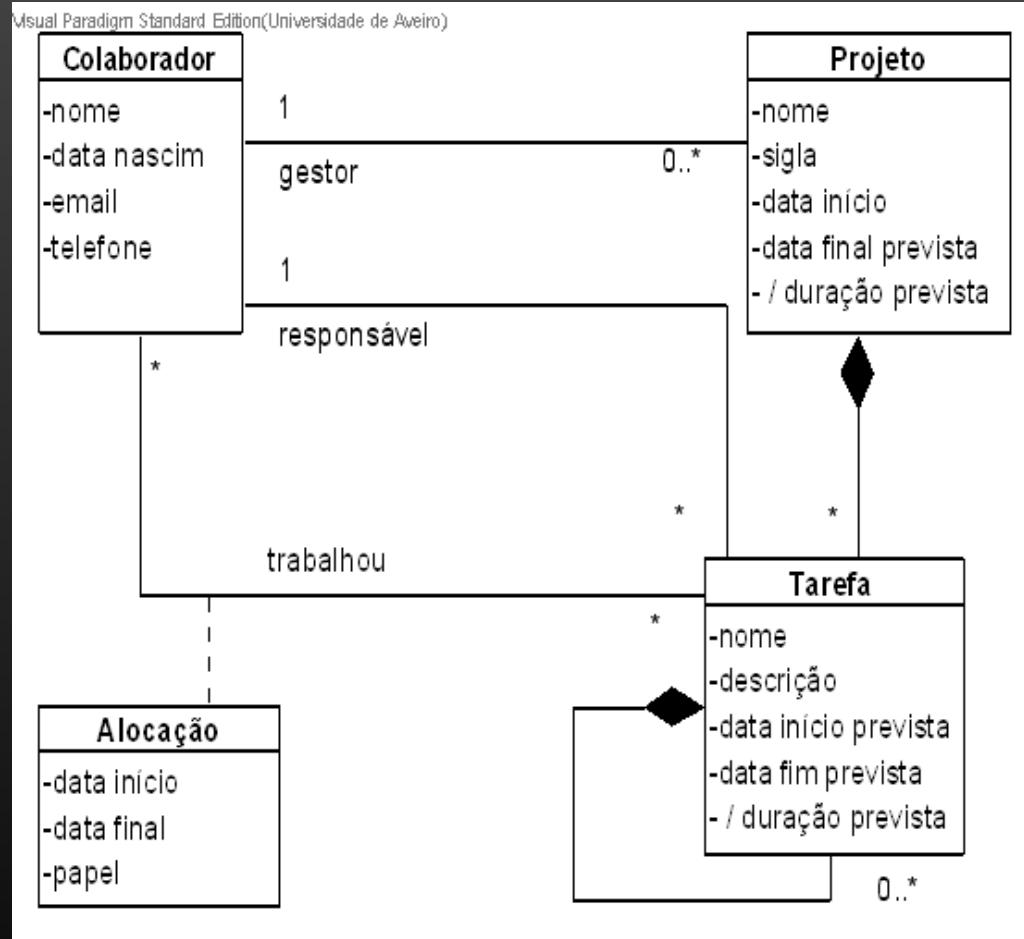


Vários objetos (de tipos diferentes) relacionados ao mesmo tempo, numa associação

Uma aula (presencial...) associa, ao mesmo tempo, um docente, uma sala, uma turma, e um semestre.

Gestão de projetos (exemplo)

Visual Paradigm Standard Edition(Universidade de Aveiro)



Role

Visibility

Multiplicity

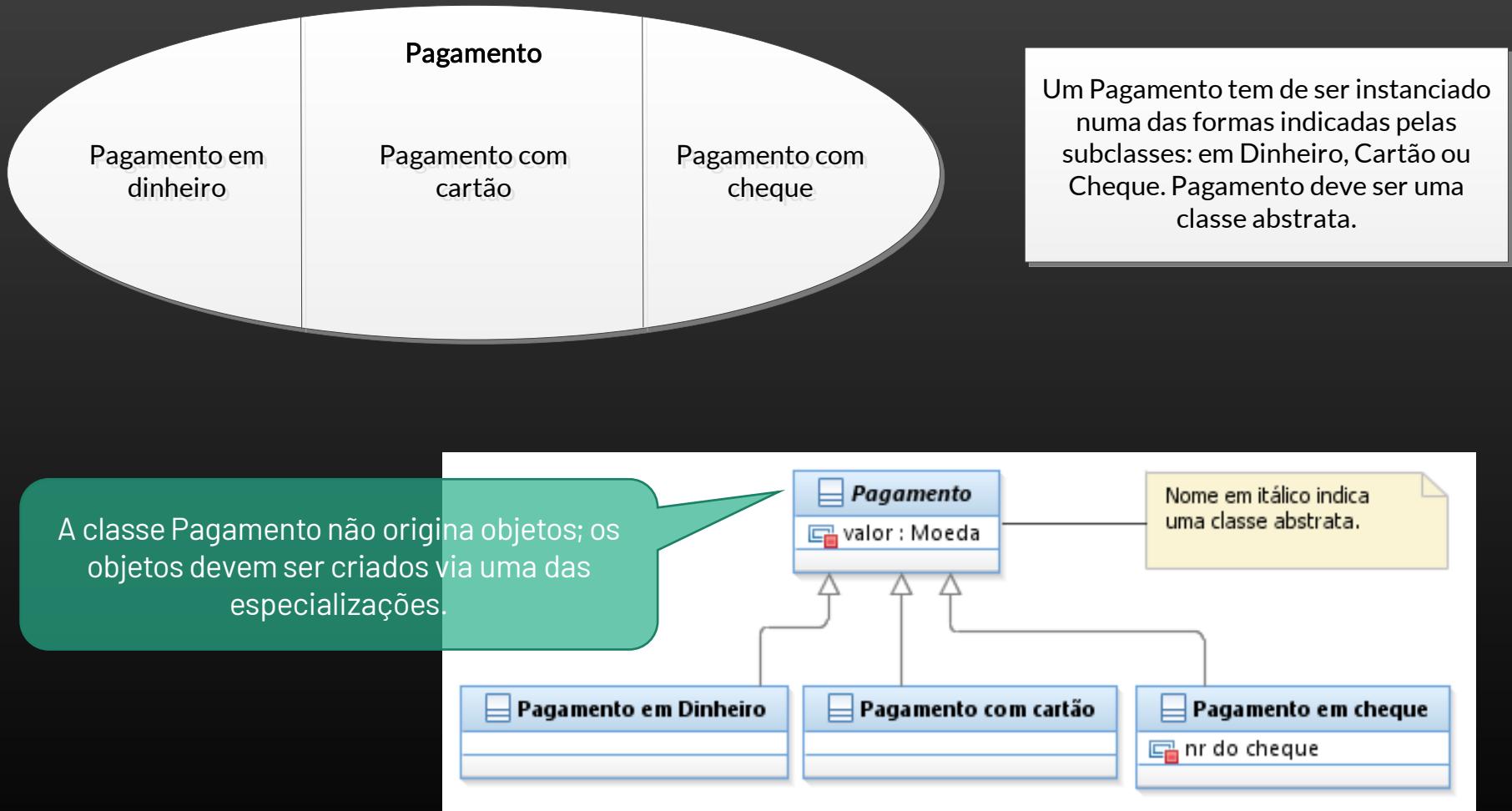
Derived attribute

Composition

Association class

Reflexive association

Uma classe abstrata não é instanciada diretamente



Diagramas de objetos



Um objeto (anónimo) do tipo Transferência usa dois objetos do tipo Conta, designados conta-destino e conta-ordenante.

Cada “caixa” representa uma instância (no exemplo, 3 instâncias de 2 classes diferentes)

41951- ANÁLISE DE SISTEMAS

Introdução à análise [orientada] por objetos

Ilídio Oliveira | v2022/03/15

Objetivos de aprendizagem

Explicar a técnica de modelação por classes, como uma forma de decompor um problema em entidades (objetos)

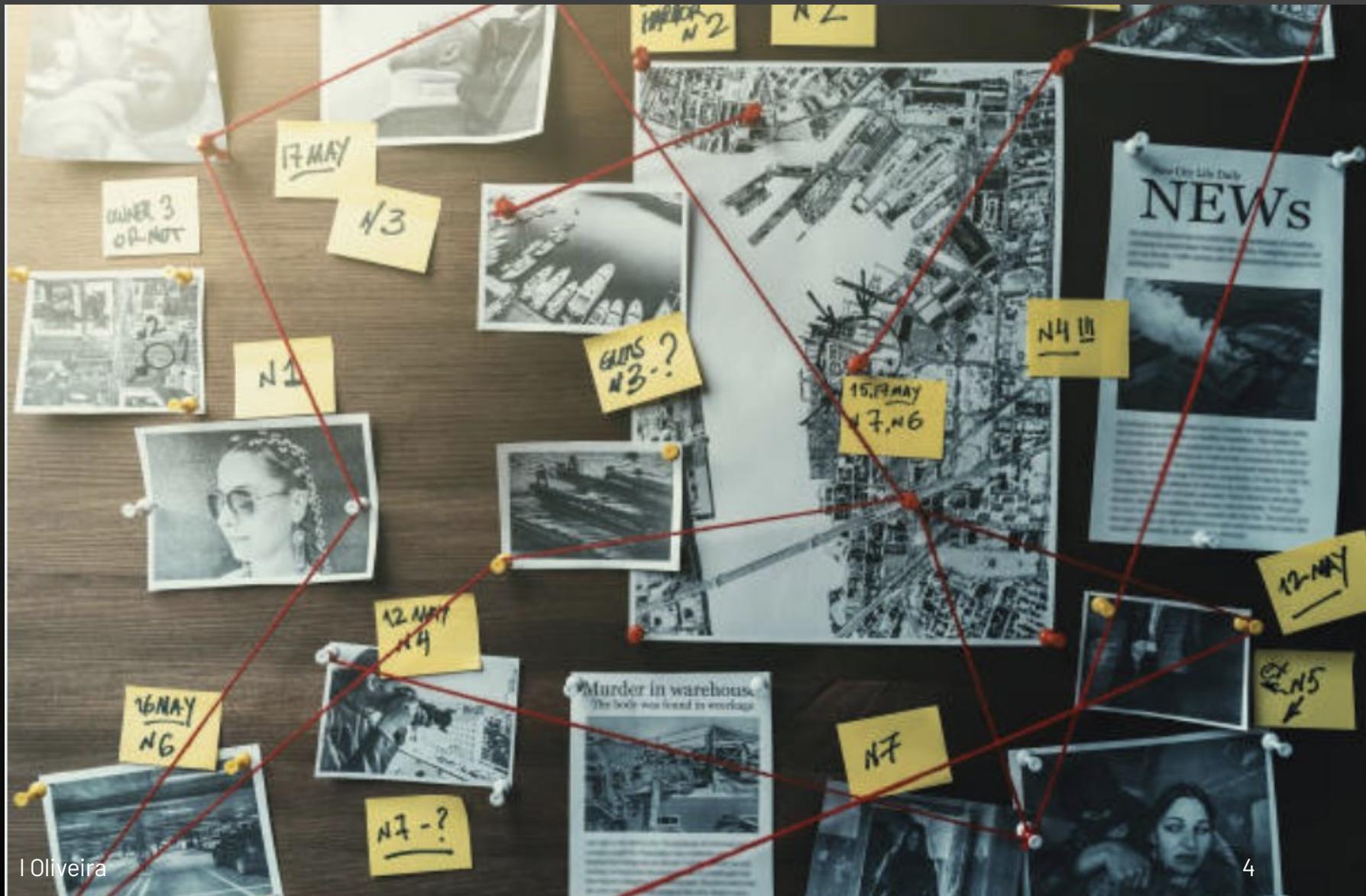
Interpretar modelos com diferentes relacionamentos: associação, agregação, generalização.

Explicar os conceitos de estado e de operações em relação a um objeto, e a forma como estão relacionados

Representar classes em diagramas.

Objetos: classificar as “coisas” do mundo

Crime scene investigation...



Investigação ~~do~~ crime domínio

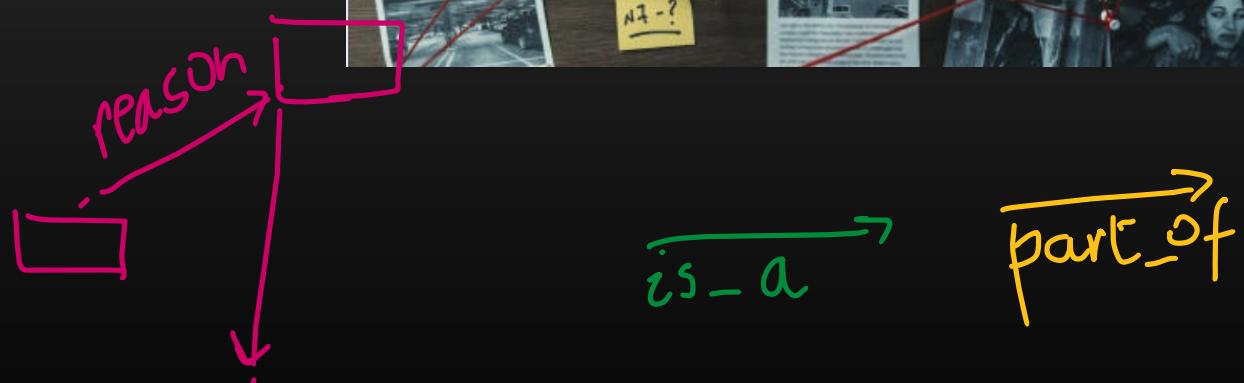
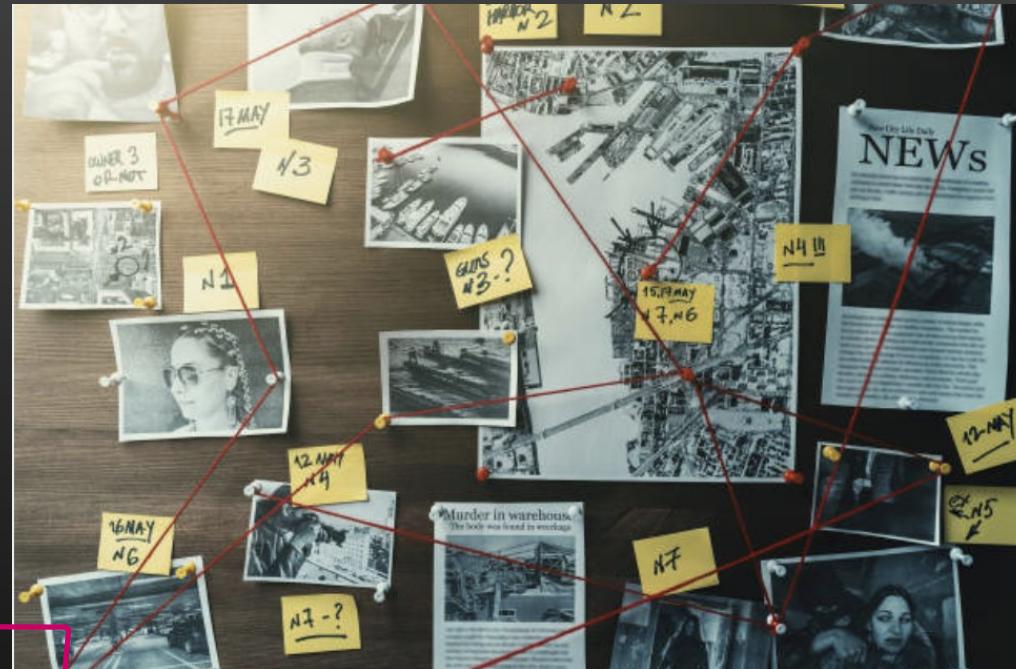
App de podcasts

→ Programas, Episódios, Autores, Categorias, Media, Subscrições, Utilizador(=ouvinte),...

Gestão académica?

→ Alunos, Turmas, UC, Cursos, Docentes, Diretor de curso, Departamentos,...

Netflix?...

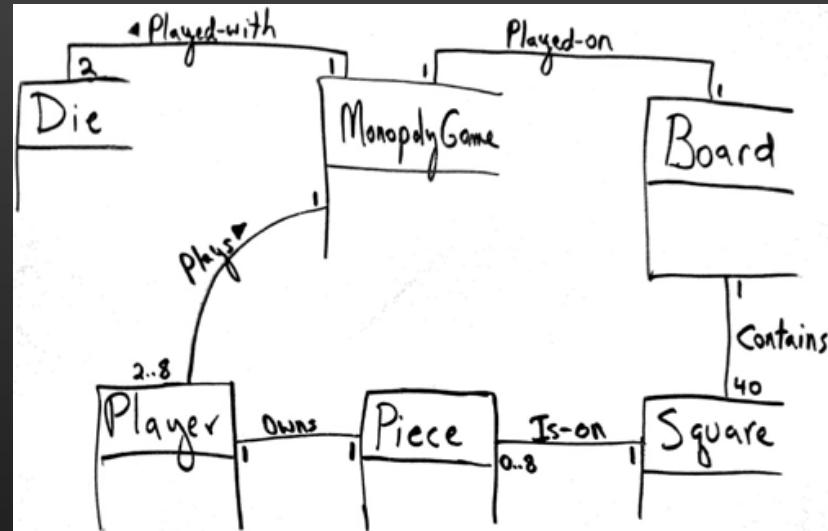


As “coisas” do domínio

Um **modelo de domínio** é uma representação visual das classes conceptuais ou objectos reais de um domínio/área.

Centra-se na explicação de 'coisas' e produtos importantes para um domínio de negócio.

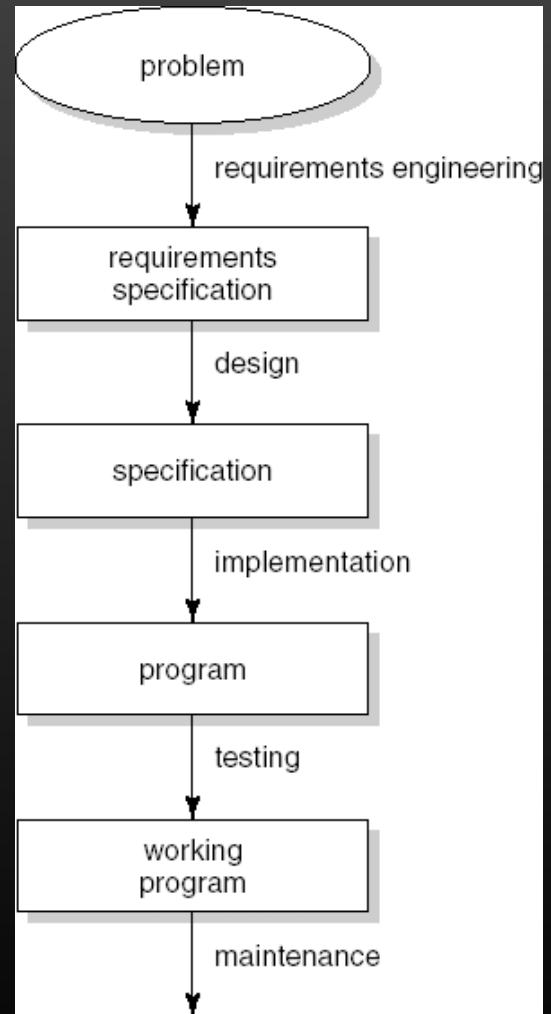
→ Não é (ainda) uma visão do software ou base de dados...



O passo fundamental da análise orientada por objectos é a decomposição de um domínio em conceitos relevantes → **objetos**.

Fluxo do desenvolvimento

onde entram os “objetos”?



During *object-oriented analysis* there is an emphasis on finding and describing the objects—or concepts—in the problem domain. For example, in the case of the flight information system, some of the concepts include *Plane*, *Flight*, and *Pilot*.

During *object-oriented design* (or simply, object design) there is an emphasis on defining software objects and how they collaborate to fulfill the requirements. For example, a *Plane* software object may have a *tailNumber* attribute and a *getFlightHistory* method (see [Figure 1.2](#)).

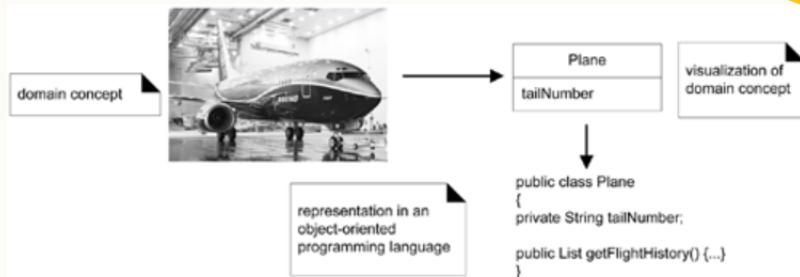
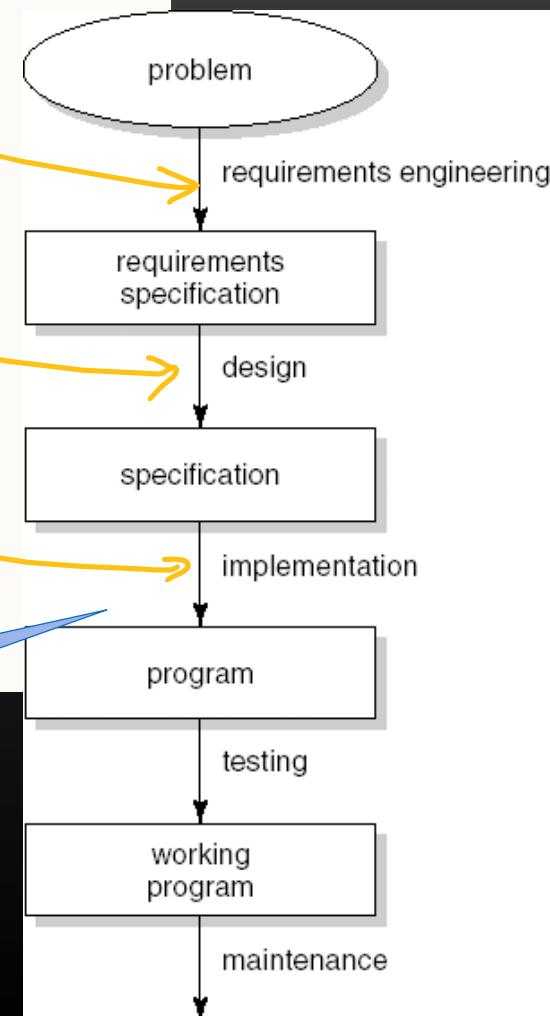


Figure 1.2. Object-orientation emphasizes representation of objects.

Finally, during implementation or *object-oriented programming*, design objects are implemented, such as a *Plane* class in Java.

Motivação: será possível manter o **mesmo esquema mental** para representar as “coisas” do problema, ao longo do SDLC?
(baixar o gap representacional com modelação OO)



3 mecanismos principais (para modelar/gerir a complexidade)

Abstração



- Tratar os objetos que
são semelhantes como
um “tipo”/categoria

Encapsulamento



- O objeto é uma unidade
que esconde o seu
estado interno

- A interação com o
objeto é feita através de
“pontos de acesso”

Hierarquia



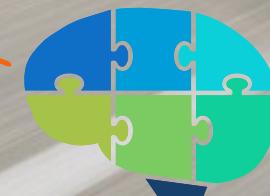
- Procurar relações
“é-um” (is-a)

- Procurar relações
“parte-de” (part-of)

Muitas instâncias.. do mesmo conceito (objeto)



Abstracção



Veiculo
-matricula
-cor
-lotação
-velocidade atual
+travar()
+acelerar()

Mecanismo: abstração

Abstração decorre do reconhecimento de semelhanças entre certos objetos, situações ou processos no mundo real, e a decisão de se concentrar nestas semelhanças → um *tipo de coisa*

Uma abstração denota as características essenciais de um objeto que o distingue de todos os outros tipos de objetos.

Abstraction: Temperature Sensor

Important Characteristics:

temperature
location

Responsibilities:

report current temperature
calibrate

Figure 2–6 Abstraction of a Temperature Sensor

Abstraction: Heater

Important Characteristics:

location
status

Responsibilities:

turn on
turn off
provide status

Related Candidate Abstractions: Heater Controller, Temperature Sensor

Figure 2–9 Abstraction of a Heater

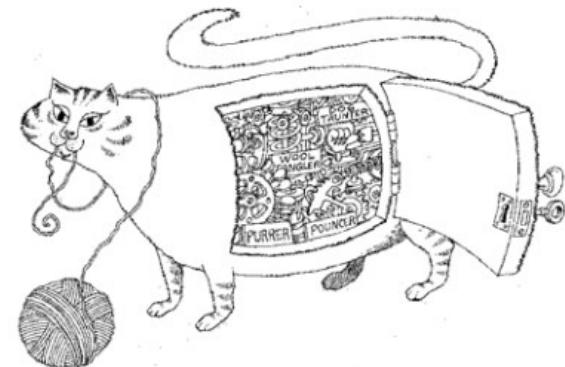
Encapsulamento

Encapsulamento incentiva a ocultação da estrutura (interna) de um objeto (a informação que mantém), bem como a implementação dos seus métodos (como faz).

Nenhuma parte de um sistema complexo deve depender do conhecimento dos detalhes internos de qualquer outra parte.

O encapsulamento permite que as alterações de um programa sejam feitas de forma fiável, com repercussões limitadas

→ mexer num módulo não deve estragar os outros



Encapsulation hides the details of the implementation of an object.

Como é que o encapsulamento contribui para a gerir a complexidade?

Encapsulamento ajuda a gerir a complexidade escondendo a dimensão “privada” (interna) das nossas abstrações.

Abstraction: Temperature Sensor

Important Characteristics:

temperature
location

Responsibilities:

report current temperature
calibrate

Figure 2–6 Abstraction of a Temperature Sensor

Abstraction: Heater

Important Characteristics:

location
status

Responsibilities:

turn on
turn off
provide status

Related Candidate Abstractions: Heater Controller, Temperature Sensor

O que é que Heater deve conhecer de Temperature Sensor?

Figure 2–9 Abstraction of a Heater

Hierarquia: a relação de herança

A herança define uma relação entre classes (tipos de coisas), em que uma classe partilha/especializa a estrutura ou o comportamento definido numa outra classe.

A herança denota uma relação semântica "é-um", e.g.:

um urso "é um" (tipo de) mamífero; uma casa "é um" tipo de ativo imobiliário; uma caravana "é um" veículo.

A herança implica, assim, uma generalização \leftrightarrow especialização

Visual Paradigm Standard(I Oliveira(Universidade



Abstraction: Temperature Sensor

Important Characteristics:

temperature
location

Responsibilities:

report current temperature
calibrate

Figure 2–6 Abstraction of a Temperature Sensor

Qual a relação semântica? Um Sensor de temperatura é um (**is-a**) Sensor.

Hierarquia: a relação de agregação

A agregação define uma relação entre categorias de coisas (i.e. classes) do tipo “parte-de”

A agregação denota a relação semântica “parte-de”, e.g.:

Um País é parte de um Continente; o Estudante é parte da Turma; uma Obra é parte de uma Coleção.

A agregação implica, assim, um agregador ←→ parte-de

Abstraction: Heater
Important Characteristics: location status
Responsibilities: turn on turn off provide status

Related Candidate Abstractions: Heater Controller, Temperature Sensor

Figure 2–9 Abstraction of a Heater

Abstraction: Temperature Sensor
Important Characteristics: temperature location
Responsibilities: report current temperature calibrate

Figure 2–6 Abstraction of a Temperature Sensor

O Temperature Sensor é parte de um Heater.
Um Estudante é parte de uma Turma.

3 mecanismos principais para modelar por objetos

Abstração



- Tratar os objetos que
são semelhantes como
um “tipo”/categoria

Encapsulamento



- O objeto é uma unidade
que esconde o seu
estado interno

- A interação com o
objeto é feita através de
“pontos de acesso”

Hierarquia



- Procurar relações “é-
um”

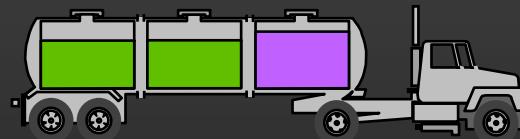
- Procurar relações
“parte-de”

Natureza e representação dos objetos

Os objetos (em OO) modelam entidades do mundo real/espaço do problema

**Observáveis no mundo físico
(coisas tangíveis)**

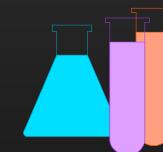
e.g.: Aluno, Avião, Veículo



Truck

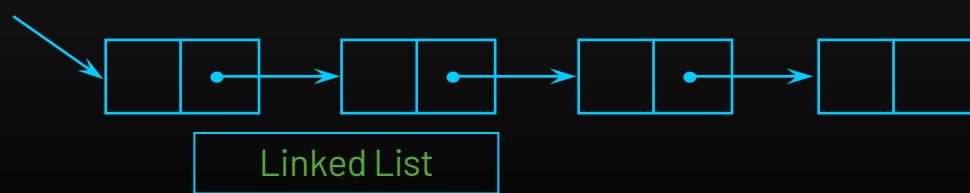
Conceitos (geram informação)

e.g.: Venda, Reserva,
Inscrição



Abstrações próprias do software

e.g.: ListaLigada, Vetor,
HashTable



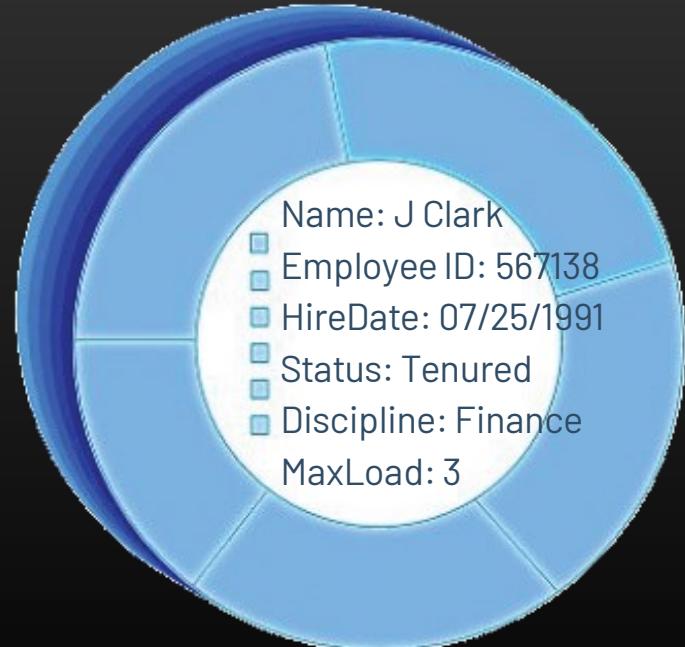
Um objeto tem um estado (interno)

O estado de um objeto corresponde a uma das condições/configurações é que é possível o objeto apresentar-se.

É normal o estado objeto mudar ao longo do tempo.



Name: J Clark
Employee ID: 567138
Date Hired: July 25, 1991
Status: Tenured
Discipline: Finance
Maximum Course Load: 3 classes



Professor Clark

Um objeto tem comportamento (funcionalidades)

O comportamento define como é que o objeto age/reage

O comportamento visível/exposto é modelado pelo conjunto de mensagens a que responde (operações)



Professor Clark's behavior
Submit Final Grades
Accept Course Offering
Take Sabbatical
Maximum Course Load: 3 classes

Objeto: estado + operações

Sabe alguma coisa

Os seus atributos

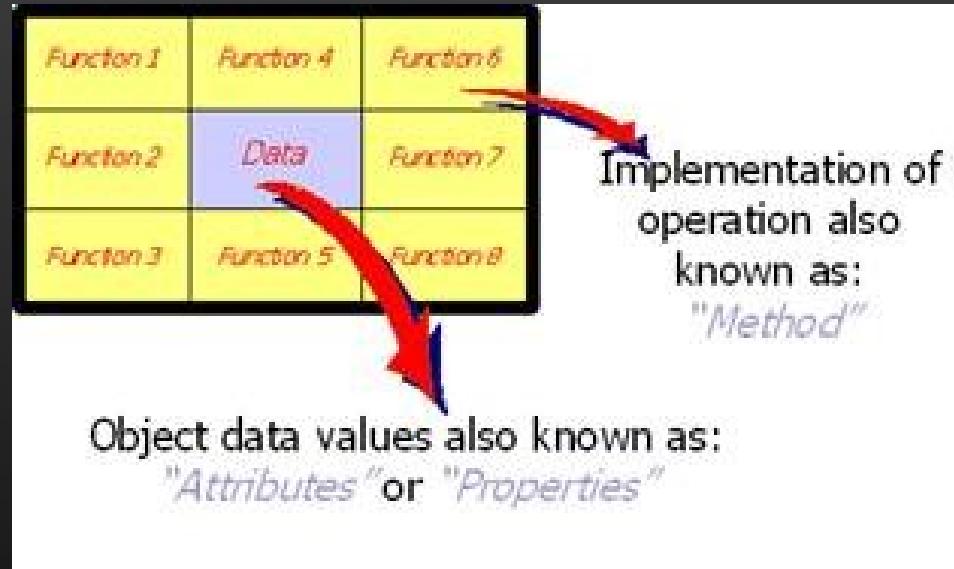
Os seus relacionamentos

Sabe fazer alguma coisa

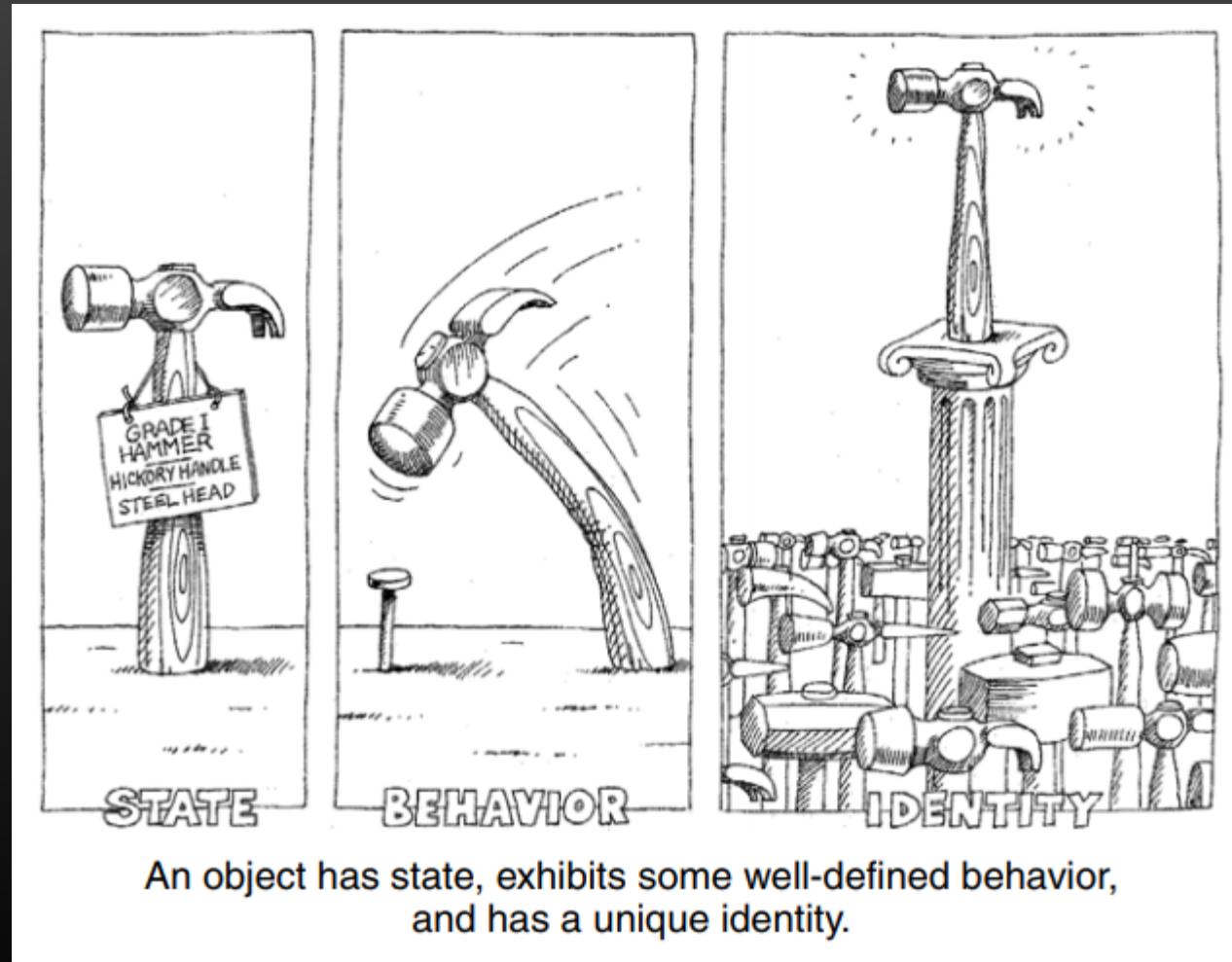
O seu comportamento, ativado através de operações

Cápsula

Os outros (objetos) não precisam de saber o que ele sabe...



Objeto: estado + operações + identidade



O que é a Classe?

Uma classe é uma categoria de objetos semelhantes que partilham os mesmos atributos, operações, relacionamentos e semântica.

O objeto é uma instância (ocorrência) de uma classe

Uma classe é uma conceitualização

Categoriza objetos semelhantes

Enfatiza as características de interesse (e suprime as outras)



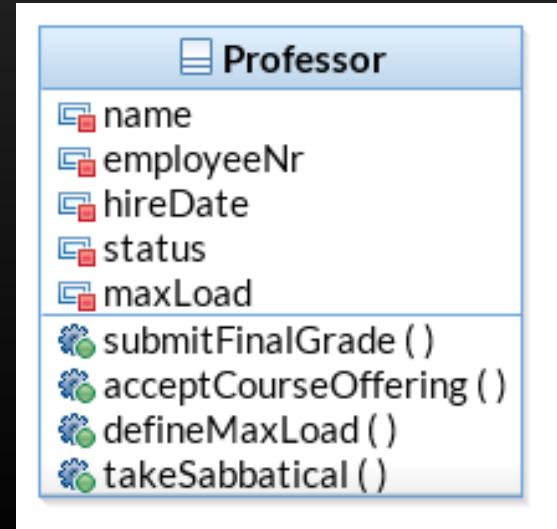
Professor Torpie



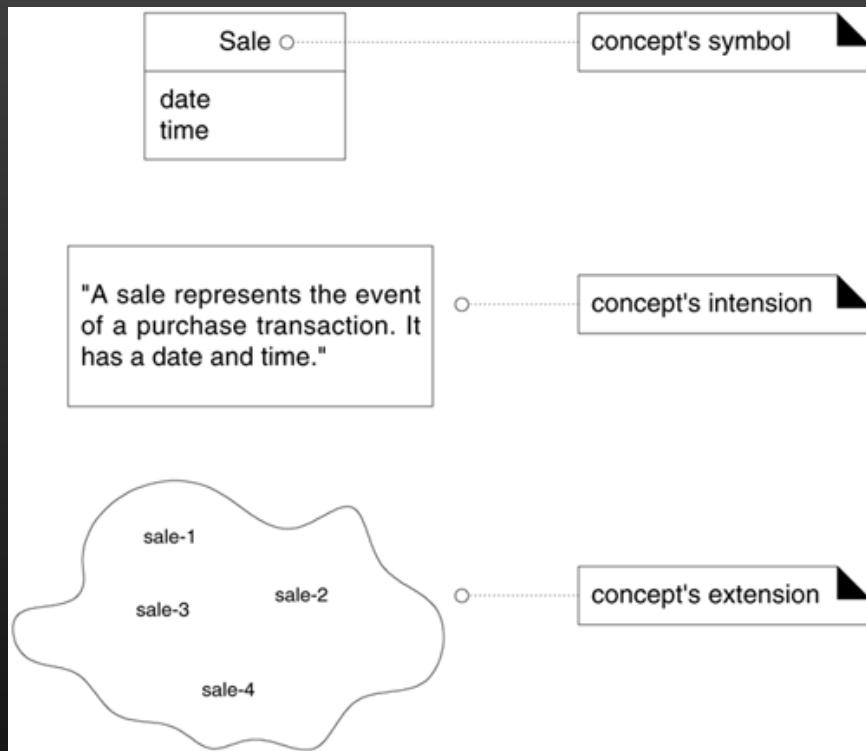
Professor Meijer



Professor Allen



Classe



- um elemento de modelação (símbolo)
- com uma interpretação
- e um conjunto de instâncias

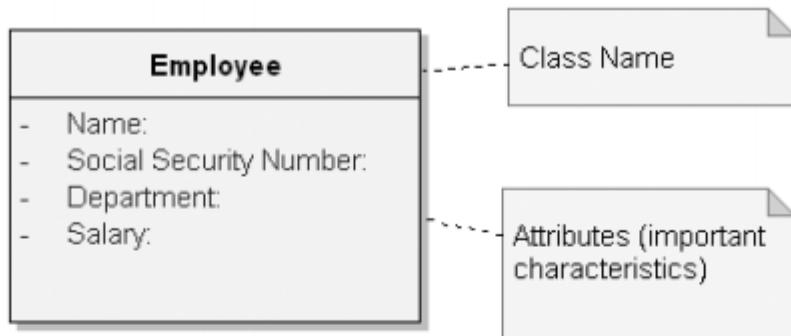
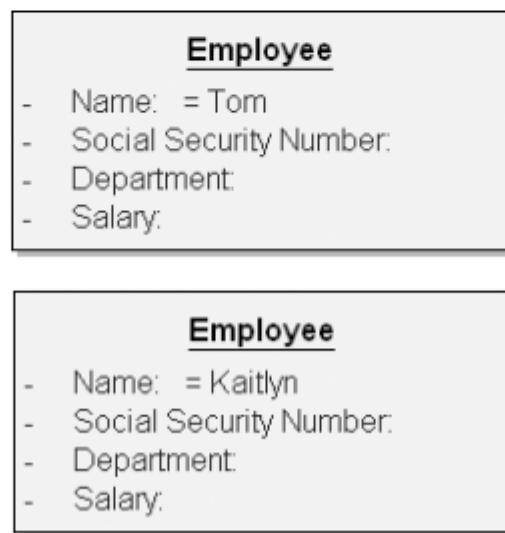


Figure 3–1 Employee Class with Attributes

specific instance. When made specific, we may have, for example, two distinct objects: Tom and Kaitlyn, each of which takes up some amount of space in memory (see Figure 3–2).



A classe define as características de um tipo/objeto. Funciona como o “molde” de um conceito.

Um objeto (ou instância) é uma ocorrência concreta de um tipo de coisas.
(realizado a partir do “molde”)

Figure 3–2 Employee Objects Tom and Kaitlyn

Modelar as relações entre classes (na UML)

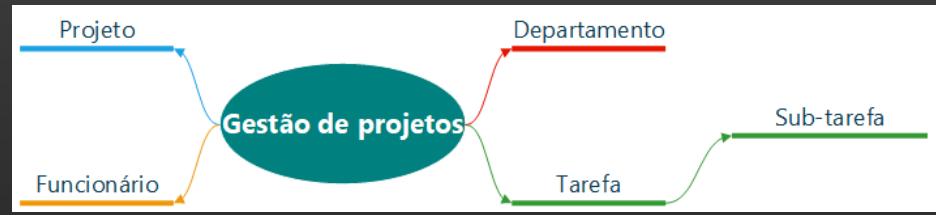
Os objetos de um domínio estão ligados numa “rede de conhecimento”

Um Projeto é coordenado por um determinado Departamento.

O Projeto tem uma equipa de vários Funcionário(s) atribuída.

Cada Projeto define várias Tarefas que, por sua vez, podem estar organizadas em sub-tarefas.

Um projeto tem um gestor.
etc.



Como transportar o conhecimento do domínio pra um modelo?
Há “coisas” de interesse e “relacionamentos” entre elas

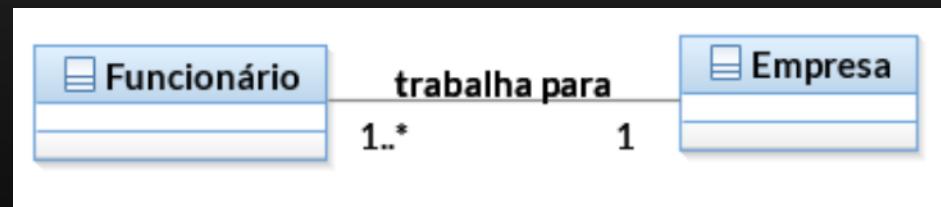
O que é a “associação”?

A relação semântica que se estabelece entre duas ou mais classes que descreve as ligações existentes entre as respetivas instâncias.

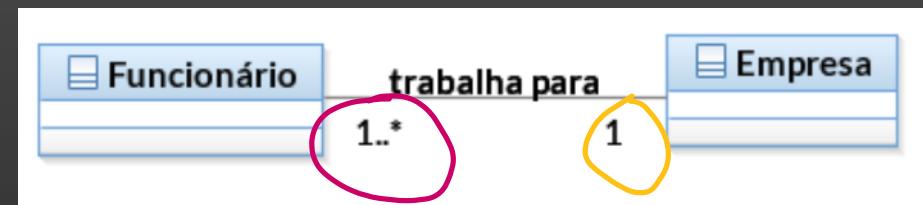
Mostra que objetos de um tipo estão ligados a objetos de outro tipo.

O tipo de “ligação” deve ser anotado com uma explicação do significado.

A classe Funcionário e Empresa estão associadas. A descrição da associação é “trabalha para”.
Portanto:
“Funcionário” → “trabalha em” → “Empresa”.



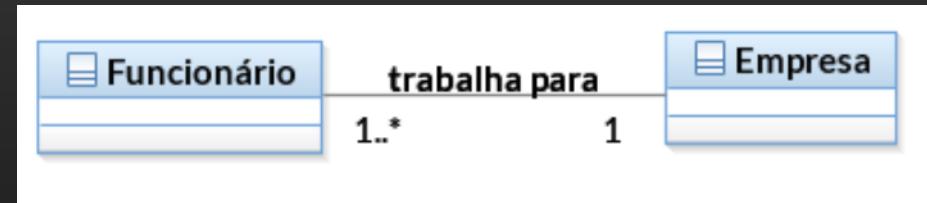
Com que “frequência” os objetos de F e E estão associados?



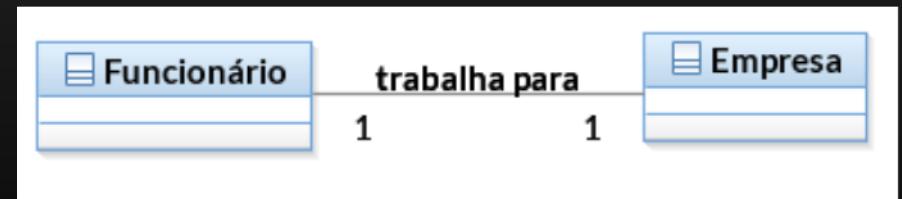
O que é a multiplicidade (de uma associação)?

Nr de instâncias de uma classe que se relacionam com uma instância da outra.

"O Funcionário só trabalha numa empresa; a Empresa tem pelo menos um Funcionário"



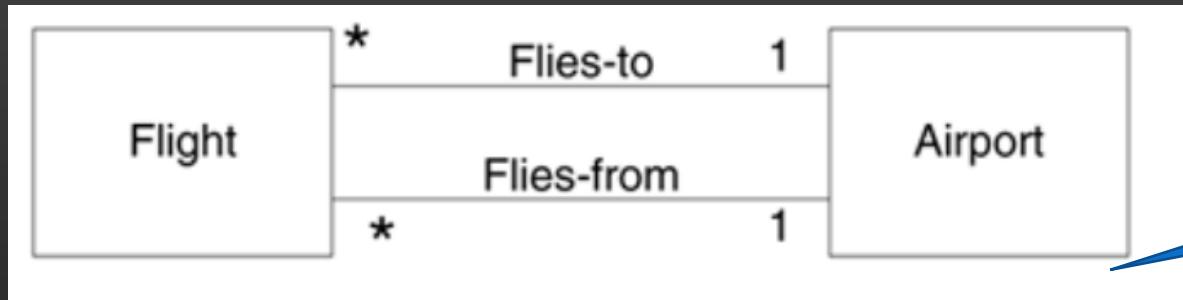
"O Funcionário só trabalha numa empresa; a Empresa só tem um Funcionário"



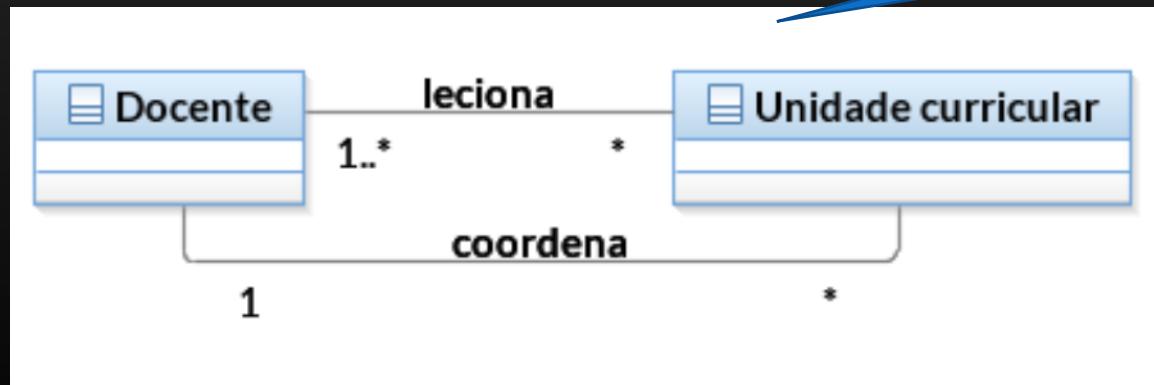
Indicação de multiplicidade

Exactly one	1	<pre> classDiagram class Department class Boss Department "1" --> "1" Boss </pre>	A department has one and only one boss.
Zero or more	0..*	<pre> classDiagram class Employee class Child Employee "0..*" --> "*" Child </pre>	An employee has zero to many children.
One or more	1..*	<pre> classDiagram class Boss class Employee Boss "1..*" --> "1..*" Employee </pre>	A boss is responsible for one or more employees.
Zero or one	0..1	<pre> classDiagram class Employee class Spouse Employee "0..1" --> "0..1" Spouse </pre>	An employee can be married to zero or one spouse.
Specified range	2..4	<pre> classDiagram class Employee class Vacation Employee "2..4" --> "2..4" Vacation </pre>	An employee can take from two to four vacations each year.
Multiple, disjoint ranges	1..3,5	<pre> classDiagram class Employee class Committee Employee "1..3,5" --> "1..3,5" Committee </pre>	An employee is a member of one to three or five committees.

Múltiplas associações entre duas classes



Um voo: parte de um Aeroporto; chega a um Aeroporto.
O “estado” de um Voo associa 2 objetos Aeroporto, por razões diferentes

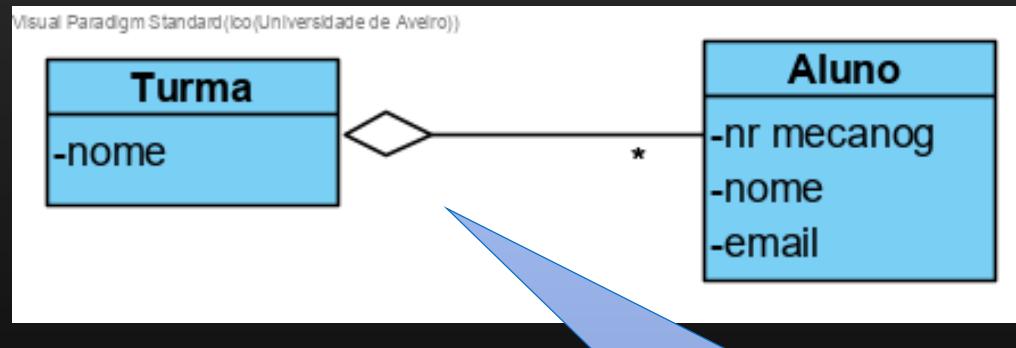


Um docente: leciona várias UC; coordena várias UC.

O que é uma agregação?

É uma forma especial de associação que modela uma relação de todo-parte, entre o agregador (“contentor”) e as suas partes constituintes

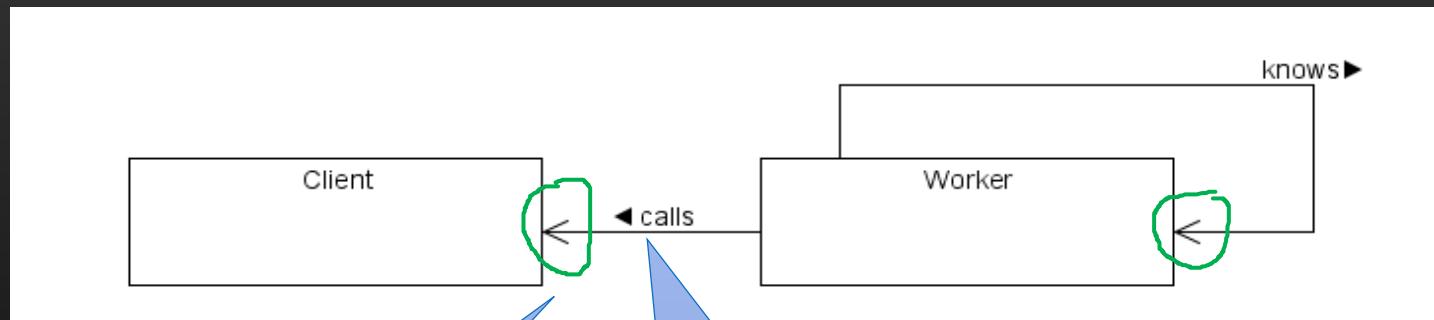
Deve ser natural ler-se “É parte de...” (sentido parte → todo); se for “forçado”, usar a associação normal



Um objeto *Turma* agrupa vários objetos *Aluno*.

O que é a navegabilidade?

Indica a possibilidade de navegar de uma classe de partida para uma classe de chegada, usando a associação



Em que sentido é que a associação é navegável/aplicável?
(Se omissos → ambos)

Notação auxiliar: em que sentido se deve ler a etiqueta. (Worker > calls > Client.
(Se omissos: cima p/ baixo; esquerda p/ direita)

Os dois primeiros casos são as situações mais comuns.



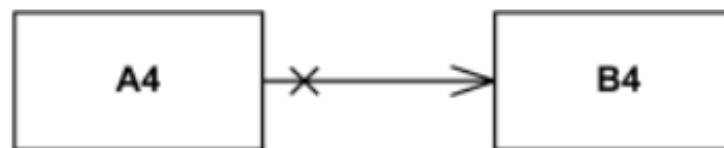
Both ends of association have unspecified navigability.



A2 has unspecified navigability while B2 is navigable from A2.



A3 is not navigable from B3 while B3 has unspecified navigability.



A4 is not navigable from B4 while B4 is navigable from A4.

O que é a generalização (=herança)

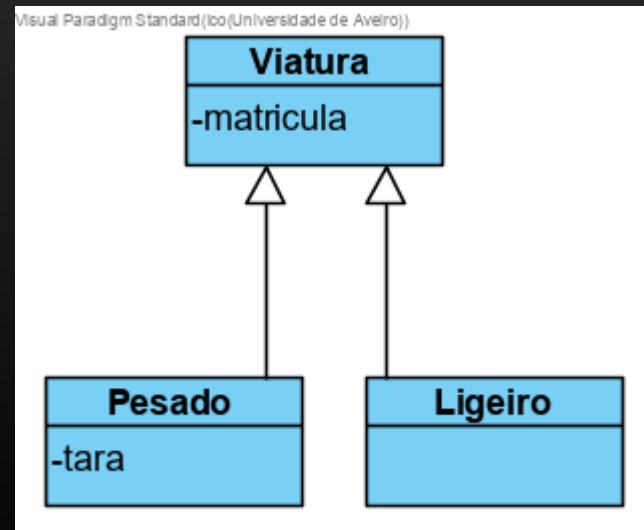
É relação entre classes em que uma especializa a estrutura e/ou comportamento de outra, partilhando todas as características

Define uma hierarquia em que a subclass recebe as características da superclasse

A subclass pode sempre ser usada onde a superclasse é usada, mas não ao contrário.

Pode ler-se “é um tipo de”

(Um *Pesado* é um tipo de *Viatura*, com matrícula e tara.)



O que é passado à subclasse?

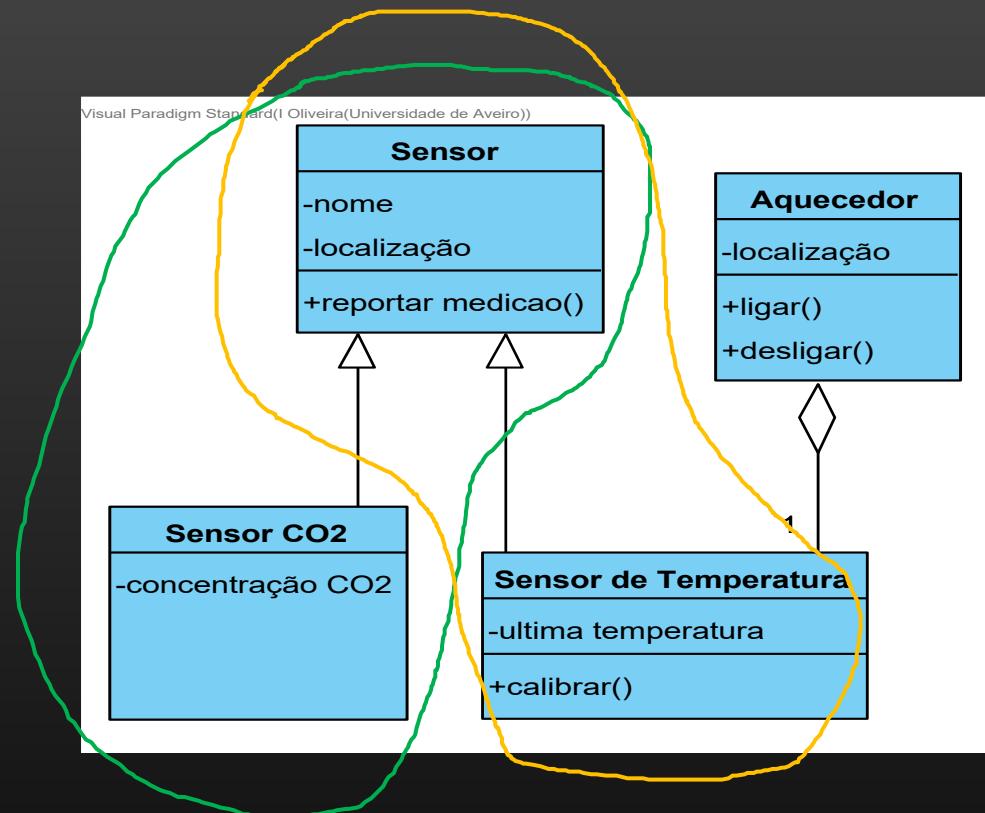
A subclasse herda os atributos, operações e relacionamentos da superclasse

A subclasse pode:

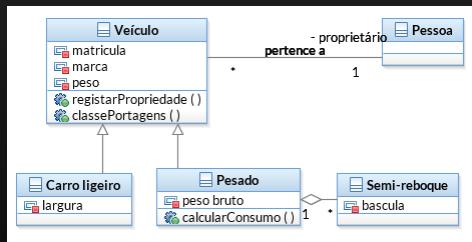
Adicionar mais atributos, operações e relacionamentos à base herdada

Redefinir as operações da superclasse

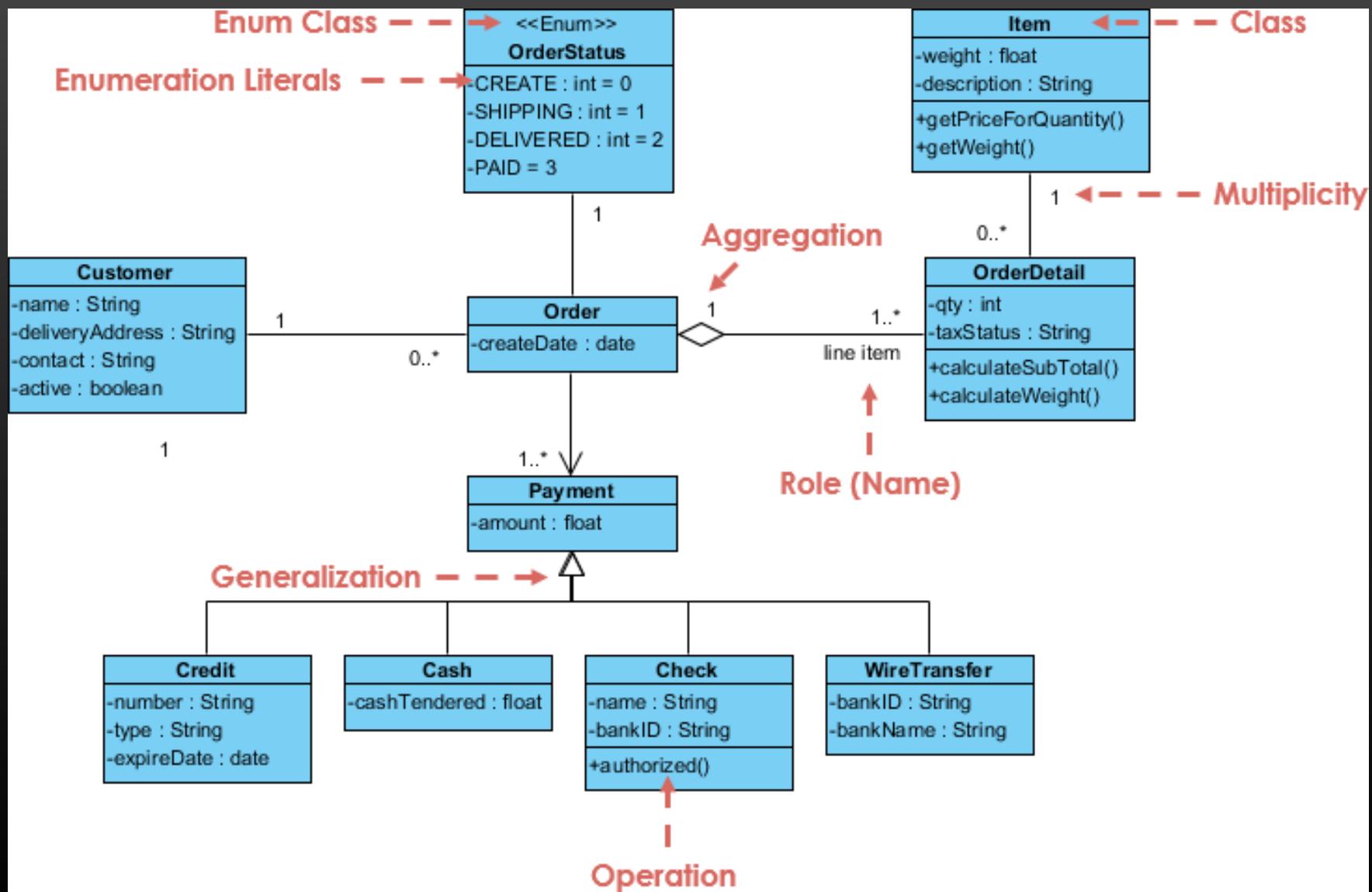
A herança põe em evidencia as características comuns entre classes



Como interpretar este modelo?



Síntese da notação básica do diagrama de classes



Decomposição por objetos está na base da Análise e Desenho por Objectos (OOA, OOD)

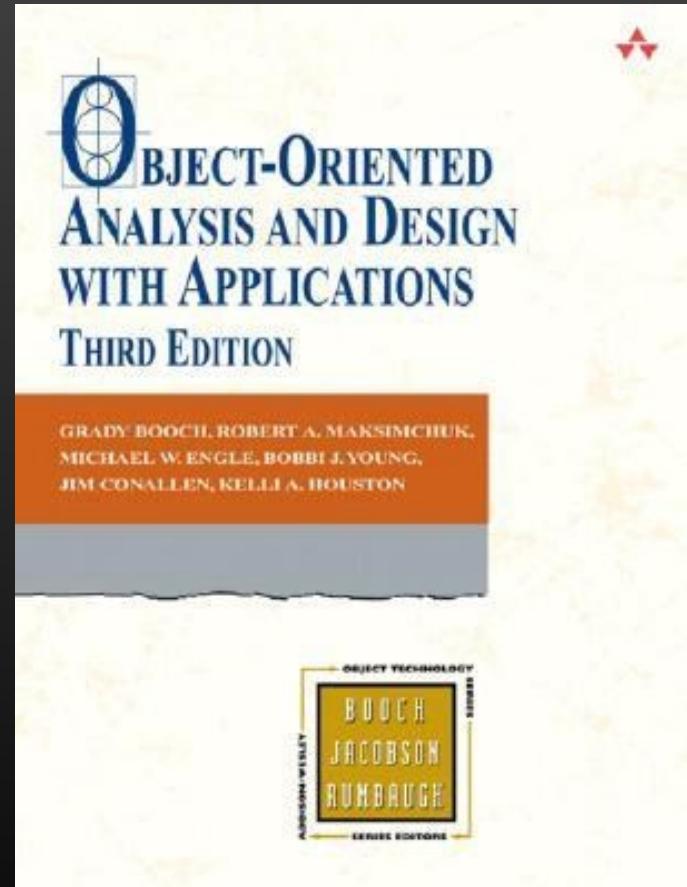
AOO/AOD: encaramos o mundo como um conjunto de entidades autónomos que colaboram para realizar algum comportamento de nível superior.

Assim, as operações não existem como um pequeno algoritmo independente; pelo contrário, as operações estão definidas no contexto de objetos delimitados. Cada objeto na solução (software) encarna o seu próprio comportamento único e cada um modela algum objeto/conceito no mundo real.

Nesta perspetiva, um objeto é simplesmente uma entidade que tem estado (guarda informação) e exibe algum comportamento bem definido.

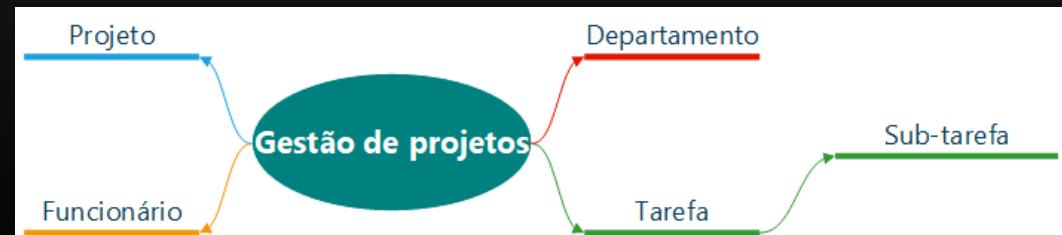
Os objetos fazem as coisas, e a forma de solicitar que façam as operações é enviando-lhes mensagens. Como a nossa decomposição é baseada em objetos e não em algoritmos, chamamos a isto uma decomposição por objetos

<http://www.drdobbs.com/windows/software-complexity-bringing-order-to-ch/199901062>



Exercício: domínio da gestão de projetos

- a) Os projetos têm um título descritivo, uma data de início e de fim.
- b) Um Projeto é implementado por um determinado Departamento.
- c) Cada Projeto define várias Tarefas que, por sua vez, podem estar organizadas em sub-tarefas.
- d) Um projeto tem uma equipa atribuída; as equipas têm vários colaboradores e um gestor.
- e) Os projetos externos são contratados por um cliente. Os internos, são pedidos por um Departamento.
- f) As tarefas têm data de início e duração estimada.



Referências

[PRE'10] Pressman, R. S. (2010). Software Engineering: a practitioners approach (seventh ed). McGraw Hill.

→ Chap. 5

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

→ Chap. 3

[LAR'04] Larman, C. (2004). Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development, 3rd ed. Pearson Education.

→ chap. 5.

41951- ANÁLISE DE SISTEMAS

Máquinas de estados - diagrama de estados

Ilídio Oliveira

v2022-03-22

Máquinas de estado

Os objectos podem mudar de estado em resposta a um evento

mostrar os diferentes estados pelos quais um objecto passa durante a sua vida

pode incluir as respostas e acções do objecto

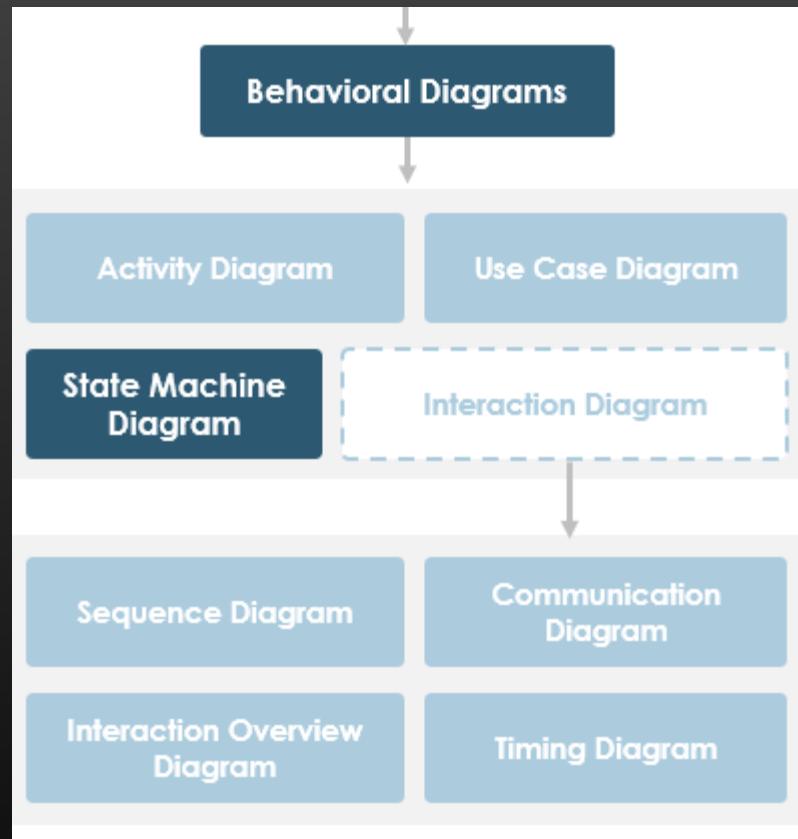
Exemplo: estados de um utente

Novo paciente - ainda não foi visto

Paciente coreente - está agora a receber tratamento

Antigo paciente - já não é visto ou tratado

Tipicamente utilizado apenas para objectos complexos



Utilização do DE

Estados de uma entidade com “ciclo de vida” relevante

E.g.: estados de um dispositivo, estados de um contrato,...

Justifica-se para máquinas de estados não-triviais

Estados

Um estado é uma situação isolada no ciclo de vida de um objeto, em que uma restrição se mantém, o objeto executa uma atividade ou aguarda por um evento.

À representação dos estados possíveis de uma entidade e das regras de transição de um estado para o outro, dá-se o nome de **máquina de estados**.

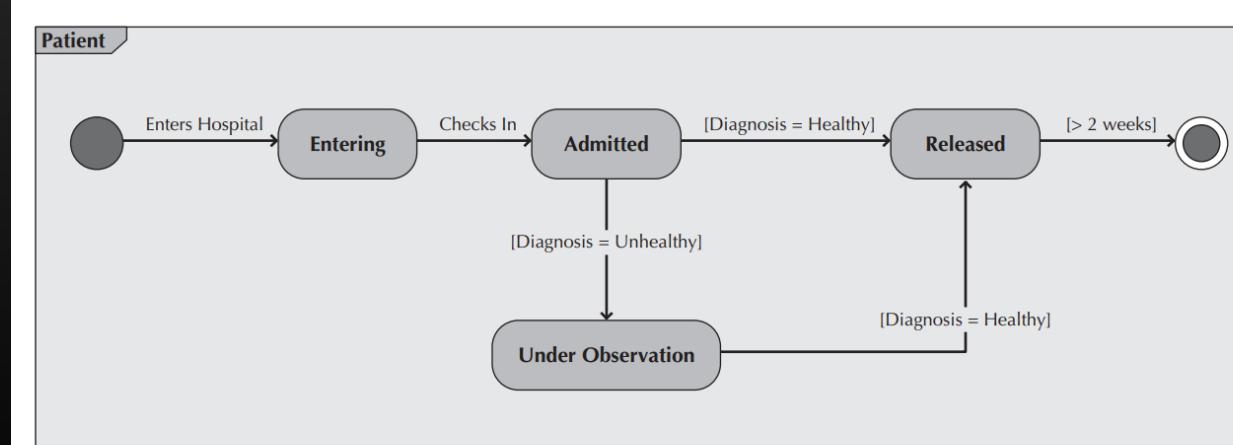
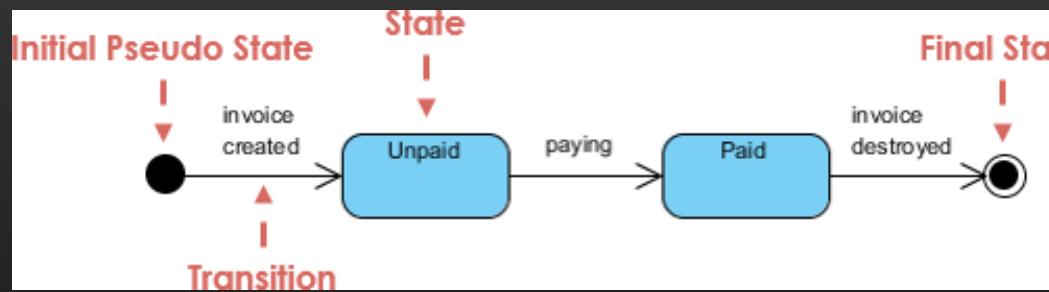
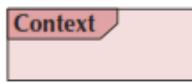
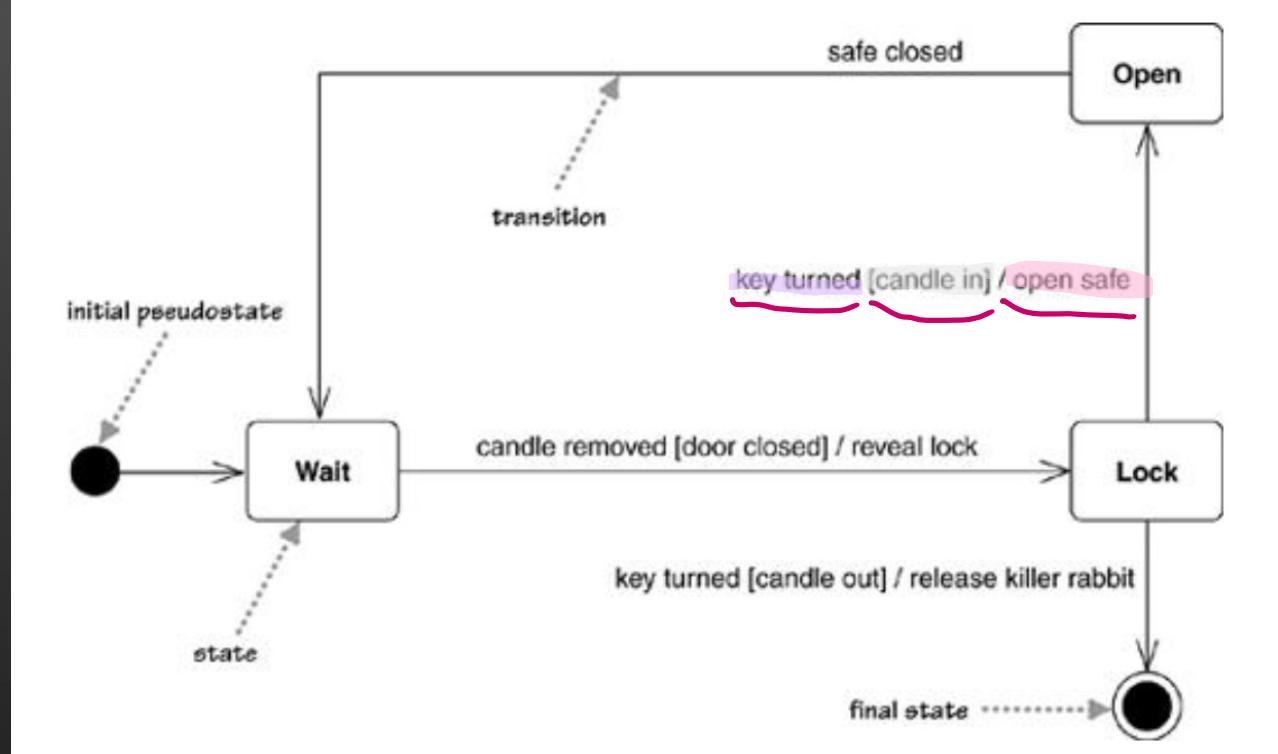


FIGURE 6-16 Sample Behavioral State Machine Diagram

Sintaxe

Term and Definition	Symbol
<p>A state:</p> <ul style="list-style-type: none">■ Is shown as a rectangle with rounded corners.■ Has a name that represents the state of an object.	
<p>An initial state:</p> <ul style="list-style-type: none">■ Is shown as a small, filled-in circle.■ Represents the point at which an object begins to exist.	
<p>A final state:</p> <ul style="list-style-type: none">■ Is shown as a circle surrounding a small, filled-in circle (bull's-eye).■ Represents the completion of activity.	
<p>An event:</p> <ul style="list-style-type: none">■ Is a noteworthy occurrence that triggers a change in state.■ Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time.■ Is used to label a transition.	anEvent
<p>A transition:</p> <ul style="list-style-type: none">■ Indicates that an object in the first state will enter the second state.■ Is triggered by the occurrence of the event labeling the transition.■ Is shown as a solid arrow from one state to another, labeled by the event name.	
<p>A frame:</p> <ul style="list-style-type: none">■ Indicates the context of the behavioral state machine.	



A transição indica um movimento de um estado para outro.

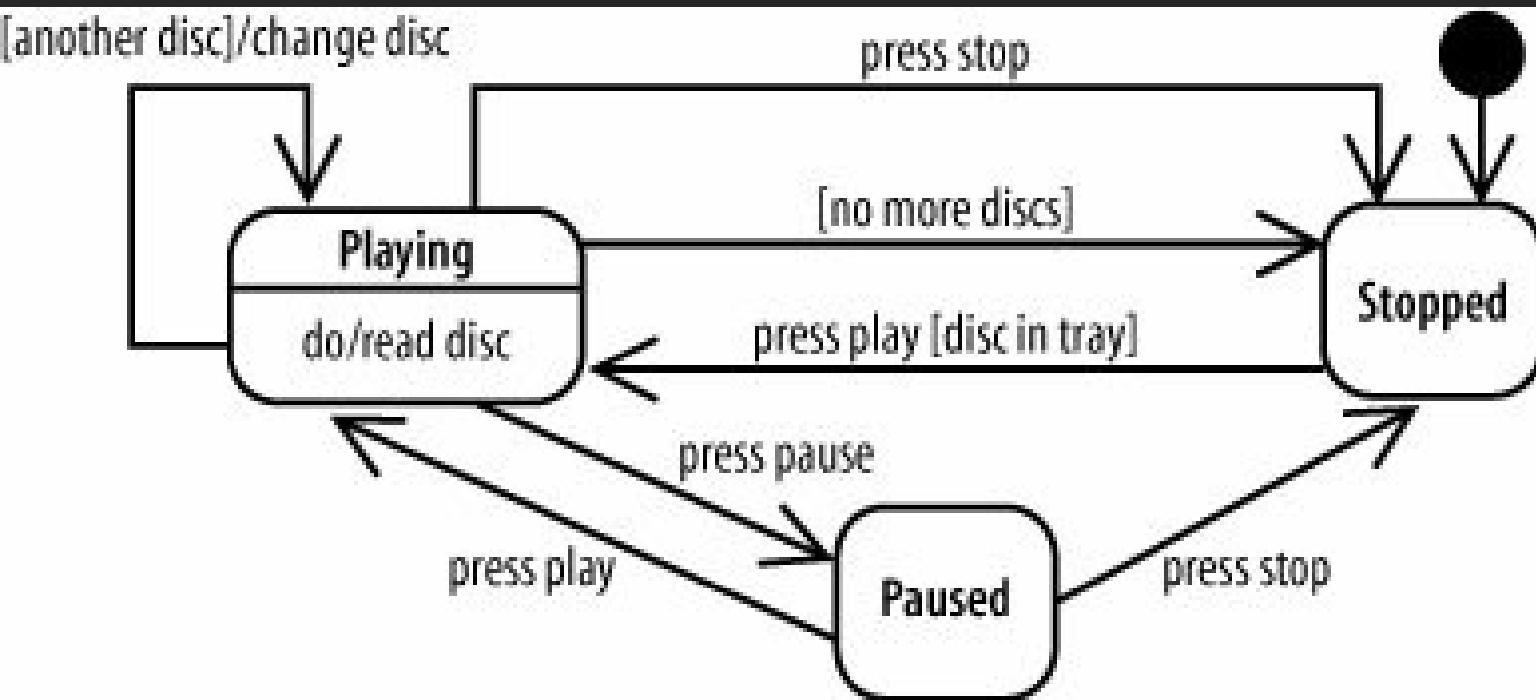
Cada transição tem uma etiqueta com três partes:
activação [guarda]/actividade. Todas as partes são opcionais.

A ativação (gatilho) é normalmente um único evento que desencadeia uma potencial mudança de estado.

A condição de acesso (guarda), se presente, é uma expressão booleana que deve ser verdadeira para que a transição seja efectuada.

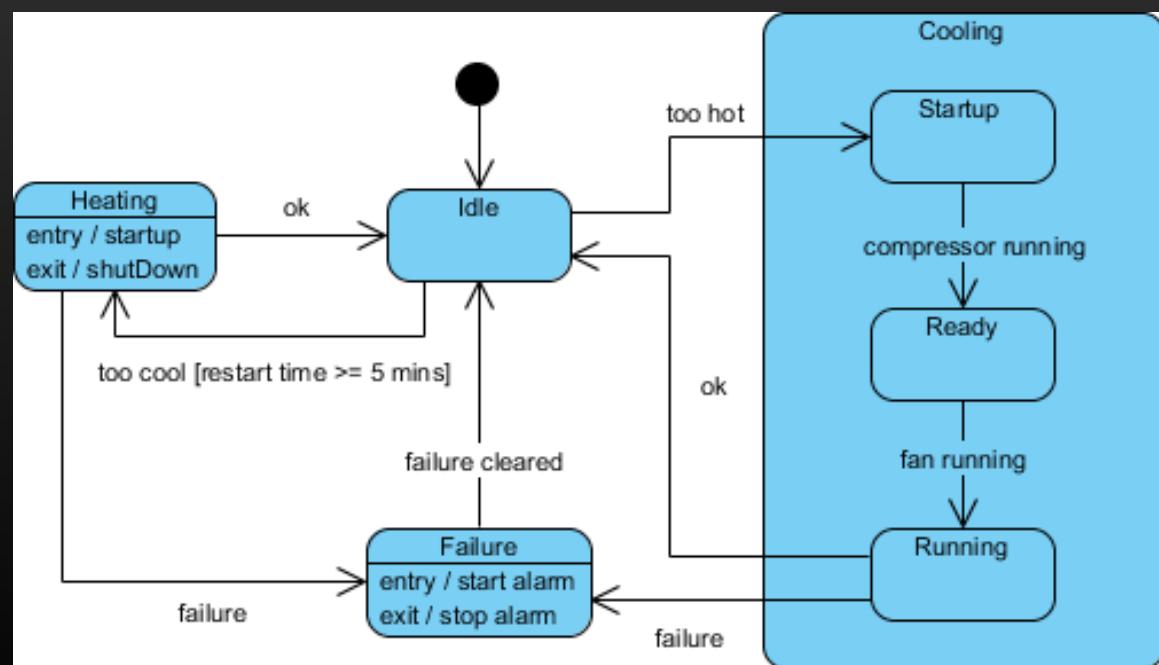
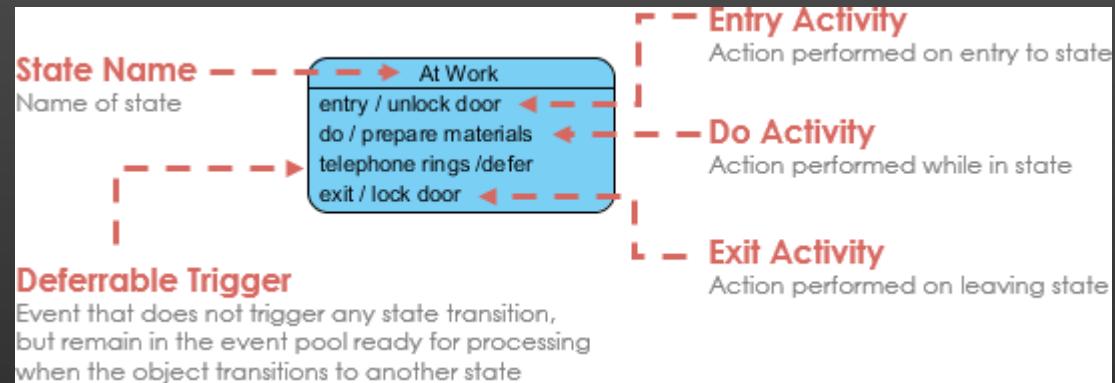
A actividade é algum comportamento que é executado durante a transição.

Exemplo



Actividades internas

Os Estados podem reagir a eventos sem transição, utilizando actividades internas: colocando o evento, guarda e actividade dentro da própria caixa do estado



Quando utilizar DE?

Usar diagramas de estado apenas para situações que exibem evolução de comportamento interessante

a construção do diagrama do estado ajuda a entender o que se passa/transições

Exemplos:

- Comportamento de um controlador de um equipamento/dispositivo (de um ATM, de um elevador, de um sensor com alarmes,...)
- Comportamento de um controlador da interação utilizador/interface gráfico
- Protocolo de comunicação entre dispositivos (e.g.: ler de um sensor de pulsação por Bluetooth)
- Objetos (do domínio) com regras relevantes (e.g.: estados de uma encomenda)

Quando usar?

Objetos com comportamento dependente do estado

Exemplos:

Controlador de um dispositivo físico (hw)

Lógica de objetos de negócio
(Venda, Reserva,...)

Protocolo de comunicação entre dispositivos (e.g.: ler de um sensor de pulsação por Bluetooth)

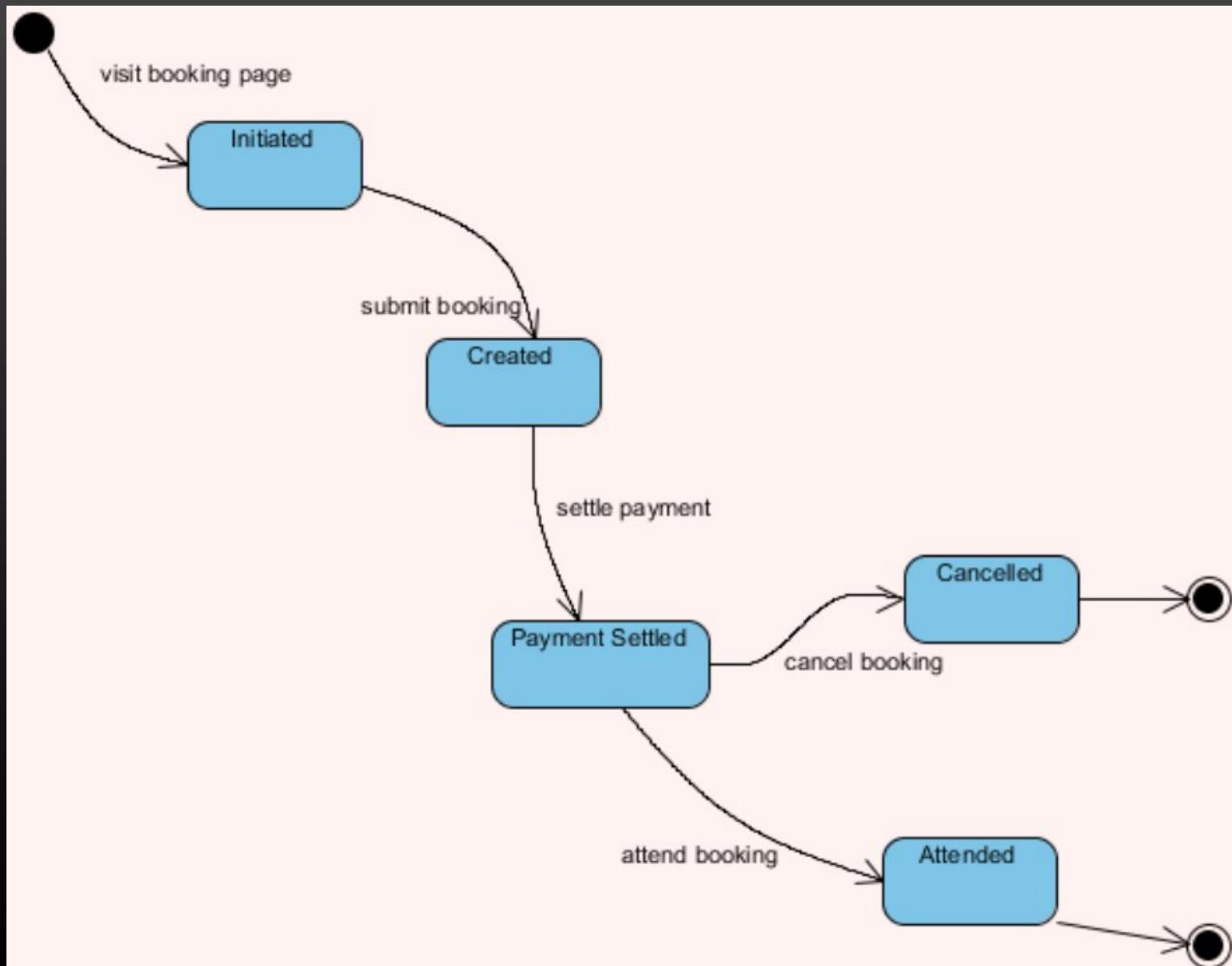
Comportamento de um controlador da interação utilizador/interface gráfico

No modelo do domínio:

Caraterizar os estados de uma classe complexa

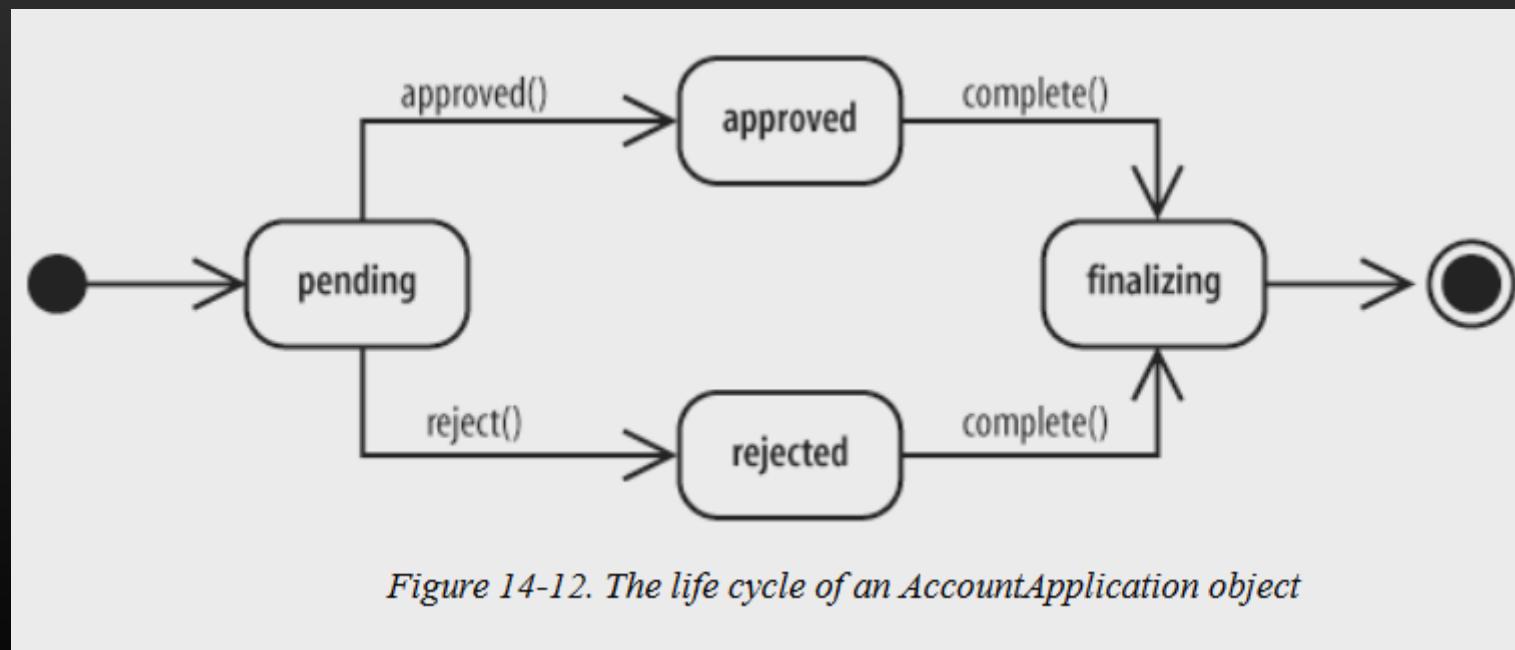
e.g.: reserva, inscrição,...

Domínio do problema



Ciclo de vida de um objeto

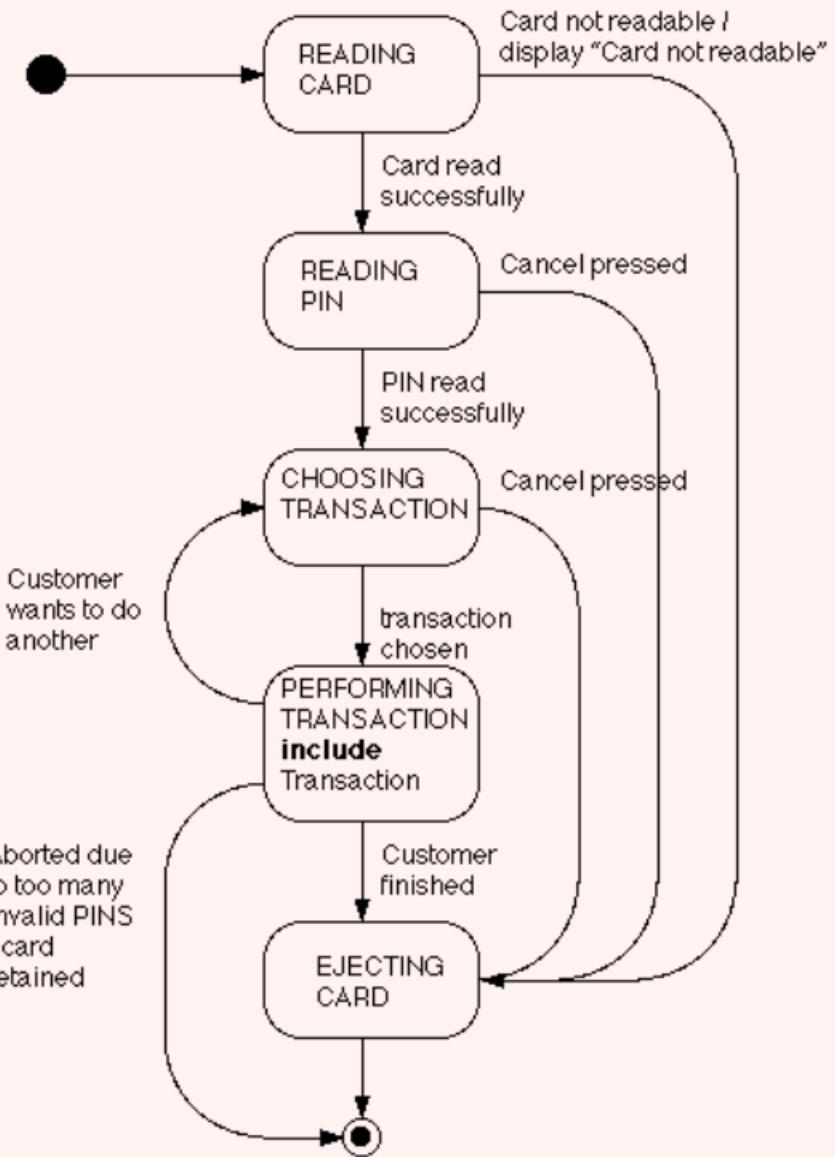
Nível do software



Controladores

Nível dos dispositivos.

State-Chart for One Session



41951- ANÁLISE DE SISTEMAS

Modelos de interação - diagrama de sequência

Ilídio Oliveira

v2022-03-22

Objectivos de aprendizagem

Compreender o papel da modelação do comportamento no SDLC

Compreender as regras e as diretrizes de estilo para sequência, comunicação e diagramas de estado

Compreender a complementaridade entre os diagramas de sequência e comunicação

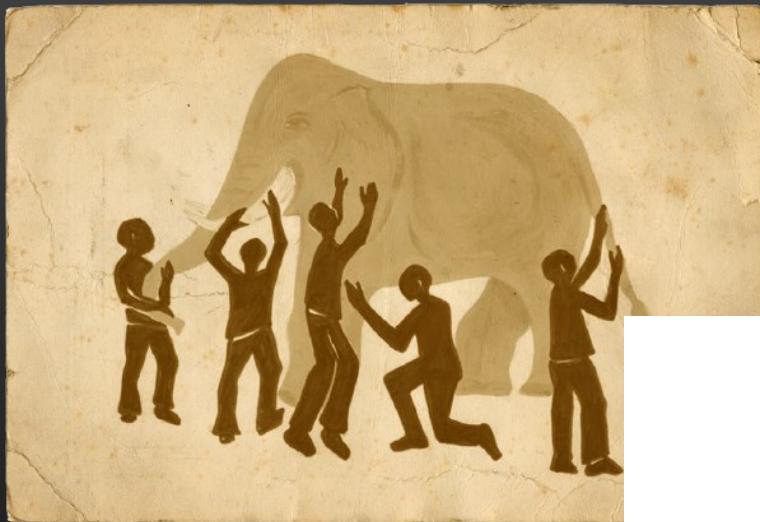
Explicar a relação entre modelos de função, estruturais e de comportamento

Os modelos são vistas parciais

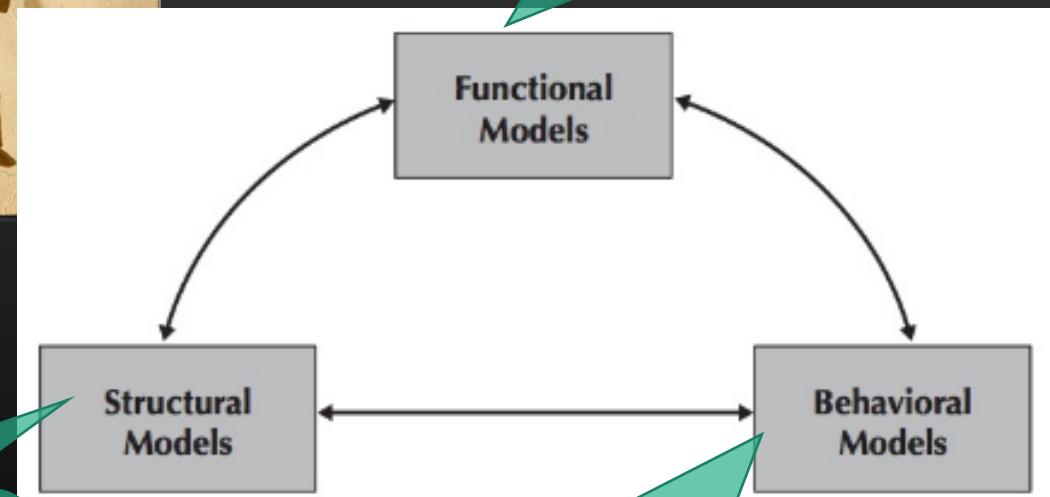


Num projeto, recorremos a
vários modelos
complementares

Três categorias de modelos



O que é que o Sistema deve fazer?
[do ponto de vista do observador externo;
caixa fechada]



Quais são as partes/"coisas"
que constituintes e o
relacionamento estrutural
entre elas?

Como é que as partes
colaboram/interagem ao
longo do tempo? [perspetiva
dinâmica]

O que tratam os modelos comportamentais

Os modelos comportamentais descrevem os aspectos dinâmicos de um sistema de informação.

Durante a análise:

os modelos comportamentais descrevem qual é a lógica interna dos processos sem especificar como os processos vão ser implementados.

Observação do exterior
(e.g.: casos de utilização)

Durante o Desenho/Implementação:

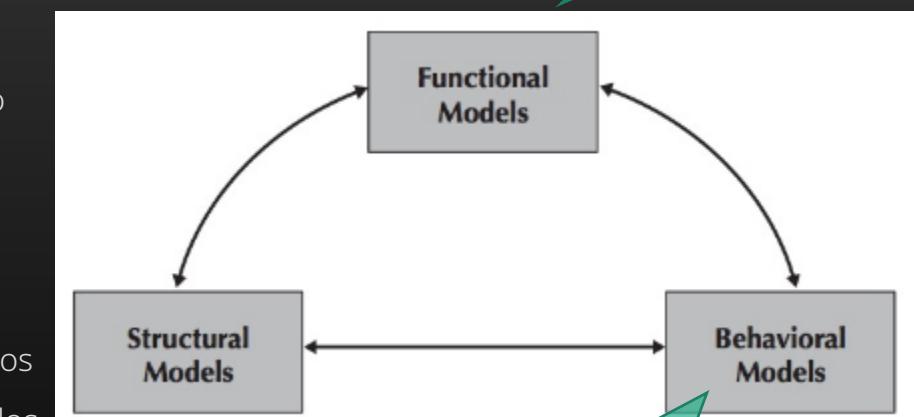
os modelos comportamentais descrevem a interação entre módulos de software

A modelação comportamental pode ser conduzida pelos casos de utilização:

Um dos principais objetivos é mostrar como os objetos de um domínio trabalham em conjunto para formar uma colaboração que realiza cada um dos cenários dos casos de utilização.

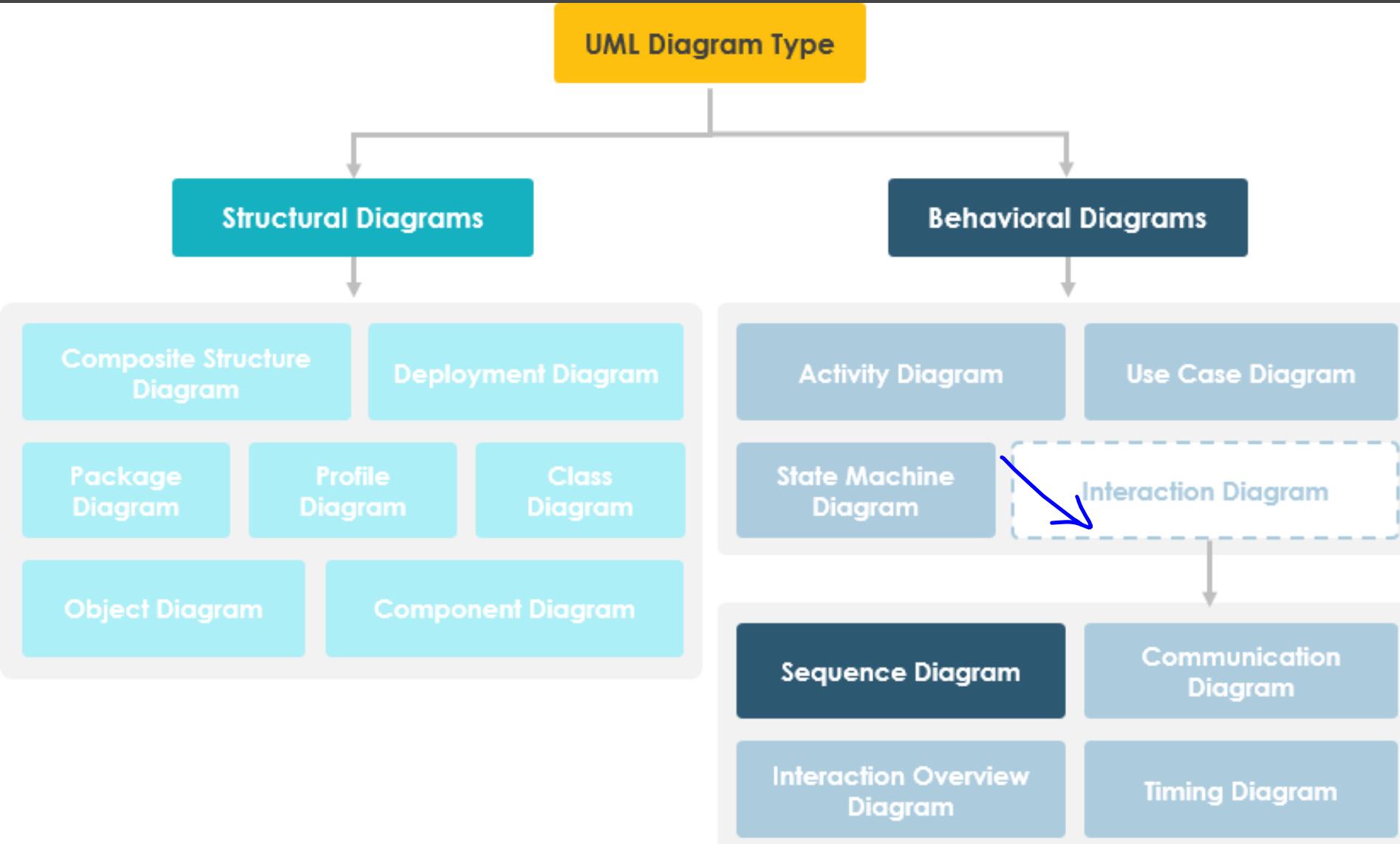
Modelos estruturais → os objetos e as relações

Modelos comportamentais → visão interna de um processo.



Explicação de uma colaboração
(e.g.: interação entre objectos para realizar um CaU)

Partes e relacionamentos estruturais entre elas (e.g.: d. de classes)



Tipos de modelos comportamentais

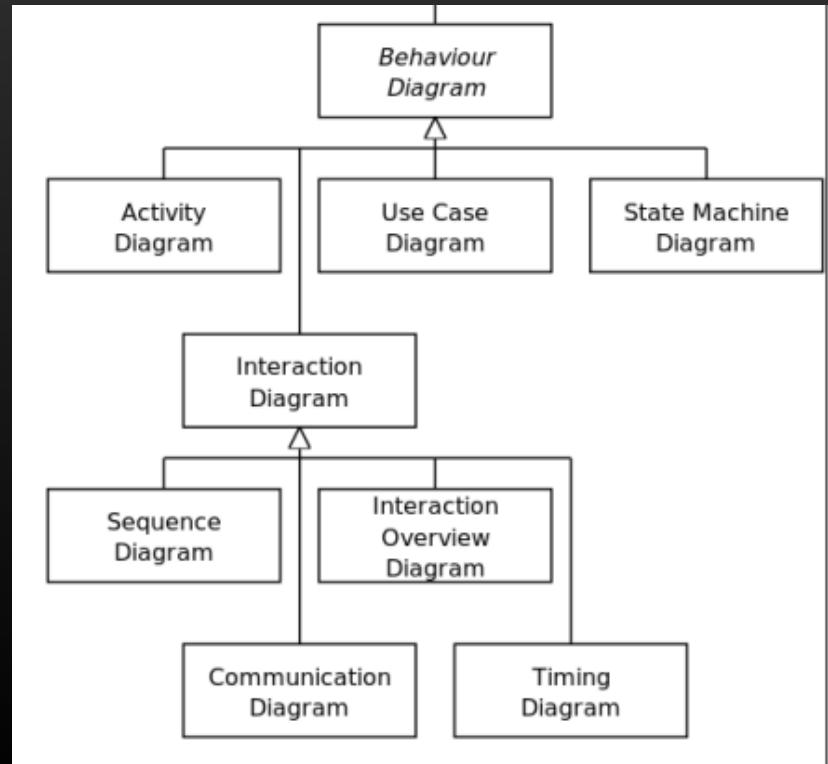
Representação dos detalhes de um processo

Diagramas de interação (Sequência & Comunicação)

Mostra como os objetos colaboram para fornecer a funcionalidade definida nos casos de utilização.

Representações de alterações nos dados (Estado)

Máquinas de estado



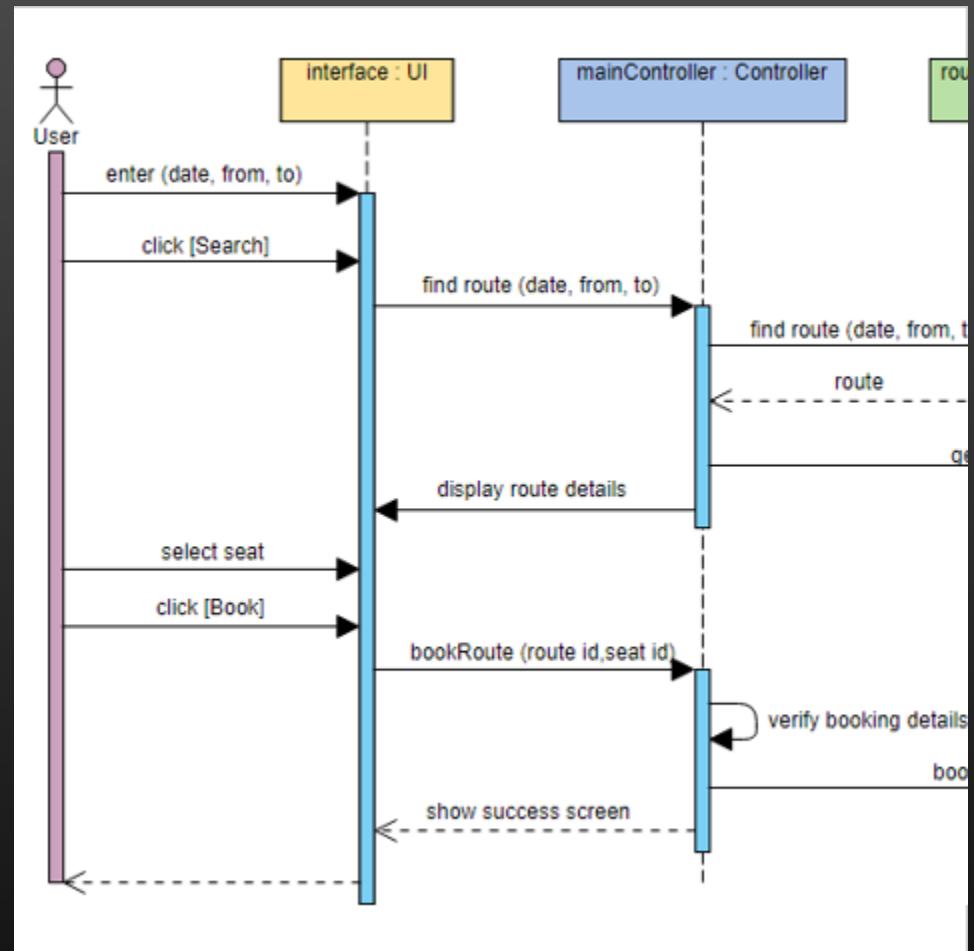
Cenário do restaurante: envio de mensagens



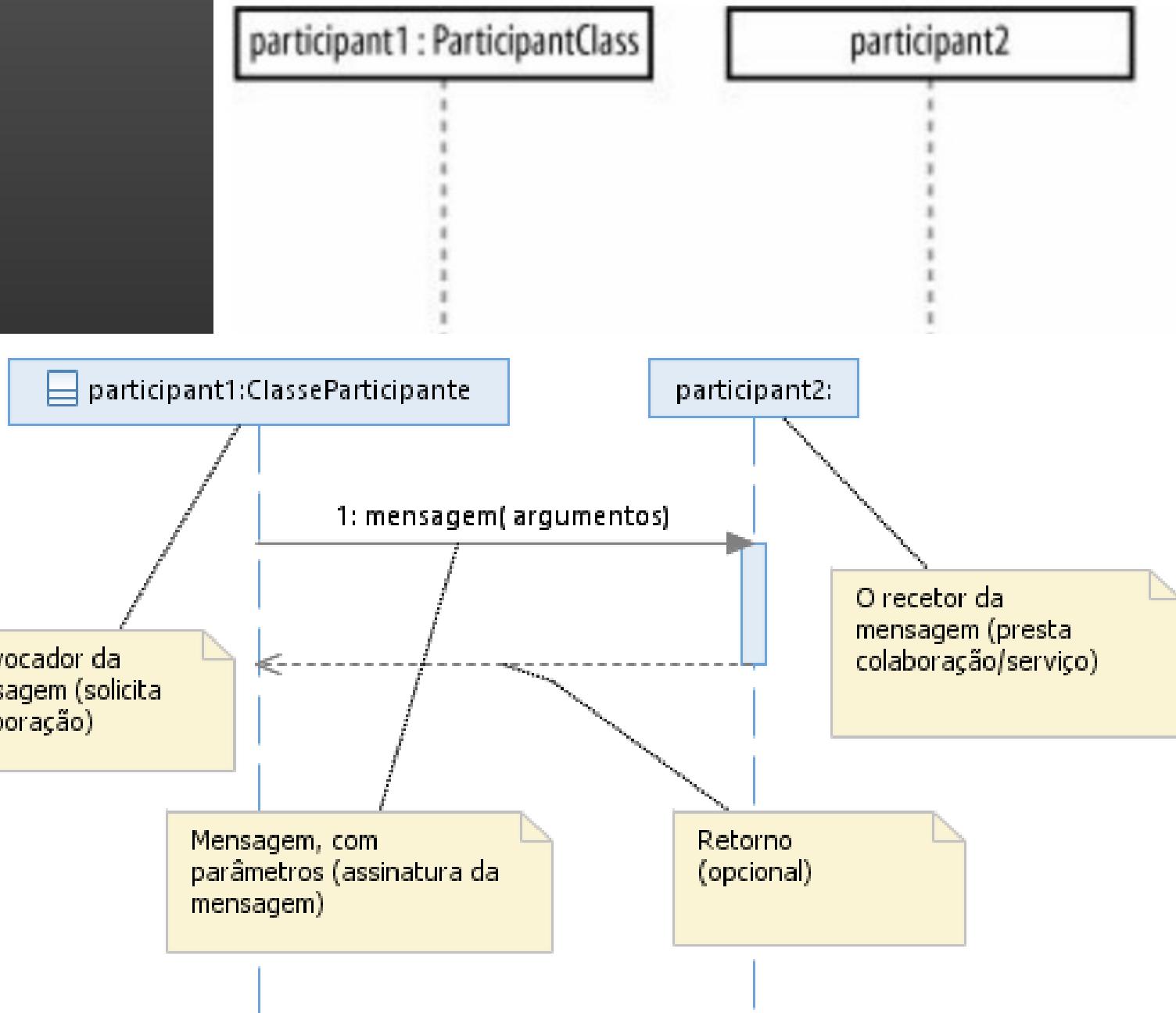
Diagrama de sequência

Ilustrar os objetos que participam numa colaboração (por exemplo: caso de utilização) e as mensagens que passam entre eles ao longo do tempo.

Um diagrama de sequência é um modelo dinâmico que mostra a sequência explícita de mensagens que são passadas entre objetos numa interação definida.



<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>



Foco na colaboração ao nível do objeto

Diagramas de classe

O foco de modelação dos diagramas de classe está no nível de classe (classificador).

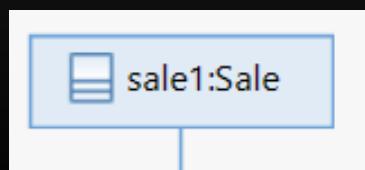
Cada tipo de objetos tem atributos que descrevem informação (estado) sobre o objeto.

Diagramas de interação

Os diagramas de interação focam-se no **nível do objeto/instância**

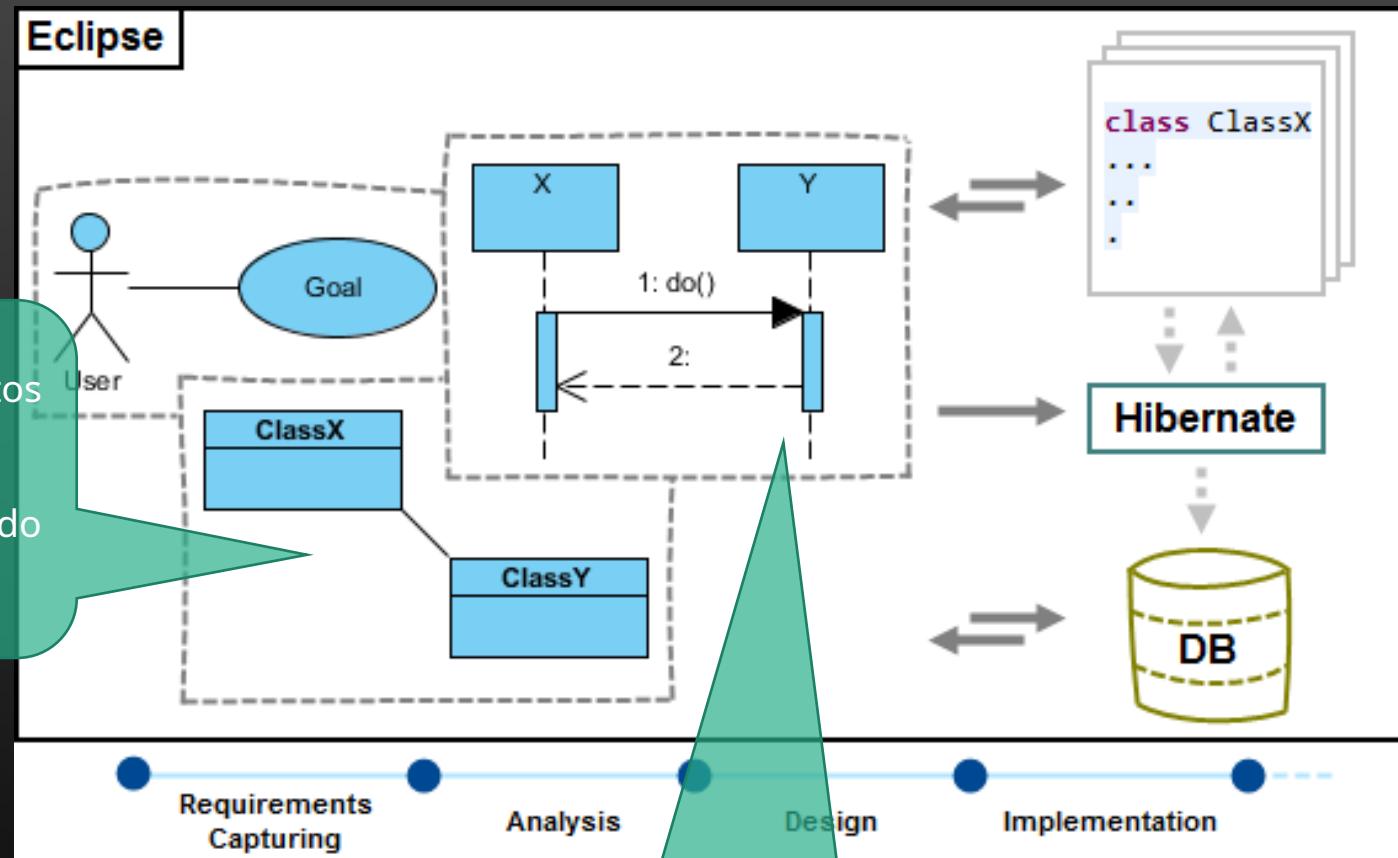
Cada objeto também tem comportamentos. Os comportamentos são descritos pelas operações. Uma operação é uma ação que um objeto pode realizar.

Cada objeto também pode **enviar e receber mensagens**. As mensagens são solicitações enviadas a objetos para que executem um dos seus comportamentos. Uma mensagem é uma *chamada* de um objeto para outro objeto.



Nível de abstração: análise ou implementação?

As Classes definem características dos objetos de um tipo (*molde*).
As classes podem representar entidades do domínio ou da implementação.



O D. de Sequência mostra como é que os objetos (que segem uma Classe) colaboram com outros, para realizar uma atividade. Também aqui, essa atividade pode ser no nível do “domínio” ou no nível do código.

O modelo do domínio e a implementação situam-se em planos de abstração diferentes

Diagramas de sequência: etapa de análise

Explicar a **colaboração** observada para realizar um processo do domínio.

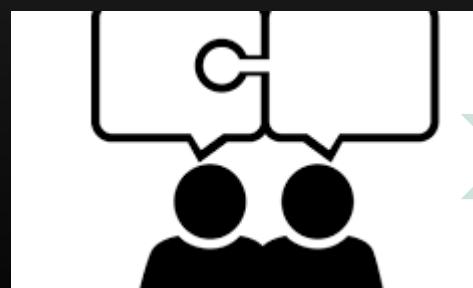
Os “objetos” são tipicamente entidades de alto nível:

- papéis de pessoas ou subsistemas

Diagrama de sequência: perspetiva de implementação

Explicar a colaboração entre objetos de código para realizar uma operação (da implementação)

Os objetos são instâncias numa linguagem OO (e.g. Java)



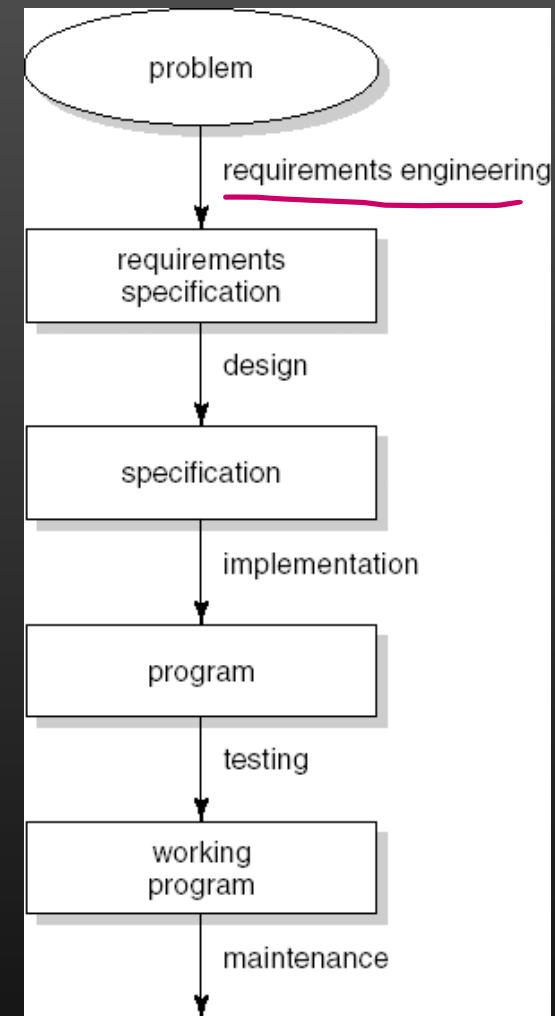
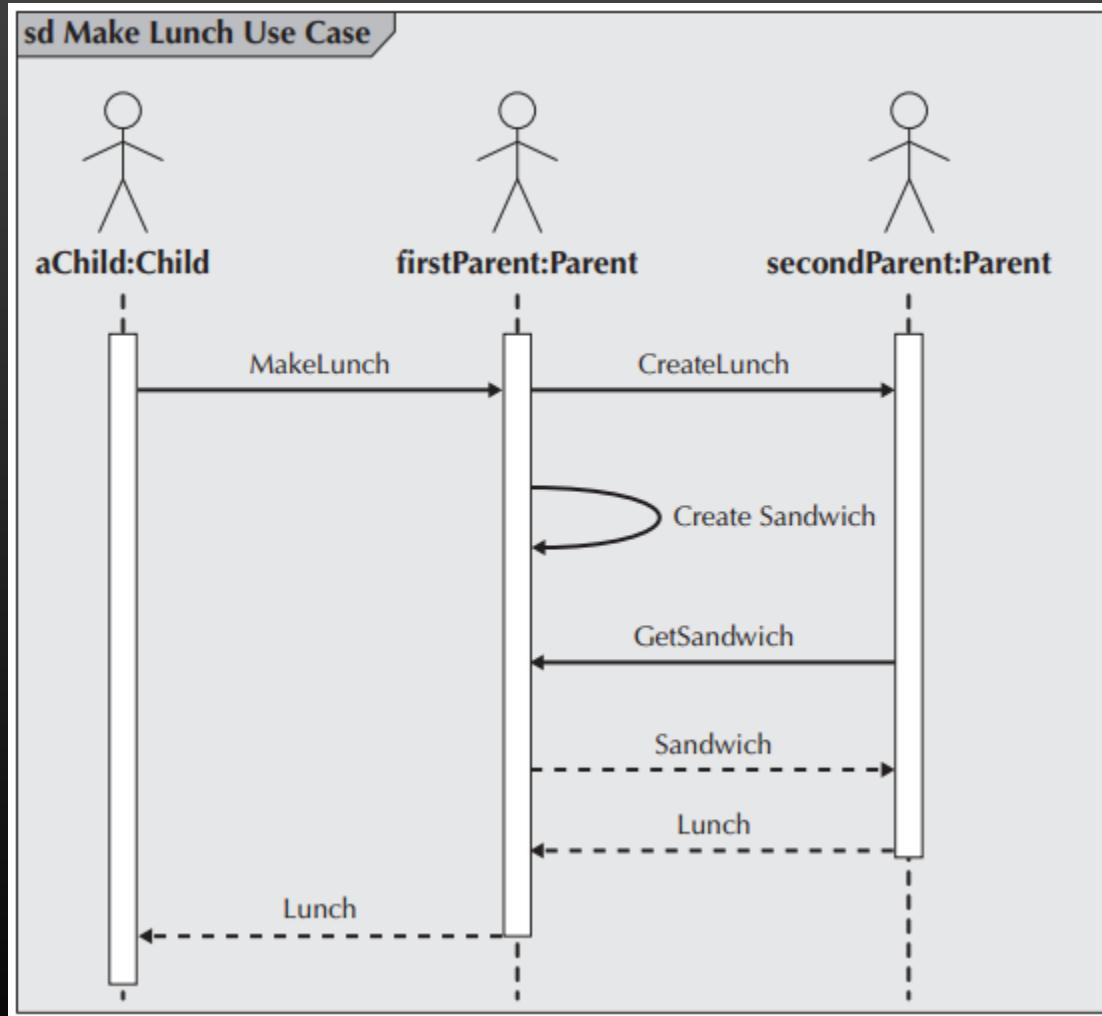
Análise

Especificação



Implementação

Colaboração entre atores/fluxo de um CaU



Mostrar a realização de um caso de utilização

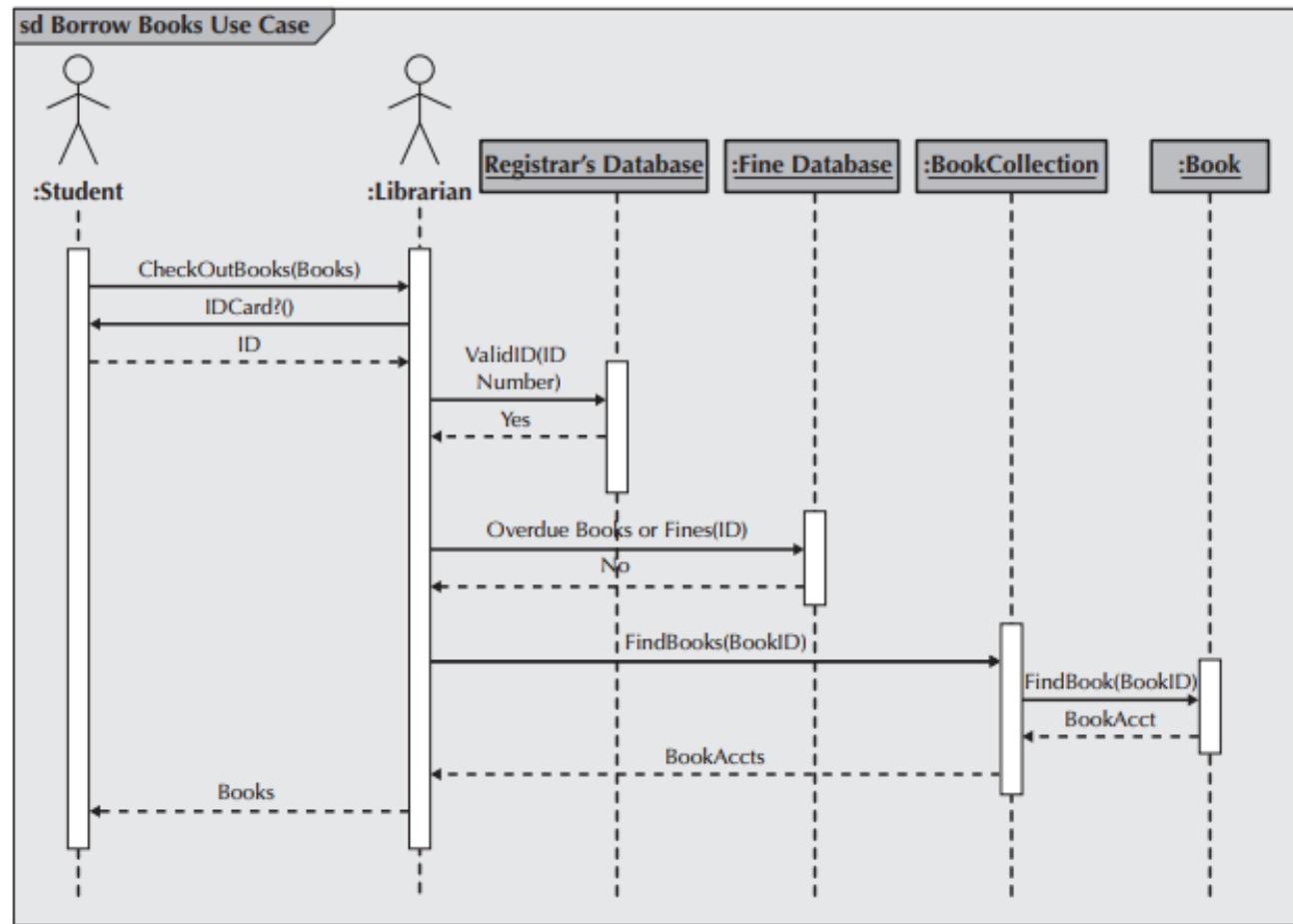
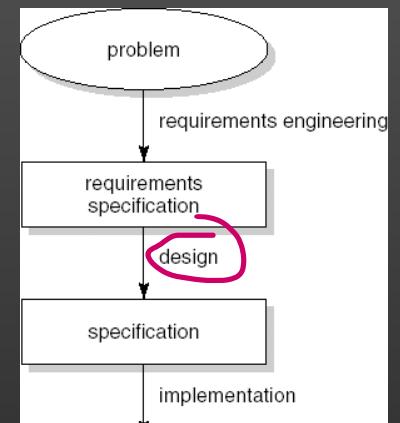
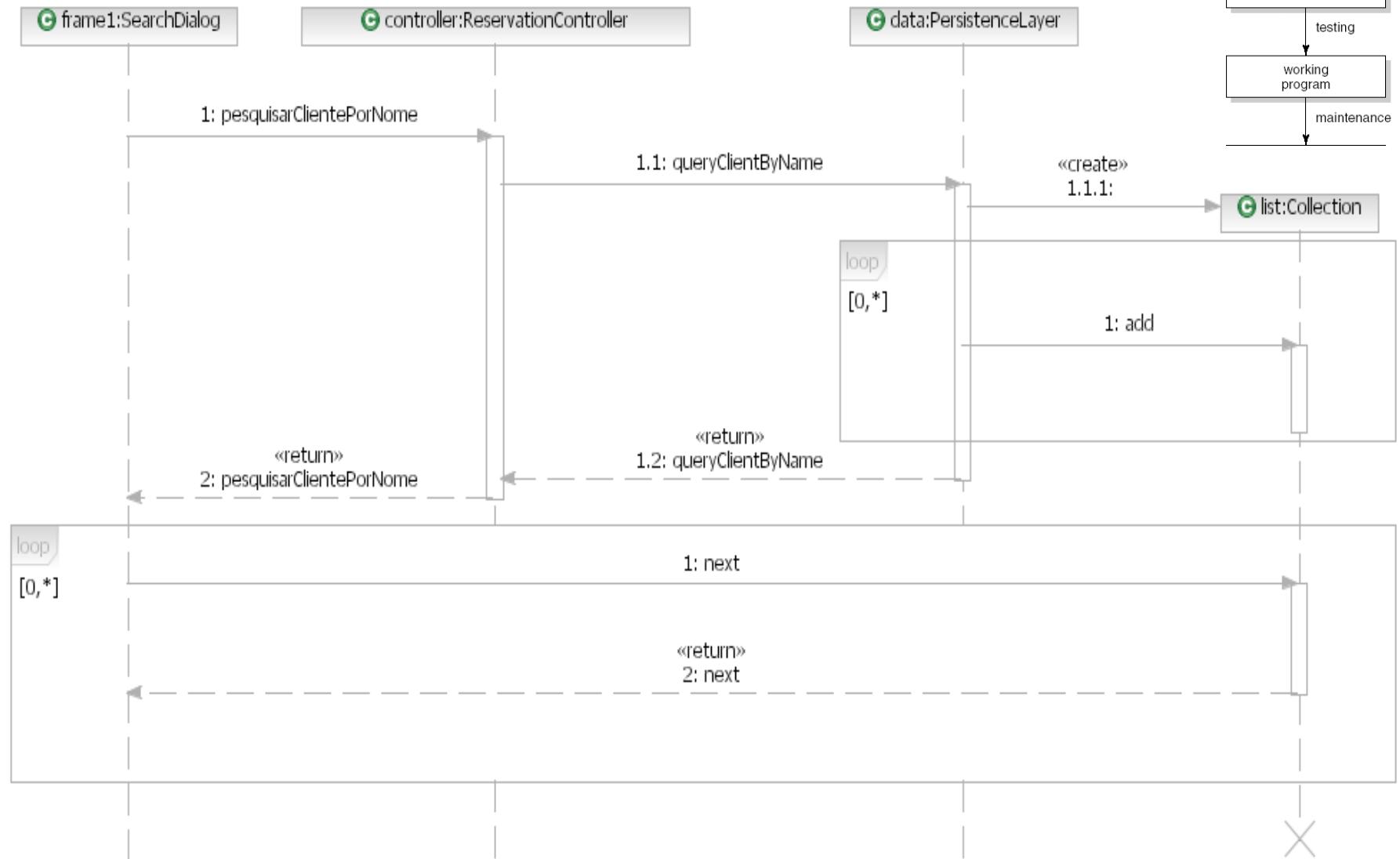
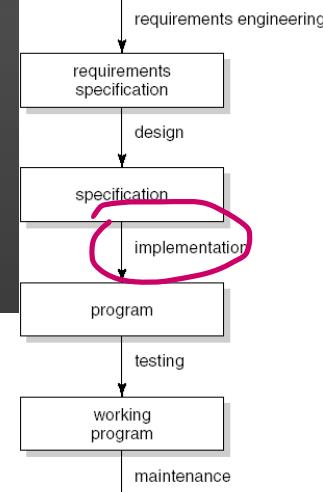
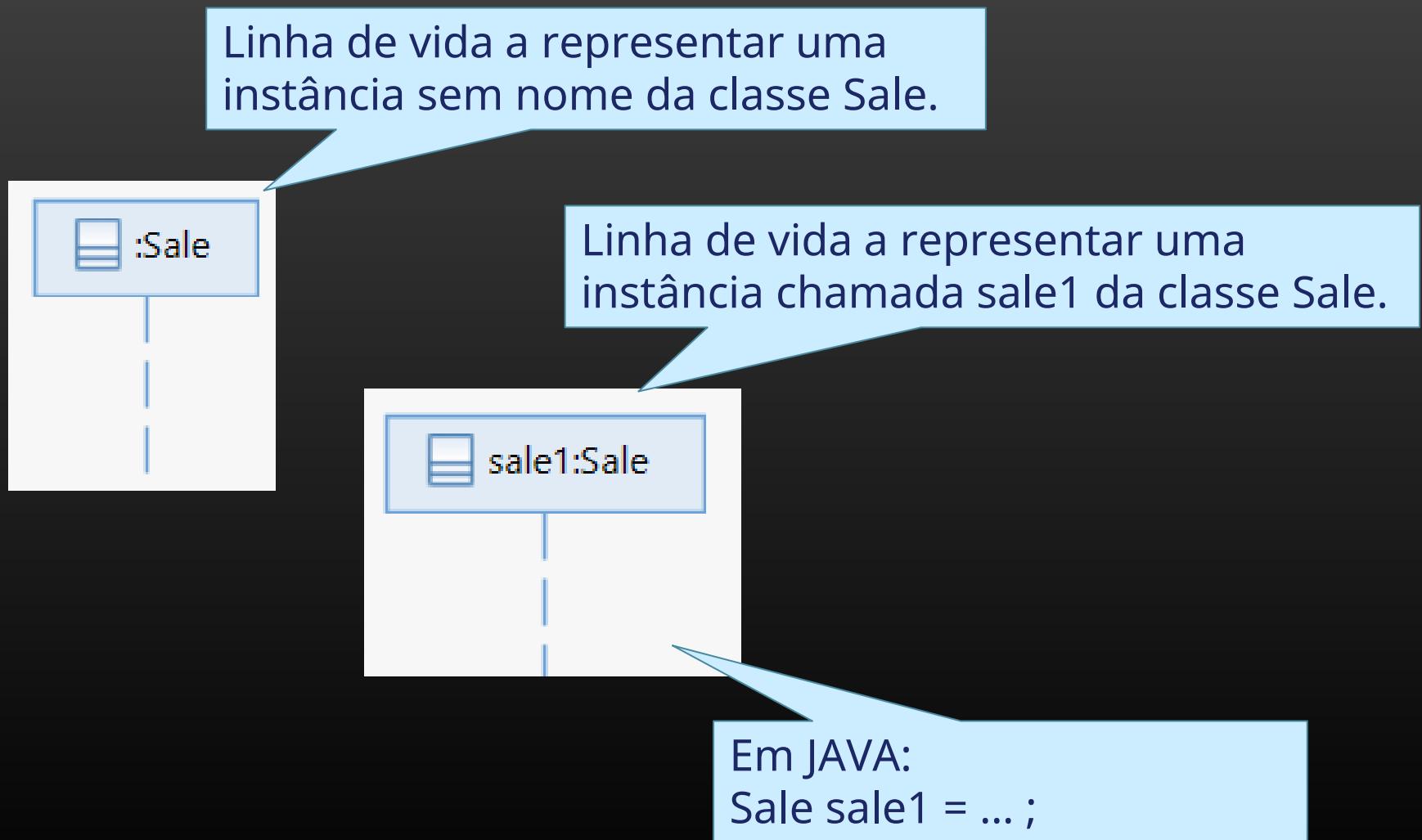


FIGURE 6-9
Sequence Diagram
of the Borrow
Books Use Case
for Students with
a Valid ID and No
Overdue Books
or Fines

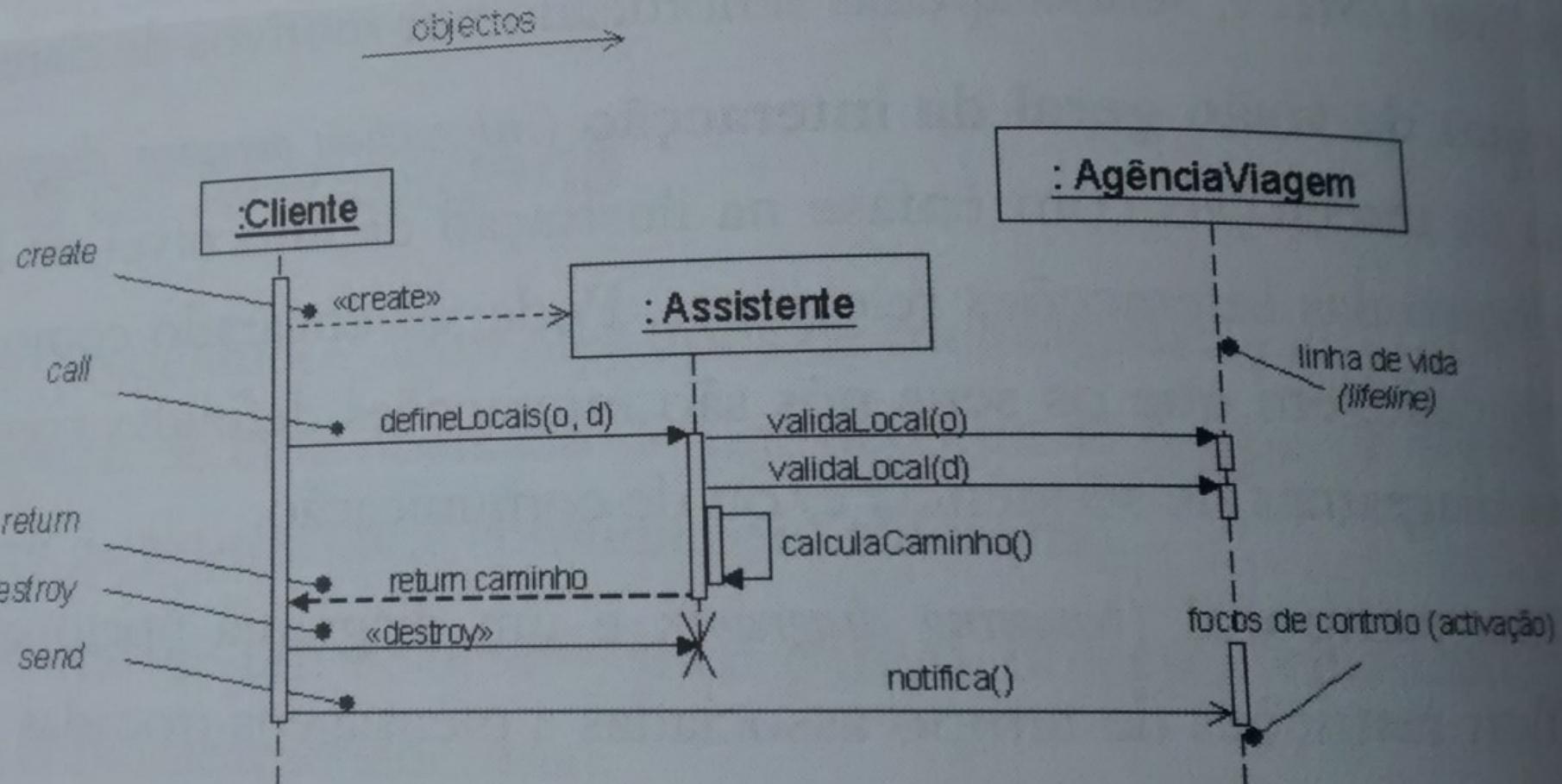
Mostrar a colaboração entre objetos de código



Aspetos notacionais dos DS



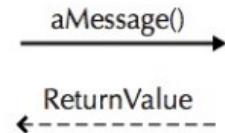
Colaboração entre objetos por mensagens (síncronas)



Term and Definition	Symbol
<p>An actor:</p> <ul style="list-style-type: none"> ■ Is a person or system that derives benefit from and is external to the system. ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram. ■ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <><> in it (alternative). 	 anActor <div data-bbox="1411 626 1699 727" style="border: 1px solid black; padding: 5px; width: fit-content;"> <><> anActor </div>
<p>An object:</p> <ul style="list-style-type: none"> ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram. 	<div data-bbox="1411 856 1699 928" style="border: 1px solid black; padding: 5px; width: fit-content;"> anObject : aClass </div>
<p>A lifeline:</p> <ul style="list-style-type: none"> ■ Denotes the life of an object during a sequence. ■ Contains an X at the point at which the class no longer interacts. 	

A message:

- Conveys information from one object to another one.
- A operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.



A guard condition:

- Represents a test that must be met for the message to be sent.



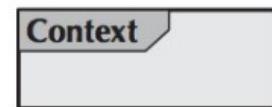
For object destruction:

- An X is placed at the end of an object's lifeline to show that it is going out of existence.

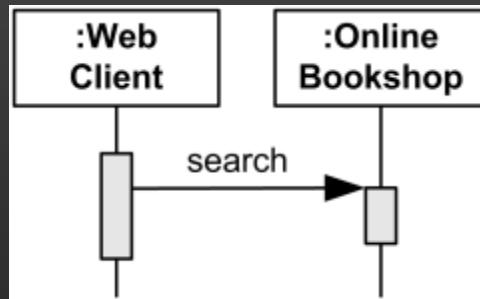
X

A frame:

- Indicates the context of the sequence diagram.



Semântica da invocação

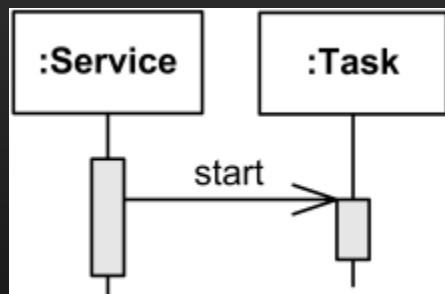


síncrona

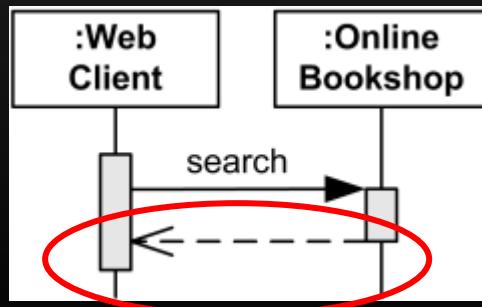
...

```
result = B.search();
```

...

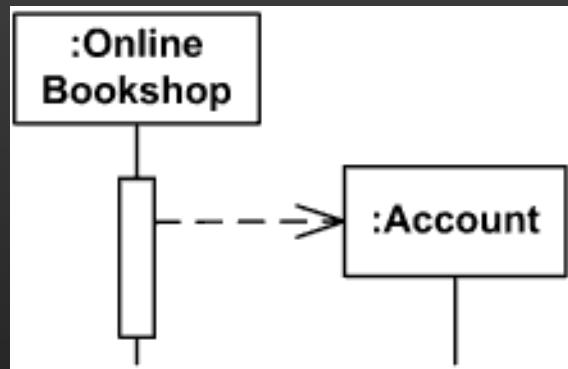


assíncrona



retorno

Modelar a criação/destruição do participante

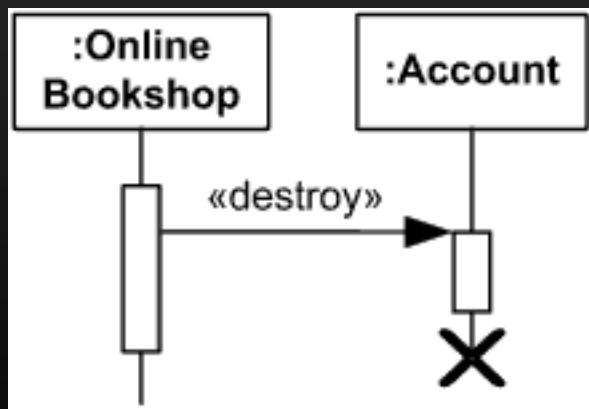


create

...

```
Account a= new Account()
```

...



destroy

...

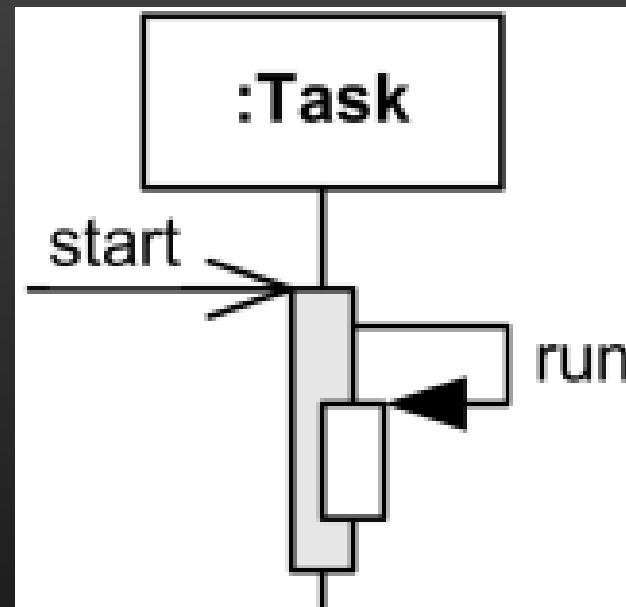
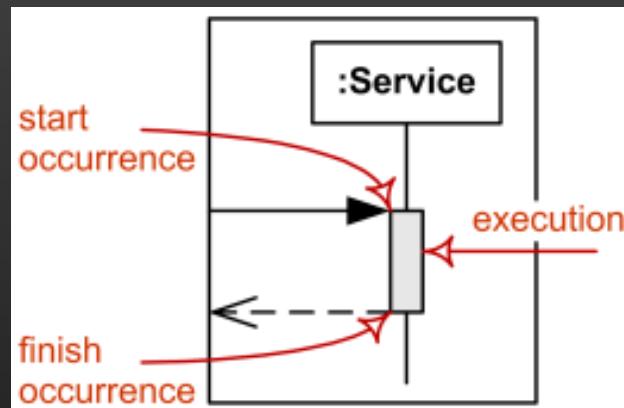
```
a=null; // in java
```

```
[a release] // objective C
```

```
free a; // C
```

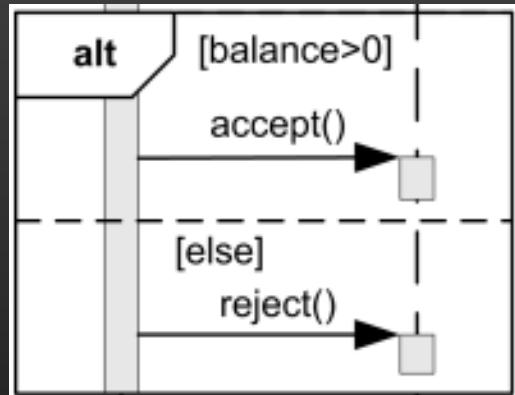
From <http://www.uml-diagrams.org/sequence-diagrams.html>

Ativação



From <http://www.uml-diagrams.org/sequence-diagrams.html>

Fragments para mostrar alternativas

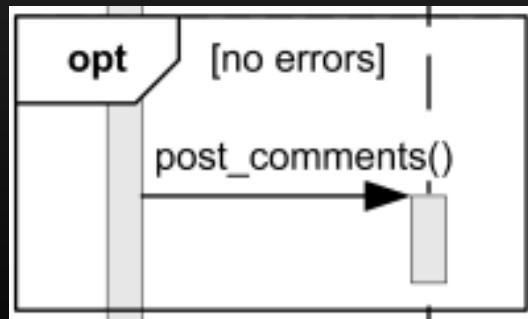


If ($\text{balance} > 0$)

otherObj.Accept()

Else

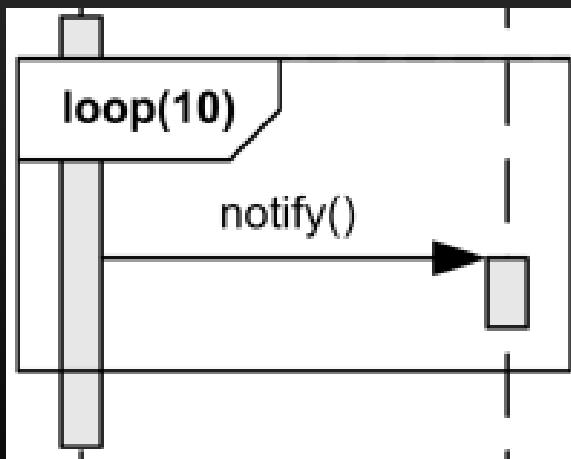
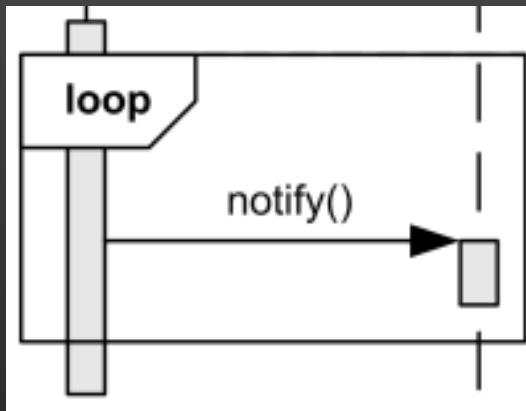
otherObj.Reject()



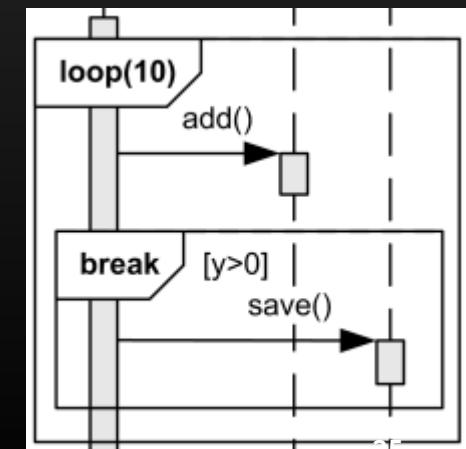
If (no errors)

otherObj.Post_comments()

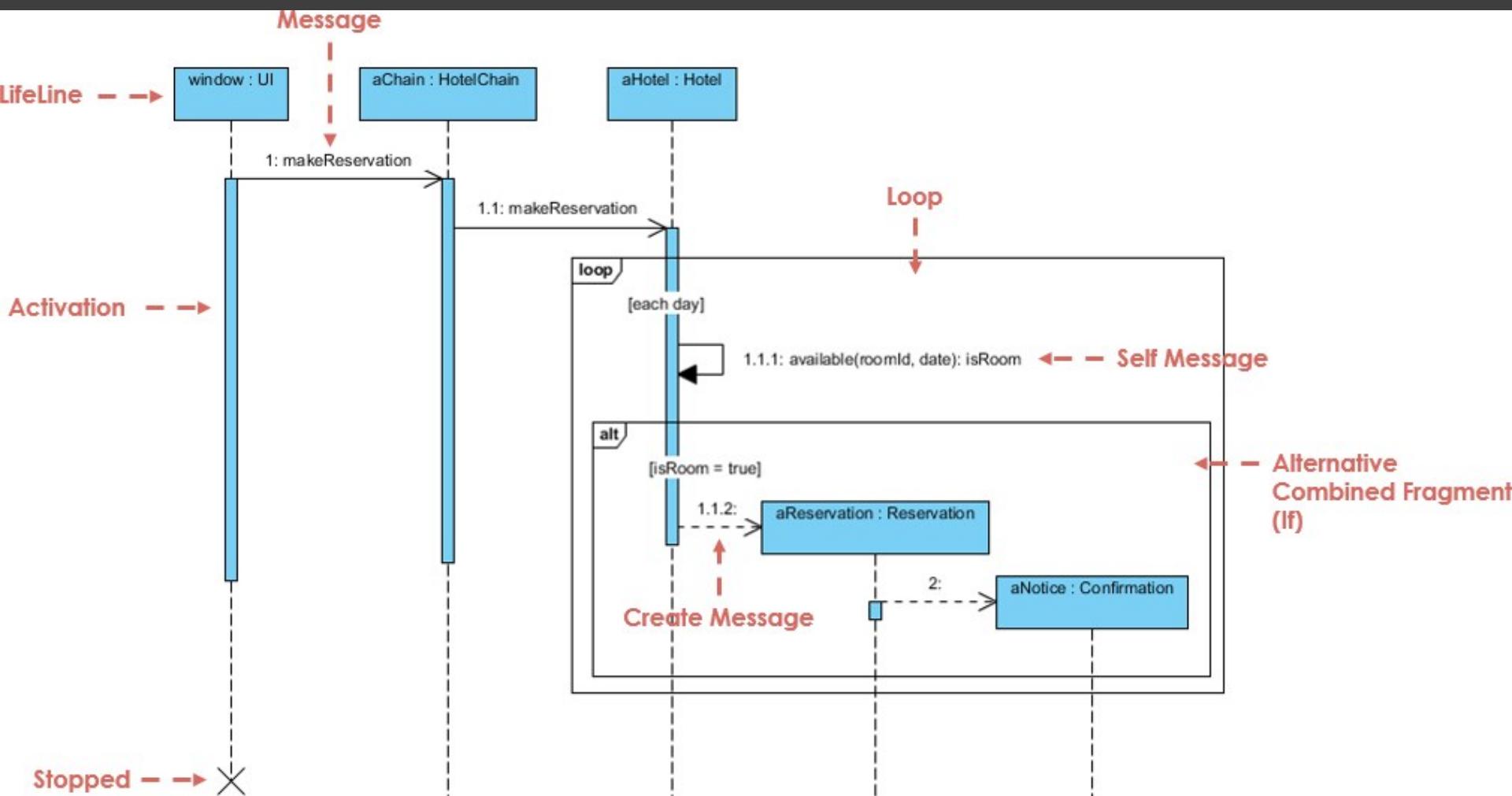
Fragmentos para mostrar loops



```
i=0;  
While( i<10 )  
{  
    otherObj.Add()  
    If ( y>0 ) break;  
    i++  
}
```



Principais elementos notacionais



Quatro tipos de diagramas de iteração disponíveis

D. Sequência

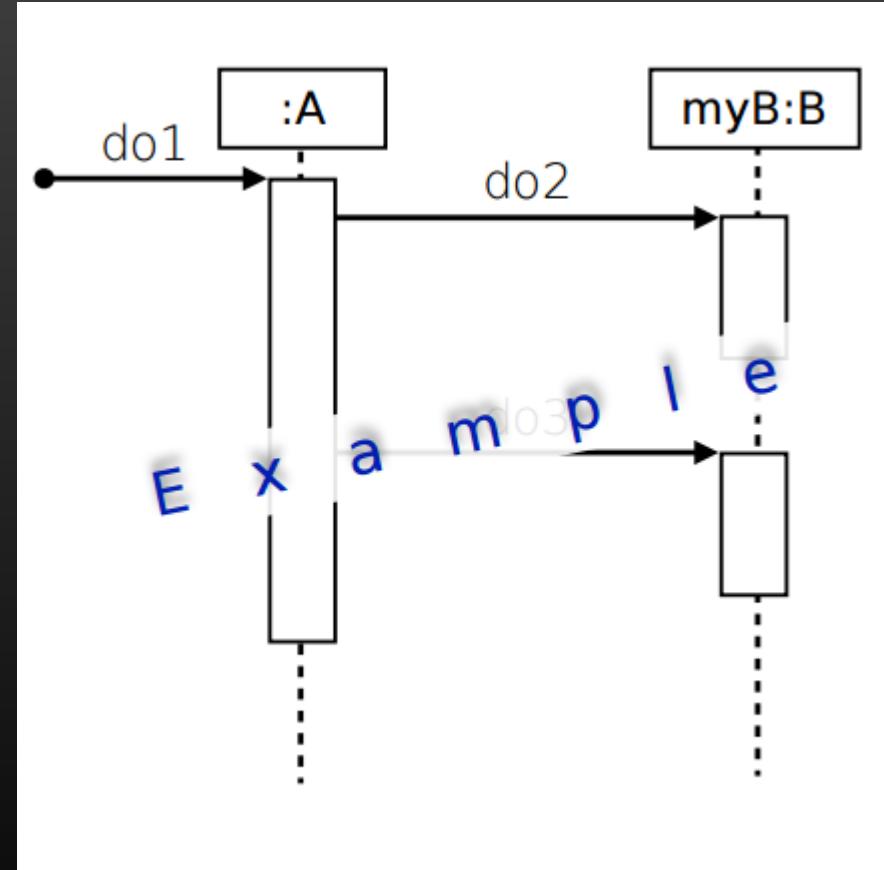
→ Formato alinhado

D. Comunicação

Formato grafo

Diagrama temporal (*timming*)

Diagrama de visão geral da
interação (*interaction overview*)



Quatro tipos de diagramas de iteração disponíveis

D. Sequência

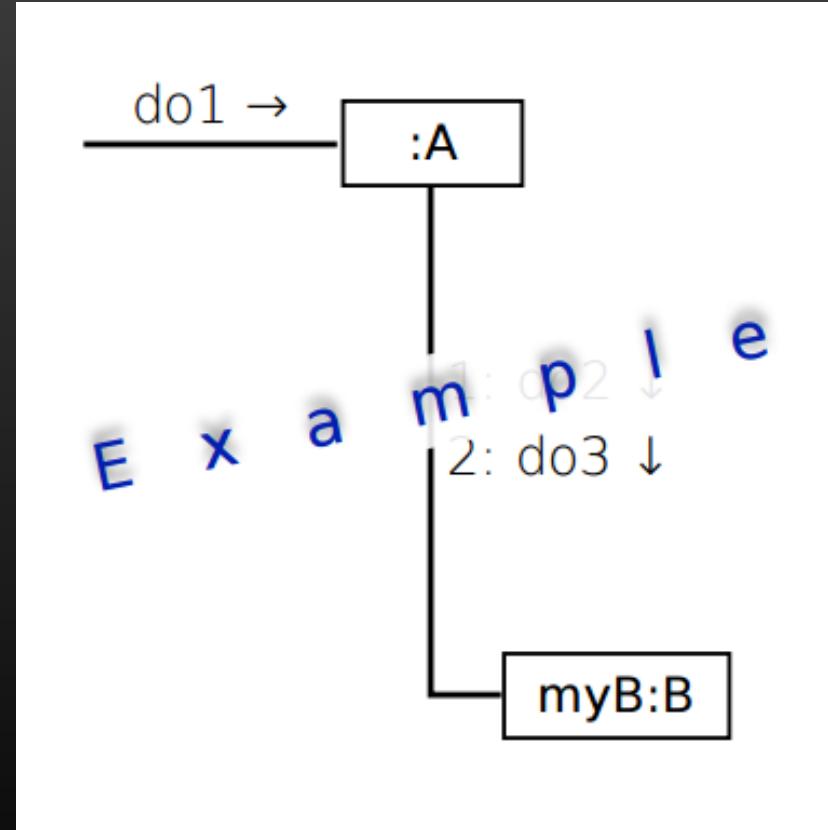
Formato alinhado

→ D. Comunicação

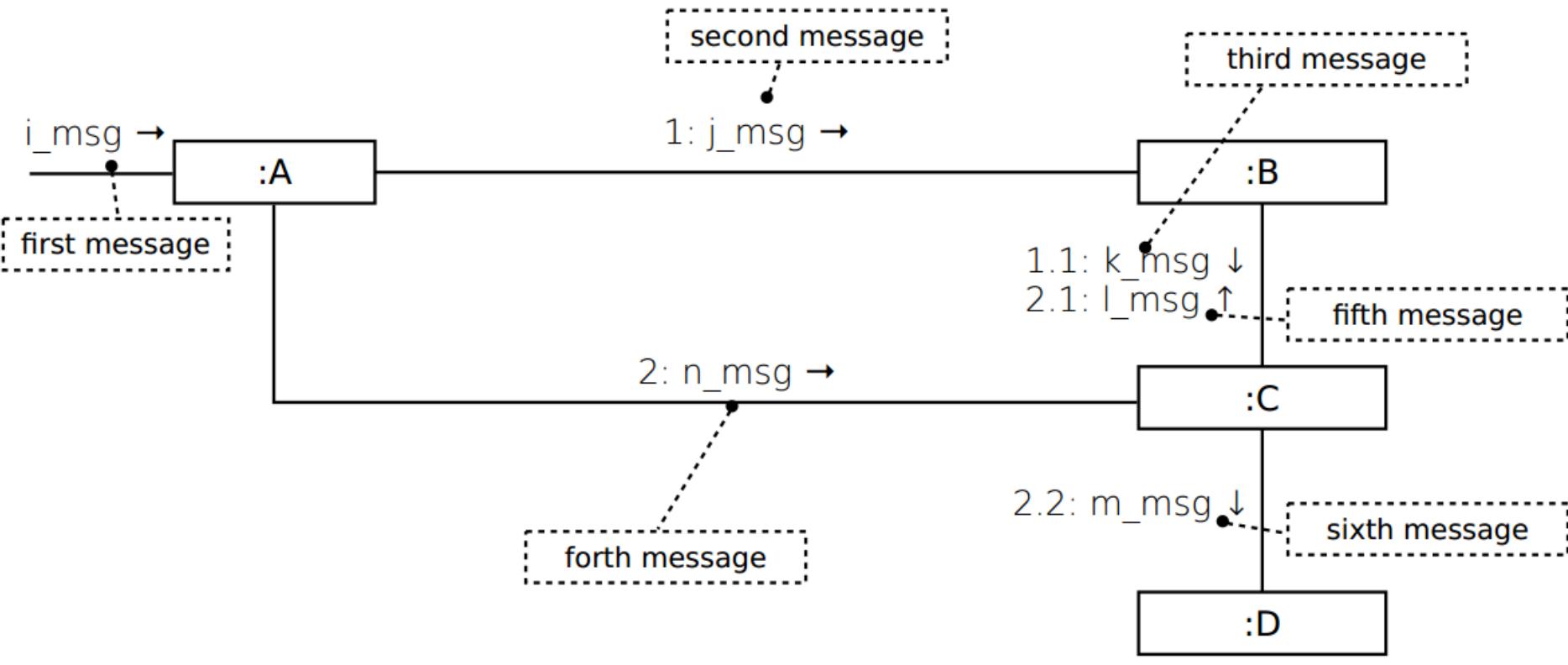
Formato grafo

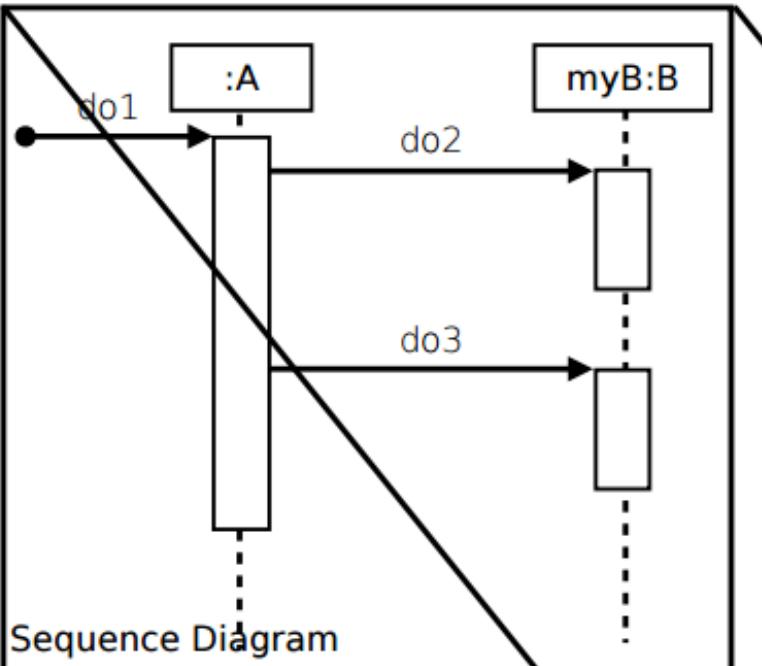
Diagrama temporal (*timming*)

Diagrama de visão geral da interação (*interaction overview*)



Diagramas de comunicação





Os diagramas de podem ser usados para visualizar a colaboração entre objetos em Java.

```
public class A {  
    private B myB = ...;  
  
    public void do1() {  
        myB.do2();  
        myB.do3();  
    }  
}
```

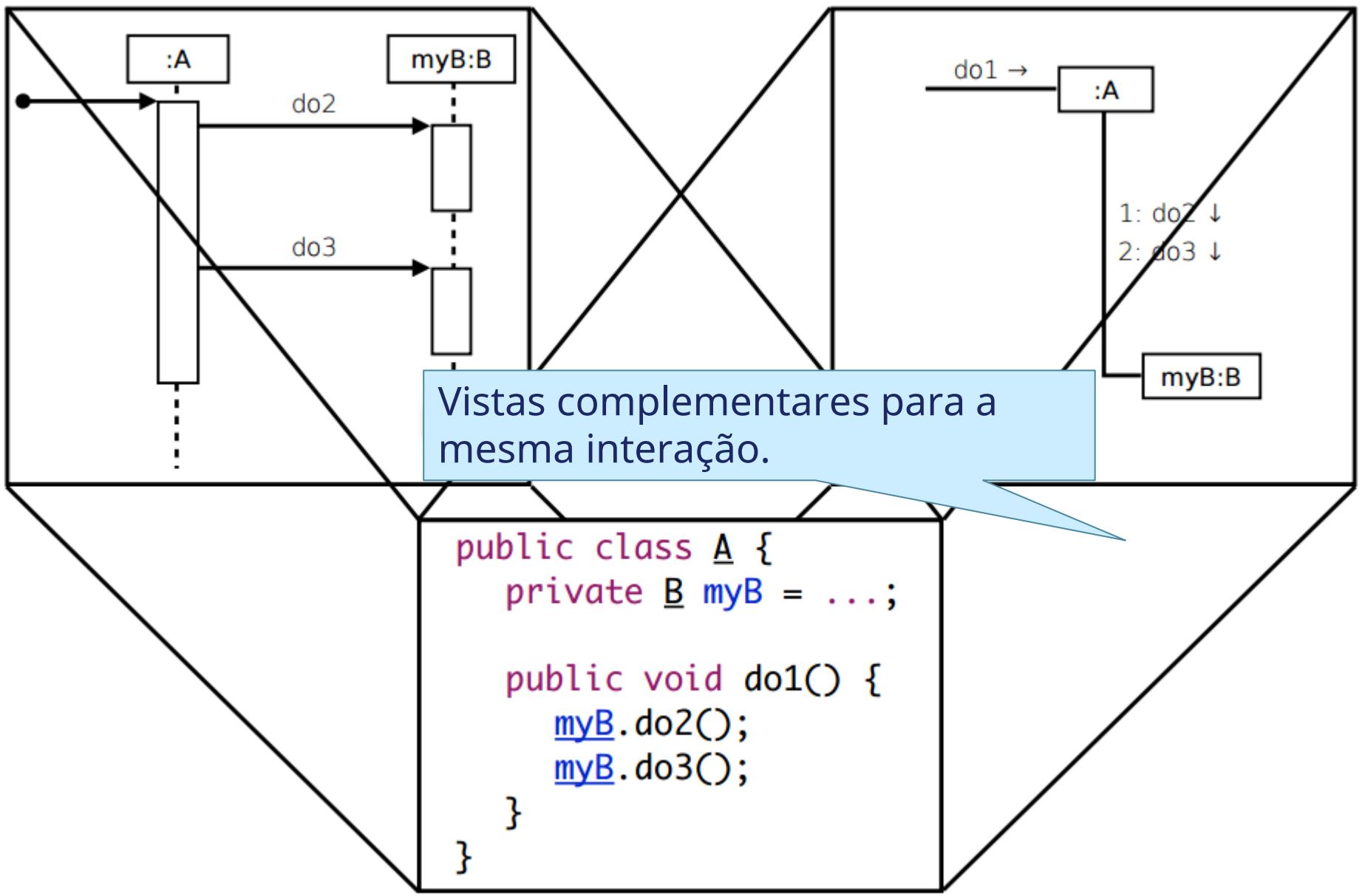


Figure Communication Diagram for a Scenario of the Make Patient Appt Use Case

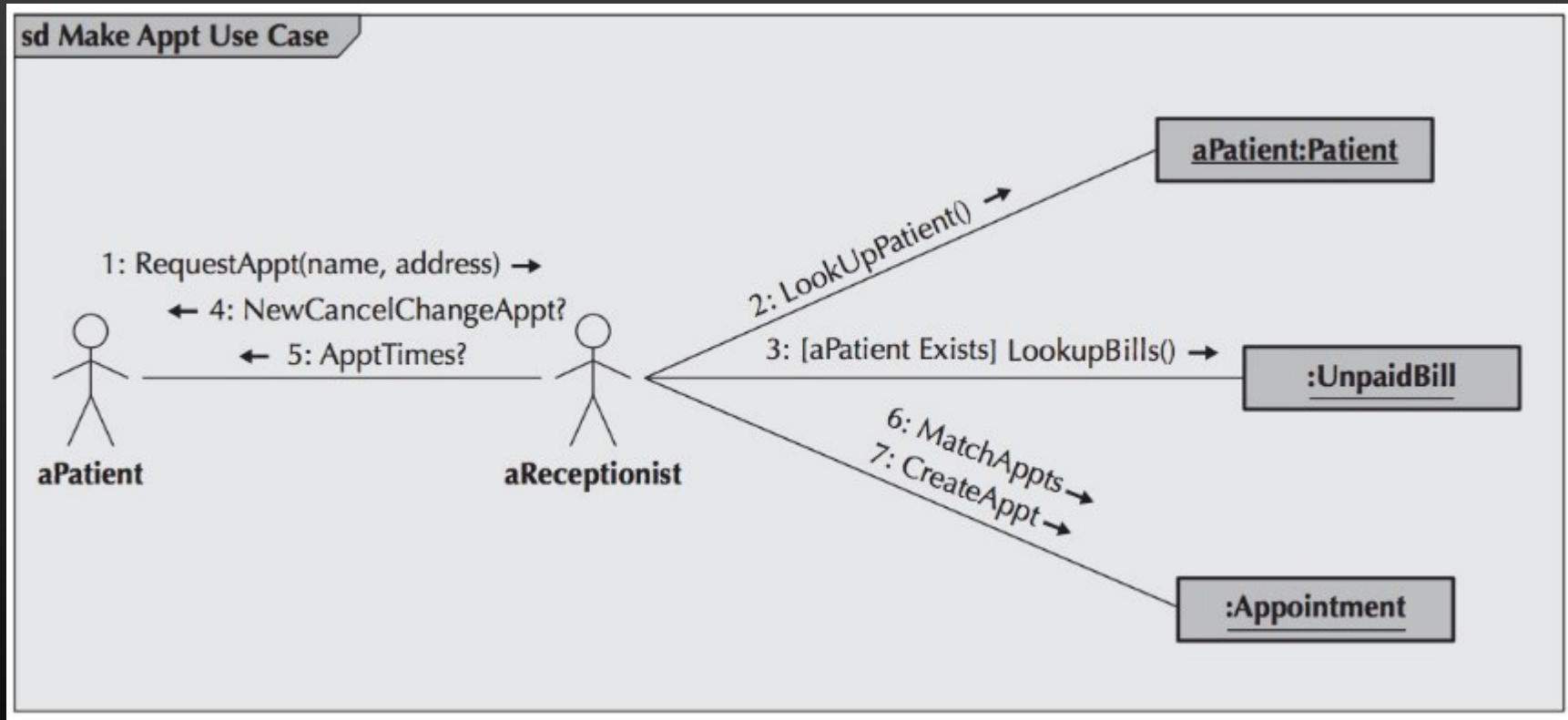
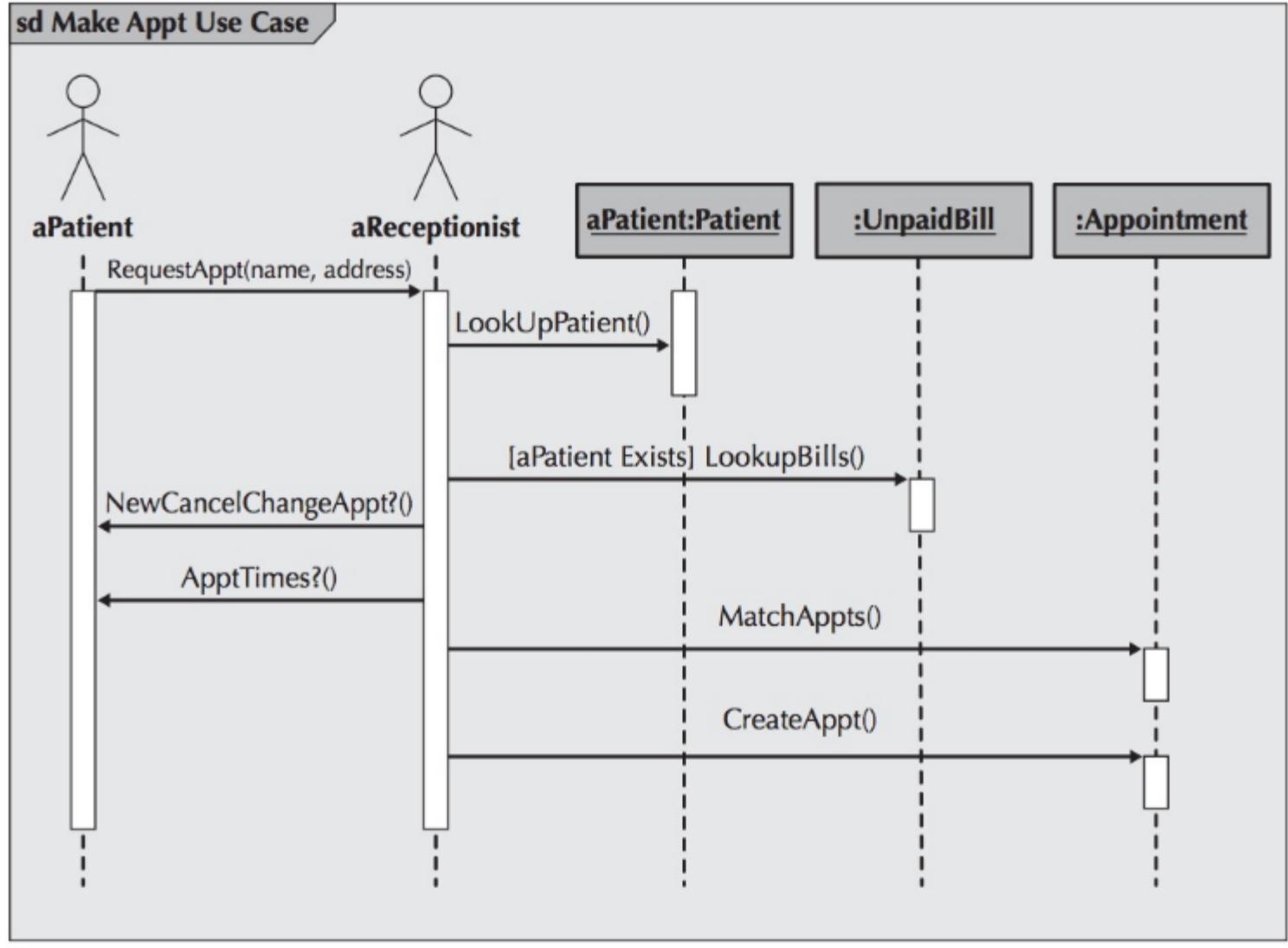


Figure Sequence Diagram for a Scenario of the Make Patient Appt Use



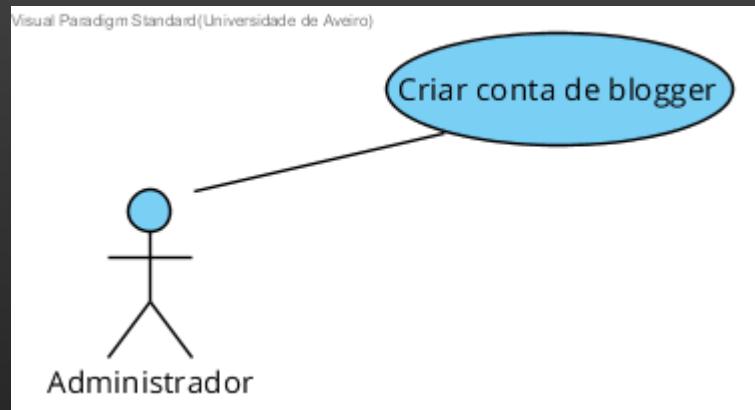
Diagramas de sequência vs diagramas de comunicação

	Benefícios	Limitações
Diagrama de sequência	<ul style="list-style-type: none">• Mostra claramente a sequência/ordem temporal das mensagens• Possibilidades de notação alargadas	<ul style="list-style-type: none">• Cresce para a direita à medida que se acrescentam objetos
Diagrama de comunicação	<ul style="list-style-type: none">• Mais fácil de desenhar (objetos podem ser adicionado em qq parte)	<ul style="list-style-type: none">• Menos expressivo (ordem temporal)• Menos opções de notação• Pouco suportado nas ferramentas UML

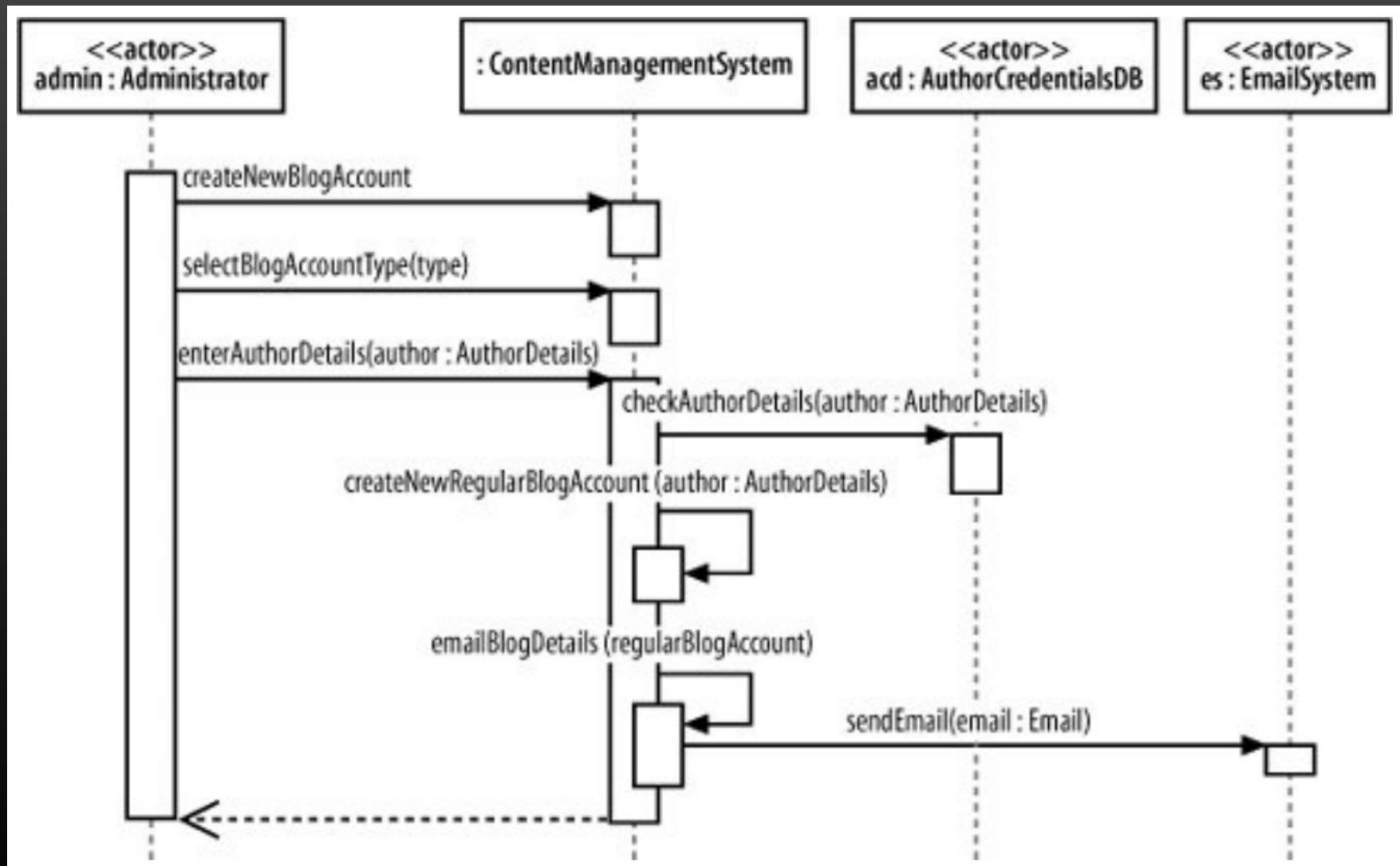
Um caso de utilização é **realizado pela colaboração entre atores e sistema**

FLUXO TÍPICO

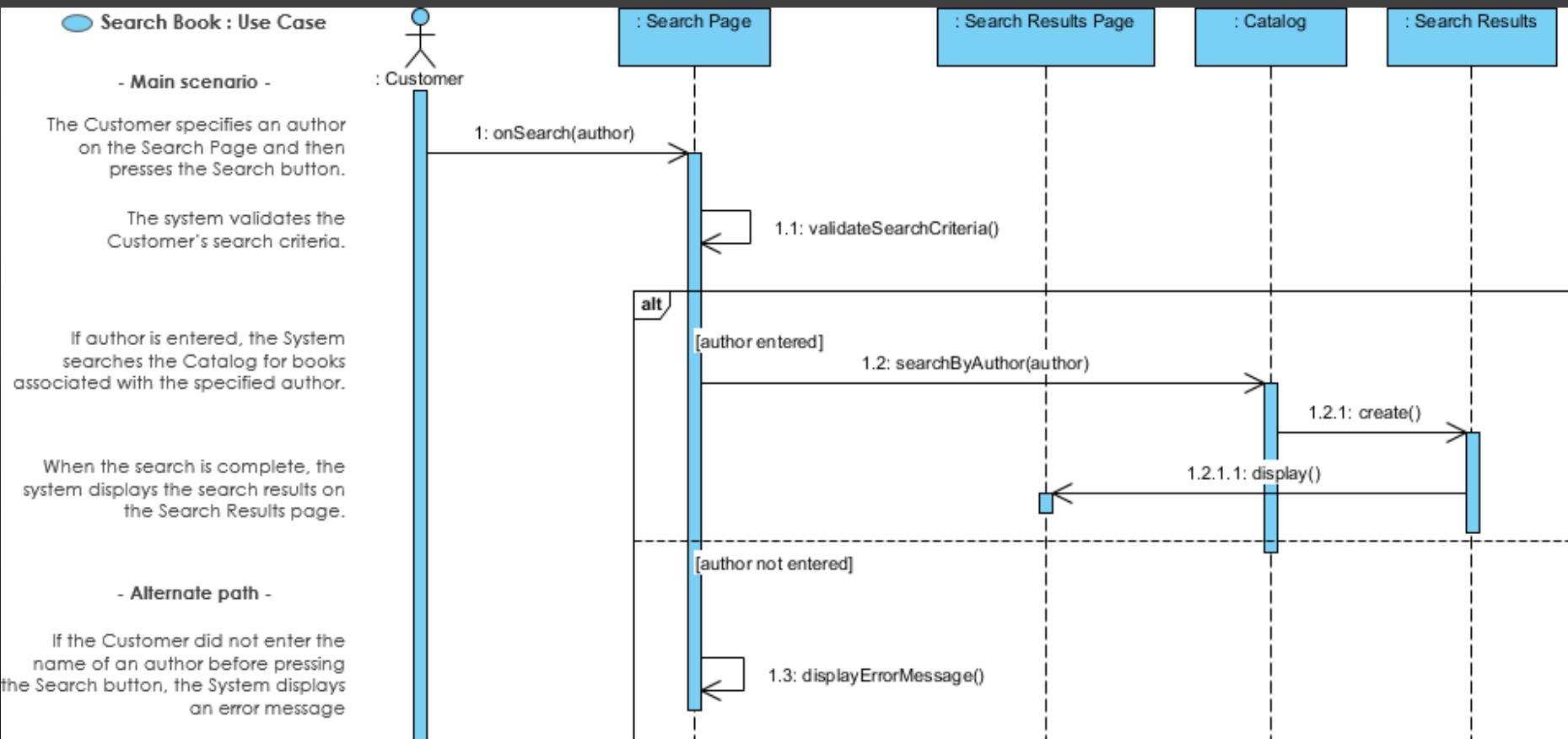
1. O Administrador pede ao sistema (CMS) para criar uma nova conta de *blogger*
2. O Administrador escolhe o tipo Normal
3. O Administrador fornece os detalhes do autor do blog
4. O CMS verifica os detalhes (se já existe) na base de Dados de Autores
5. O CMS cria a nova conta normal.
6. Um sumário dos detalhes da nova conta são enviados por email ao autor.



Realização do caso de utilização como uma colaboração entre objetos



A realização de um caso de utilização



Dos casos de utilização para o sistema (método Larman)

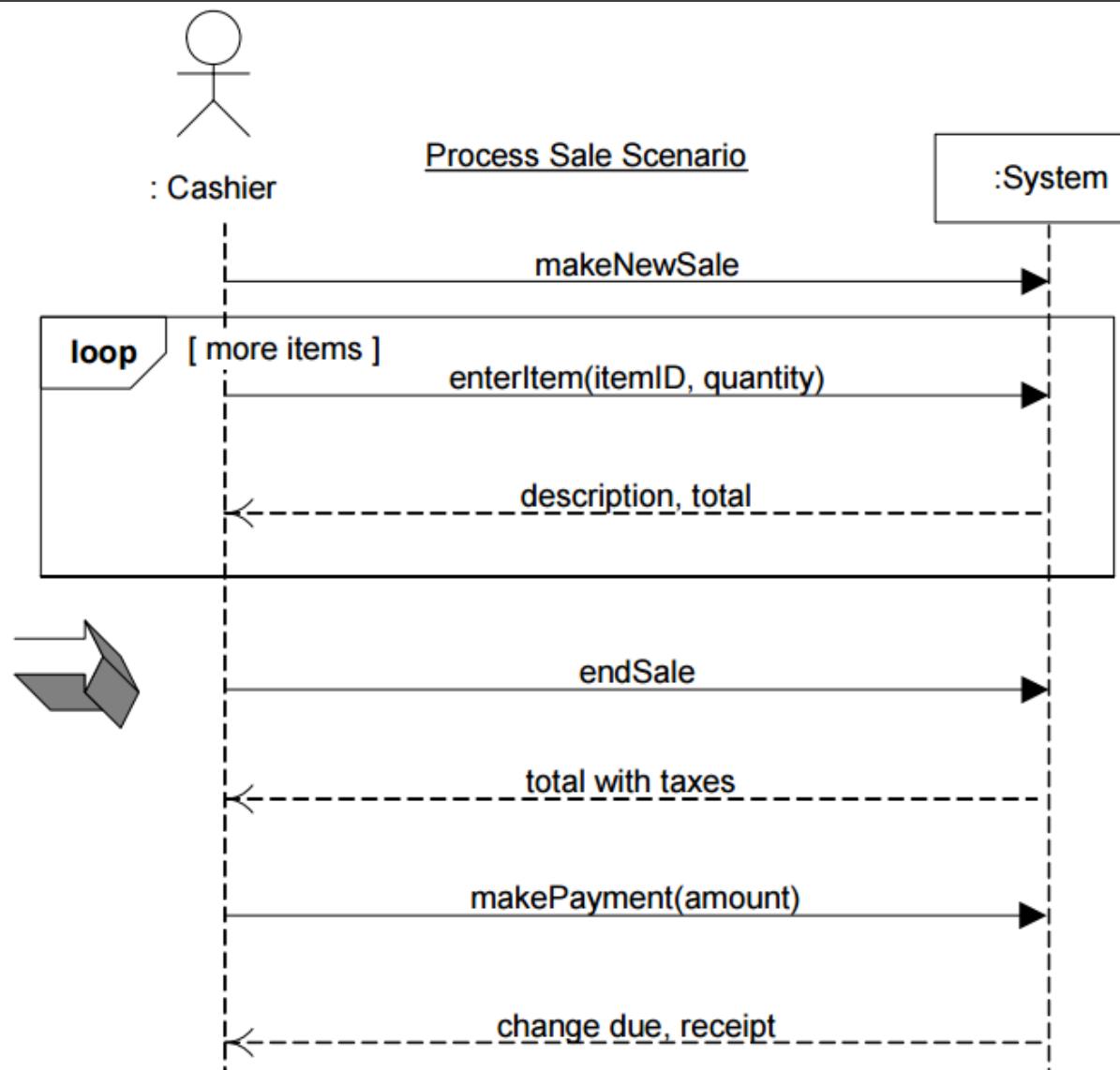
Simple cash-only Process Sale scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.

...



Diagramas de sequência de sistema (DSS)

Mostram, para um cenário de um CaU:

- os eventos que os atores externos geram,
- a sua ordem temporal,
- as necessidades de integração entre sistemas

Estilo para a construção

O sistema é tratado como uma caixa-fechada

- Não mostra componentes internos

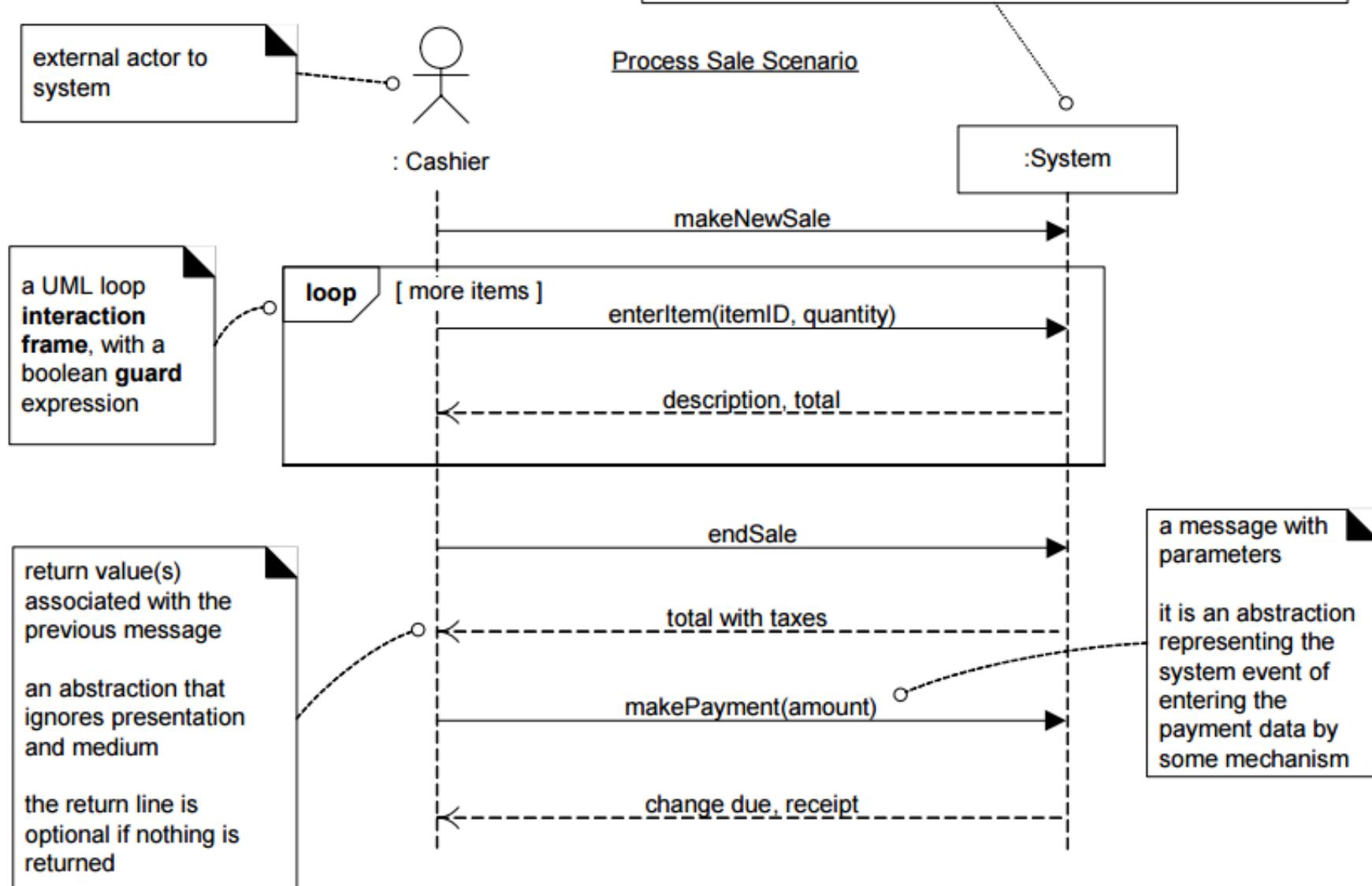
Mensagens que chegam ao Sistema são modeladas como operações

- Os serviços solicitados/invocados pelos atores

system as black box

the name could be "NextGenPOS" but "System" keeps it simple

the ":" and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML



um novo diagrama de sequência de sistema (DSS) p/ cada CaU

uma *lifeline* para o actor primário (ou actores) e o sistema.

O sistema é modelado por uma classe que o representa globalmente

Opcionalmente, pode-se incluir o texto da descrição do CaU



Os DSS mostram operações de sistema

Uma **operação de sistema** (OpS) corresponde a um ponto de entrada no sistema

Encapsula um conjunto de interações subjacentes entre objetos, necessárias para realizar essa operação.

Etapa seguinte na construção do modelo dinâmico:

mostrar, para cada OpS, a rede de objetos que vai ser ativada.

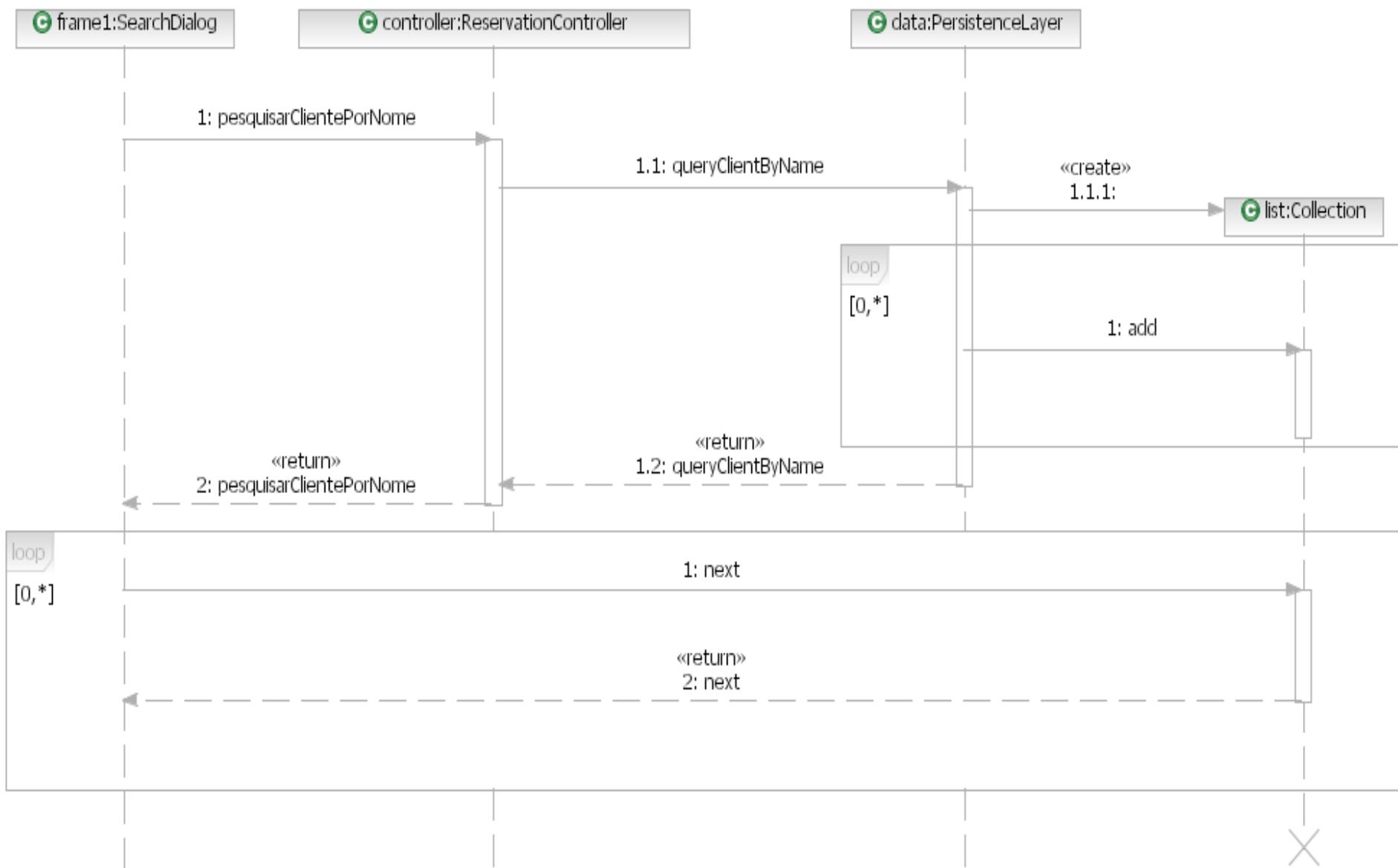
Estes objetos são instâncias de classes que devemos identificar

Como escolher as classes que participam na solução?

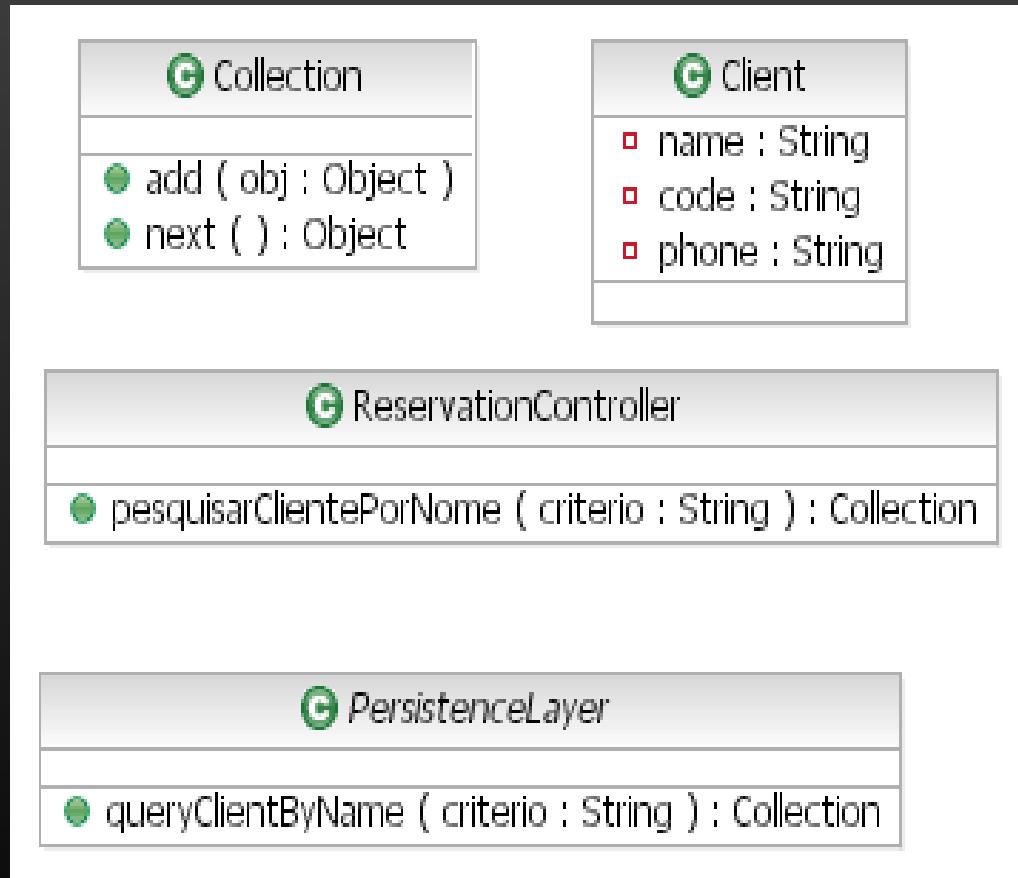
Essa atividade é o que designamos por desenho

Ponto essencial: **distribuir responsabilidades pelos objetos**

seguindo os princípios do desenho por objectos (*Object-Oriented Design*).

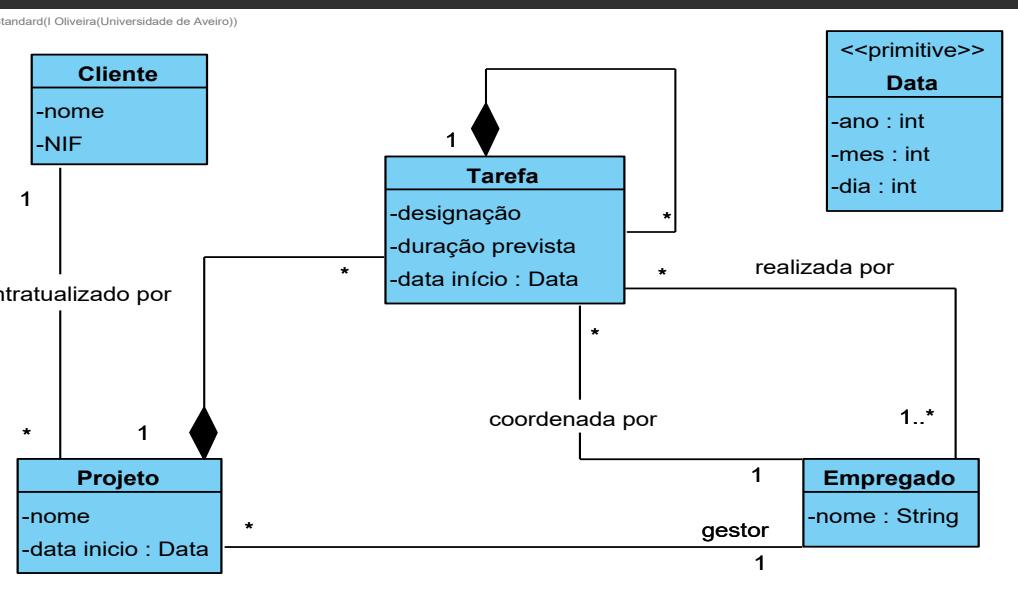


Os diagramas de interacção ajudam a distribuir responsabilidades → encontrar os métodos das classes



Distribuição de responsabilidades pelos objetos

Visual Paradigm Standard (Oliveira/Universidade de Aveiro)



checkIn() / checkOut()

Começar a trabalhar numa tarefa / terminar

reportarEstadoProjeto()

calcula a % de execução do projeto (tarefas já concluídas / previstas)

formatarISO8601()

Representa uma data no formato aaaa/mm/dd

atribuirGestor(Empregado emp1)

Define o gestor de um projeto

contaCorrente()

Lista os movimentos lançados para um cliente, no presente ano fiscal.

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. 6

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

→ chap. 10, chap. 15.

41951- ANÁLISE DE SISTEMAS

Arquitetura do software e a UML

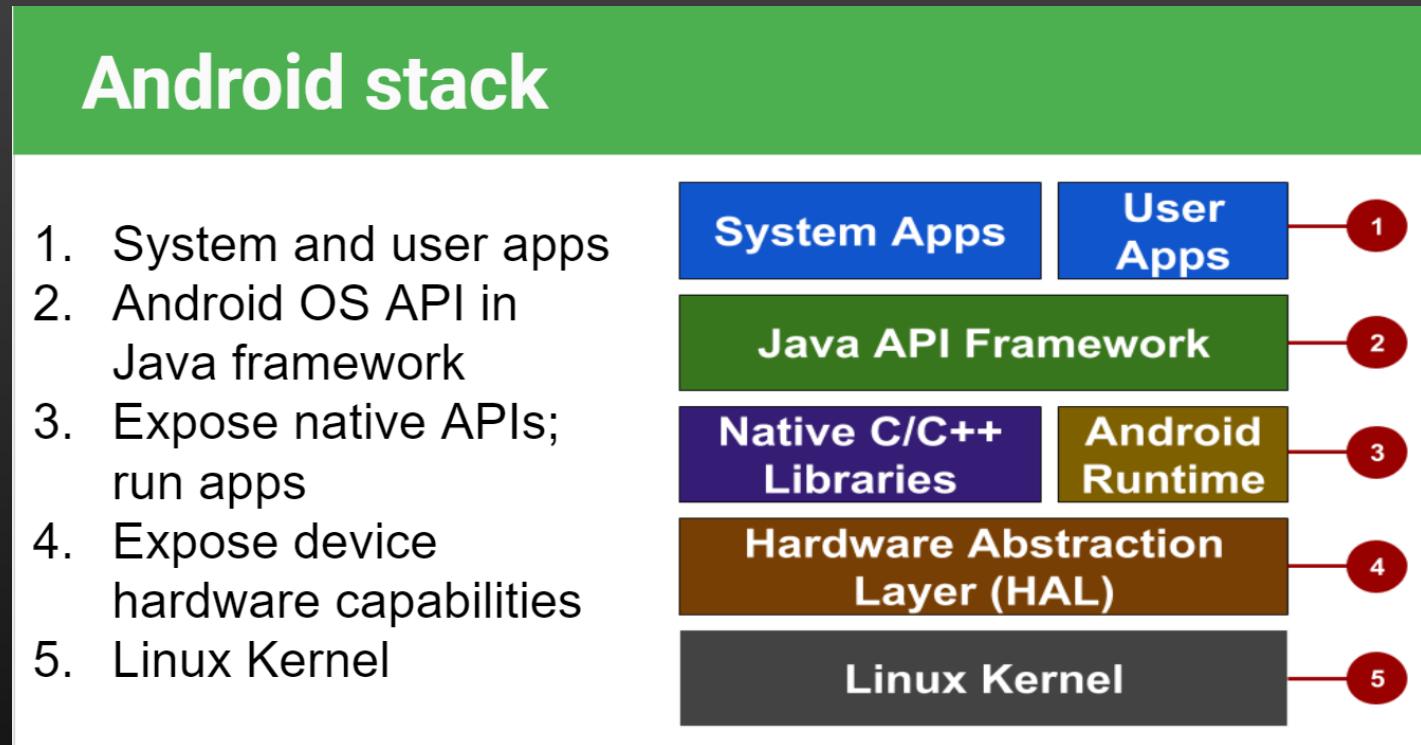
- D. de pacotes, components e de instalação

Ilídio Oliveira | v2022/03/29

Objetivos de aprendizagem

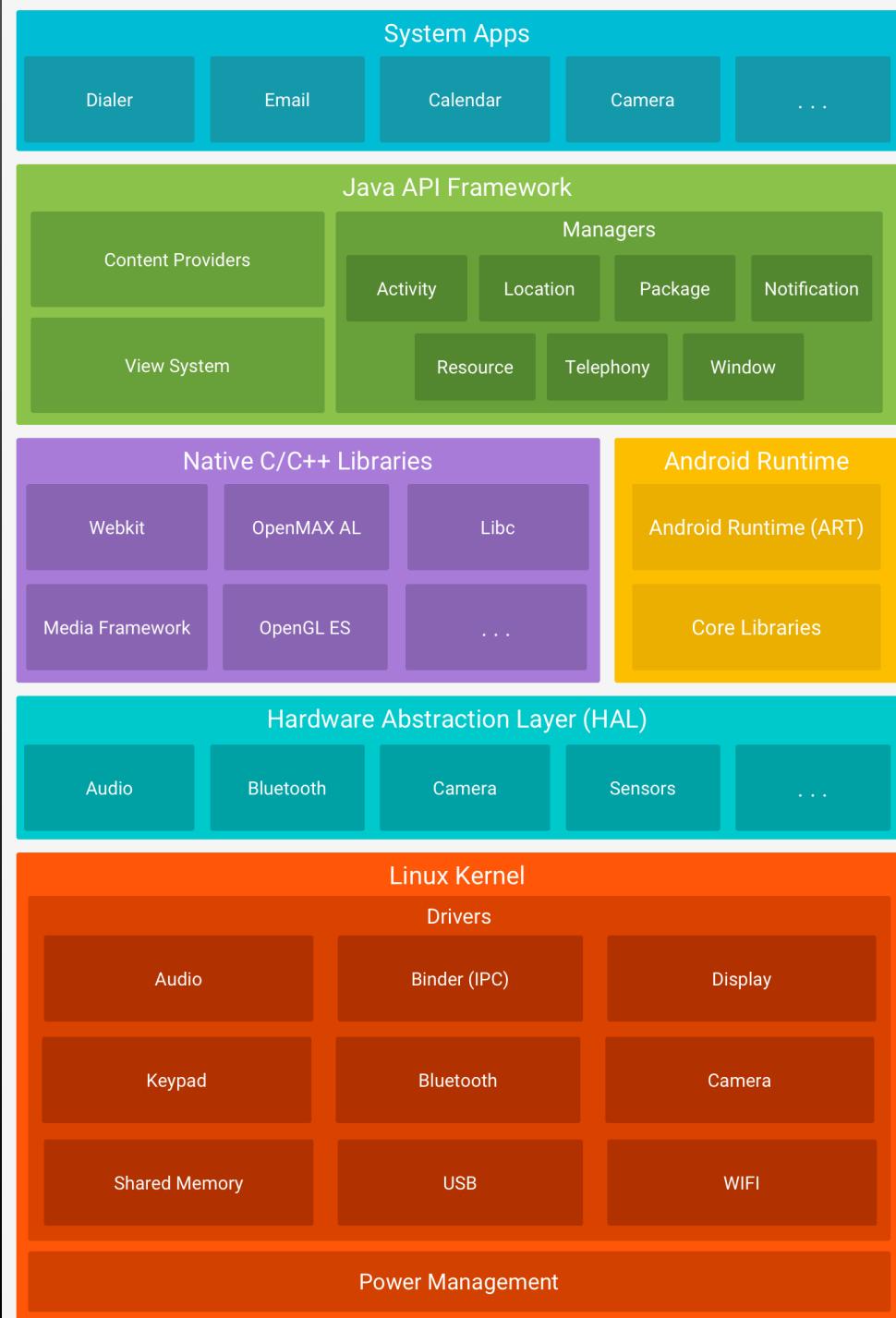
- Explicar as atividades do desenvolvimento associadas à arquitetura de software
- Definir a prática de Arquitetura Evolutiva, conforme especificado no OpenUP
- Identificar os elementos abstratos de uma arquitetura de software
- Identificar requisitos de arquitetura significativos num domínio e relacionar 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)



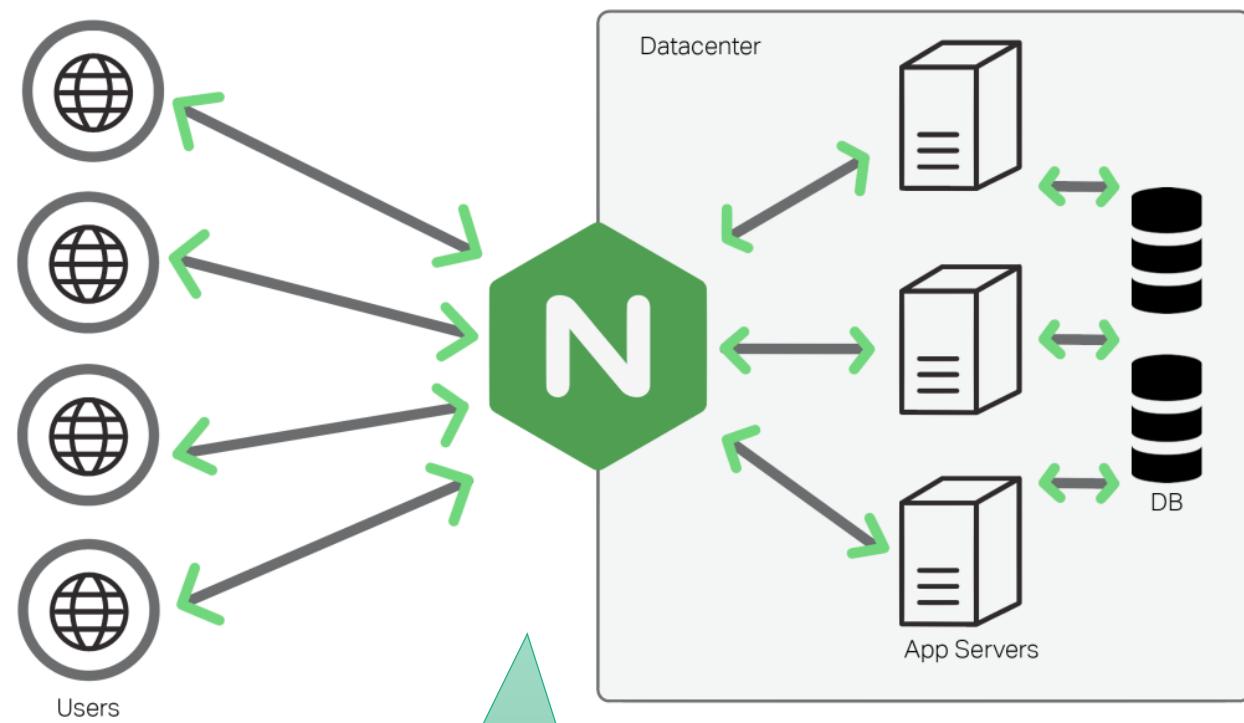
Um exemplo de arquitetura: a organização do sistema Android.

Android stack



Módulos em cada camada
(partições).

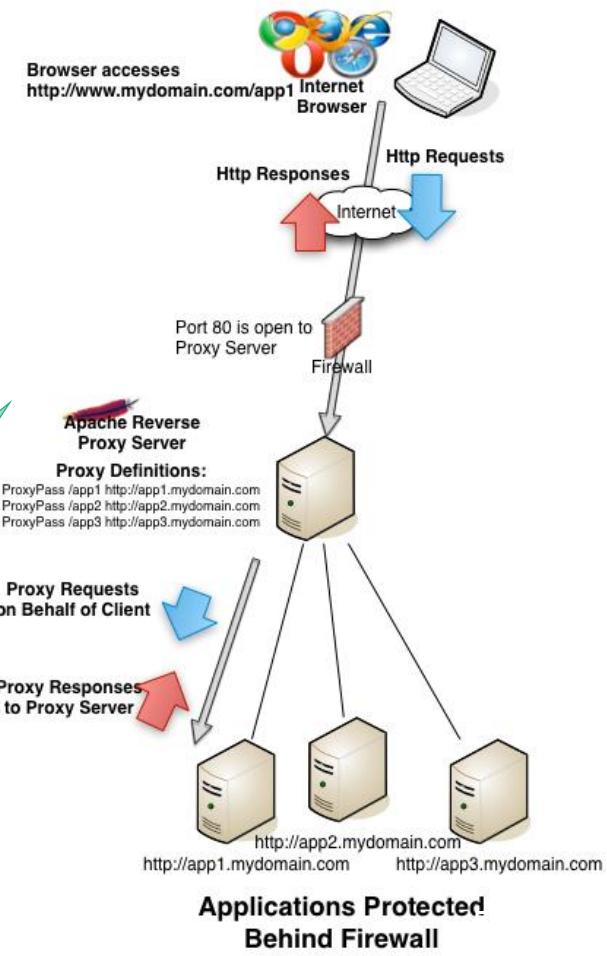
I Oliveira



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

Organização dos serviços de rede numa configuração de "reverse proxy"

Reverse Proxy Diagram



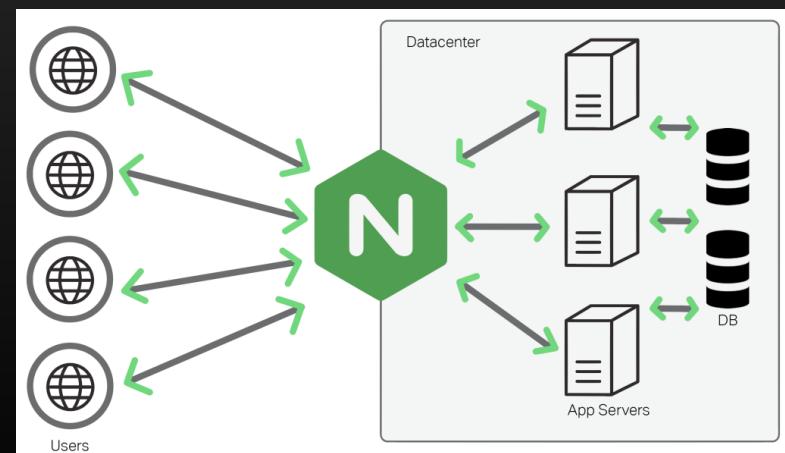
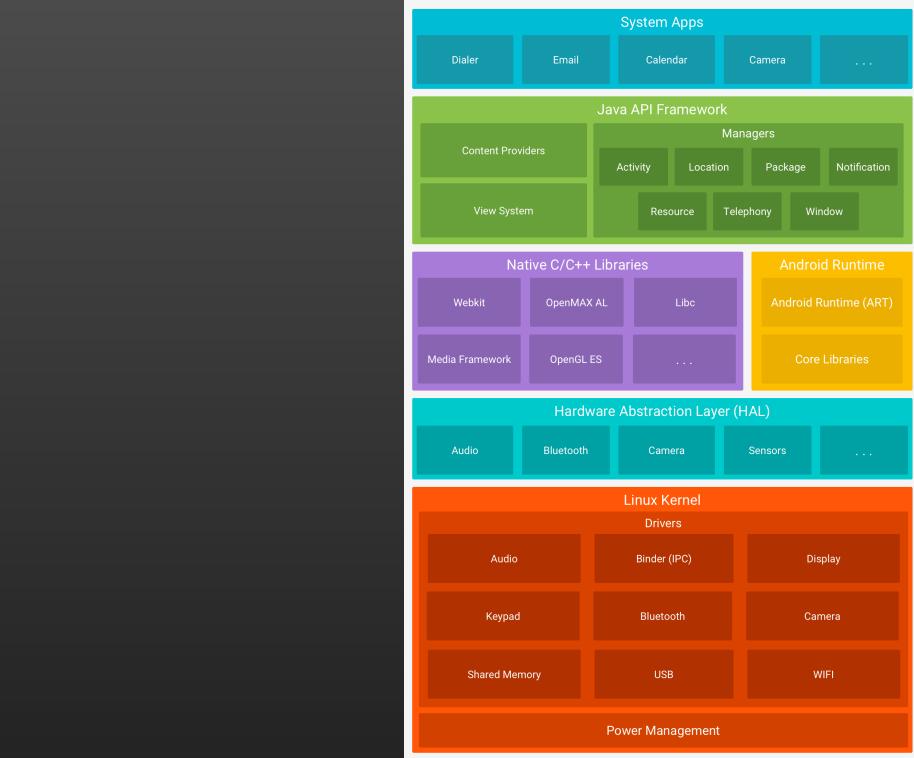
Elementos comuns

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)



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 Ar

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

Arquitetura de Software

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

...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

Plano (decisões) com a estratégia para a implementação.



Who Needs an Architect?

Outra maneira de colocar a questão...

In the words of Ralph Jordan:

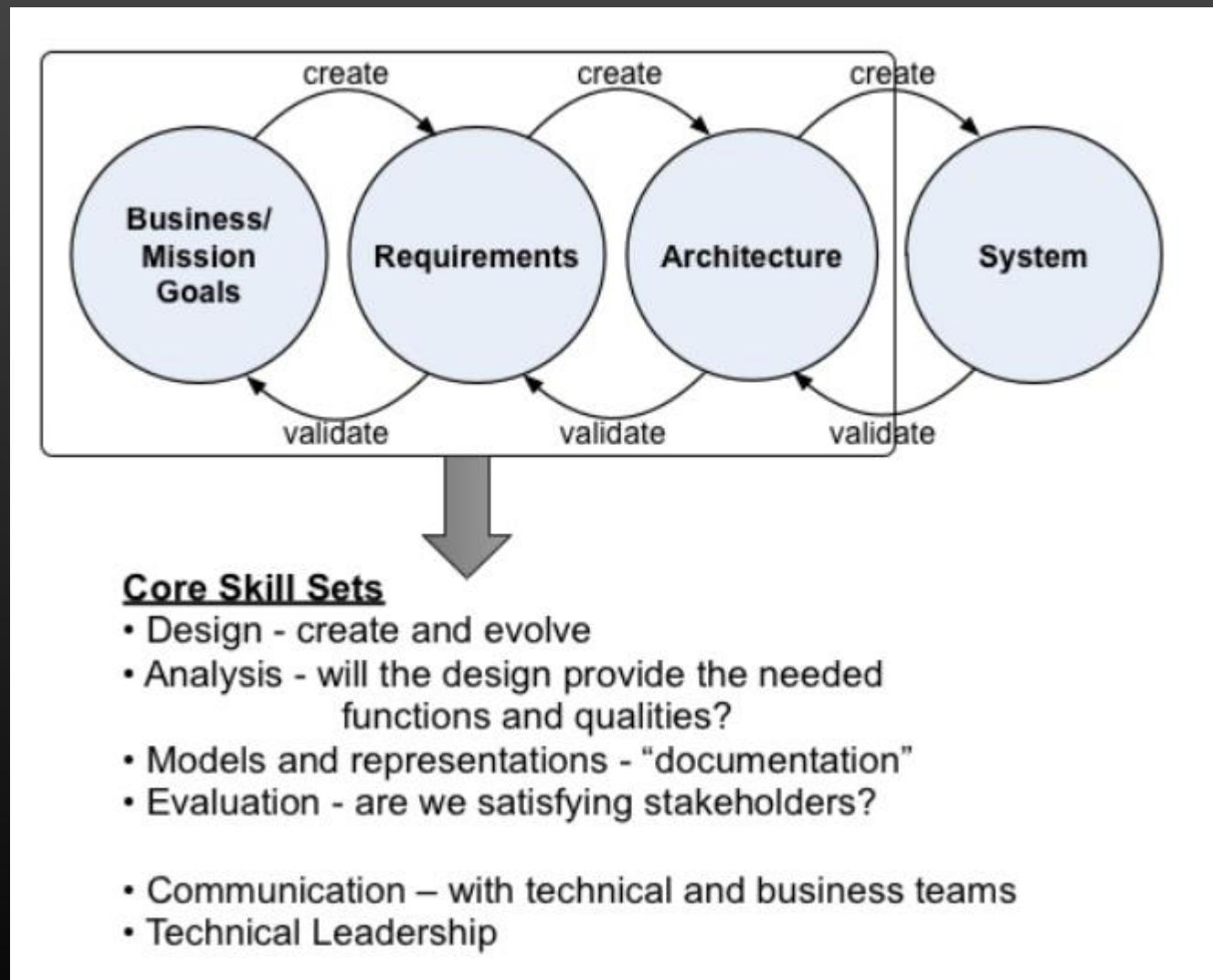
So, a better definition would be "In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.' This understanding includes how the system is divided into components and how the 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."

Martin Fowler

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

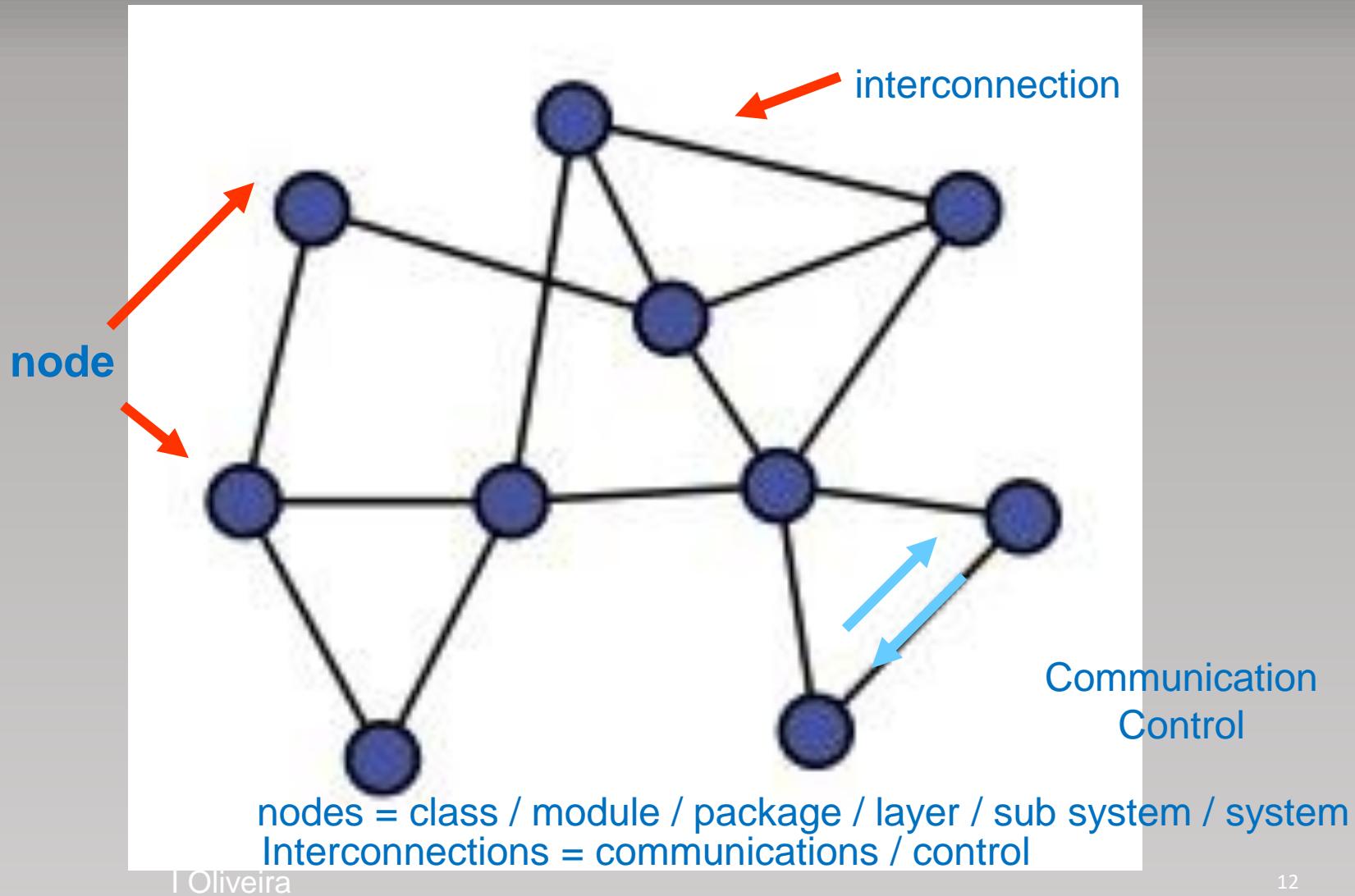
There is another style of definition of architecture which is something like "architecture is the set of design decisions that must be made early in a project." I complain about that one, too, saying that architecture is the decisions that you wish you could get right early in a project, but that you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

Papel do arquiteto (de software)



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

Elementos de uma arquitetura

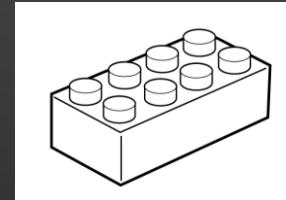


O que compõe uma “arquitetura”?

Elementos

Abstrações/blocos usados na construção do sistema

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

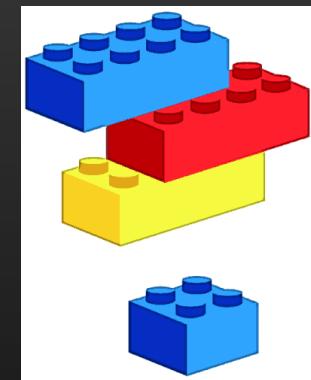


Interações de elementos

interagir através de interfaces públicas

Relacionamentos normalmente são concretizados com interfaces

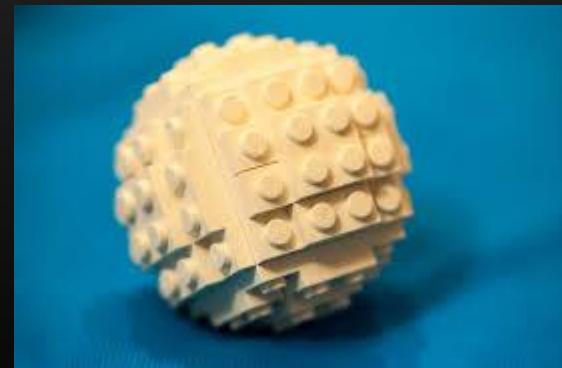
Arquitetura trata as interfaces públicas



Estruturas

Defina a partição estática e atribuição de funcionalidade

Suporta a atribuição de comportamentos

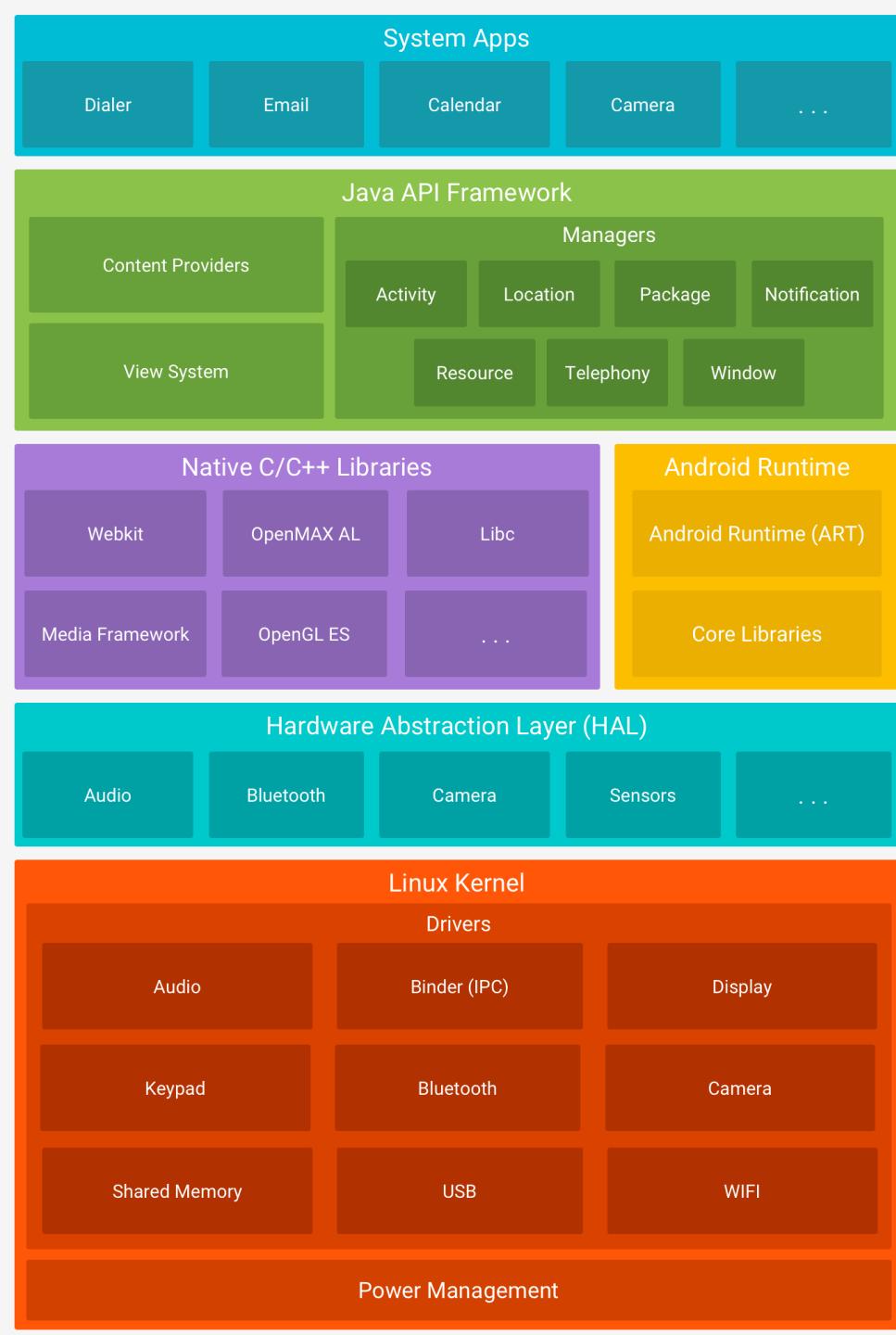


<http://www.columbia.edu/cu/gsapp/BT/BSI/SUSPENSI/suspensi.html>

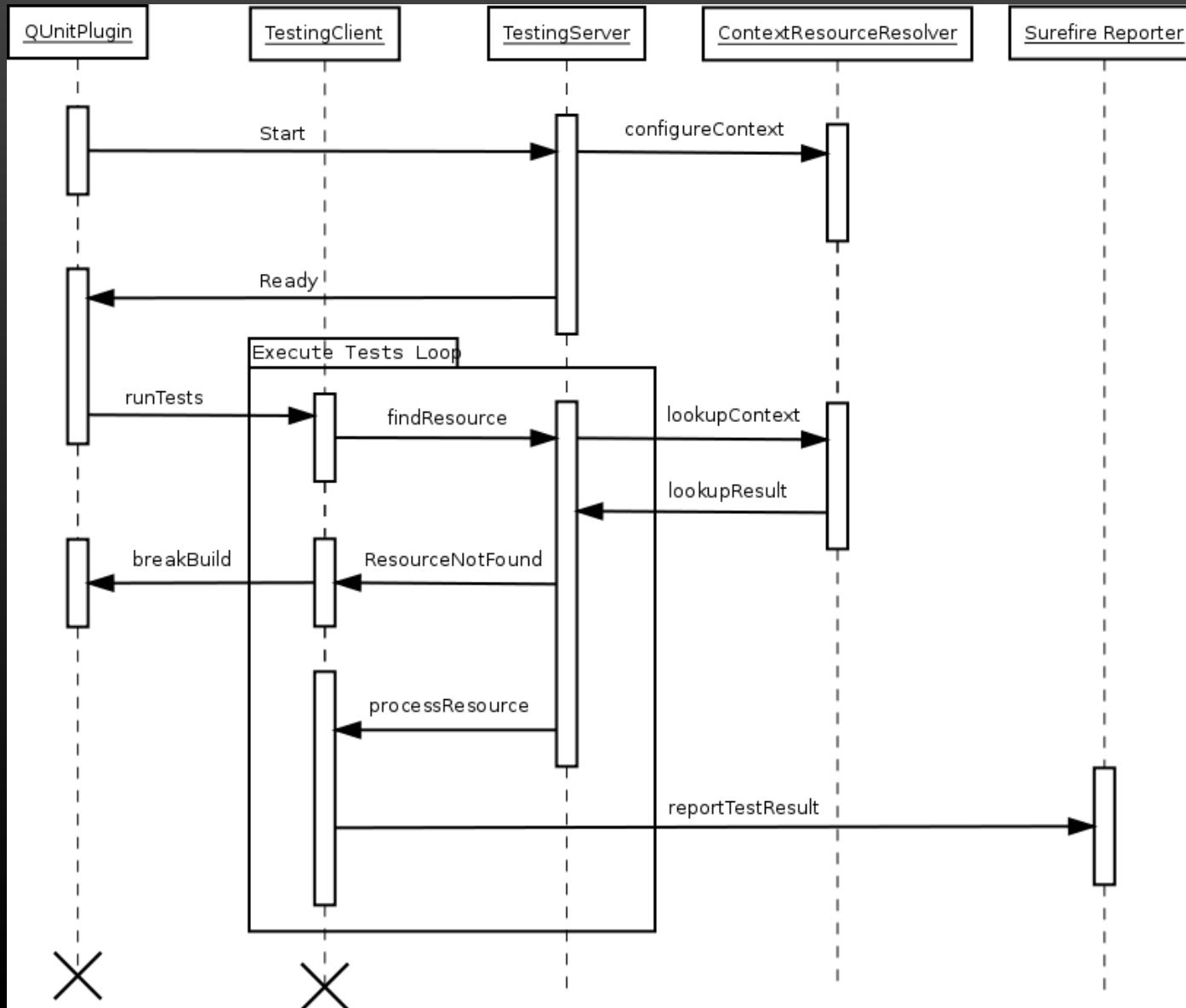
Vista estrutural (as partes constituintes)

Android platform architecture

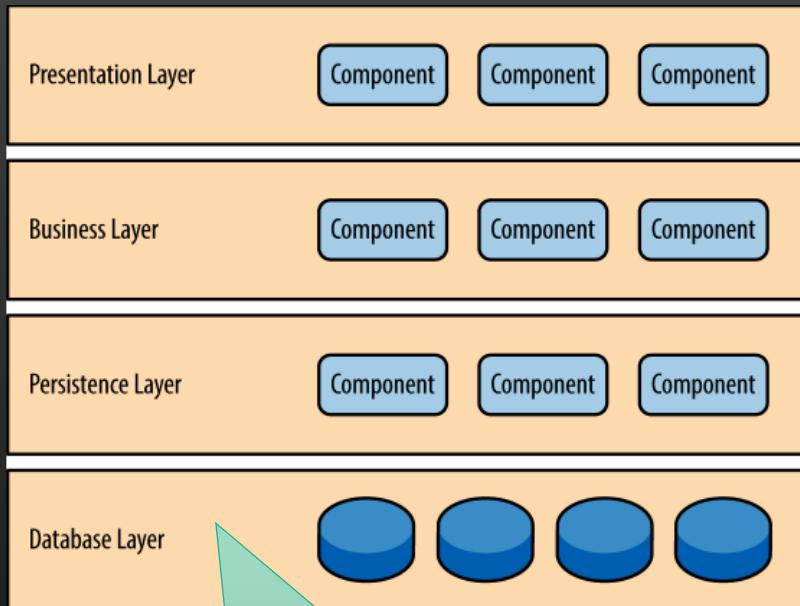
I Oliveira



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



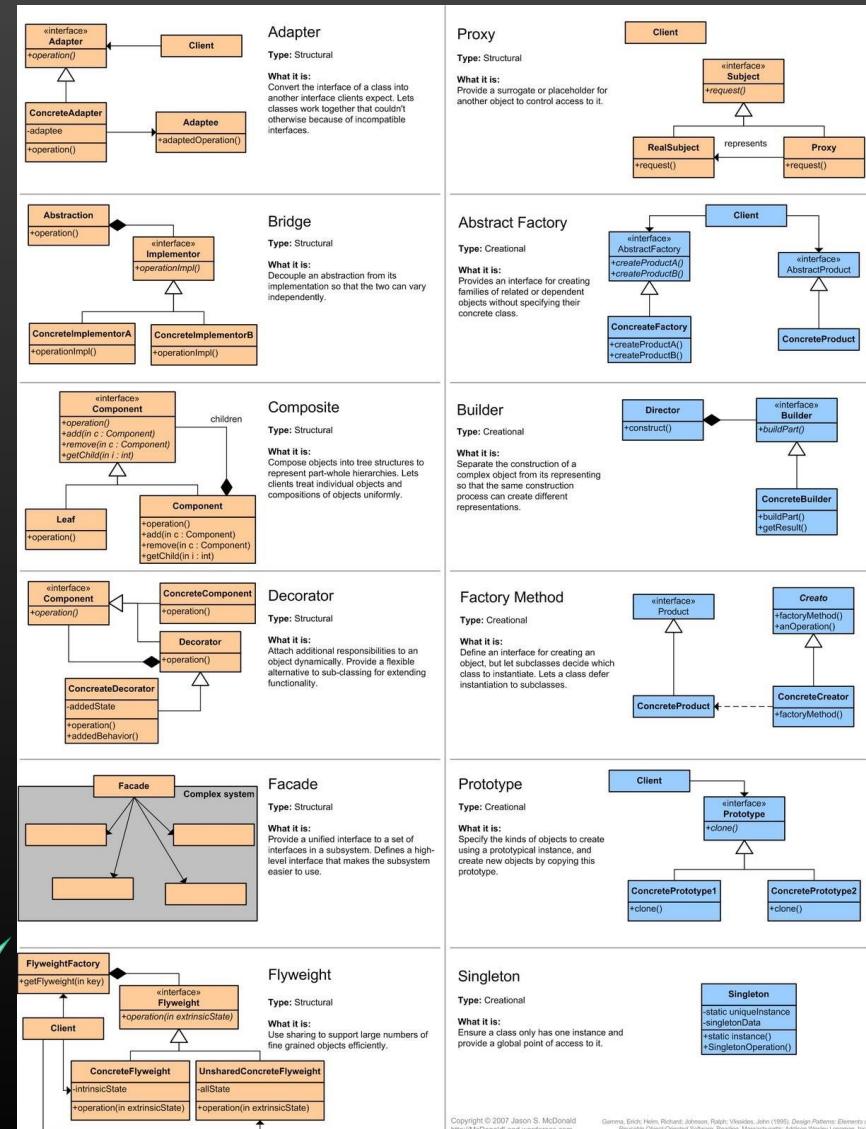
Arquitetura, componentes, classes



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



**A arquitetura é consequência dos
requisitos e estabelece compromissos**

Decisões de arquitetura

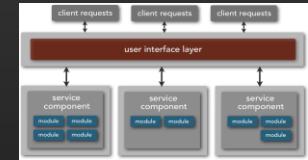
A decisão de usar uma aplicação web como camada de interação da solução



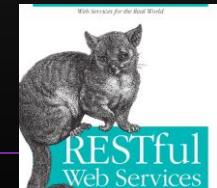
A decisão de usar Java Server Faces para o “web framework” de desenvolvimento



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

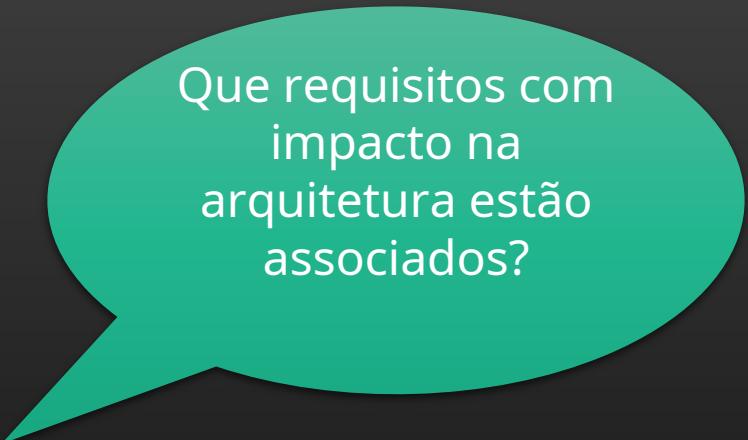
<https://feedzai.com>

- 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



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 faturaçāo (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

Interações 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 evolution, 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 perspetivas de análise

① Arquitetura lógica do software

Organização geral dos blocos de software

Independente da tecnologia de implementação

② Arquitetura de componentes do software

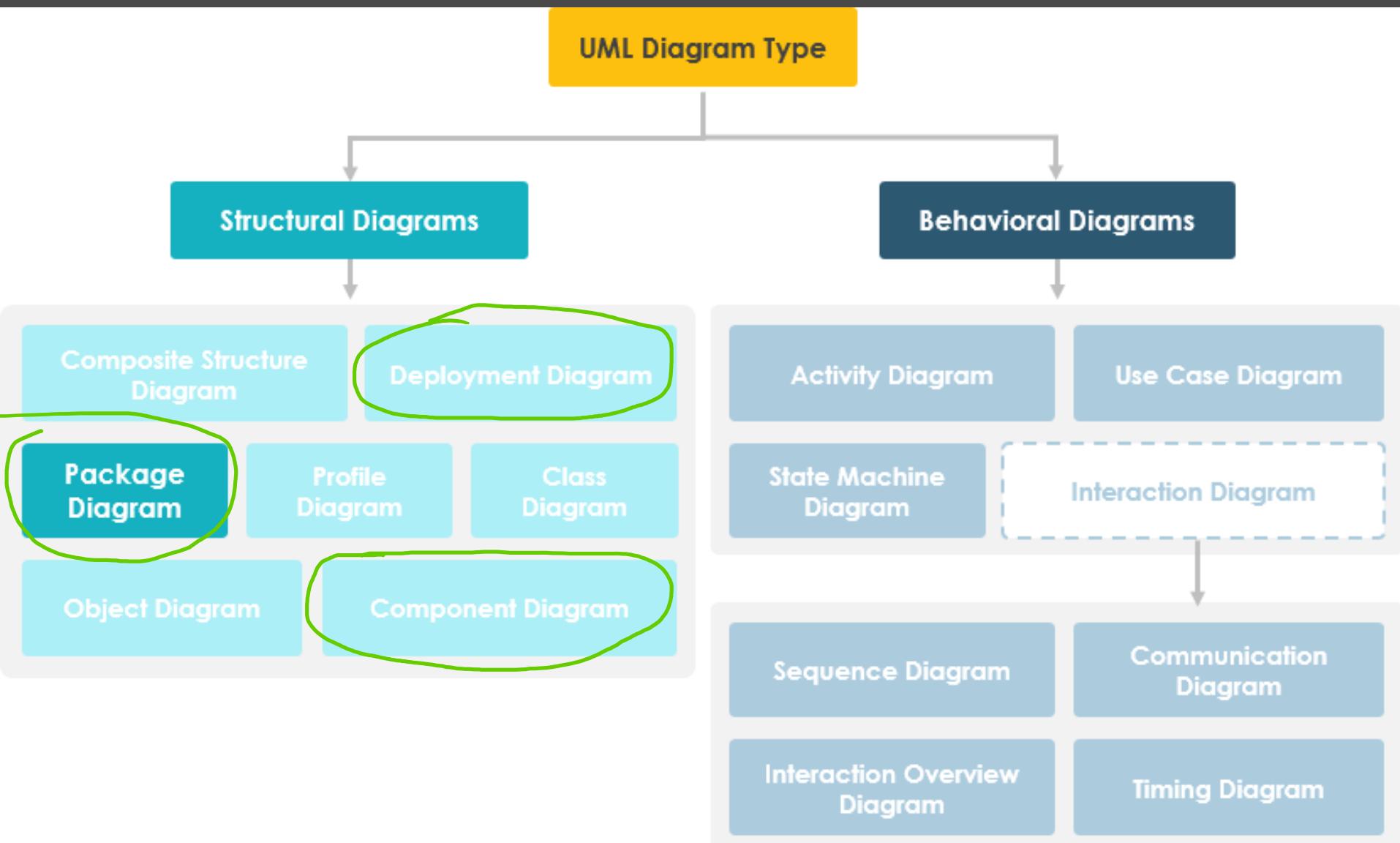
Peças construídas com uma tecnologia concreta

Construção “modular”

- E.g.: existem pré-feitos?

③ Arquitetura de instalação

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



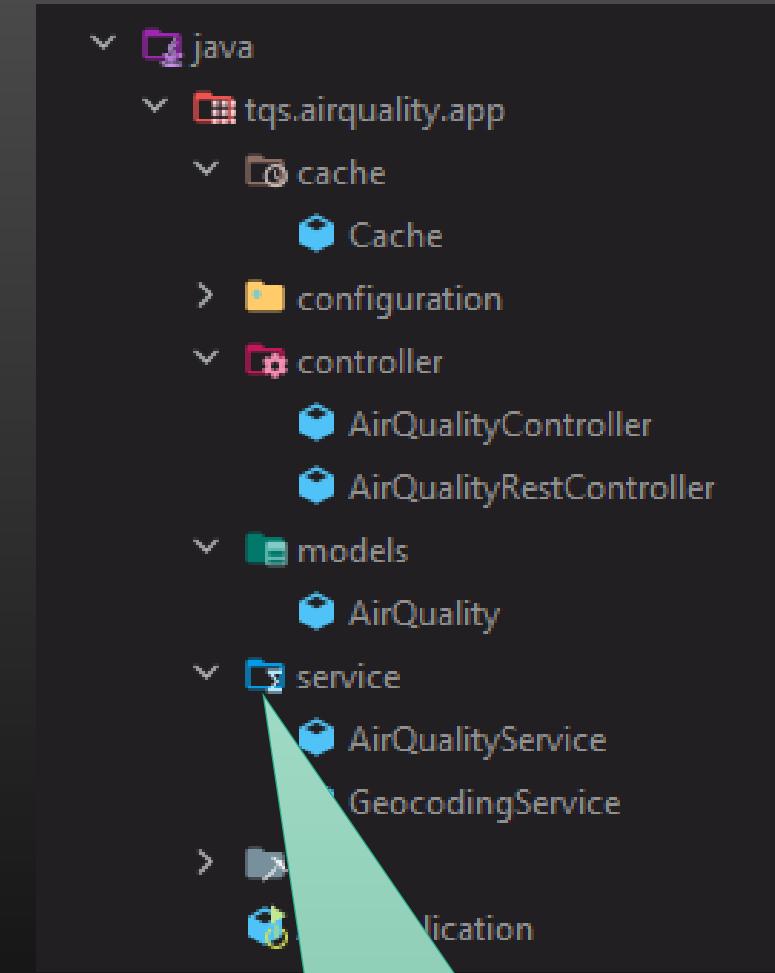
Arquitetura lógica

Organização geral da solução em blocos (*packages*)

Os *packages* podem representar agrupamentos muito diferentes.

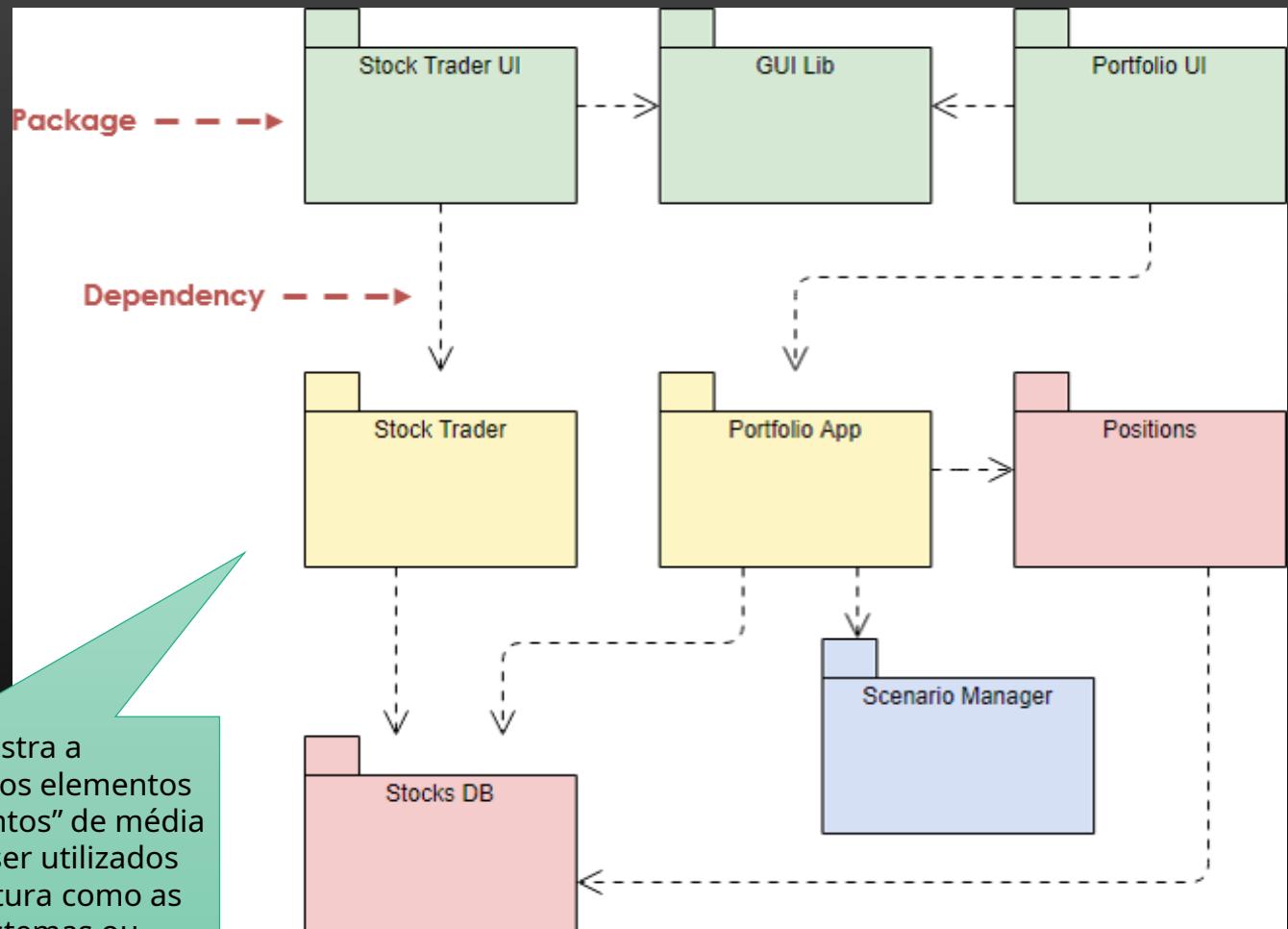
E.g.:

- 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

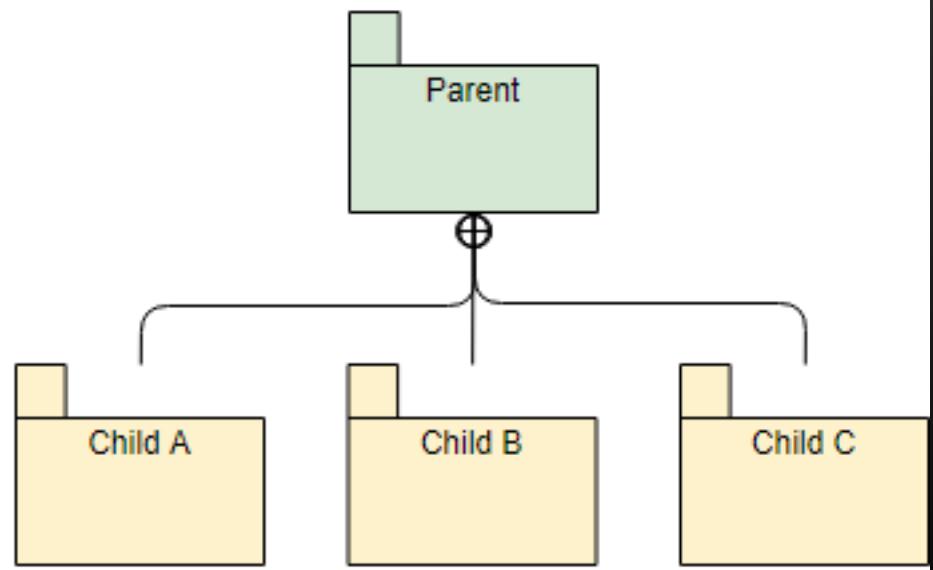
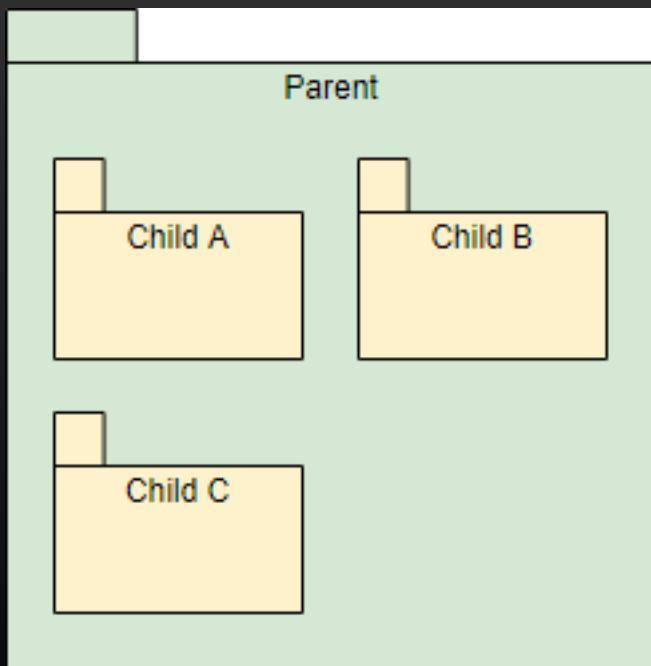


O diagrama de pacotes mostra a disposição e organização dos elementos do modelo em “agrupamentos” de média a larga escala que podem ser utilizados para mostrar tanto a estrutura como as dependências entre sub-sistemas ou módulos.

Comunicar a arquitetura lógica com pacotes (packages)

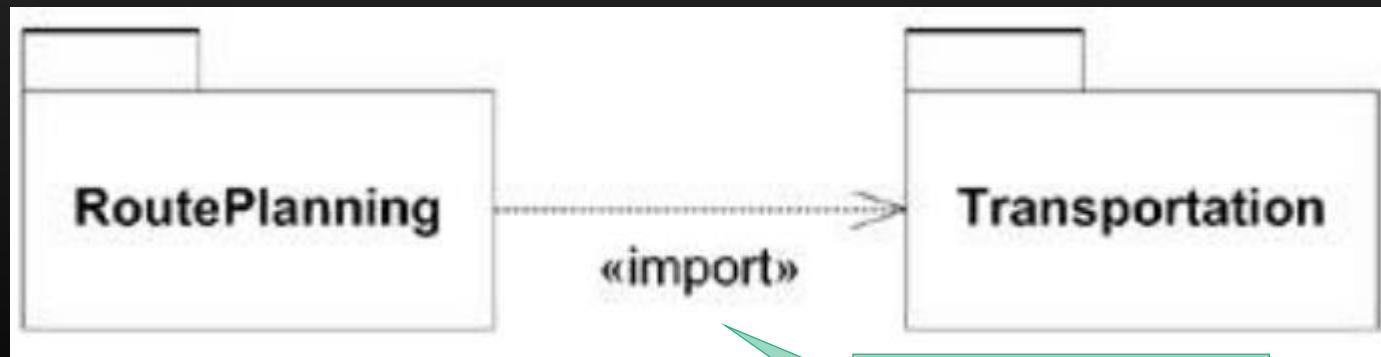
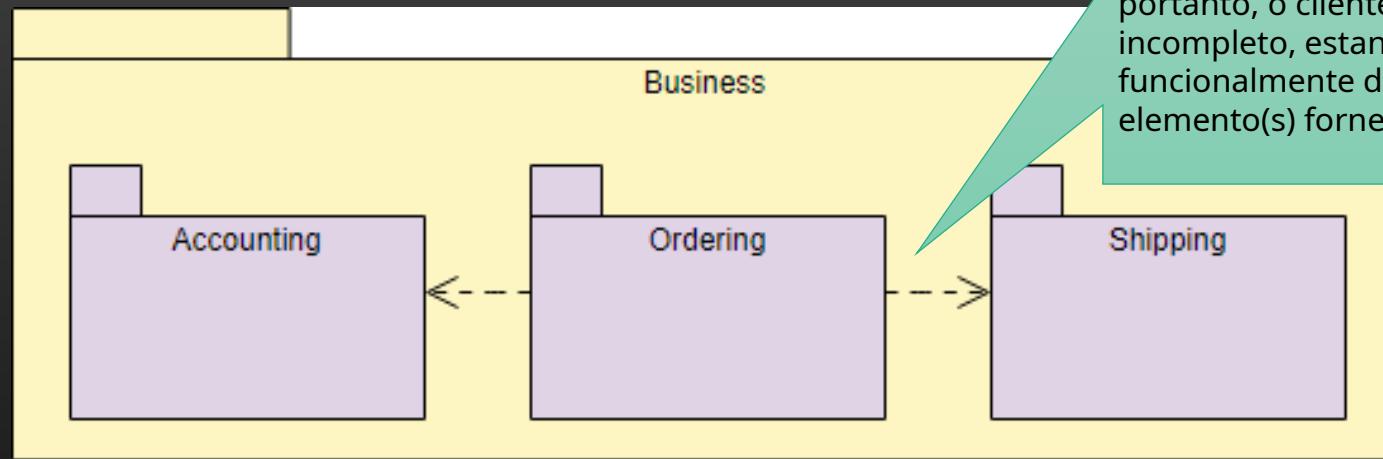
"Contido em"

- duas representações alternativas

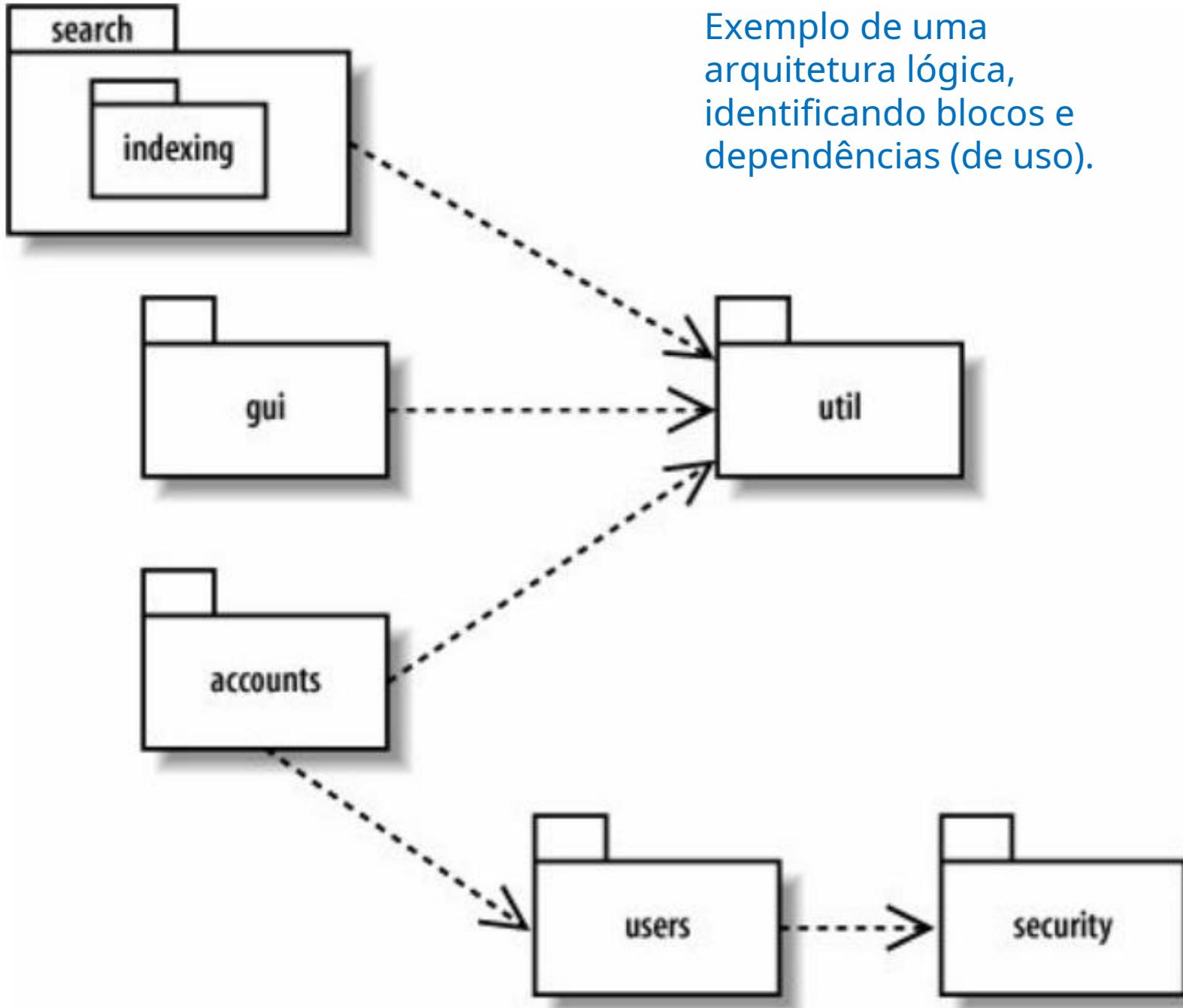


Associações entre pacotes

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



A arquitetura do sistema aborda diferentes perspetivas 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,...)

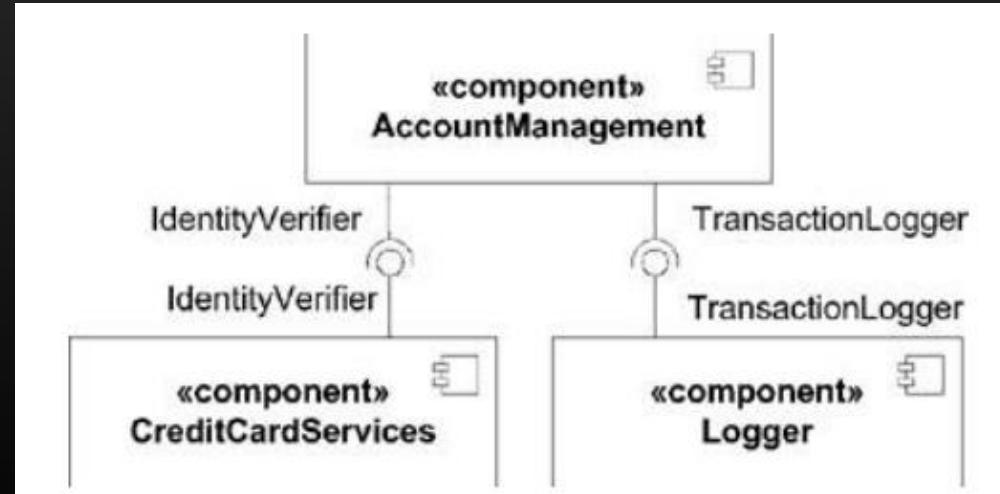
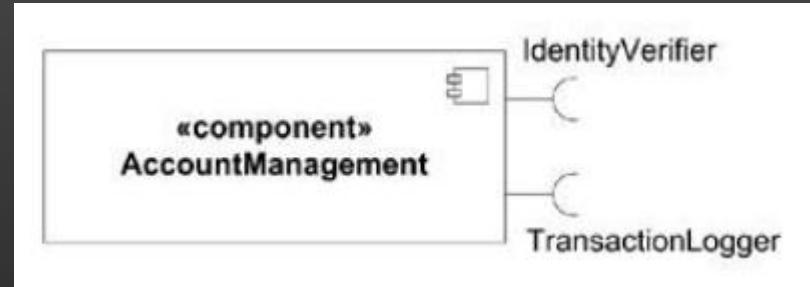
Componente

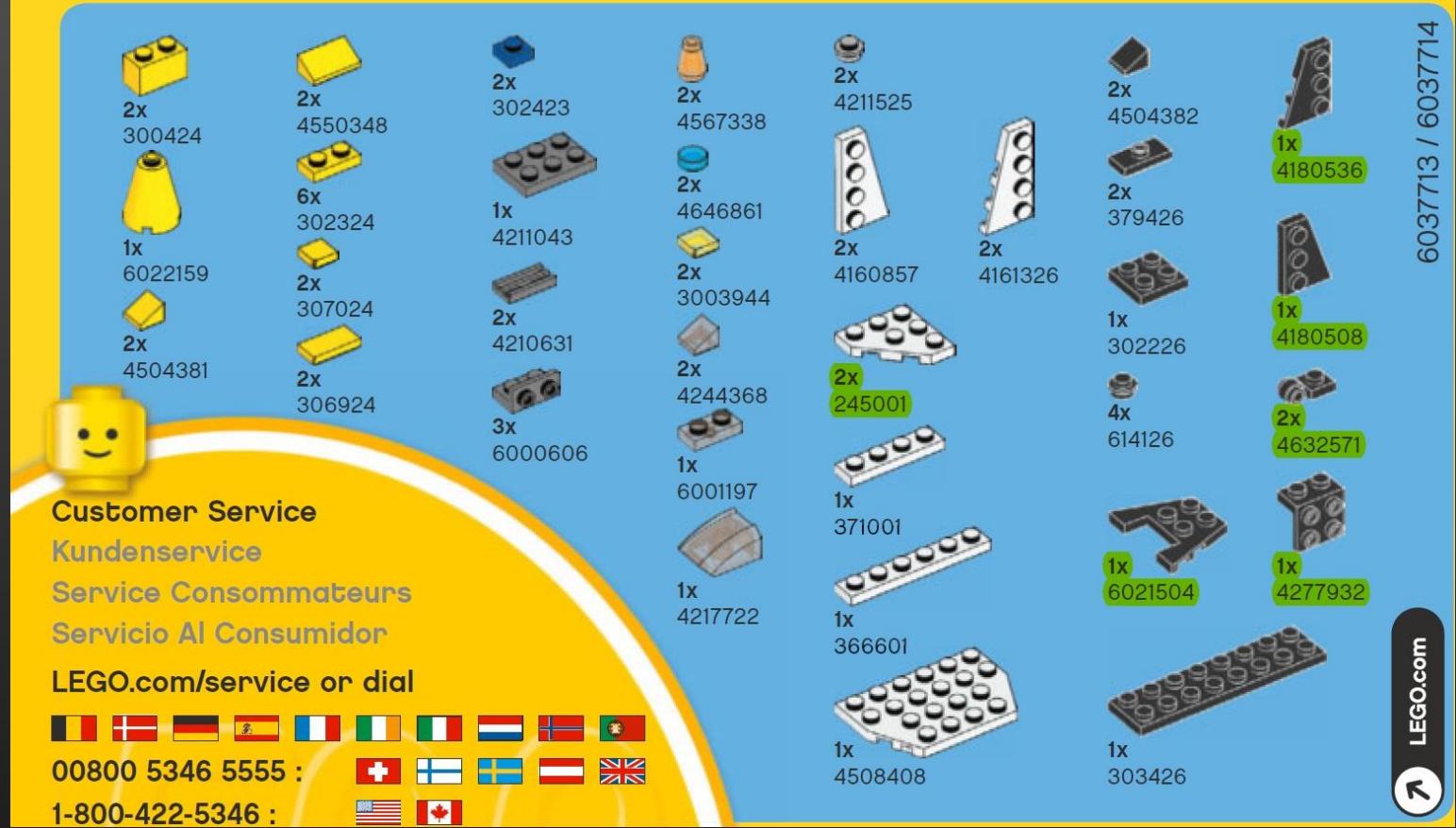
É normal dividir sistemas complexos em subsistemas mais geríveis

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 *application servers*)



<https://mvnrepository.com>

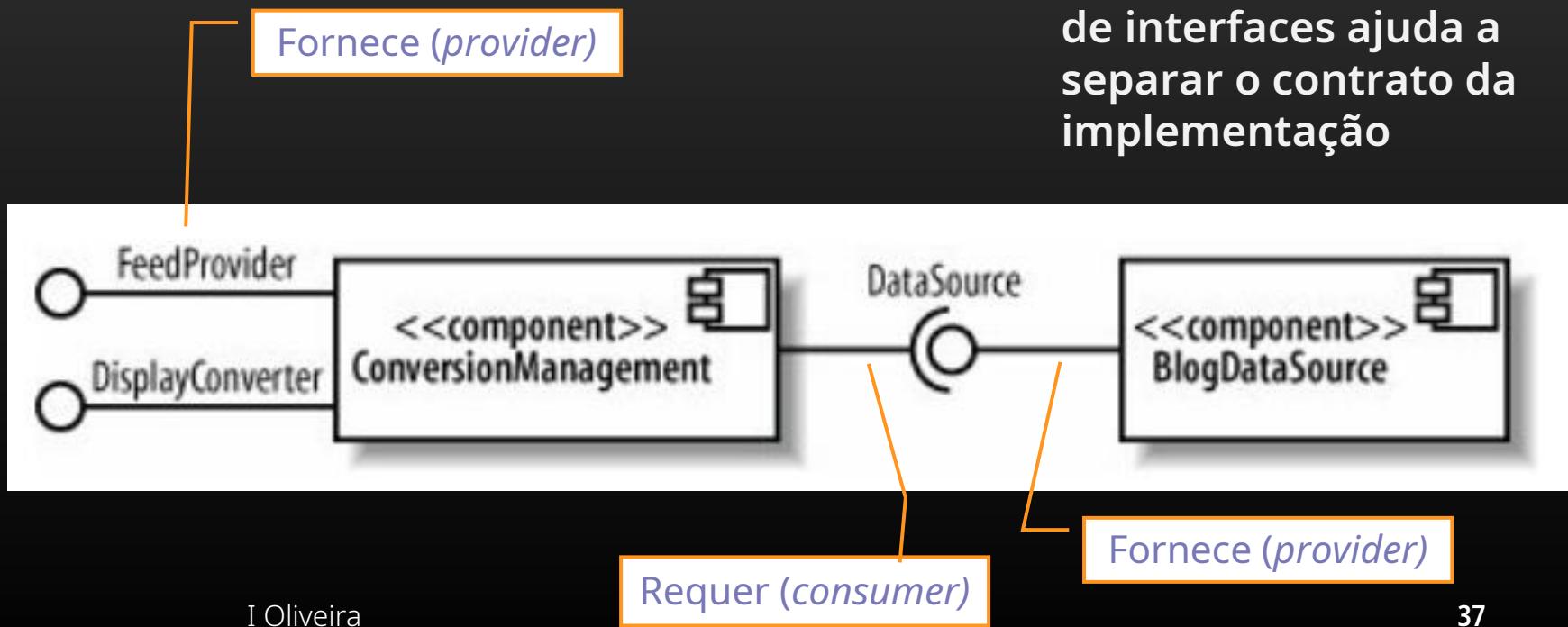
Uma “loja” de componentes, encapsulados, reutilizáveis e substituíveis.

Figure

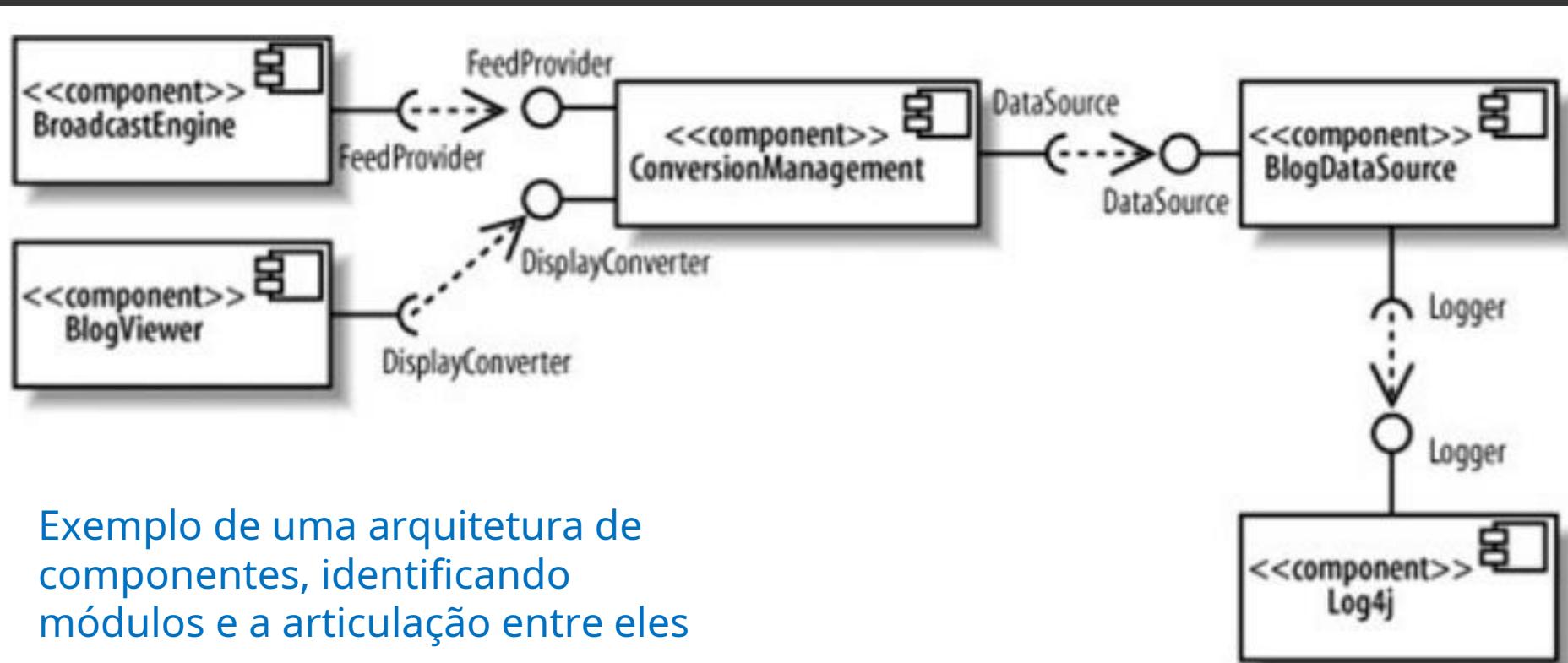
Solução modular com componentes

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

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

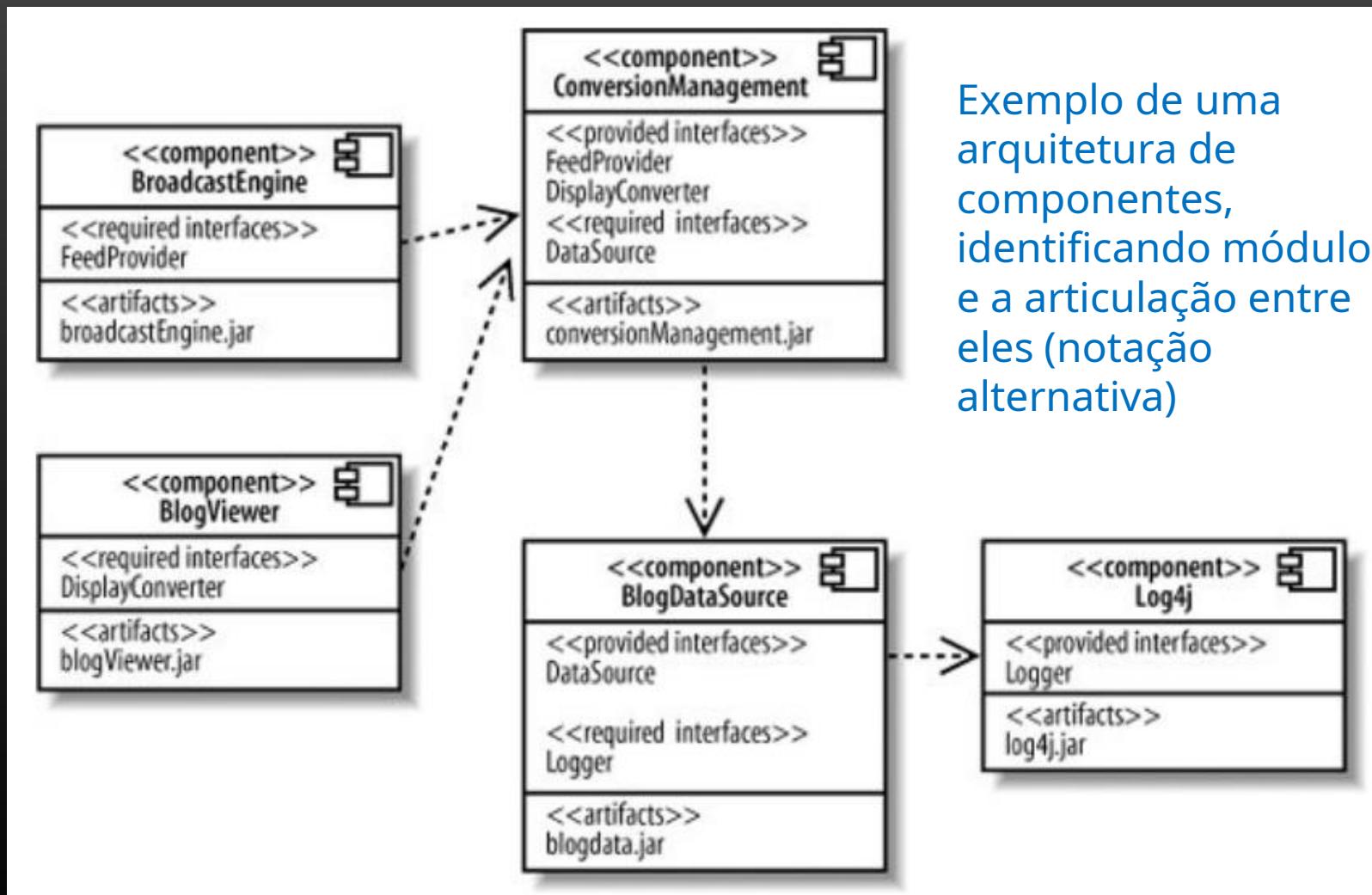


O mesmo modelo, notação ligeiramente diferente 1

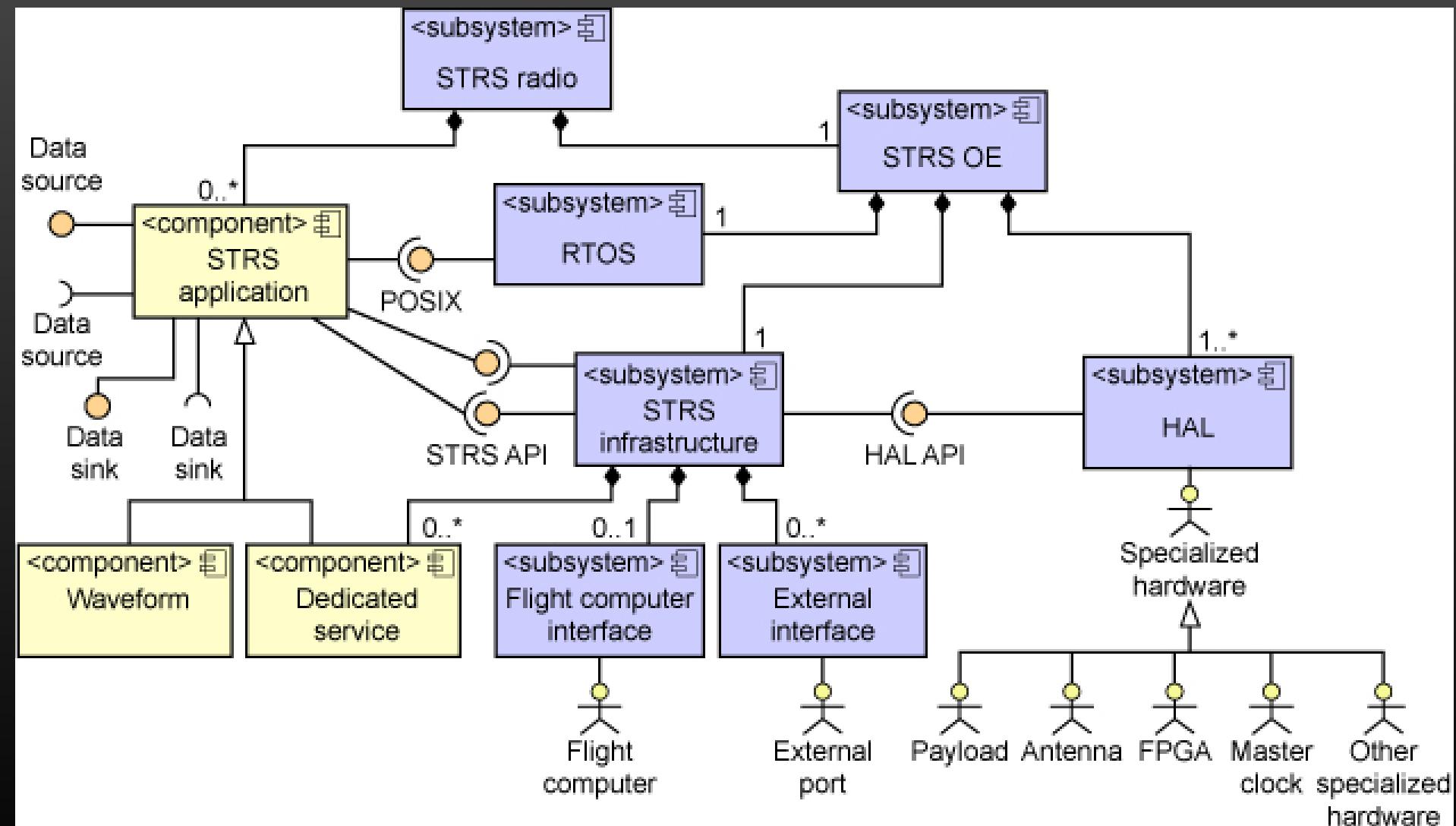


Exemplo de uma arquitetura de componentes, identificando módulos e a articulação entre eles

O mesmo modelo, notação ligeiramente diferente 2

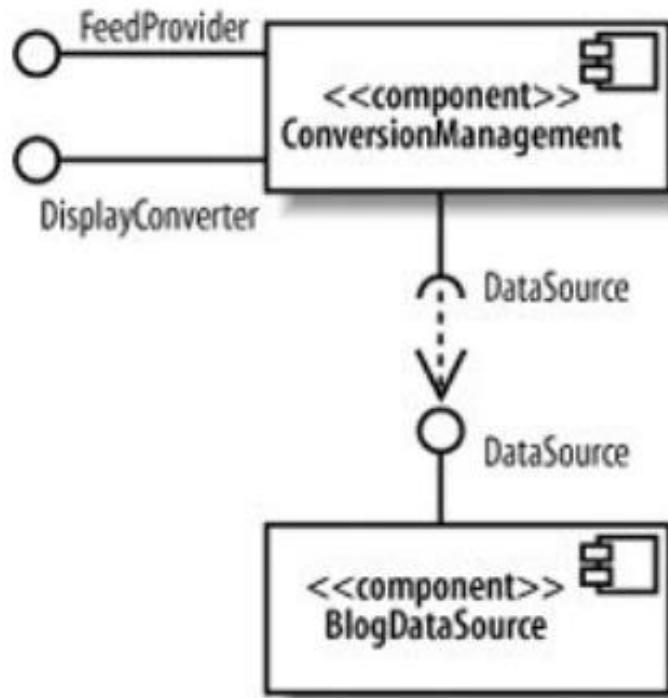


An example from NASA: Radio System

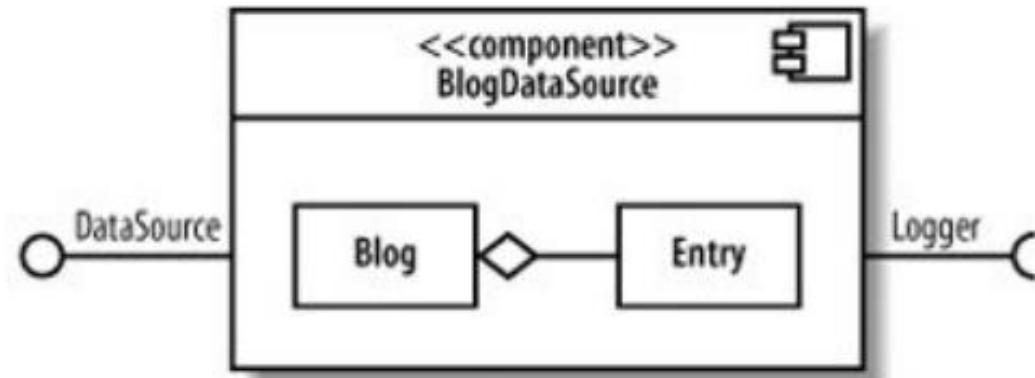


Notação “caixa fechada” / “caixa aberta”

Example Black-Box Component View



Example White-Box Component View



A arquitetura do sistema aborda diferentes perspetivas 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,...)

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.

<<stereotype>>
Node Name

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.

<<stereotype>>
Artifact Name

A node with a deployed artifact:

- Portrays an artifact being placed on a physical node.

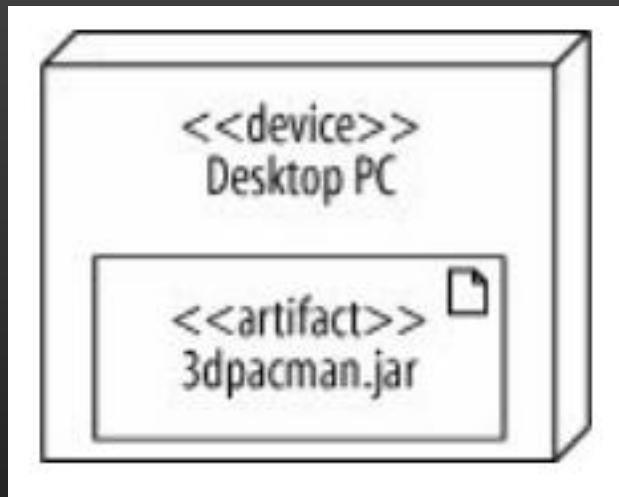
<<stereotype>>
Node Name
<<stereotype>>
Artifact Name

A communication path:

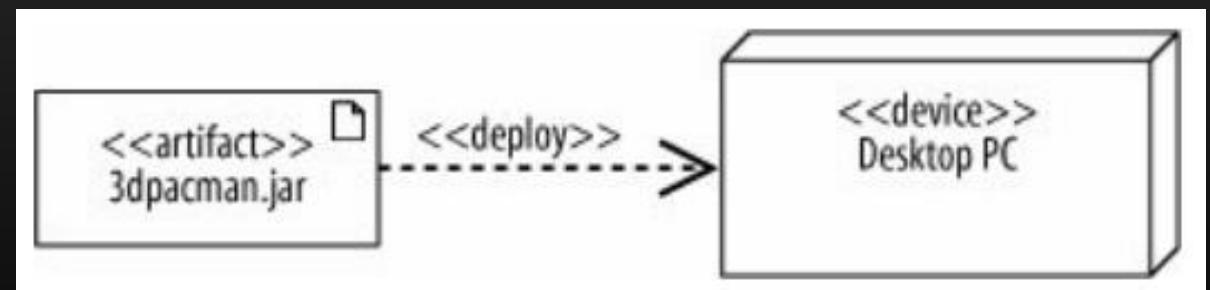
- Represents an association between two nodes.
- Allows nodes to exchange messages.
- May contain a stereotype to specifically label the type of communication path being represented, (e.g., LAN, Internet, serial, parallel).

<<stereotype>>

Os artefactos são executados em nós



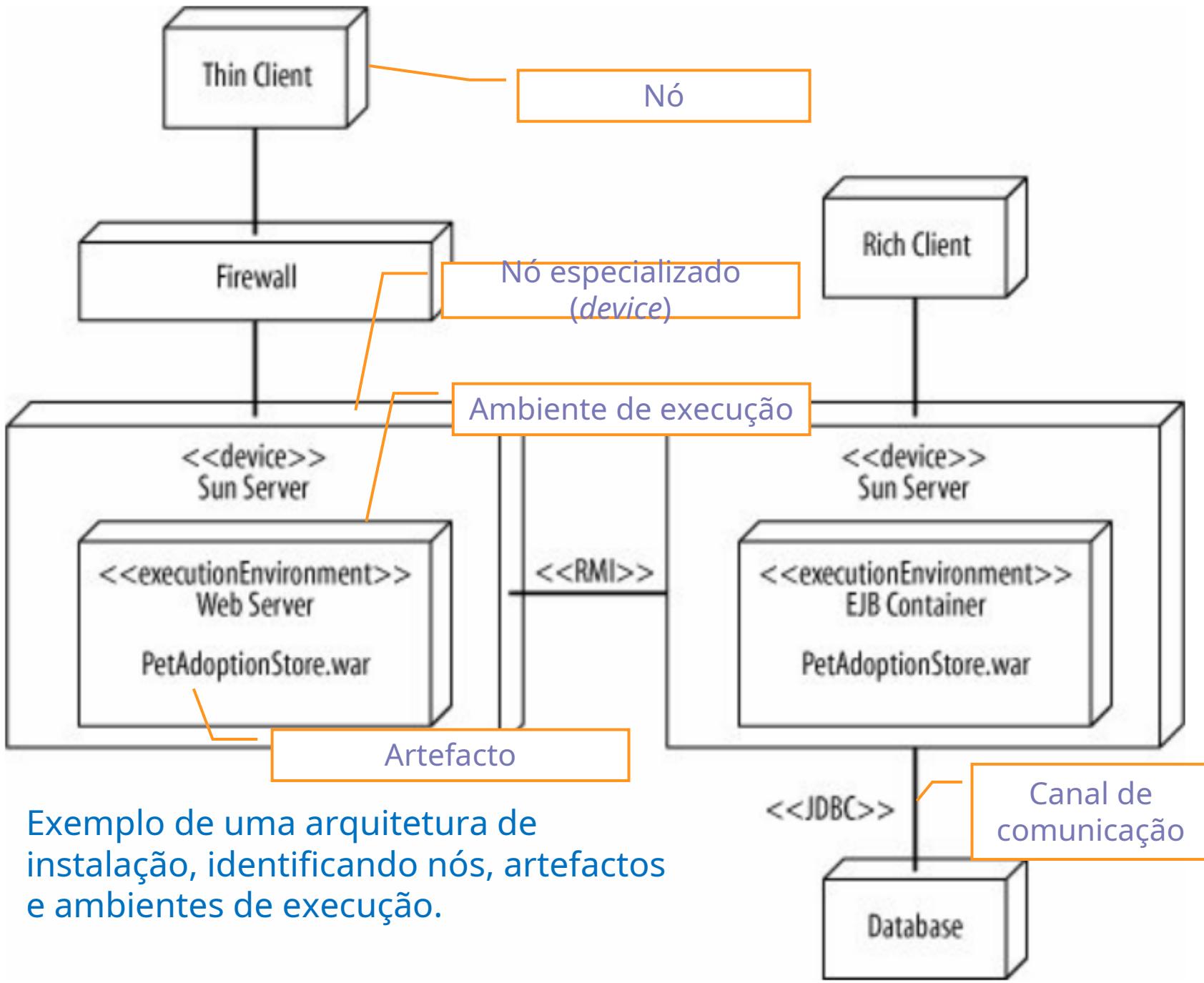
Notações alternativas.



Rastreabilidade até aos componentes



A relação “manifest” permite associar componentes a artefactos.



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

Exemplos (diagrama de instalação)

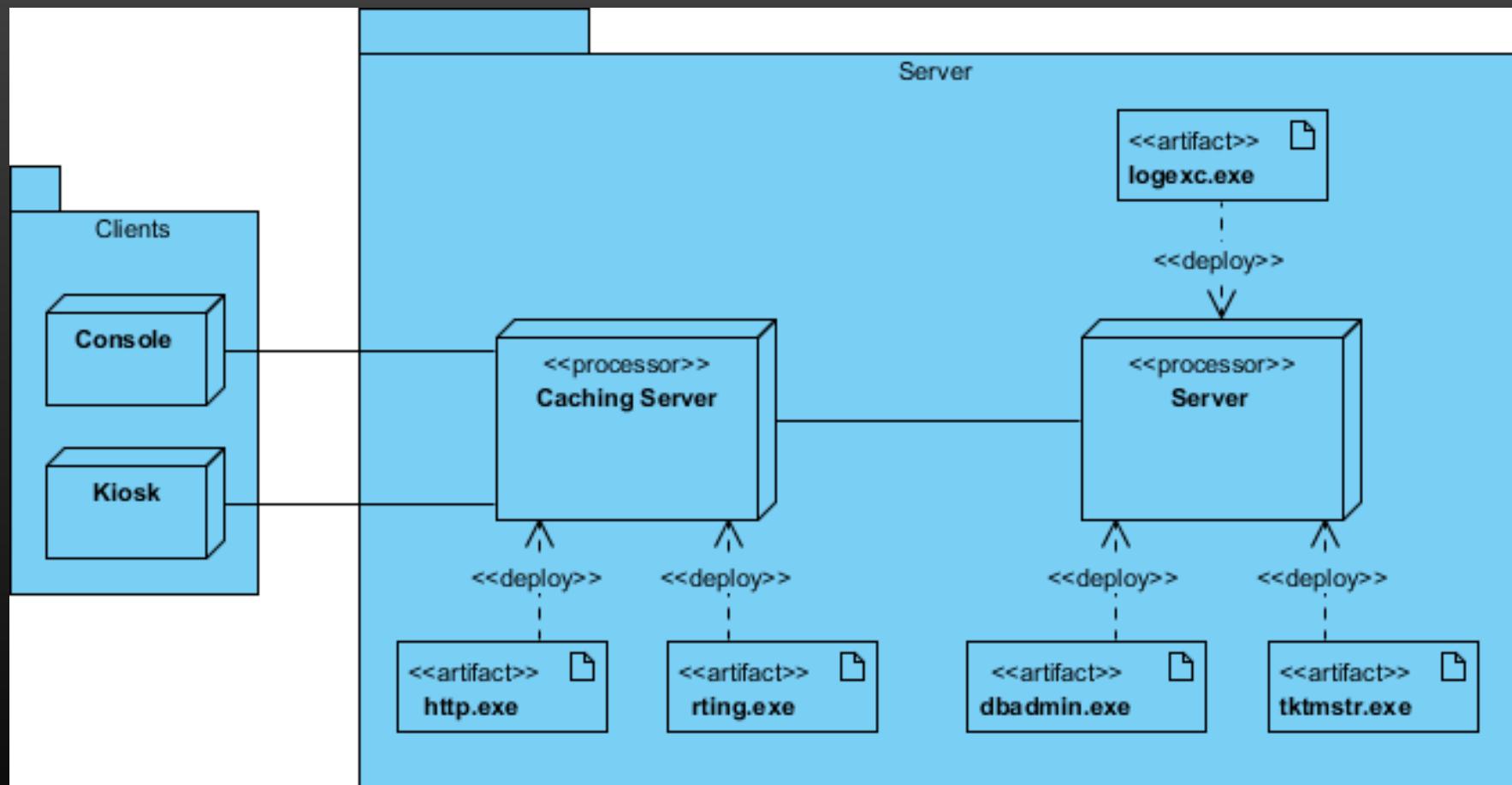
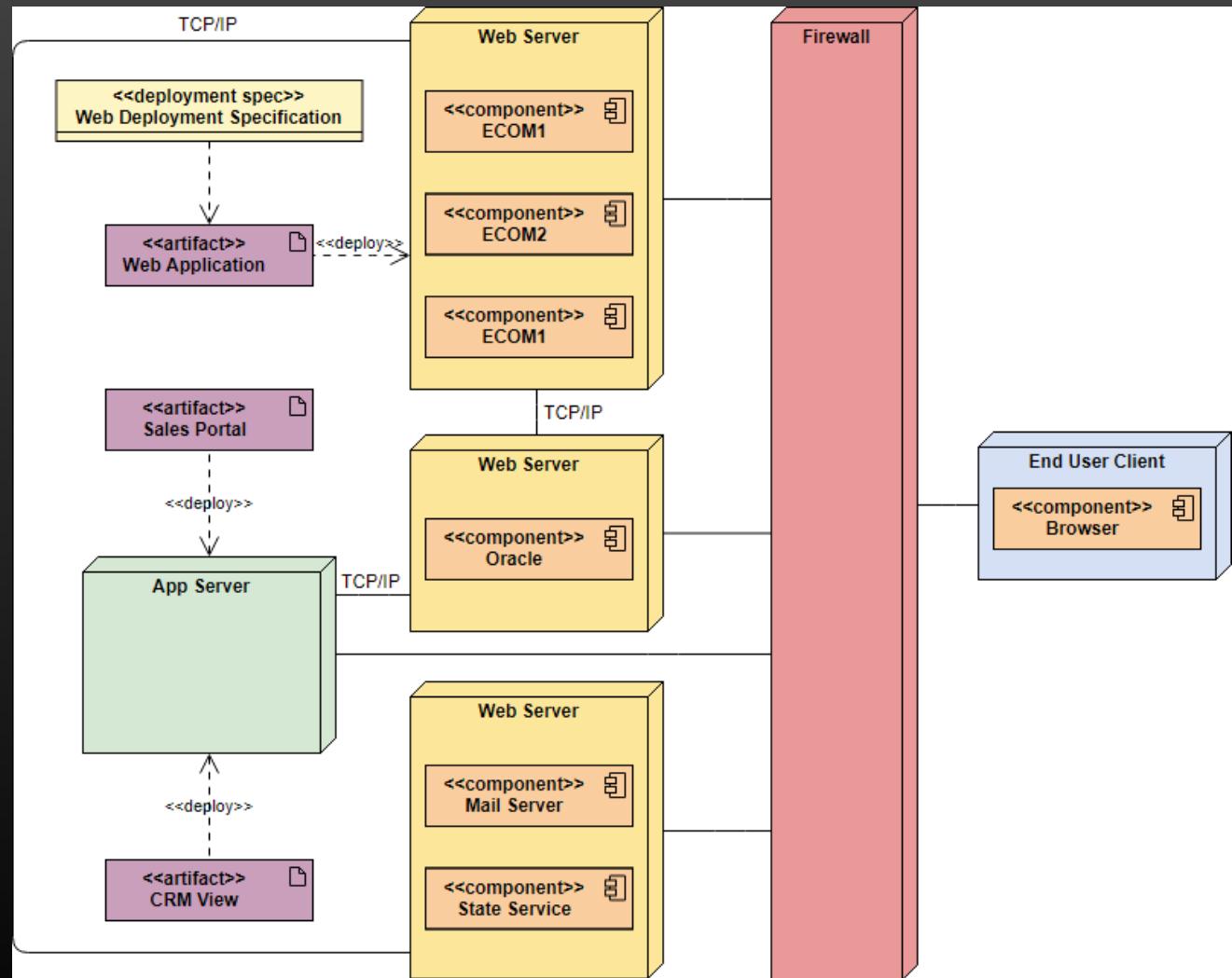


Diagrama de instalação



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/

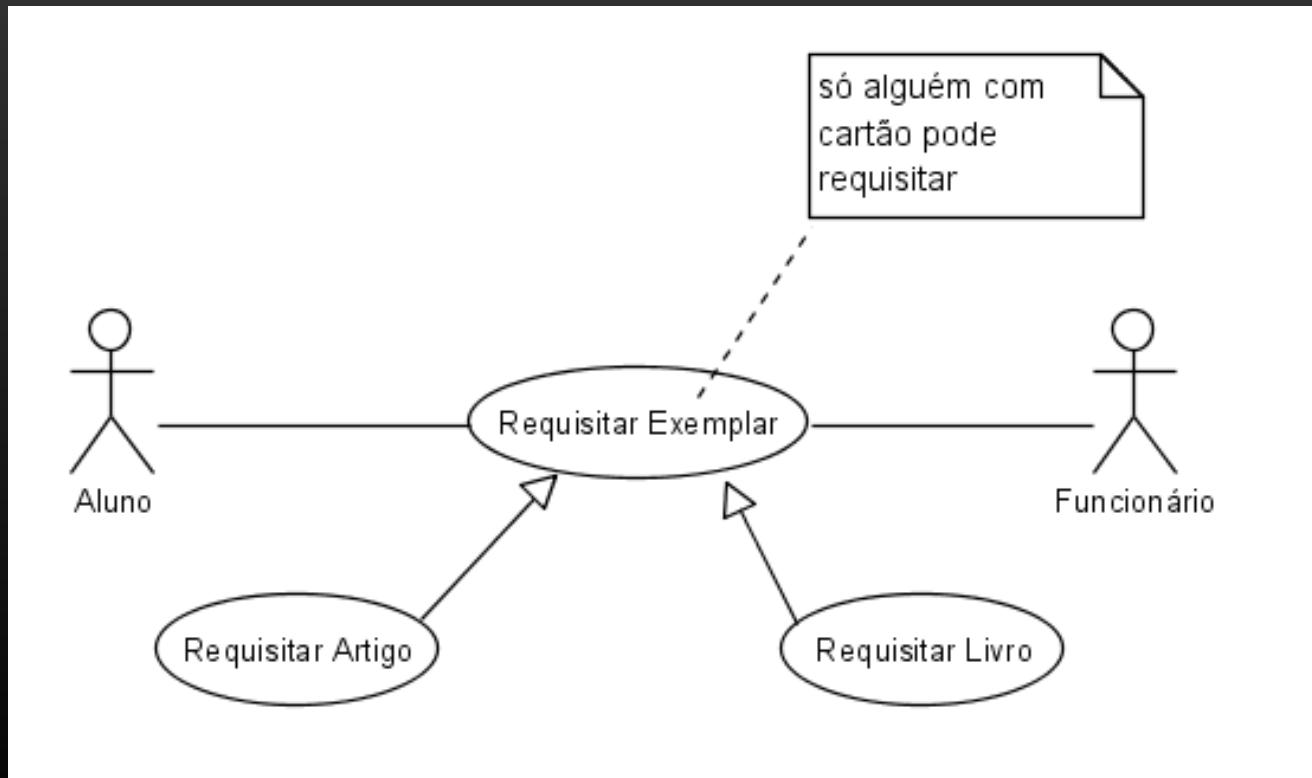
41951- ANÁLISE DE SISTEMAS

Elementos comuns da UML

Ilídio Oliveira | v2022/04/05

Anotações

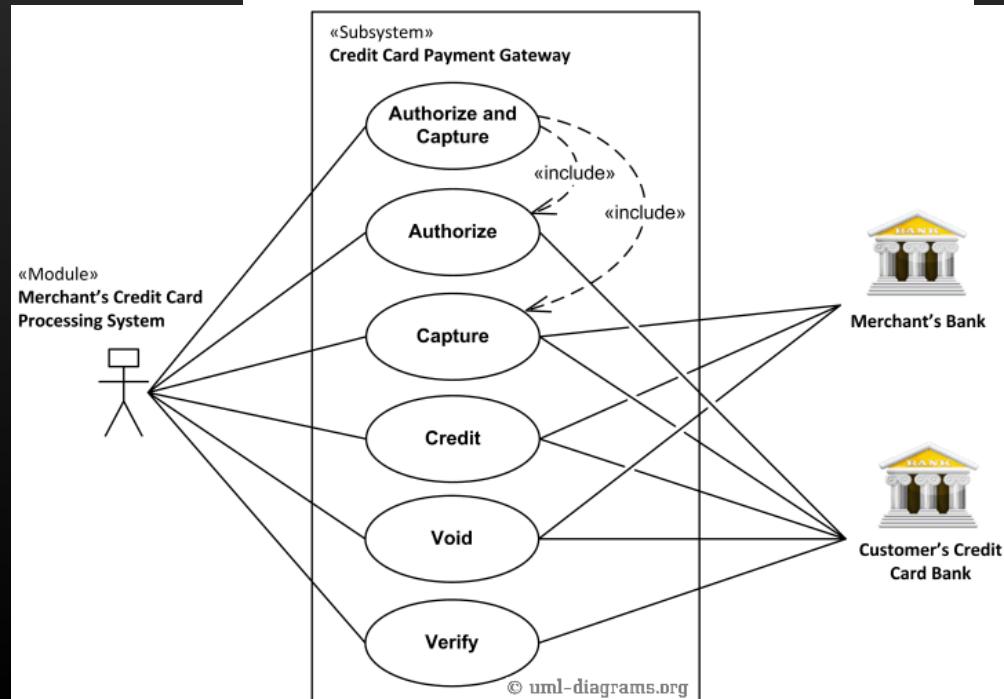
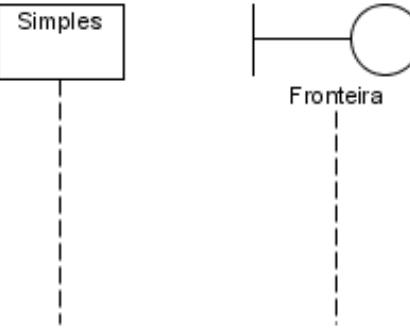
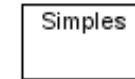
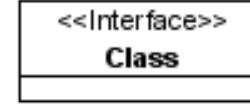
Um comentário que pode ser usado para anotar qualquer elemento



Estereotipo (*stereotype*)

uma especialização da semântica de um elemento do modelação

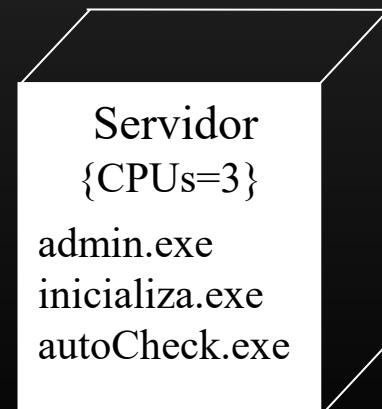
marcada com «...» ou com a alteração da decoração



Valores etiquetados (*tagged values*)

Estender elementos do modelo
com uma linguagem “computável”
(pares atributo/valor)

```
«Computer»  
{Vendor = "Acer",  
CPU = "AMD Phenom X4",  
Memory = "4 GB DDR2"}  
Aspire X1300
```

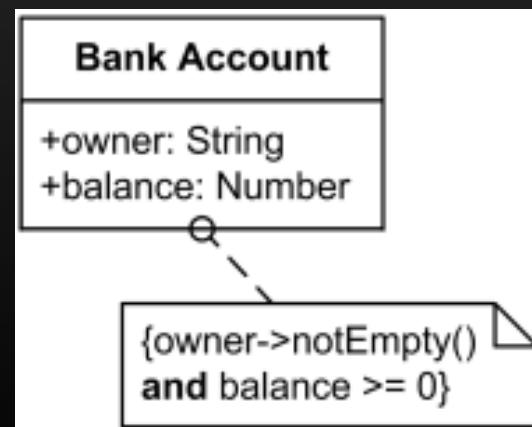
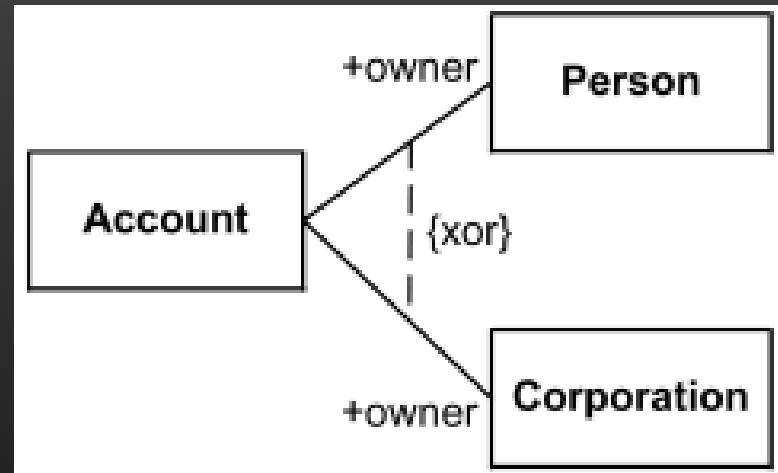


Restrições

Adicionar regras ao modelo ou condicionar a sua interpretação

condição ou restrição relacionada com um ou mais elementos

Linguagem própria para declarar restrições (OCL)

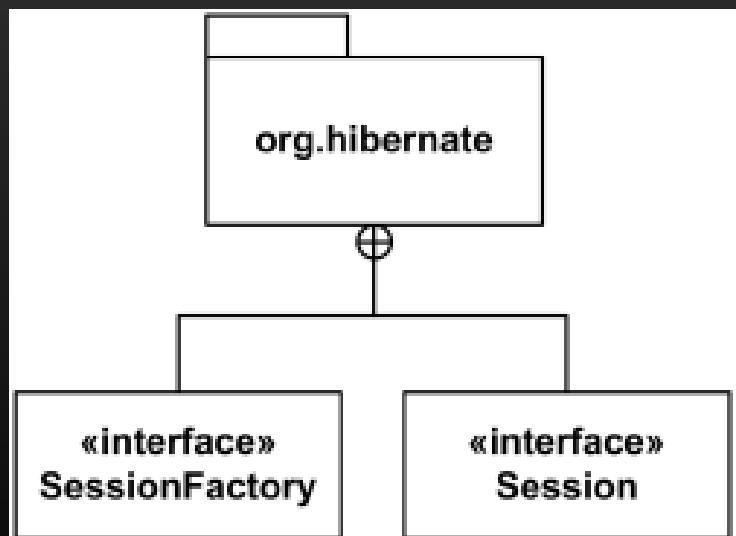
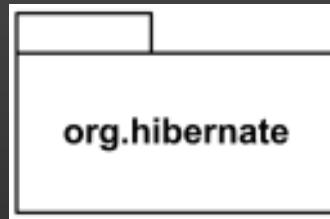


Pacotes

Um mecanismo para dividir
um modelo em partes

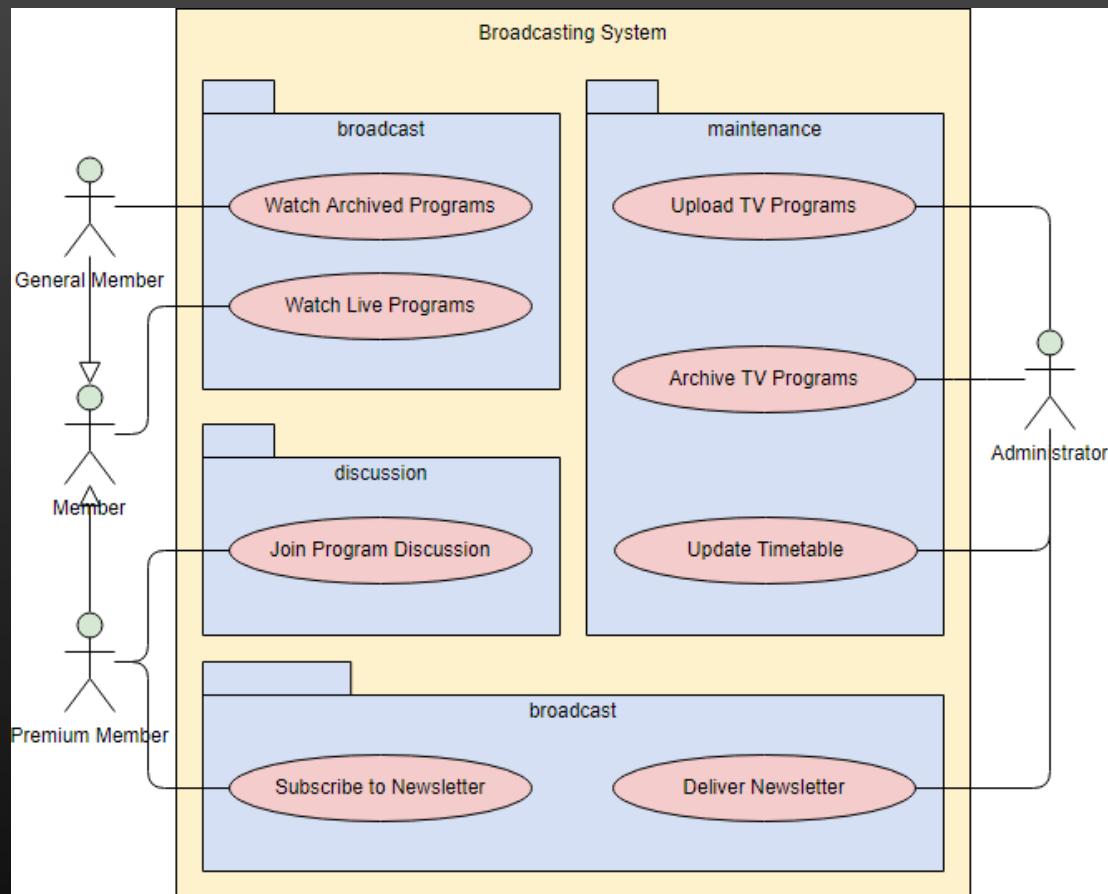
serve como mecanismo genérico
para fazer agrupamentos

Aplicação “especial” nas
vistas de arquitetura (módulos
lógicos)



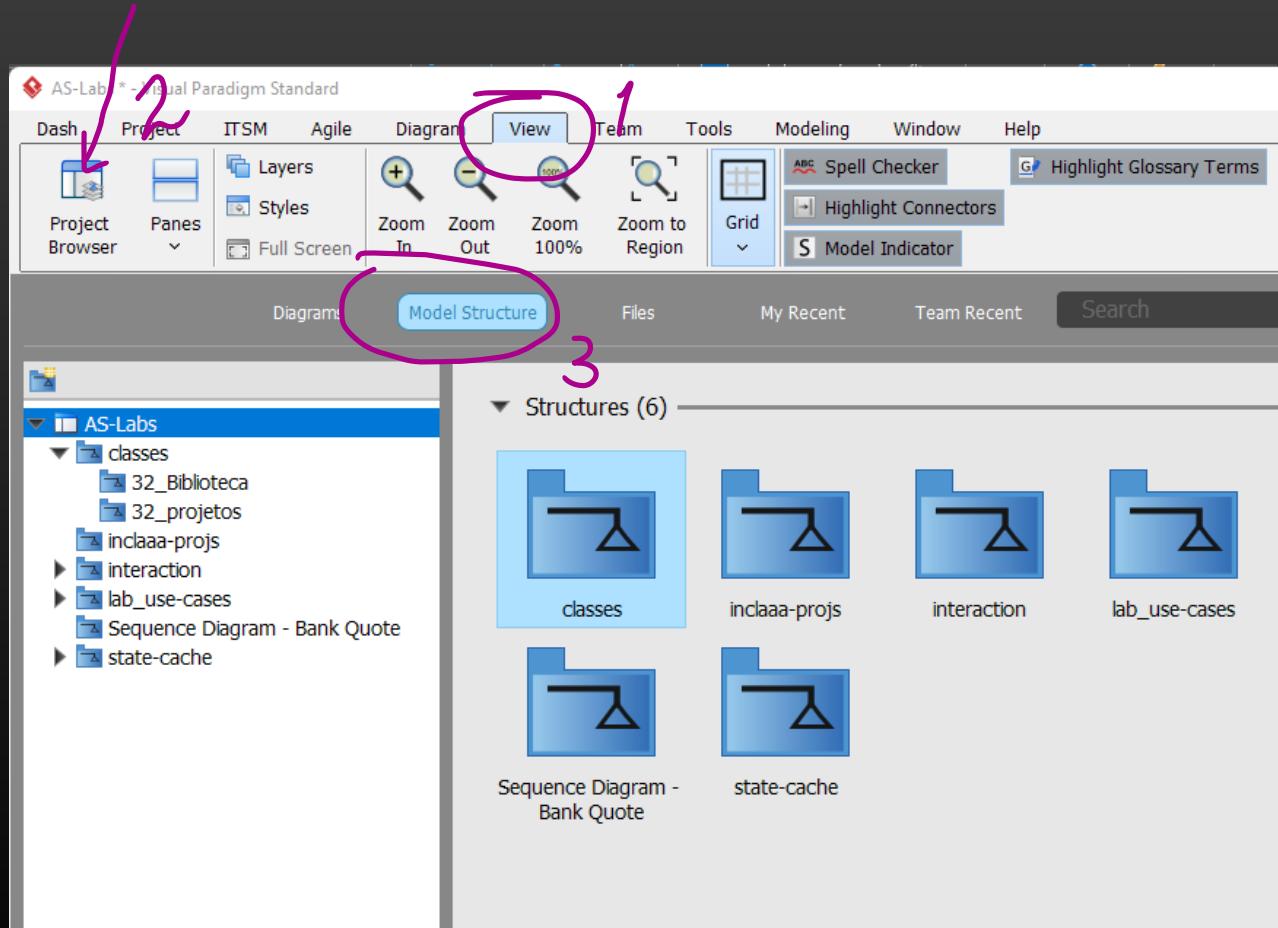
Casos de utilização

Agrupamento sugere modularização/subsistemas



Estrutura do modelo no VP

Modelos hierárquicos



UML®

Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.



Specification

Title: Unified Modeling Language

Acronym: UML®

Version: 2.5.1

Document Status: formal ⓘ

Publication Date: December 2017

Categories: [Modeling](#) [Software Engineering](#) [Platform](#)

IPR Mode ⓘ

RF-Limited ⓘ

TABLE OF CONTENTS

- [About the Specification](#)
- [Issues associated with this specification](#)
- [Specification Documents](#)
 - [Normative Documents](#)
 - [Normative Machine Readable Documents](#)
 - [Informative Documents](#)
- [History](#)
 - [Formal Versions](#)
- [Links](#)

ISSUES ASSOCIATED WITH THIS SPECIFICATION



Issues Reported in
this Specification



Issues Fixed in this
Specification



[Report an issue](#)



Issue Tracker
(Members Only)



41951- ANÁLISE DE SISTEMAS

Processos de software

Ilídio Oliveira | v2022/04/05

Objetivos de aprendizagem

- Identificar atividades comuns a todos os projetos (ciclo de vida)
- Distinguir projetos sequenciais de projetos evolutivos
- Descrever a estrutura do Unified Process (fases, objetivos, iterações)
- Identificar as principais atividades exigidas na atribuição do projeto
- Mapear disciplinas técnicas nas fases do OpenUP

Fases do SDLC: um modelo geral para a engenharia de software

SDLC - Systems Development lifecycle (a.k.a. Software Development lifecycle)

Quatro fases fundamentais: planeamento, análise, desenho e implementação.

Diferentes projetos podem enfatizar diferentes partes do SDLC, mas todos os projetos têm elementos destas quatro fases.

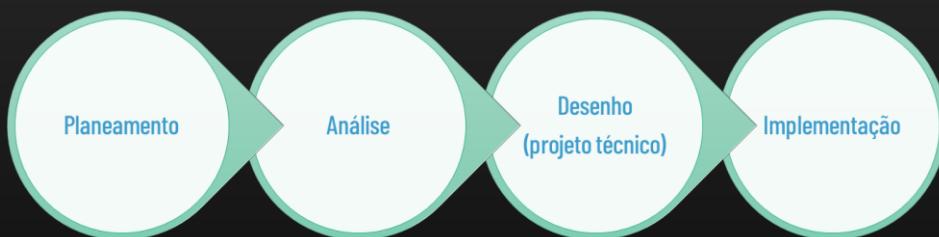
Cada fase é composta por uma série de atividades, em que aplica disciplinas técnicas para produzir resultados previstos



Fases fundamentais: planeamento, análise, desenho e implementação

A fase de planeamento é o processo fundamental de compreensão do porquê de um sistema de informação dever ser construído e determinar como a equipa do projeto irá construí-lo.

Definir a transformação digital pretendida.



PASSOS PRINCIPAIS:

1. Arranque do projeto

- o valor do sistema para a organização é identificado.
- Um pedido de um novo sistema ("caderno de encargos") apresenta um breve resumo de uma necessidade de negócio, e explica como um sistema que suporta a necessidade irá criar valor de negócio.
- Os pedidos do sistema e a análise da viabilidade são apresentados a um comité de aprovação (ao nível de direção), que decide se o projeto deve ser realizado.

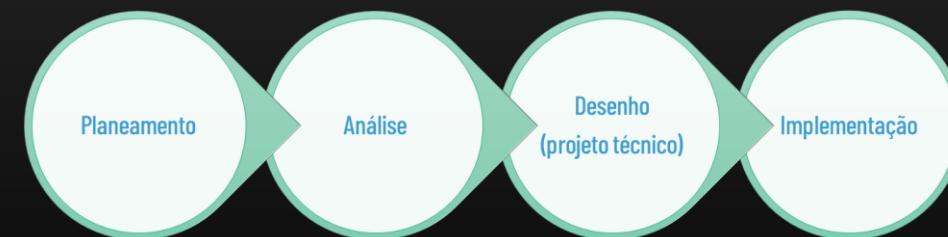
2. Gestão do projeto

- O gestor do projeto cria um plano de trabalho, atribui uma equipa, e coloca em prática as técnicas para monitorizar e dirigir o projeto através de todo o SLC.

Fases fundamentais: planeamento, análise, desenho e implementação

A fase de análise responde às perguntas de quem irá utilizar o sistema, o que é que o sistema deve fazer, e onde e como será utilizado.

Durante esta fase, a equipa do projeto investiga quaisquer sistemas atuais, identifica oportunidades de melhoria e desenvolve um conceito para o novo sistema.



PASSOS PRINCIPAIS:

1. Estudo do domínio/área e análise dos sistemas existentes

Como é que as pessoas vão trabalhar e como é que o novo sistema ajuda nisso?

2. Levantamento de requisitos

Trabalho de ligação com os stakeholders para sistematizar as necessidades (capacidades pedidas para novo sistema)

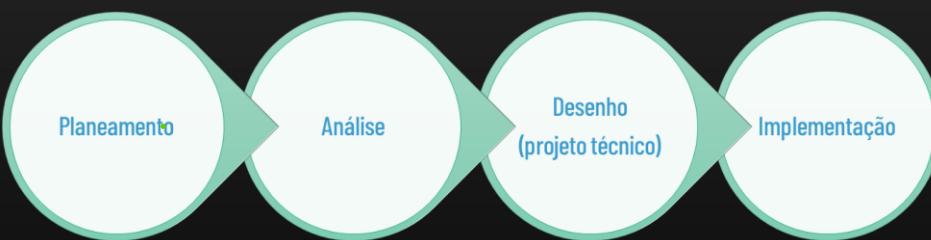
3. Conceito para a solução (proposta do sistema)

Proposta de uma solução (sistema) que resolve as necessidades identificadas.

Fases fundamentais: planeamento, análise, desenho e implementação

A fase de desenho (=plano técnico para a implementação) decide como o sistema será construído, em termos de hardware, software, infraestrutura de rede; a interface, formulários e relatórios do utilizador; e os programas específicos, bases de dados e ficheiros que serão necessários.

Escolha de tecnologias e “tática” para a equipa de desenvolvimento.



PASSOS PRINCIPAIS:

1. Estratégia de desenvolvimento

Desenvolvimento interno ou contratualizado?

2. Conceção da arquitetura do sistema

e.g.: cloud ou desktop? Componentes distribuídas em vários servidores ou num nó central?...

3. Conceção do modelo de dados

Detalhe das estruturas de dados que devem ser mantidas, e.g.: modelo da base de dados PostgreSQL

3. Desenho das entidades de software (programas) e seleção de frameworks

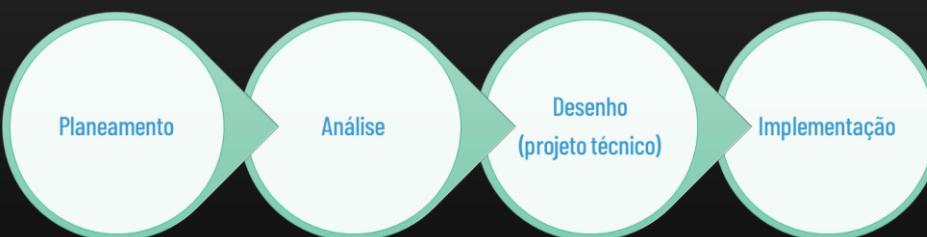
Aplicar princípios e boas práticas para estruturar os programas (código)

A generalidade dos sistemas usa/integra “plataformas” de software existente, para resolver alguns problemas comuns (e.g.: desenvolver ou usar uma solução de pagamentos online?)

Fases fundamentais: planeamento, análise, desenho e implementação

Na fase de implementação, o sistema é de facto construído (ou adquirido, no caso de pacotes pré-feitos), com a escrita do código, integração de sistemas, desenvolvimento das bases de dados, verificação do software (testes),...

Inclui também a transição para o ambiente de produção.



PASSOS PRINCIPAIS:

1. Implementação de sistemas (construção e garantia da qualidade)

Desenvolver o código, realizar os testes, integrar módulos e frameworks, desenvolver as interfaces do utilizador,...

2. Instalação e transição

Colocar em “produção”

3. Plano de suporte (revisões pós-instalação e gestão de modificações)

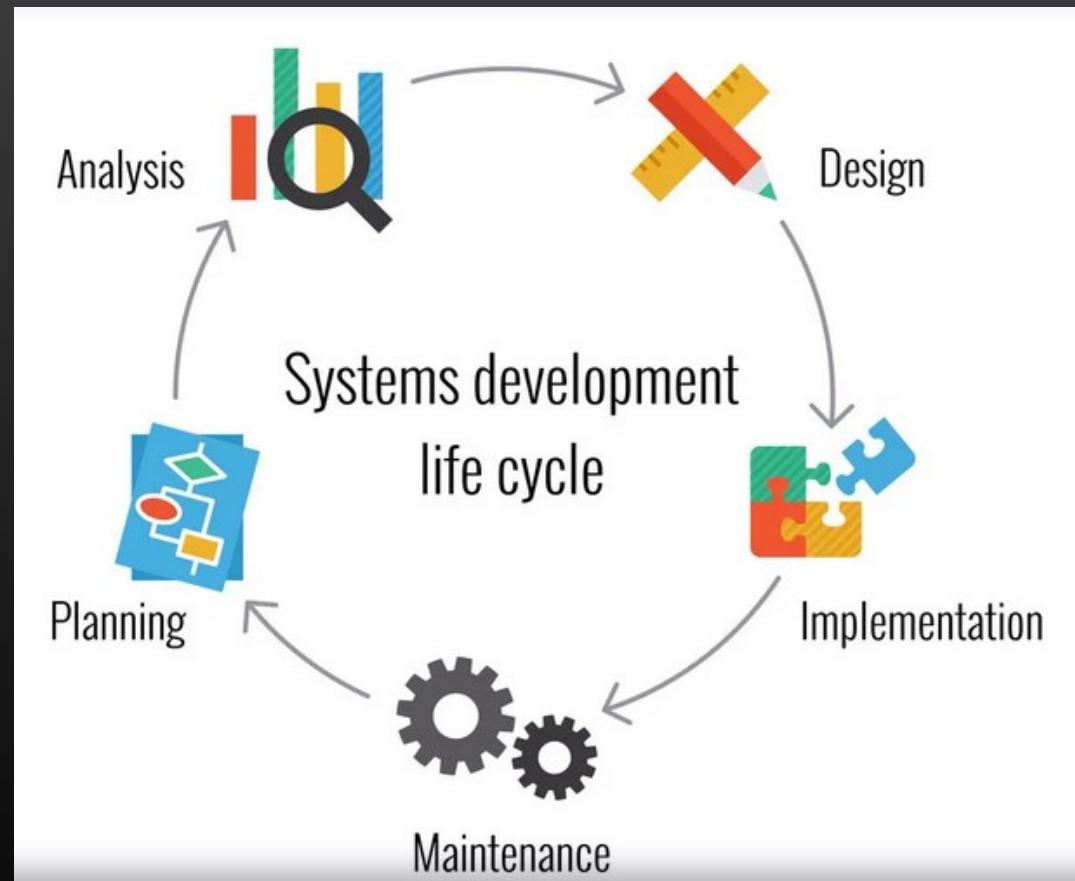
A vida do software continua após a entrada em produção...

.

Alguns autores representam a fase de manutenção

Ideia de evolução do produto

Verdadeiro “ciclo”



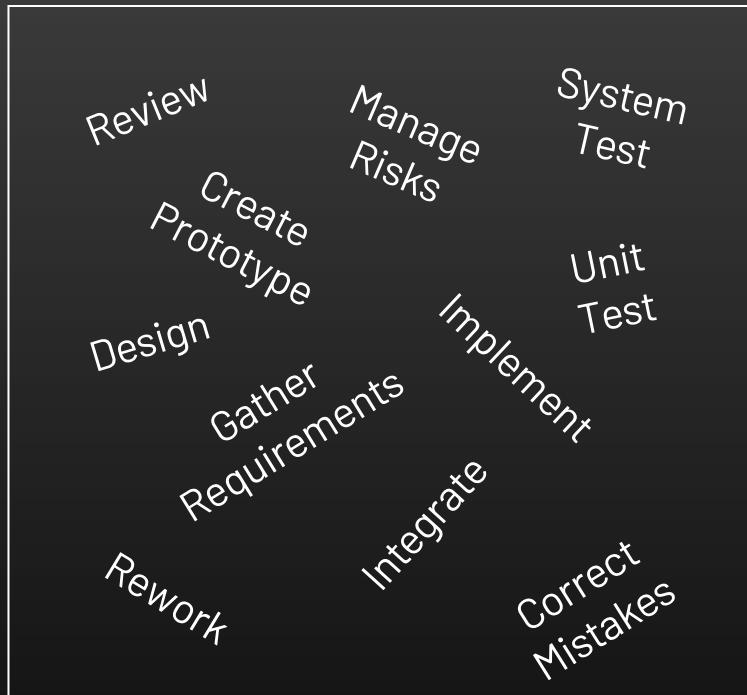
E.g.: Hoffer et al, "Modern Systems Analysis and Design", 5th ed.

Table 1-2 Products of SDLC Phases

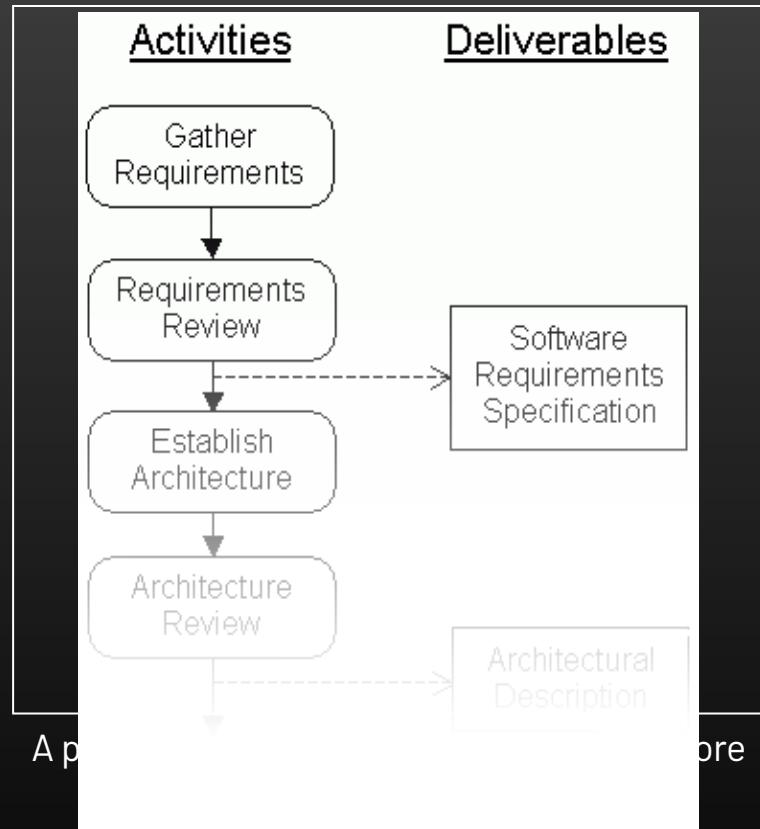
Phase	<i>Products, Outputs, or Deliverables</i>
Planning	Priorities for systems and projects; an architecture for data, networks, and selection hardware, and IS management are the result of associated systems; Detailed steps, or work plan, for project; Specification of system scope and planning and high-level system requirements or features; Assignment of team members and other resources; System justification or business case
Analysis	Description of current system and where problems or opportunities are with a general recommendation on how to fix, enhance, or replace current system;
Design	Explanation of alternative systems and justification for chosen alternative Functional, detailed specifications of all system elements (data, processes, inputs, and outputs); Technical, detailed specifications of all system elements (programs, files, network, system software, etc.); Acquisition plan for new technology
Implementation	Code, documentation, training procedures, and support capabilities
Maintenance	New versions or releases of software with associated updates to documentation, training, and support

Credit: Hoffer et al, "Modern Systems Analysis and Design", 5th ed.

Como aplicar o SDLC de forma sistemática?



Developing software without a defined process is chaotic and inefficient



A p

ore

"It is better not to proceed at all, than to proceed without method." -- Descartes

Processos de software

Sistematização das atividades, papéis e resultados do processo de desenvolvimento.

Software process

O SDLC é concretizado usando um processo de software sistemático.

Um processo de software é um guia para as atividades e tarefas que são necessárias para construir software de qualidade.

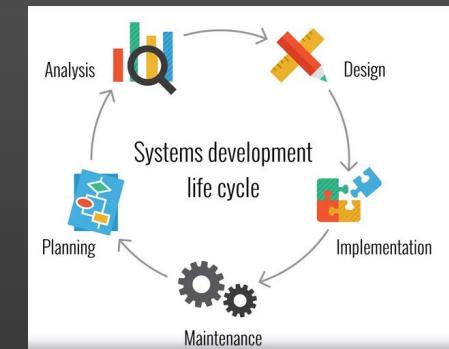
Por que precisamos de um processo explícito?

As falhas ocorrem frequentemente

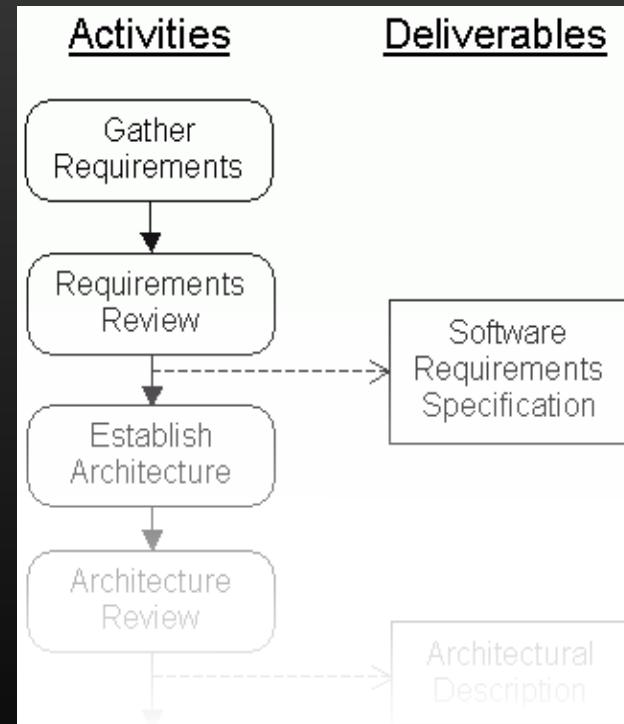
Criar sistemas complexos não é intuitivo

Os projetos apresentam problemas de implementação (terminados fora do prazo, acima do orçamento ou entregues com menos funcionalidades do que o previsto)

SDLC (visão geral do desenvolvimento)



Processo (dá o “guião”)



Plan-driven or evolutionary processes?

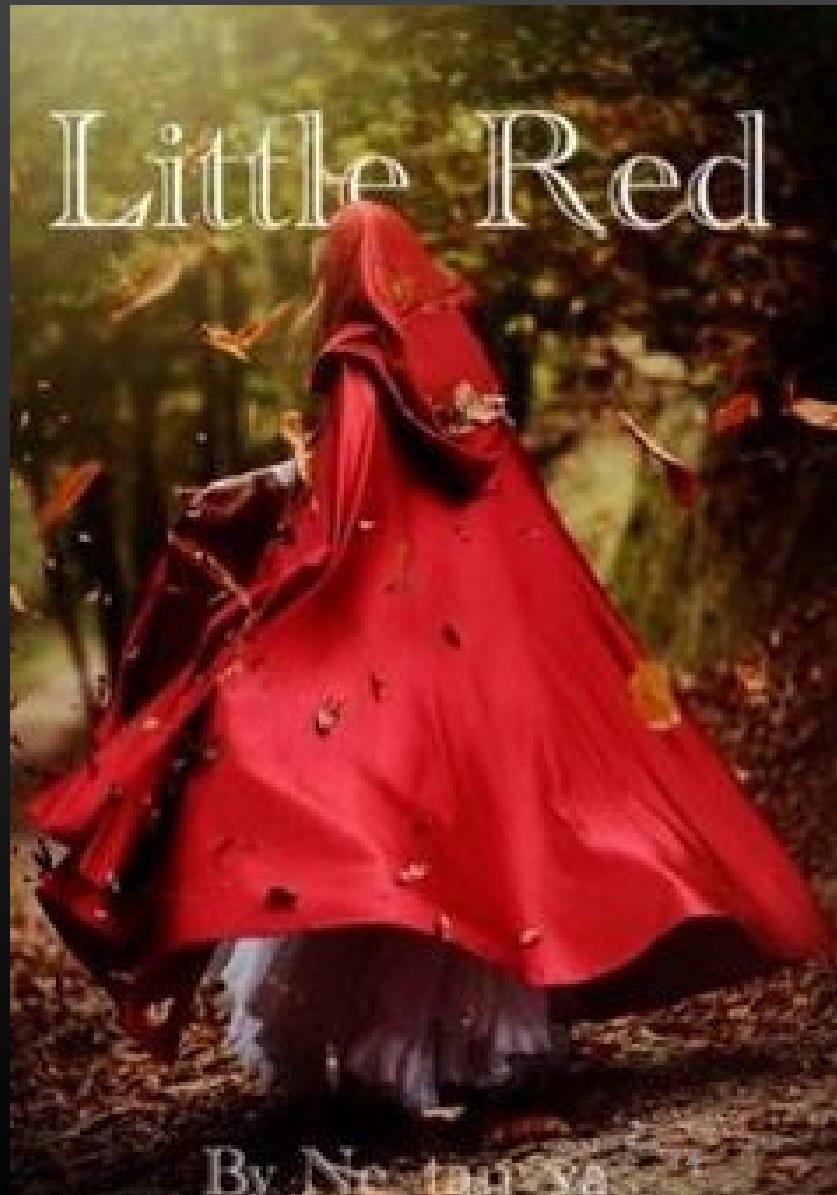
Plan-driven/prescriptive processes

all of the process activities are planned in advance and progress is measured against this plan.

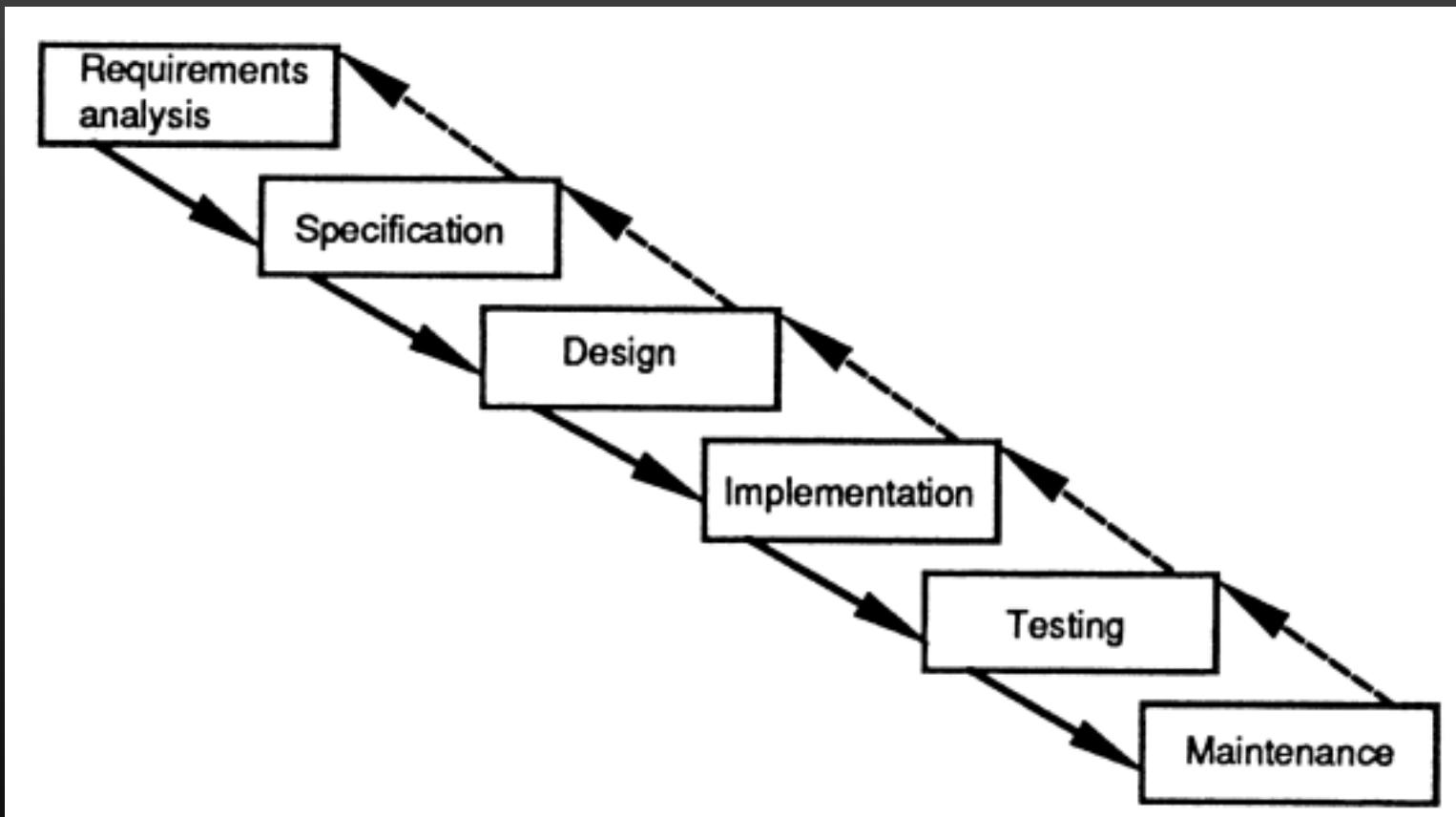
Evolutionary processes

planning is incremental and it is easier to adapt to changing customer requirements.

<https://bit.ly/3DGbEsa>



“Classical” engineering approach: Waterfall model



W. Royce, “Managing the Development of Large Software Systems,” *Proc. Westcon*, IEEE CS Press, 1970, pp. 328-339.

Waterfall model advantages

Simple and easy to understand and use.

Easy to plan

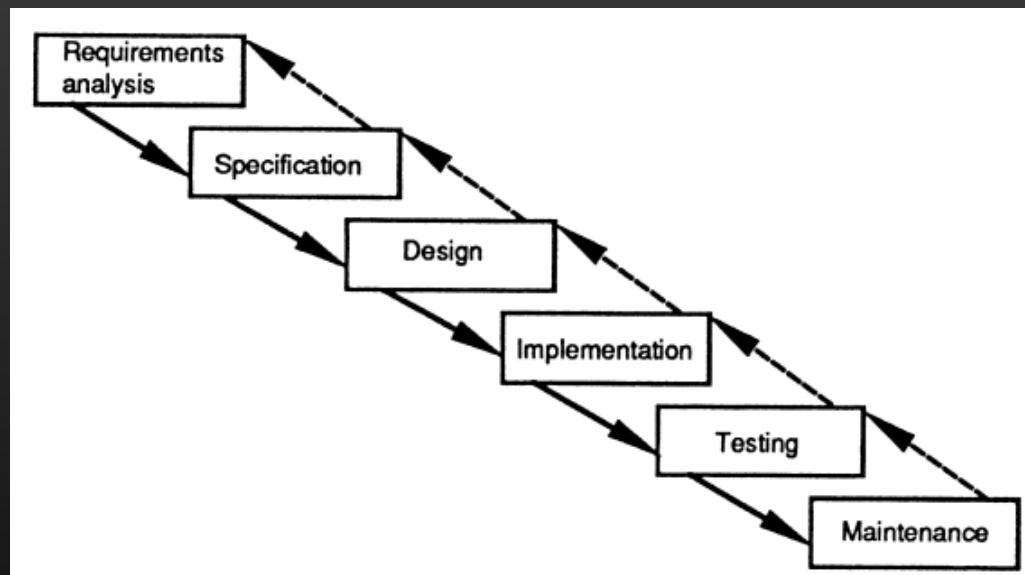
A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a car-wash, and theoretically, be delivered on time.

Easy to manage

each phase has specific deliverables and a review process.

Phases are processed and completed one at a time.

Works well where requirements are stable and well understood



Waterfall model disadvantages

Problems

Difficulty of accommodating change after the process is underway.

Poor model for long and ongoing projects.

No working software is produced until late during the life cycle.

Not suitable for the projects where requirements are uncertain or at the risk of changing.

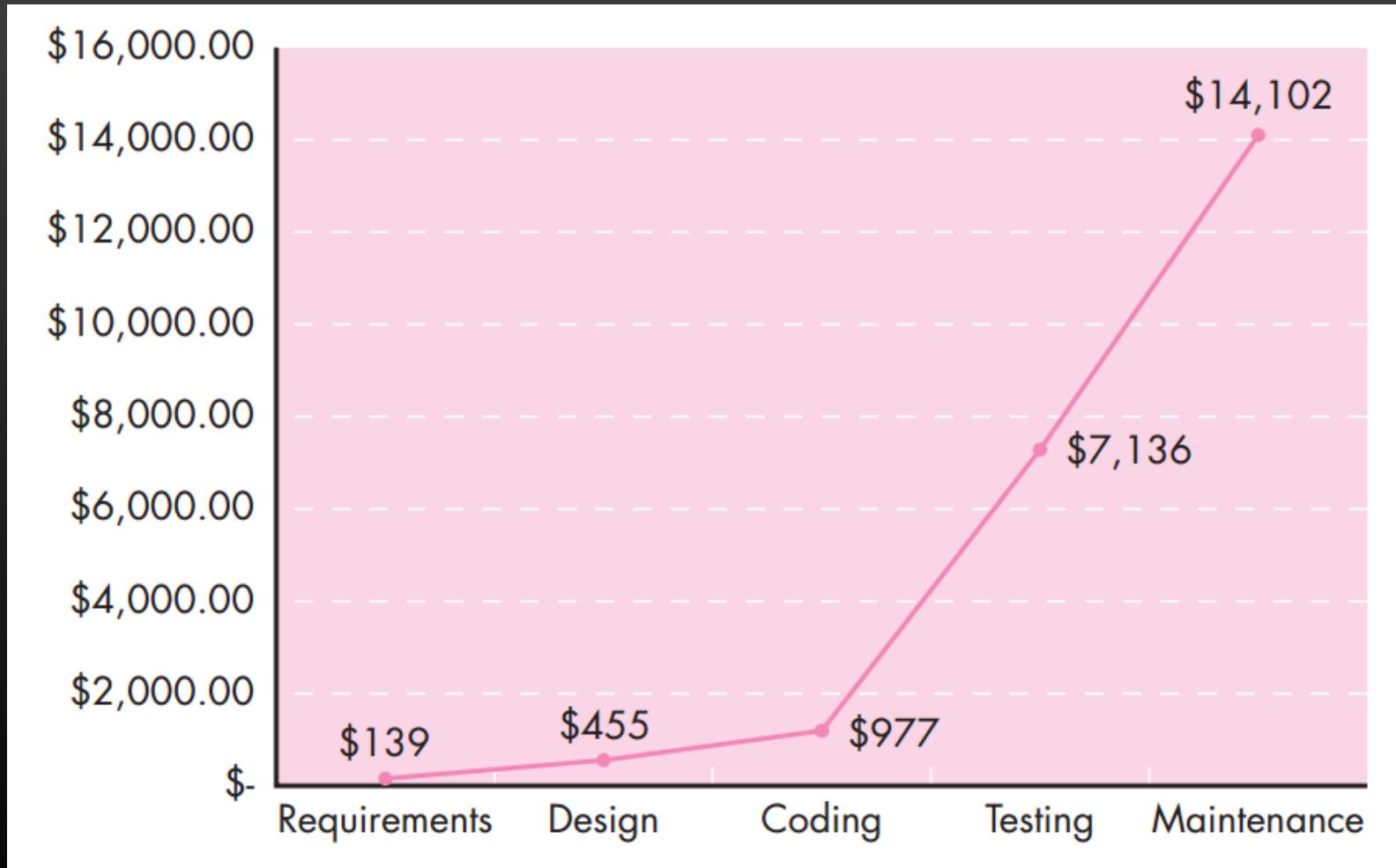
Why it may fail?

Real projects rarely follow the sequential flow that the model proposes

It is often difficult for the customer to state all requirements explicitly

The customer must have patience. A working version of the program(s) will not be available until late in the project time span

The cost of correcting an error raises exponentially along the sw lifecycle

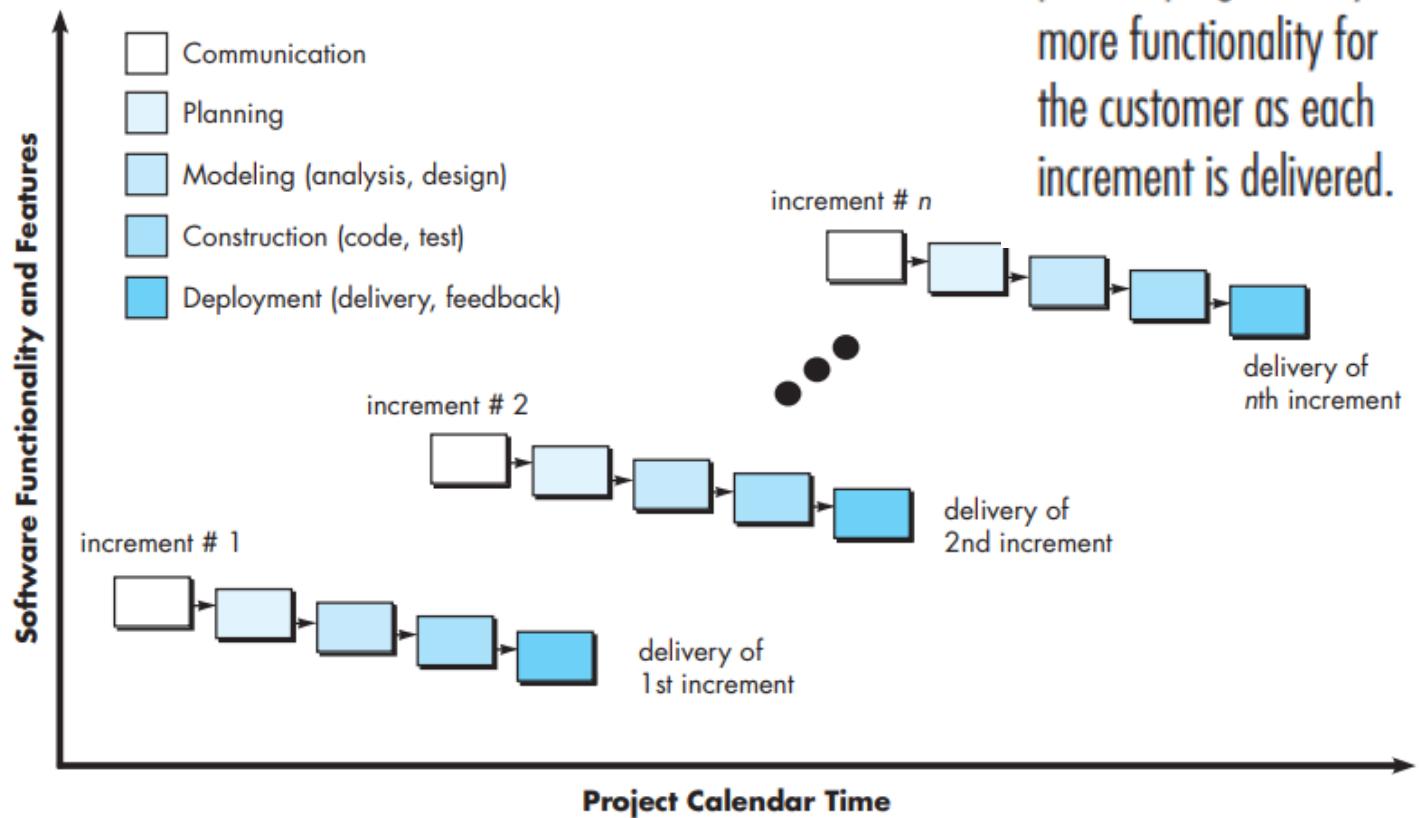


Boehm, B., and V. Basili, "Software Defect Reduction Top 10 List," IEEE Computer,
vol. 34, no. 1, January 2001, pp. 135-137. <http://doi.ieeecomputersociety.org/10.1109/2.962984>

The incremental model delivers a series of releases, called increments, that provide progressively more functionality for the customer as each increment is delivered.

FIGURE 4.3

The incremental model



Phased & incremental

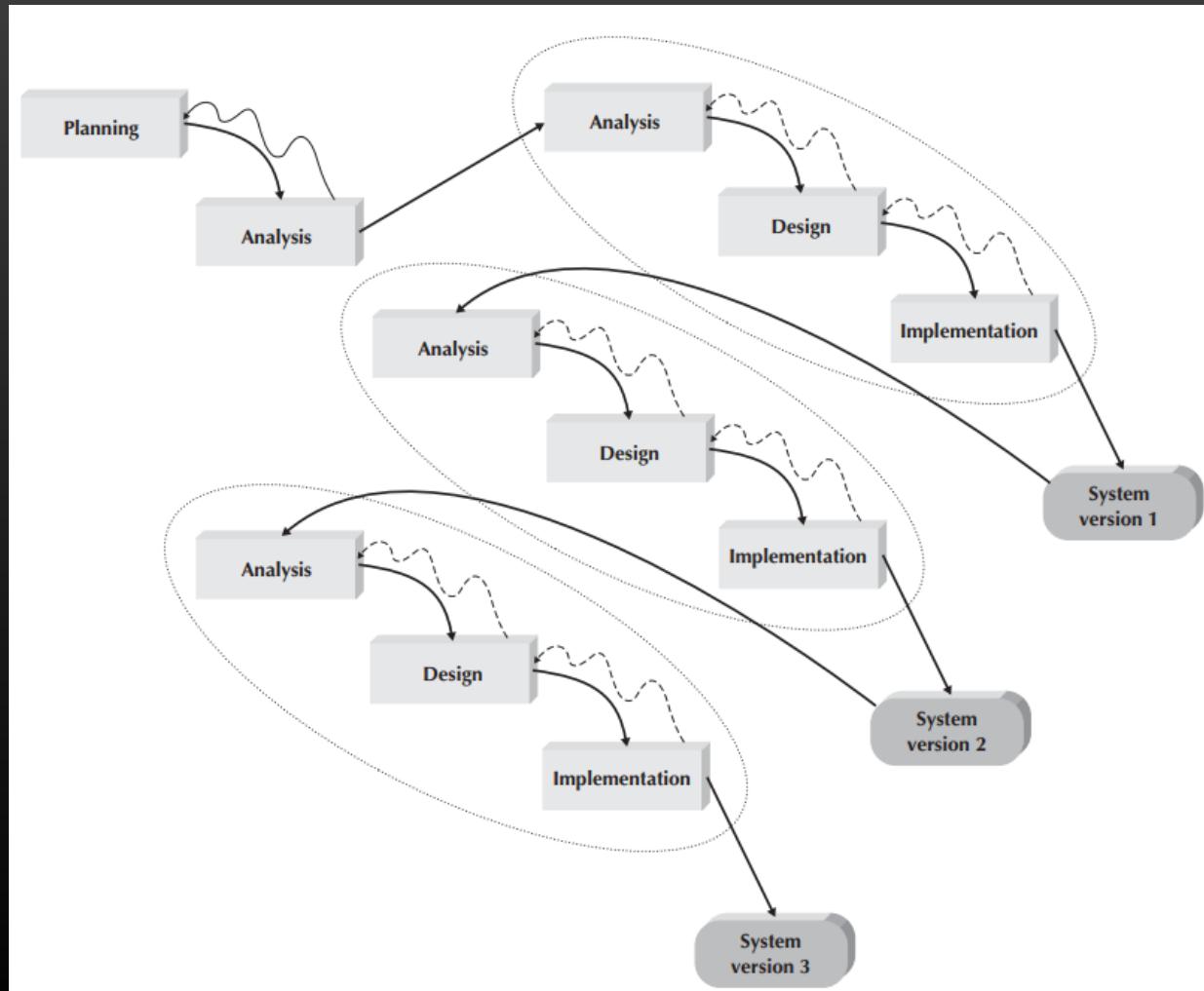
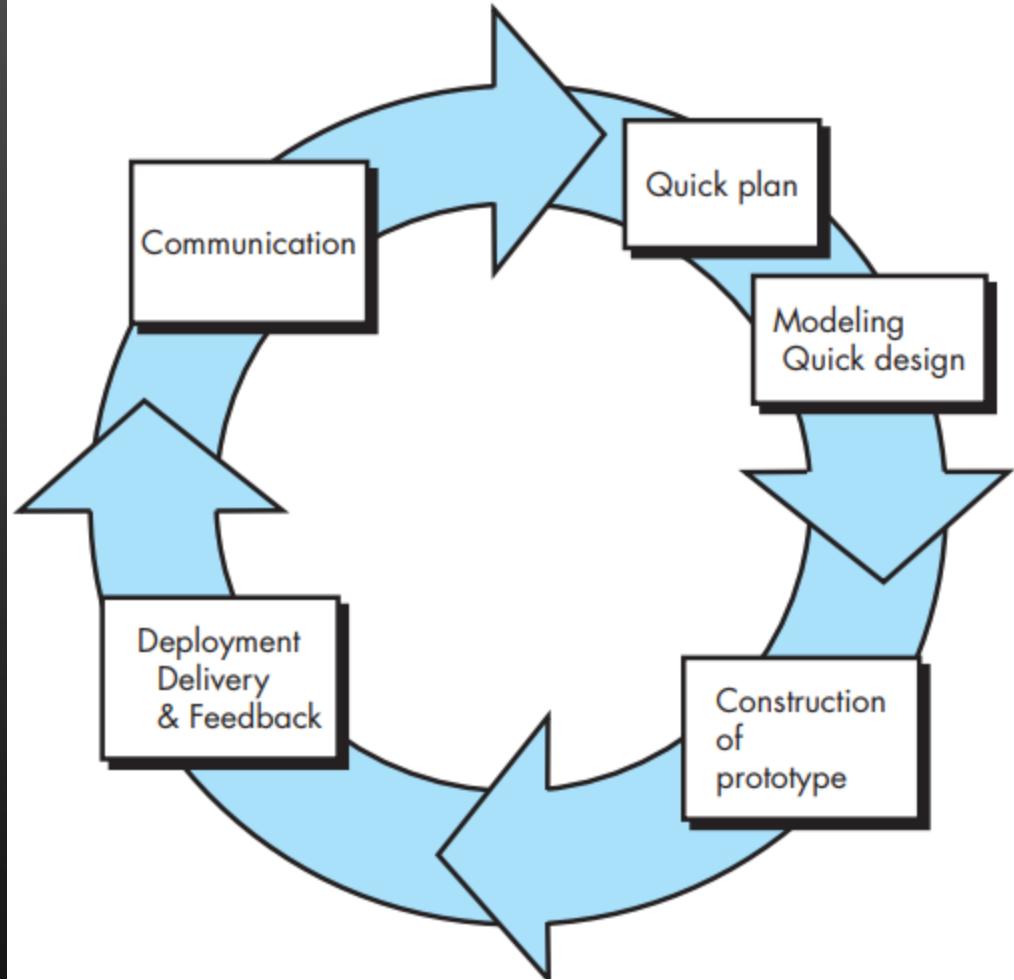


FIGURE 1-4 A Phased Development-Based Methodology

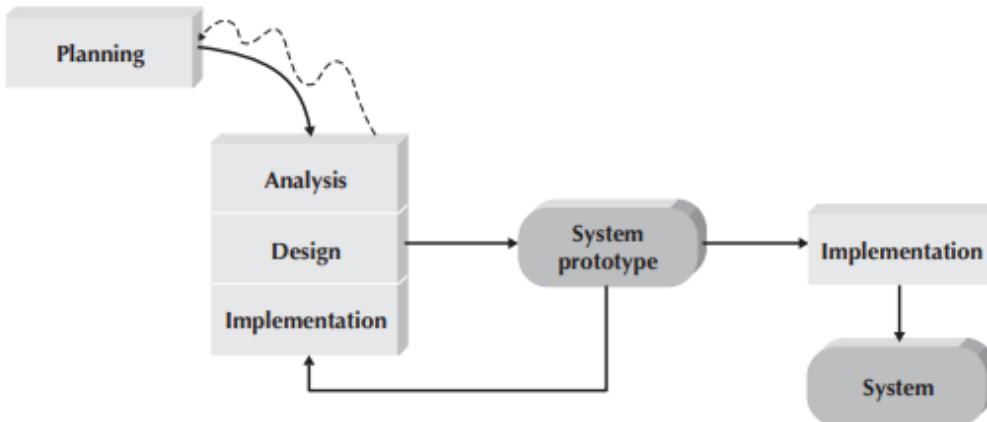
Evolutionary: prototyping



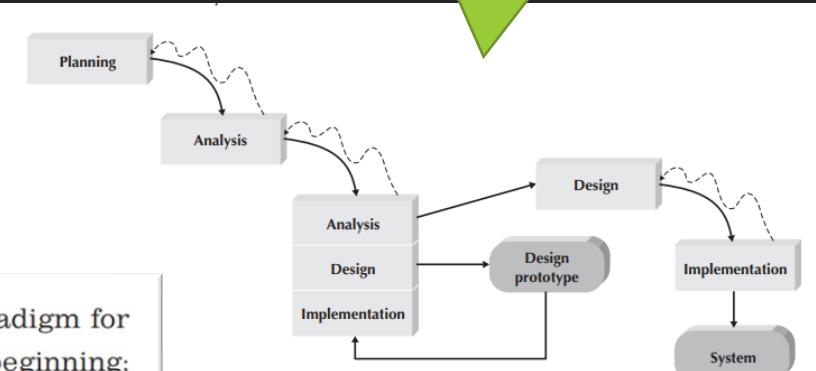
Although problems can occur, prototyping can be an effective paradigm for software engineering. The key is to define the rules of the game at the beginning; that is, all stakeholders should agree that the prototype is built to serve as a mechanism for defining requirements. It is then discarded (at least in part), and the actual software is engineered with an eye toward quality.

RAD: prototyping

FIGURE 1-5
A Prototyping-Based Methodology

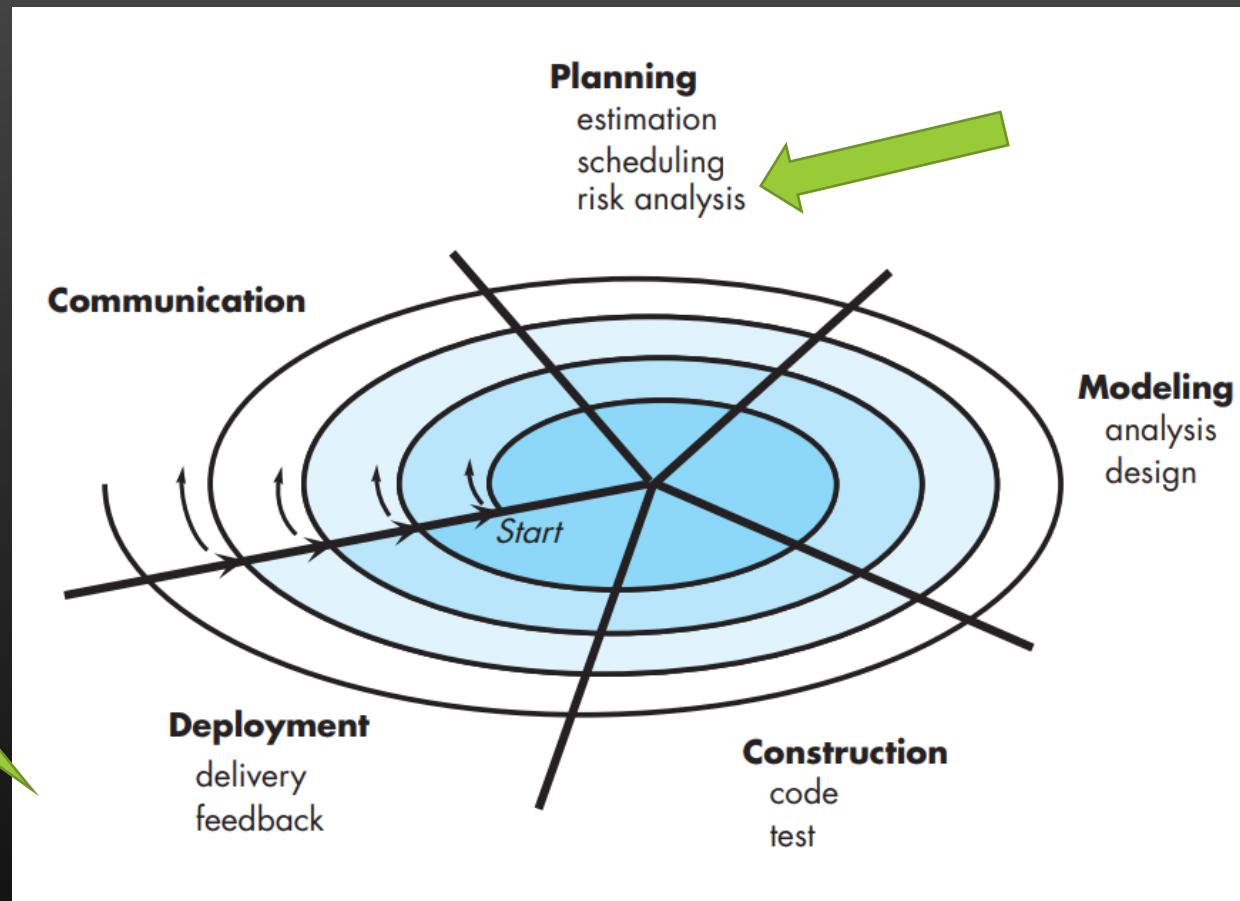


Throwaway prototyping: a variation of Prototyping in which the prototype is discarded (not used for the implementation).



Although problems can occur, prototyping can be an effective paradigm for software engineering. The key is to define the rules of the game at the beginning; that is, all stakeholders should agree that the prototype is built to serve as a mechanism for defining requirements. It is then discarded (at least in part), and the actual software is engineered with an eye toward quality.

Evolutionary: spiral model



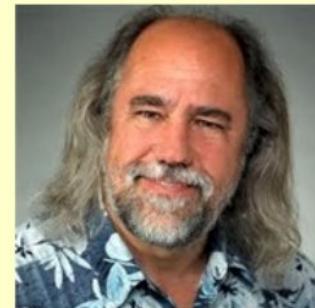
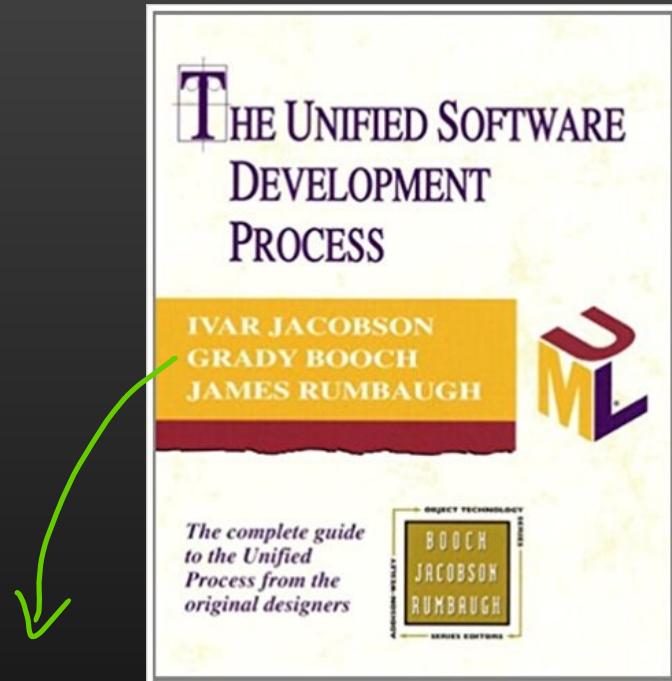
Using the spiral model, software is developed in a series of evolutionary releases. During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

Unified Process/Open Unified Process

Uma tentativa de um processo de genérico

Pode ser adaptado para projetos concretos

O OpenUP é uma versão "livre" do Unified Process



O SDLC é concretizado em processos de desenvolvimento

Um processo especifica:

O quê?

Quem?

Como?

Quando?

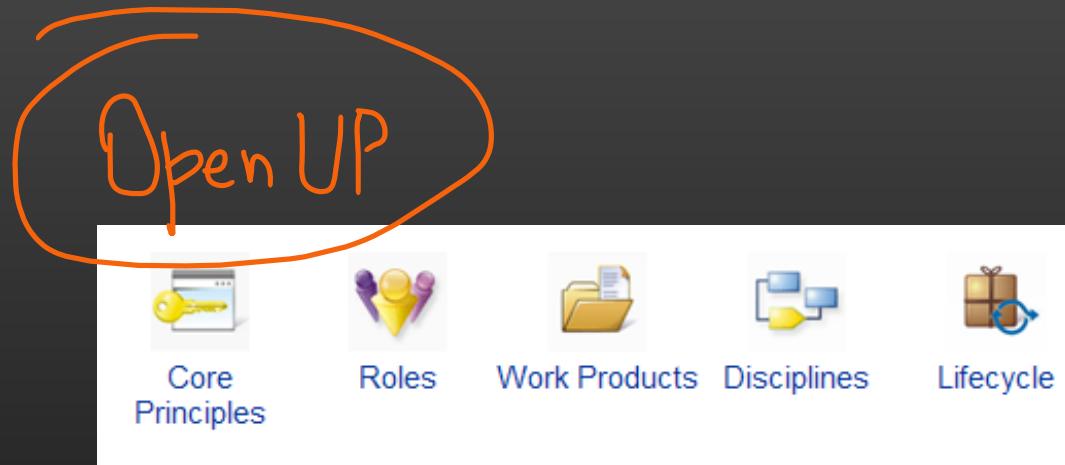
Um processo inclui:

Papéis

Fluxos de trabalho

Procedimentos

Modelos (dos resultados esperados)



http://sweet.ua.pt/ico/OpenUp/OpenUP_v1514/

Qual é o melhor processo?

Não há um único "melhor processo"

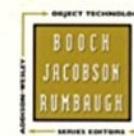
As organizações devem seleccionar
(ou personalizar) o seu processo.

THE UNIFIED SOFTWARE DEVELOPMENT PROCESS

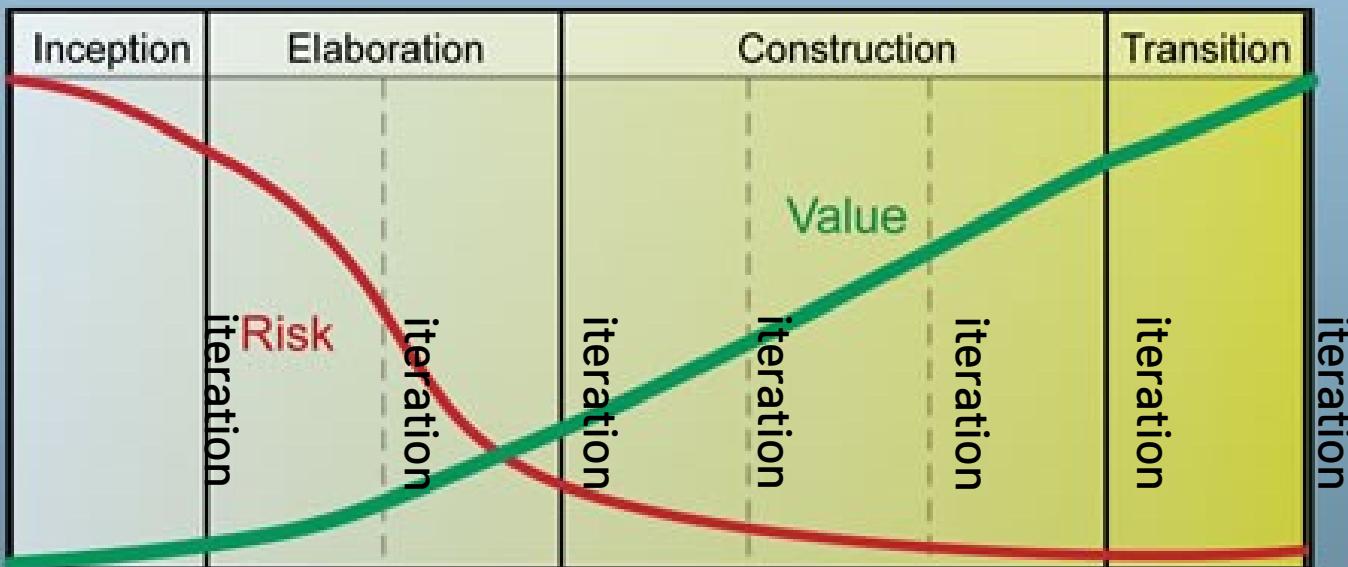
IVAR JACOBSON
GRADY BOOCHE
JAMES RUMBAUGH



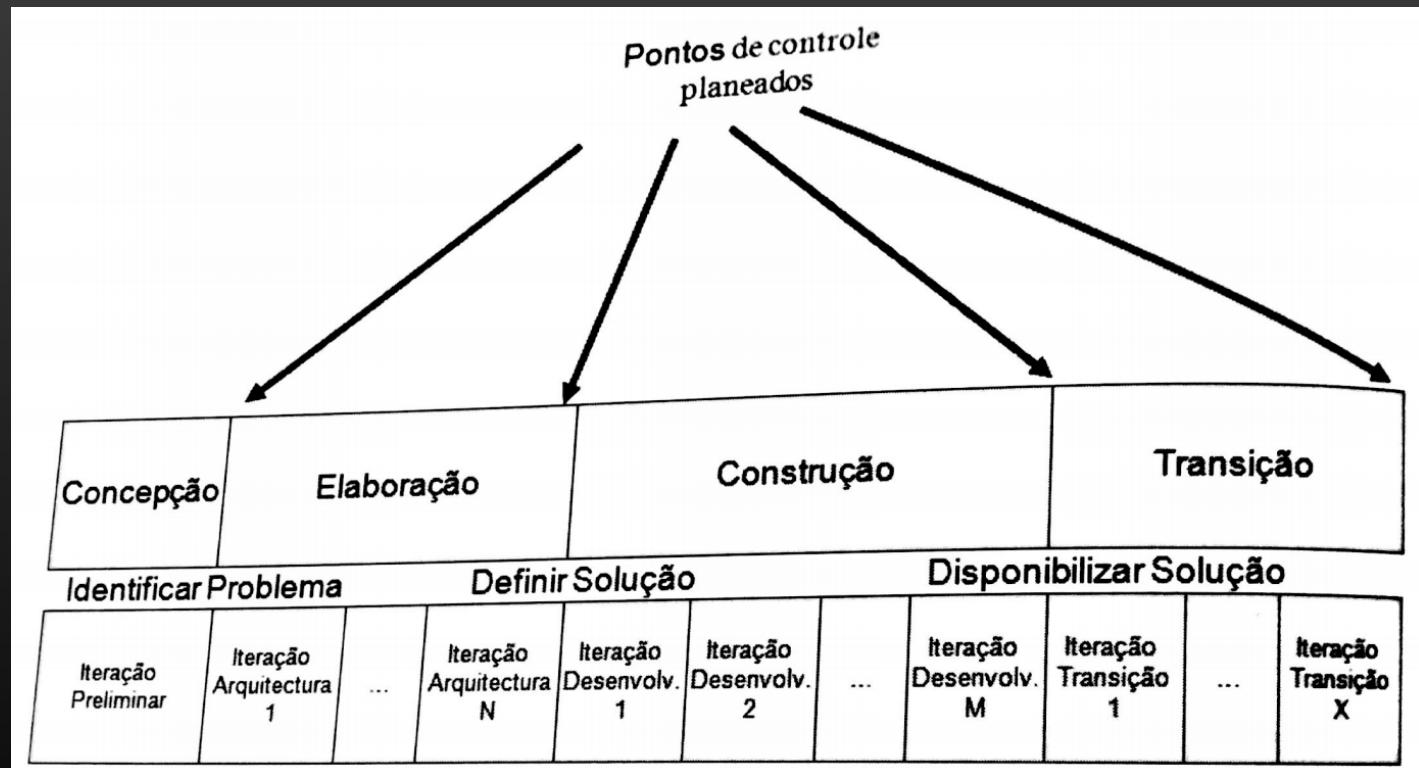
*The complete guide
to the Unified
Process from the
original designers*



Project Lifecycle



PT: Fases, iterações e pontos de controlo

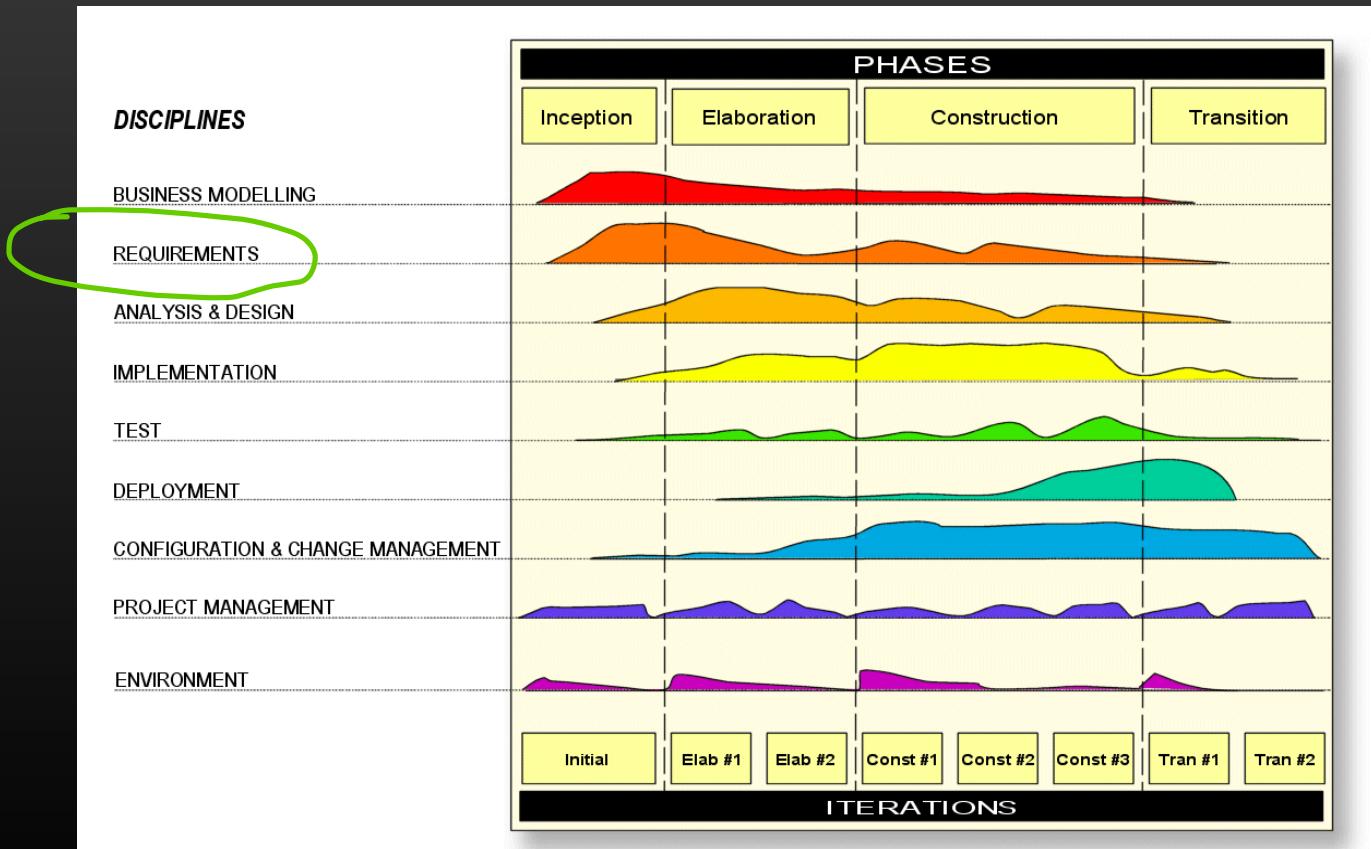


Visão geral do OpenUP/Unified Process

O UP oferece uma abordagem ao SDLC concebida como uma matriz, cruzando diferentes disciplinas técnicas com iterações (evoluções) no projecto.

(Nota: fases UP ≠ fases SDLC)

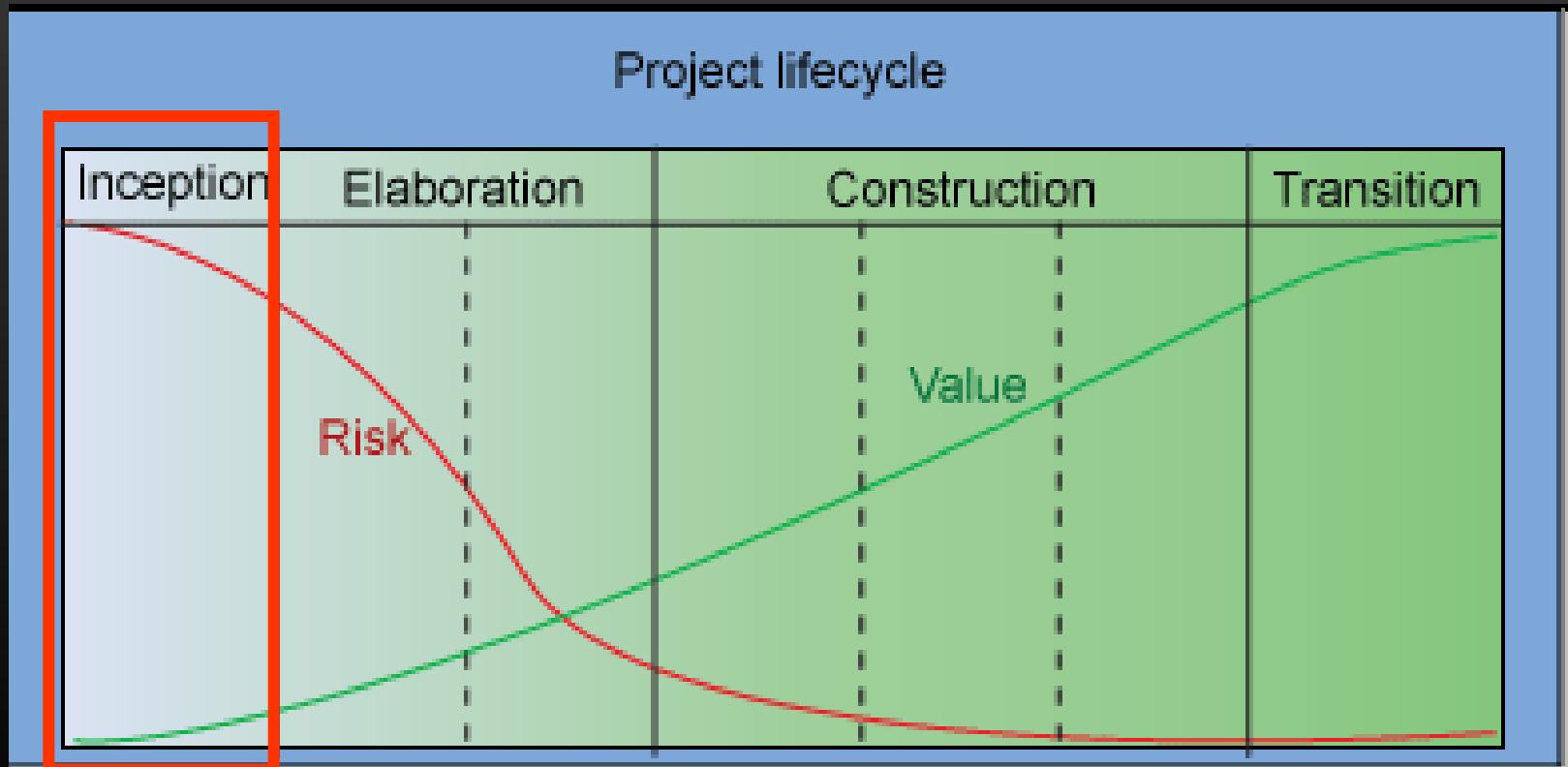
A análise dos requisitos é realizada principalmente no início do projeto (requisitos básicos), mas também durante as iterações (requisitos evolutivos).



The phases: Inception

Figura Project lifecycle

Do we agree on project scope and objectives, and whether or not the project should proceed?



Inception: Know What to Build

Typically one short iteration

Produce vision document and initial business case

Develop high-level project requirements

Initial use-case and (optional) domain models (10-20% complete)

Focus on what is required to get agreement on 'big picture'

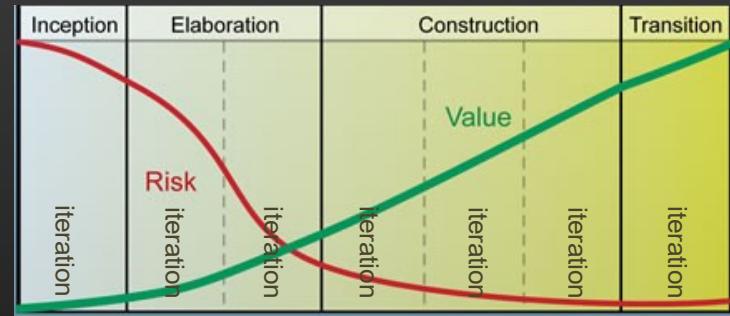
Manage project scope

Reduce risk by identifying key requirements

Acknowledge that requirements will change

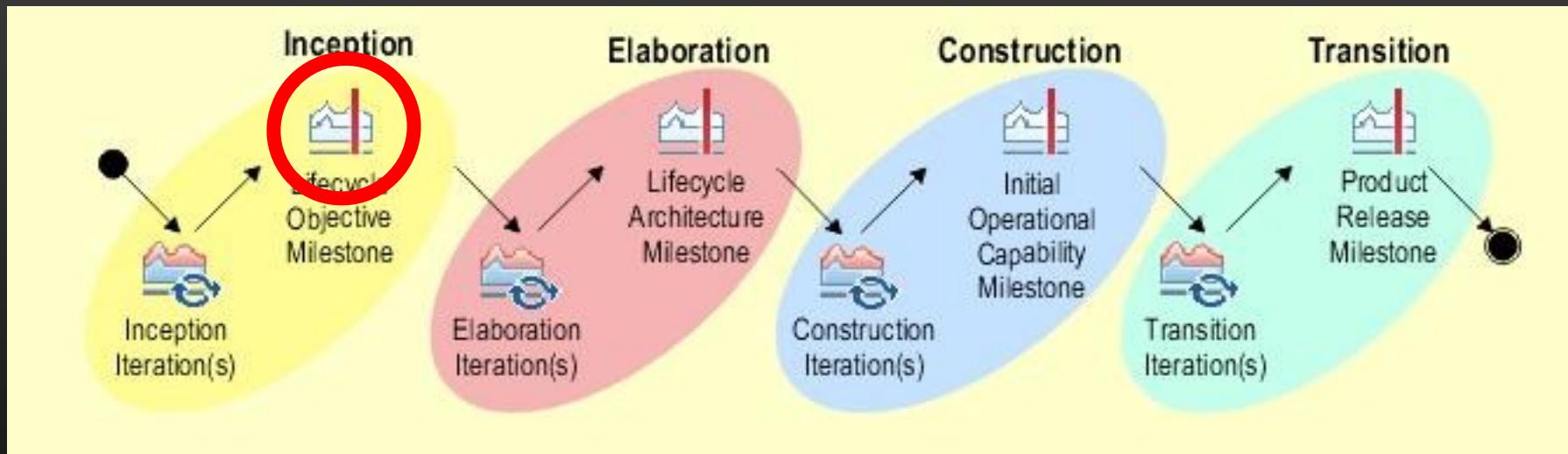
Manage change, use iterative process

Produce conceptual prototypes as needed



Credit: Per Kroll (IBM)

Milestone: Inception



Objectives Milestone. At this point, you examine the cost versus benefits of the project, and decide either to proceed with the project or to cancel it.

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

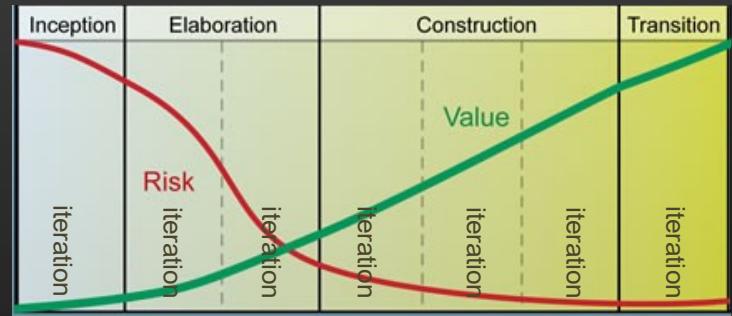
Define, implement and test interfaces of major components. Partially implement some key components.

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

Roughly 10% of code is implemented.

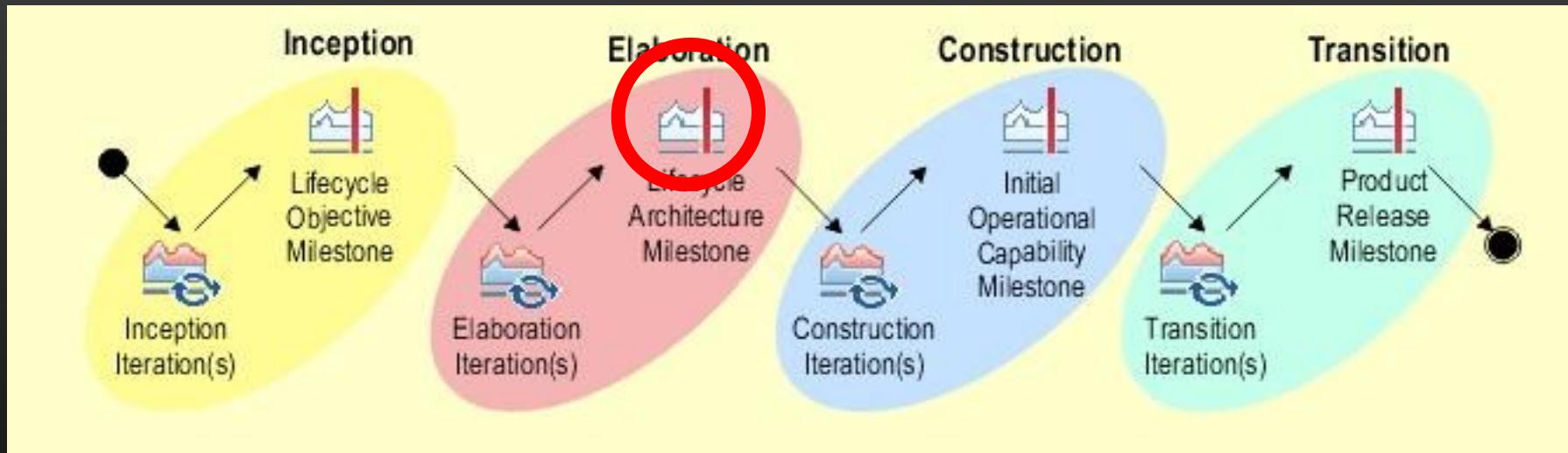
Drive architecture with key use cases

20% of use cases drive 80% of the architecture



Credit: Per Kroll (IBM)

Milestones: Elaboration



Architecture Milestone. At this point, a baseline of requirements is agreed to, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks. The milestone is achieved when the architecture has been validated.

Construction: Build The Product

Incrementally define, design, implement and test more and more scenarios

Incrementally evolve executable architecture to complete system

Evolve architecture as you go along

Frequent demonstrations and partial deployment

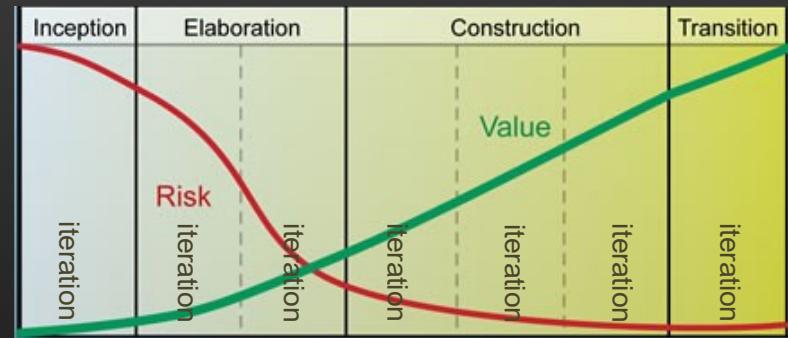
Partial deployment strategy depends greatly on what system you build

Daily build with automated build process

You may have to have a separate test team if you have

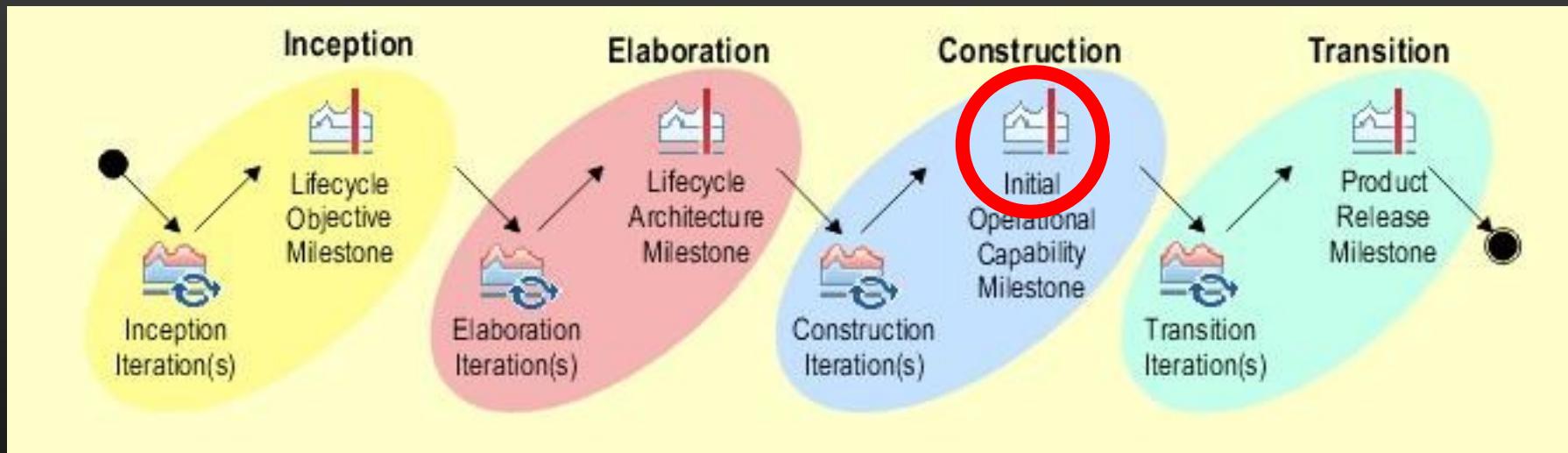
Complex test environments

Safety or mission critical systems



Credit: Per Kroll (IBM)

Milestones: Construction



Initial Operational Capability Milestone. At this point, the product is ready to be handed over to the transition team. All functionality has been developed and all alpha testing (if any) has been completed. In addition to the software, a user manual has been developed, and there is a description of the current release. The product is ready for beta testing.

Transition: Stabilize and Deploy

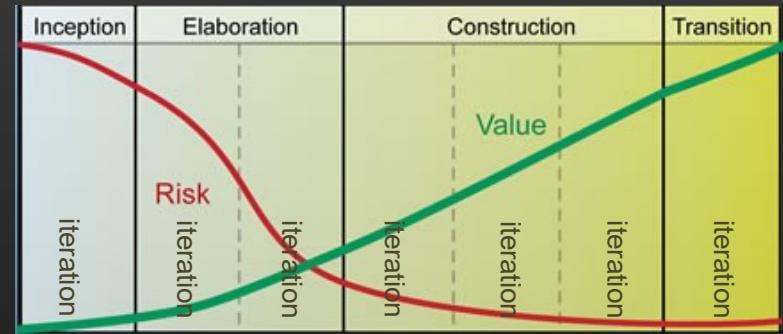
Project moves from focusing on new capabilities to **stabilizing** and tuning

Produce incremental 'bug-fix' releases

Update user manuals and deployment documentation

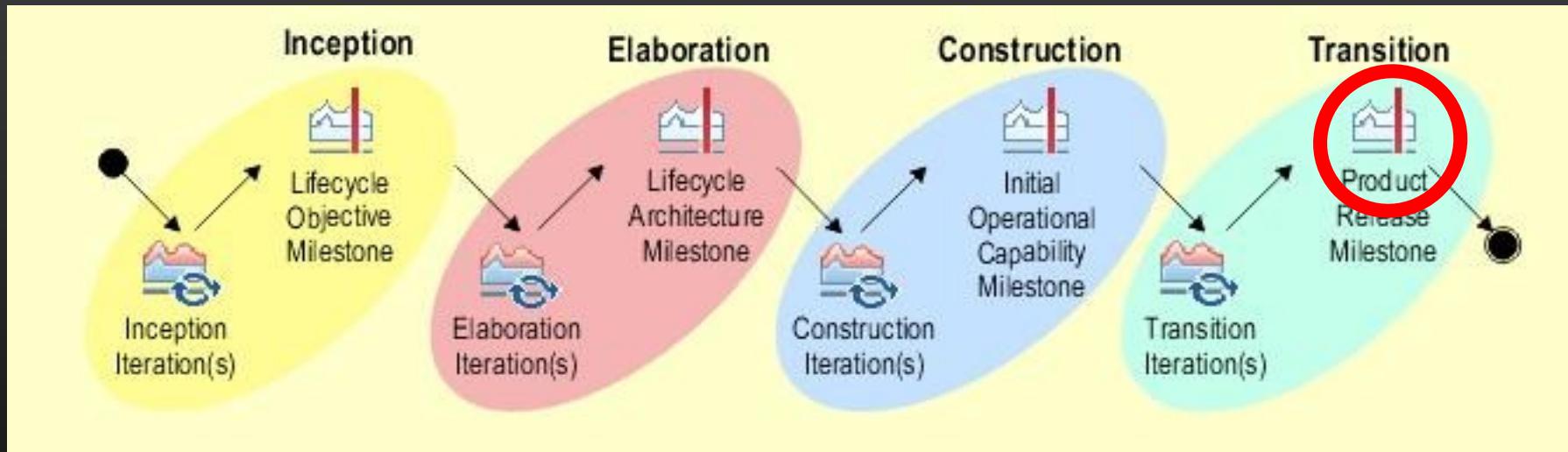
Execute cut-over

Conduct "post-mortem" project analysis



Credit: Per Kroll (IBM)

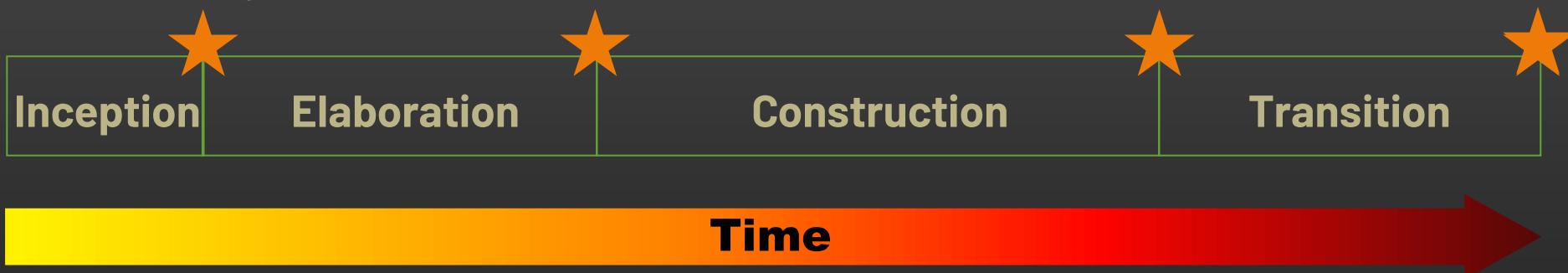
Milestones: Transition



Product Release Milestone. At this point, you decide if the objectives were met, and if you should start another development cycle. The Product Release Milestone is the result of the customer reviewing and accepting the project deliverables.

Recap main control points (lifecycle milestone)

Major Milestones



Inception: Agreement on overall scope

Vision, high-level requirements, business case
Not detailed requirements

Elaboration: Agreement on design approach and mitigation of major risks

Baseline architecture, key capabilities partially implemented
Not detailed design

Construction: Agreement on complete operational system

Develop a beta release with full functionality

Transition: Validate and deploy solution

Stakeholder acceptance, cutover to production

Na direção do desenvolvimento “ágil”

"Embrace change"

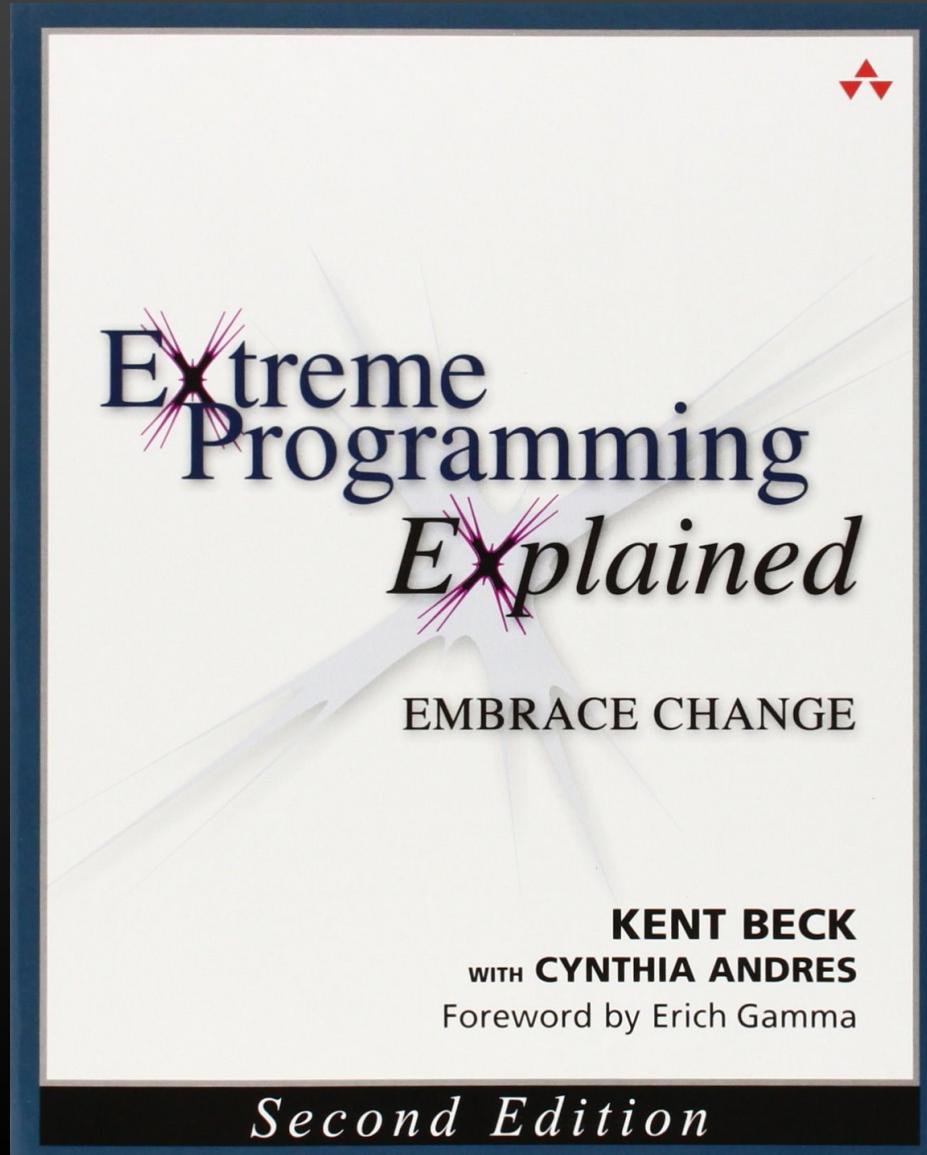
Em vez de:

- Lutar contra a inevitável a mudança que ocorre no desenvolvimento de software
- Tentando (sem sucesso) especificar, congelar e "assinar" num conjunto de requisitos congelados e conceber antes da implementação

desenvolvimento iterativo e evolutivo:

- Baseia-se numa atitude de abraçar a mudança e a adaptação como fatores inevitáveis e, na verdade, essenciais.
- Mas: isto não quer dizer que o desenvolvimento iterativo incentive um processo descontrolado e reativo.

<https://learning.oreilly.com/library/view/extreme-programming-explained/0321278658/>



Manifesto para o Desenvolvimento Ágil de Software.

Ao desenvolver e ao ajudar outros a desenvolver software,
temos vindo a descobrir melhores formas de o fazer.

Através deste processo começámos a valorizar:

Indivíduos e interacções mais do que processos e ferramentas

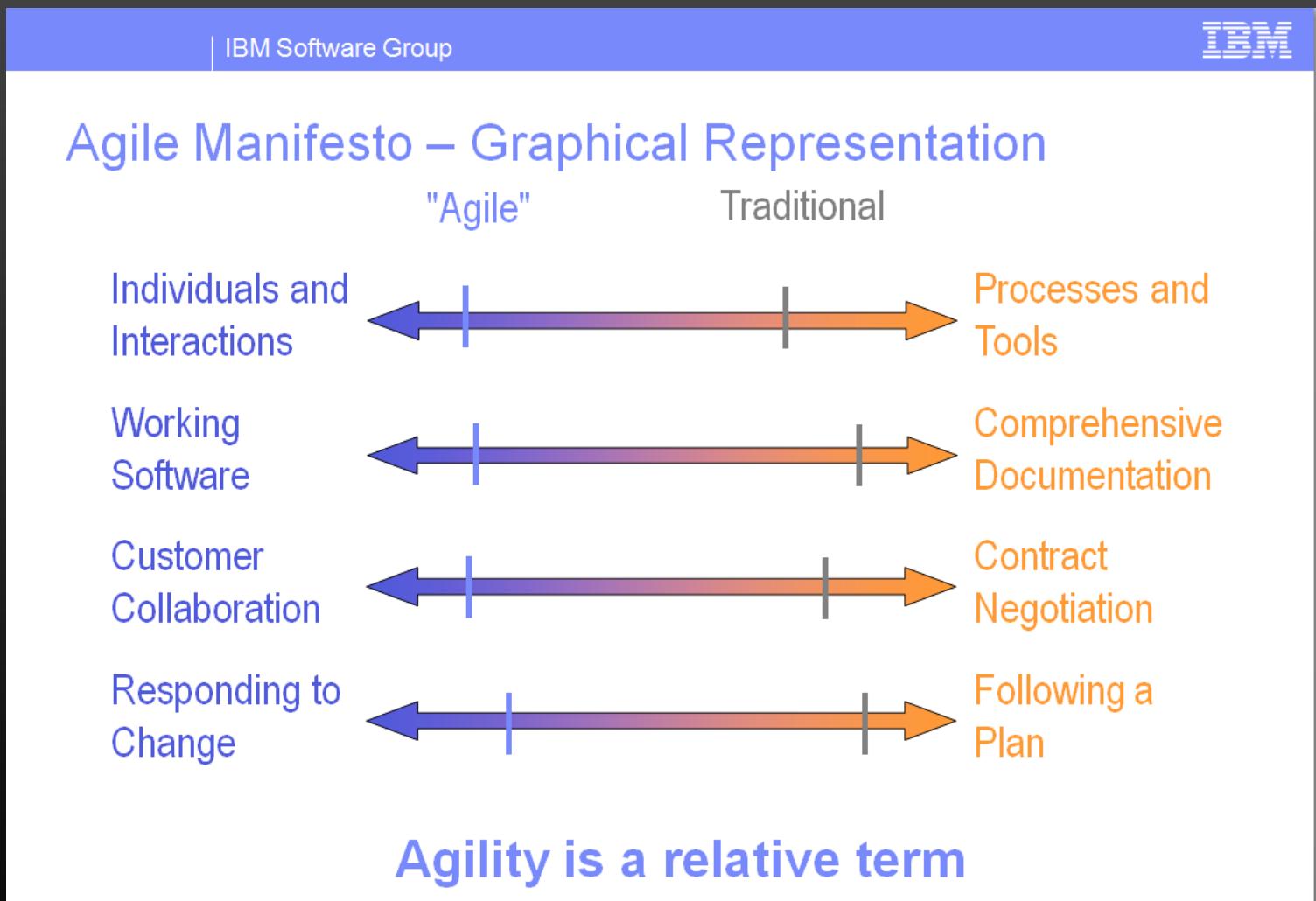
Software funcional mais do que documentação abrangente

Colaboração com o cliente mais do que negociação contratual

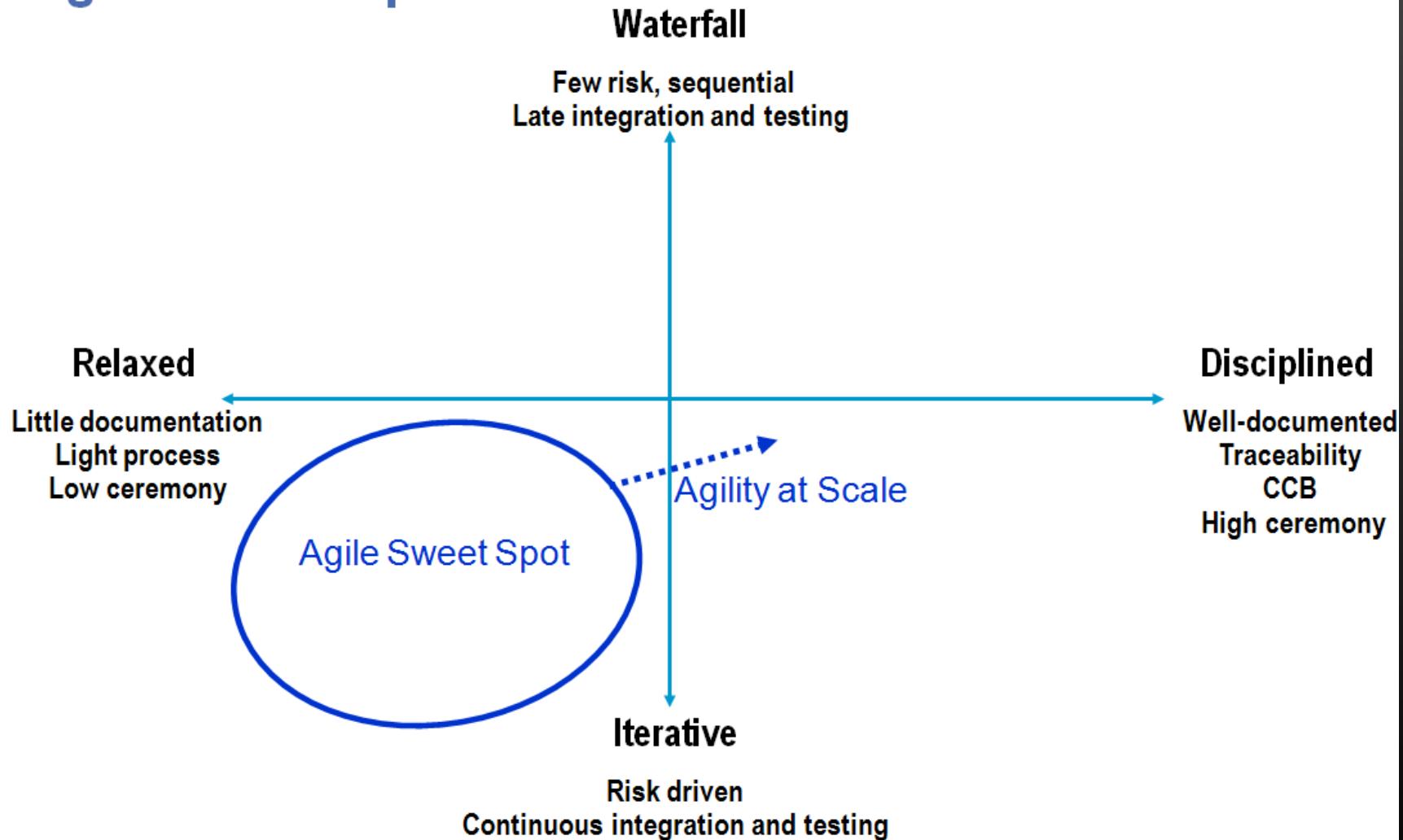
Responder à mudança mais do que seguir um plano

Ou seja, apesar de reconhecermos valor nos itens à direita,
valorizamos mais os itens à esquerda.

O desenvolvimento ágil de software

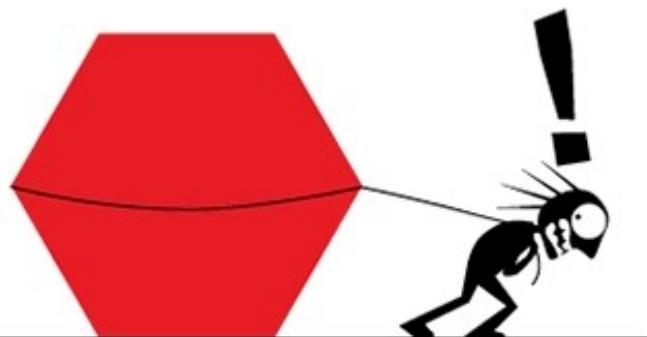


Agile Sweet Spot

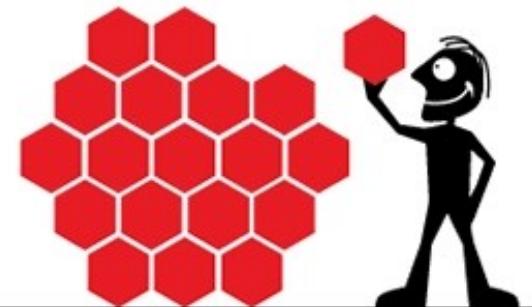


Credit: Per Kroll (IBM)

One project? Micro-projects?



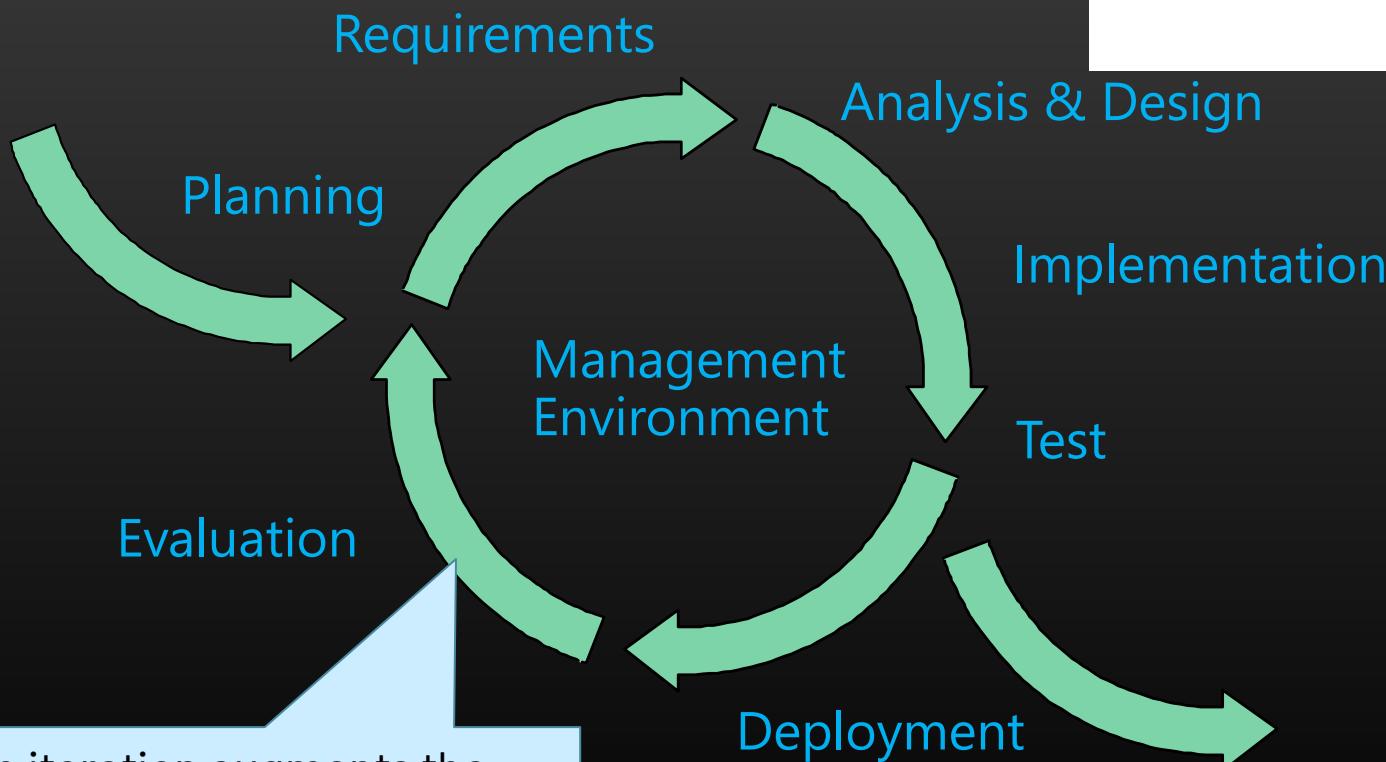
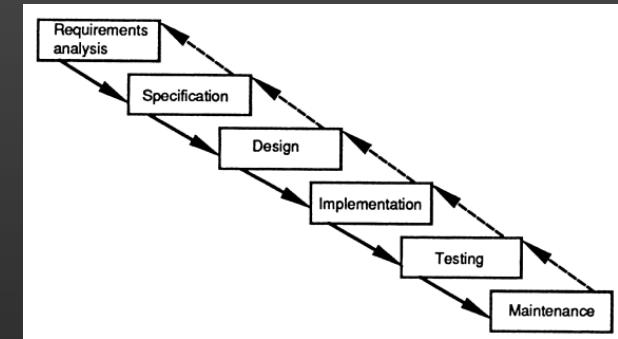
*'This project has got so big,
I'm not sure I'll be able to deliver it!'*



*'It's so much better delivering this
project in bite-sized sections'*

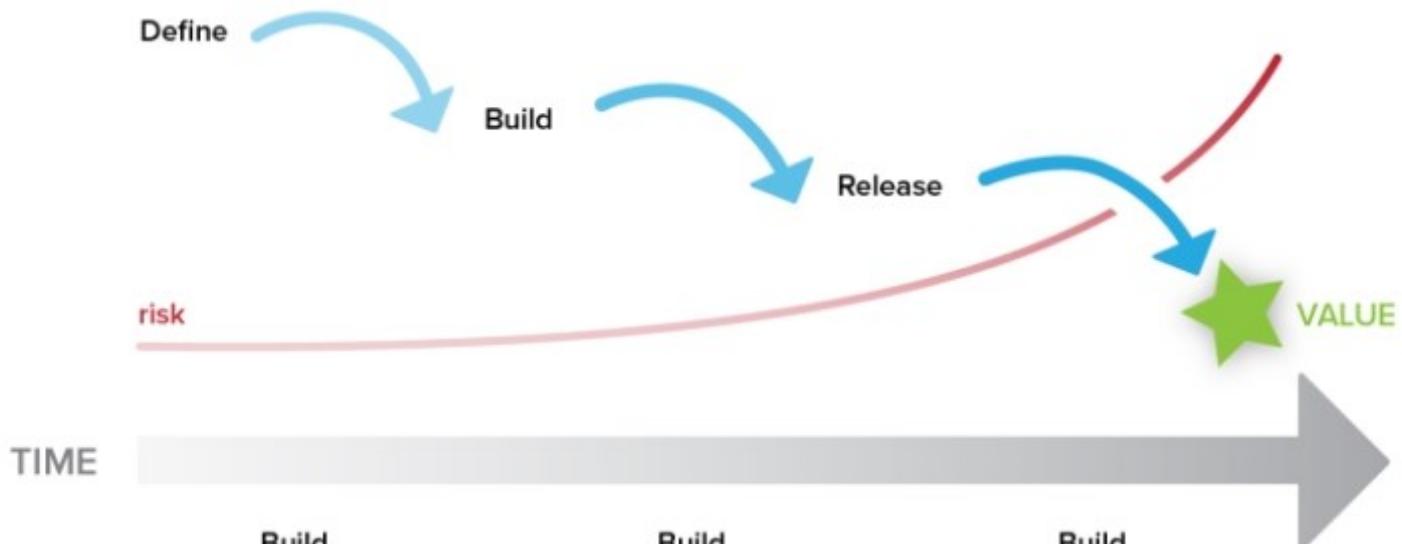
<https://blog.ganttpro.com/en/waterfall-vs-agile-with-advantages-and-disadvantages/>

Iterative development focuses on short and value-oriented cycles → Agile

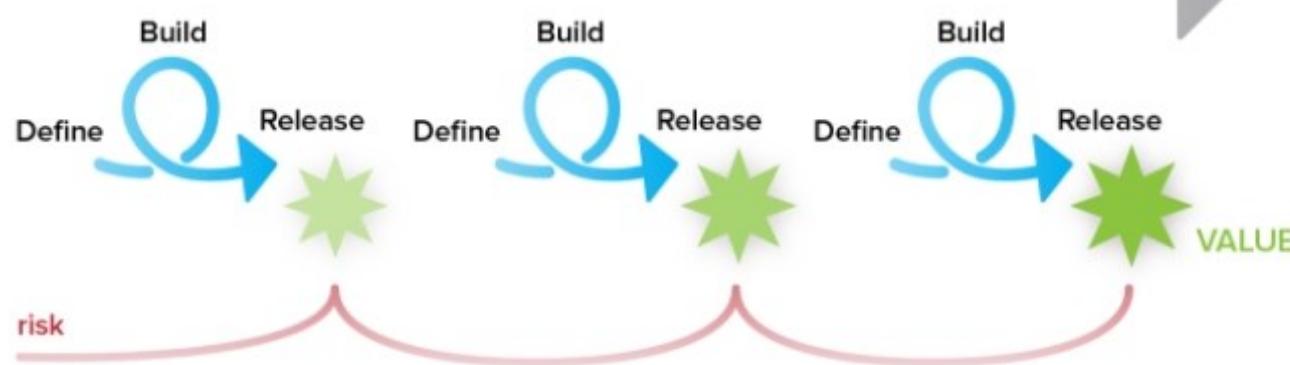


Each iteration augments the solution by integrating some executable result

WATERFALL

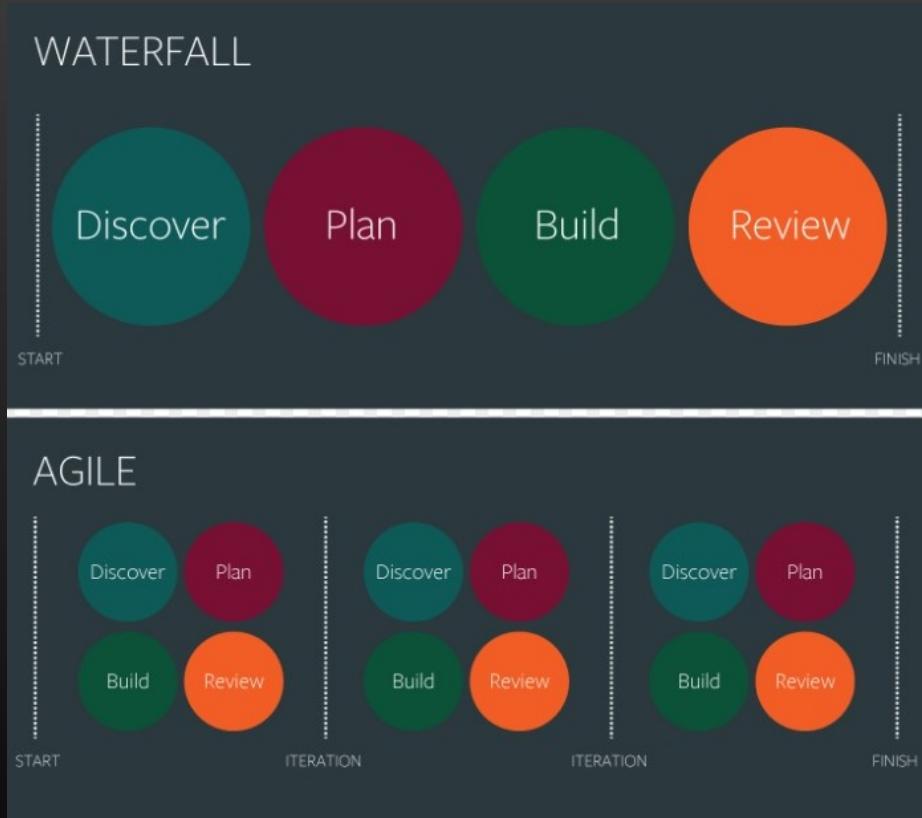


AGILE



<https://blog.ganttpro.com/en/waterfall-vs-agile-with-advantages-and-disadvantages/>

To be (agile) or not to be...



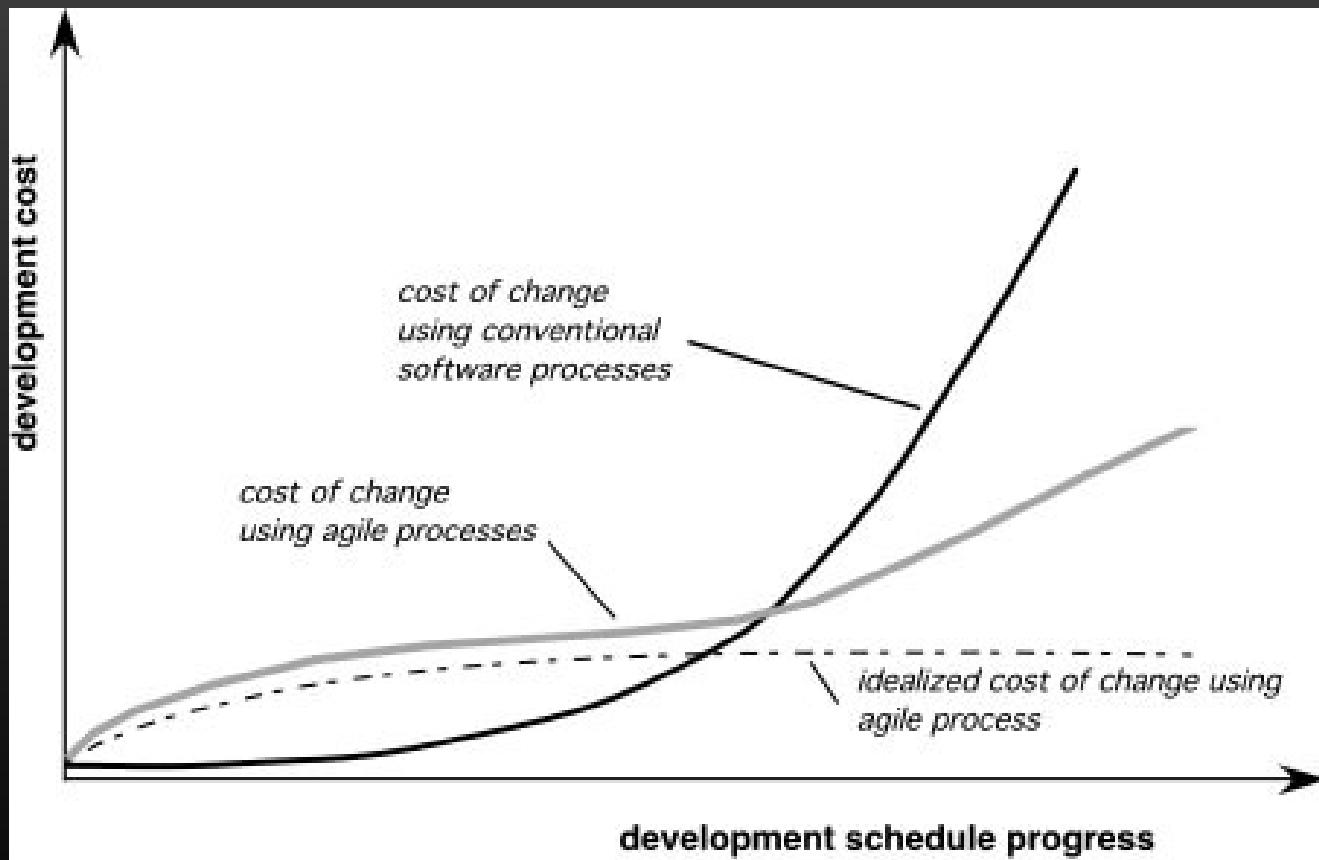
Iterative development (short cycles) vs linear development through stages?

Frequent business interaction vs fluctuations in stakeholder's participation?

Best to have good collaboration or a good plan?

Welcome changes to mitigate risk vs avoid changes to control risk?

Agility and the Cost of Change



The cost of change increases nonlinearly as the project progresses

Agility: key points

What is “Agility” in software development?

Effective (rapid and adaptive) response to change

Effective communication among all stakeholders

Drawing the customer onto the team

Organizing a team so that it is in control of the work performed

Yielding ...

Rapid, incremental delivery of software

Is OpenUP an agile process?

An Agile Process...

Is driven by customer descriptions of what is required (scenarios)

Recognizes that plans are short-lived

Develops software iteratively with a heavy emphasis on construction activities

Delivers multiple ‘software increments’

Adapts as changes occur

Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Pressman'15] – Chap. 4, 5	<ul style="list-style-type: none">• [Dennis'15] – Chap 1.

41951- ANÁLISE DE SISTEMAS

Determinação de requisitos e OpenUP

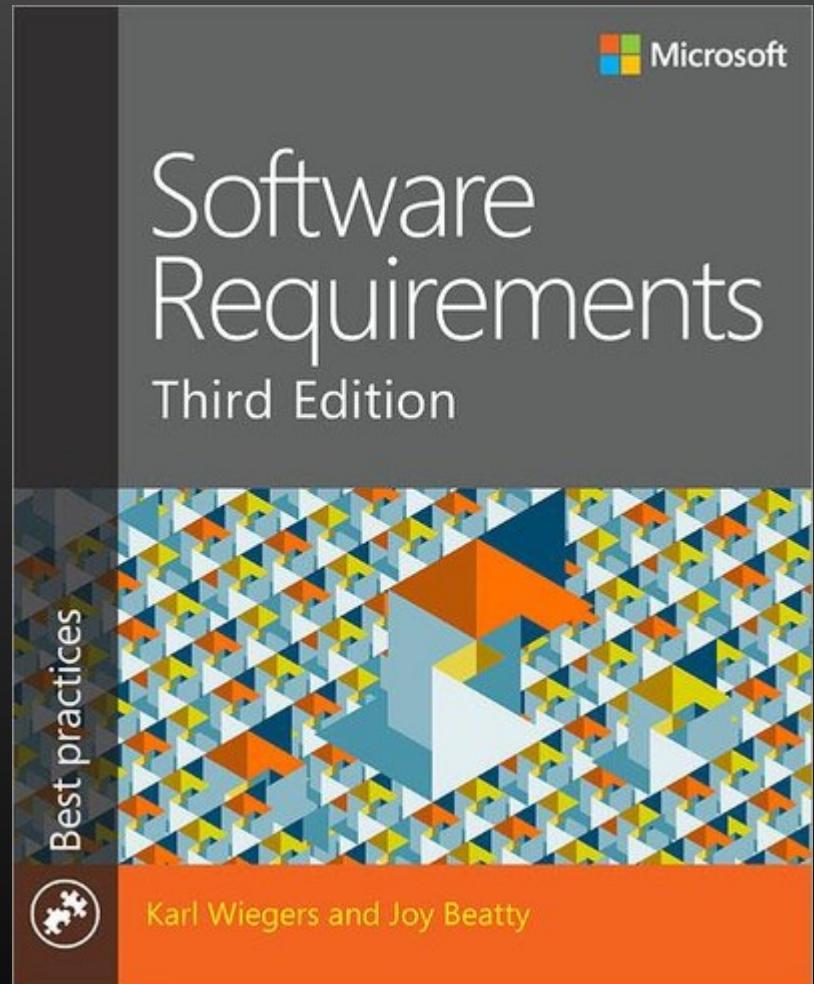
Ilídio | v2022/04/19

Objetivos de aprendizagem

- Distinguir requisitos funcionais e não funcionais
- Enumerar técnicas de recolha de requisitos e argumentar quando usar cada uma
- Descrever técnicas de documentação de requisitos
- Explicar a relação entre requisitos e casos de utilização
- Identificar as atividades e disciplinas relacionadas com os requisitos no OpenUP

Requisitos...

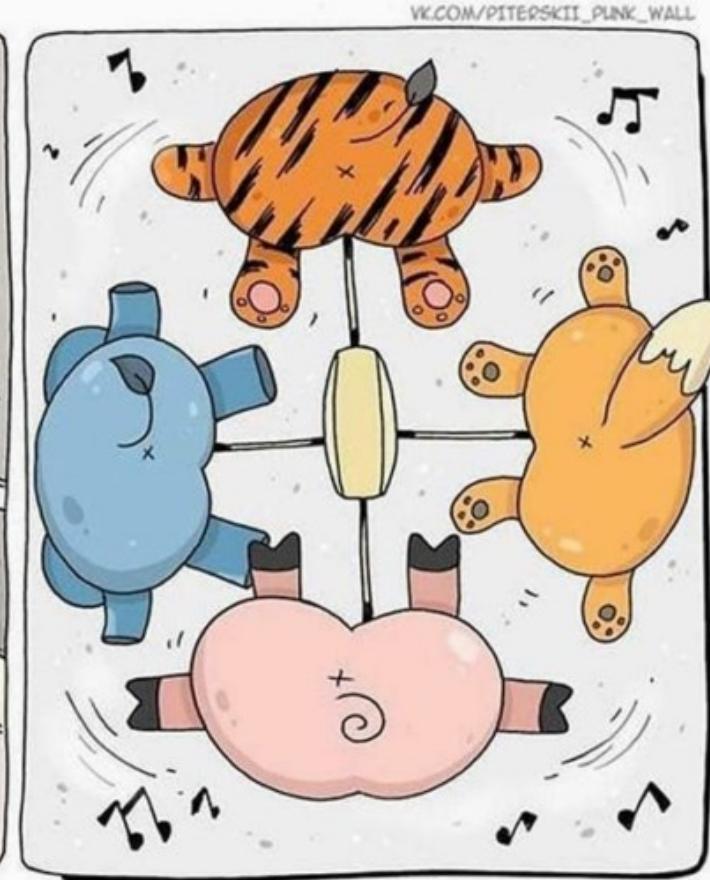
Como descobrir o que é para construir?



<https://learning.oreilly.com/library/view/software-requirements/9780735679658/>

Clients

Users

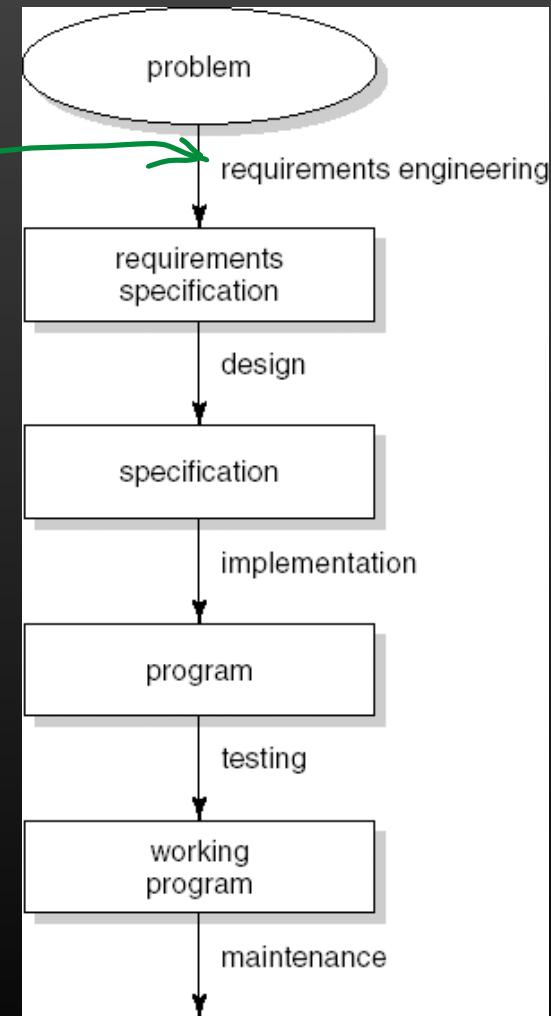


<https://www.designershumor.com/2019/05/21/ux-is-important/>

Atividades e resultados no ciclo de vida do software



Representação do SDLC, com explicitação de QA (#5) e Manutenção (#6).

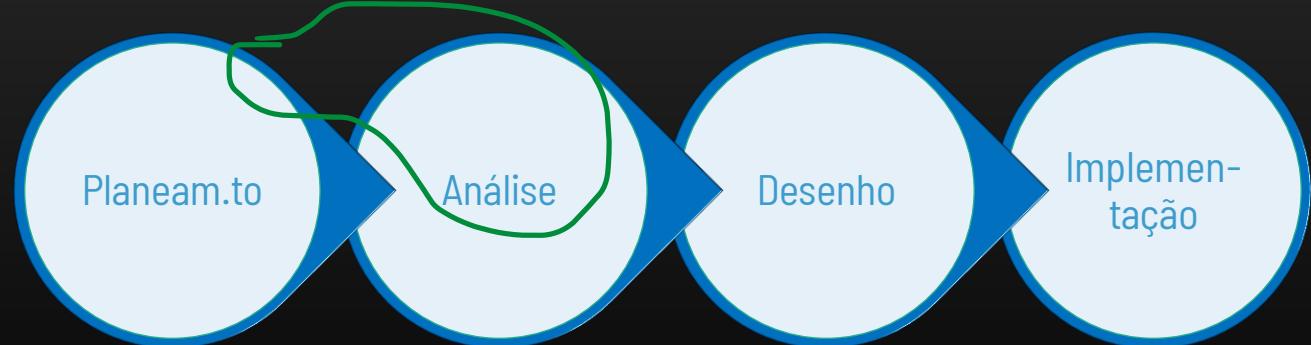


Determinação dos requisitos

Um passo crítico no SDLC

As alterações podem ser feitas facilmente nesta fase

A maioria (>50%) das falhas do sistema são devidas a problemas com os requisitos



Adapted from: Dennis et al., "Systems Analysis and Design: An Object-Oriented Approach with UML", 5th ed.

Porquê realizar actividades de "determinação de requisitos"?

Objetivo

Converter pedidos do negócio (*alto nível*) em requisitos detalhados que possam ser utilizados como inputs para a criação de modelos

O que é um requisito?

Uma declaração do que o sistema deve fazer ou uma característica que deve ter.

evoluirá mais tarde para uma descrição técnica de como o sistema será implementado.

"Missão" do projecto

Desenvolver um sistema que esteja em conformidade com os requisitos estabelecidos (capacidades e condições de operação)

Adapted from: Dennis et al, "Systems Analysis and Design: An Object-Oriented Approach with UML", 5th ed.

Requisitos são muitas vezes elencados em listas “o sistema deve...”

#	Requisito
RF.1	O sistema deve permitir a um profissional criar um novo pedido de adesão , em auto-serviço, na web.
RF.2	O sistema deve gerar credenciais temporárias para os pedidos de adesão e enviá-las, por email, aos solicitantes.
RF.3	O sistema deve permitir a pesquisa de cheques-dentista (emitidos) por número de utente do SNS.
RNF.1	As pesquisas de cheques-dentista têm de retornar resultados em <5 segundos ou um evento de tempo expirado.
...	

Os atributos de qualidade são necessários na definição do produto



As mesmas
capacidades
funcionais, mas
atributos de
qualidade diferentes

I Oliveira



Quais são os tipos de requisitos?

Requisitos funcionais

Funcional: refere-se a comportamento → um processo computacional ou tratamento de dados (*as coisas/tarefas que o sistema é capaz de realizar*)

E.g.:

- O sistema deve permitir pesquisar os pacientes pelo nome, usando pelo menos duas palavras (que devem ocorrer em sequência, em qualquer parte do nome).

Requisitos não funcionais

Não-funcional: diz respeito a uma qualidade ou propriedade do sistema (*quão bem deve o sistema fazer a operação*)

E.g.:

- As pesquisas de utentes, por nome, devem retornar os resultados em <1 seg.

A formulação “*The system shall...*” é comum nos RF.

Exemplo de Definição de Requisitos

Categorias comuns:

- Usabilidade
- Desempenho
- Segurança
- Restrições operacionais
- ...

Nonfunctional Requirements

1. Operational Requirements

- 1.1. The system will operate in Windows environment.
- 1.2. The system should be able to connect to printers wirelessly.
- 1.3. The system should automatically back up at the end of each day.

2. Performance Requirements

- 2.1. The system will store a new appointment in 2 seconds or less.
- 2.2. The system will retrieve the daily appointment schedule in 2 seconds or less.

3. Security Requirements

- 3.1. Only doctors can set their availability.
- 3.2. Only a manager can produce a schedule.

4. Cultural and Political Requirements

- 4.1. No special cultural and political requirements are anticipated.

Functional Requirements

1. Manage Appointments

- 1.1. Patient makes new appointment.
- 1.2. Patient changes appointment.
- 1.3. Patient cancels appointment.

2. Produce Schedule

- 2.1. Office Manager checks daily schedule.
- 2.2. Office Manager prints daily schedule.

3. Record Doctor Availability

- 3.1. Doctor updates schedule

Agrupamento do RF depende do problema específico

Adapted from: Dennis et al,
"Systems Analysis and Design:
An Object Oriented Approach
with UML", 5th ed.

O que pode estar envolvido no "desempenho" (atributo de qualidade)?

TABLE 14-2 Some aspects of performance

Performance dimension	Example
Response time	Number of seconds to display a webpage
Throughput	Credit card transactions processed per second
Data capacity	Maximum number of records stored in a database
Dynamic capacity	Maximum number of concurrent users of a social media website
Predictability in real-time systems	Hard timing requirements for an airplane's flight-control system
Latency	Time delays in music recording and production software
Behavior in degraded modes or overloaded conditions	A natural disaster leads to a massive number of emergency telephone system calls

Credit: Wiegers '13

Os atributos de qualidade do software, segundo Wieger (*-ies)

TABLE 14-1 Some software quality attributes

Analista/requisitos

External quality	Brief description
Availability	The extent to which the system's services are available when and where they are needed
Installability	How easy it is to correctly install, uninstall, and reinstall the application
Integrity	The extent to which the system protects against data inaccuracy and loss
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictably the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure
Robustness	How well the system responds to unexpected operating conditions
Safety	How well the system protects against injury or damage
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system
Internal quality	Brief description
Efficiency	How efficiently the system uses computer resources
Modifiability	How easy it is to maintain, change, enhance, and restructure the system
Portability	How easily the system can be made to work in other operating environments
Reusability	To what extent components can be used in other systems
Scalability	How easily the system can grow to handle more users, transactions, servers, or other extensions
Verifiability	How readily developers and testers can confirm that the software was implemented correctly

Apesar da implementação da segurança ter aspectos funcionais, é comum classificar os requisitos de segurança como RNF

Credit: Wiegers '13

Funcional ou não-funcional: como distinguir?

Requisitos funcionais

captam o comportamento pretendido

- Serviços e funções que o sistema deverá desempenhar

Podem ser encontrados em descrições de cenários de interação (e.g.: use cases)

Podem ser complementados com diagramas de comportamento: actividades, sequência,...

Requisitos não-funcionais

Restrições globais/qualidades operacionais no software

- E.g.: facilidade de utilização, portabilidade, robustez,...

a.k.a. atributos de qualidade, uma vez que se referem ao grau que se espera de uma certa qualidade pretendida

Normalmente, não se limitam a uma função/módulo, mas sim a uma característica transversal

FURPS: categorias "clássicas" para os requisitos

Functionality

- avaliado através da apreciação do conjunto de capacidades funcionais do programa.

Usability

- avaliados considerando factores humanos, estética geral, consistência, e documentação.

Reliability

- avaliado através da medição da frequência e gravidade das falhas, da exactidão dos resultados de saída, do tempo médio até à falha (MTTF), da capacidade de recuperação das falhas

Performance

- medida utilizando velocidade de processamento, tempo de resposta, consumo de recursos, rendimento e eficiência.

Supportability

- combina capacidade de extensão, capacidade de **adaptação e de manutenção** e, além disso, facilidade de teste, compatibilidade, facilidade de configuração, facilidade de instalação de um sistema e facilidade de localização de problemas.

Requirements

STRQ1: Want to be able to transfer funds from other accounts (not necessarily held with this firm) to a trading account.

STRQ2: State and federal regulations require monthly reports of account activity. Refer to specification RUFS-1234 for details of the information required.

STRQ3: The system should allow the use of any browser.

STRQ4: Customers want to manage their retirement funds.

STRQ5: Must be able to upgrade the system without taking it offline.

STRQ6: The system should allow traders to trade in multiple markets across the world.

STRQ7: Must be able to provide convenient answers to customer's most common questions.

STRQ8: The system must provide a secure environment that prohibits fraudulent access.

STRQ9: Need a way to train customers in the use of the system quickly and conveniently.

STRQ10: The system must operate on hardware that falls under the company's current maintenance contracts.

STRQ11: Need to be able to maintain the system with our current IT hardware and skills. Refer to enterprise architecture document EA-1234 for details.

STRQ12: Need account activity statements for tax reporting.

STRQ13: The system must provide all the basic capabilities of a normal stock broking firm.

STRQ14: Need to be able to perform research on any given stock.

STRQ15: The system must allow traders to obtain up-to-date news and alerts on nominated stock.

STRQ16: The system must provide current and historical information on Trading Accounts. Such as number of shares held, current price, total Trading Account value

STRQ17: The system shall provide the following types of trades: Market Trades (buy and sell), Limit Trades (buy and sell), and transfers between mutual funds.



Software designers tend to focus on the problem to be solved. Just don't forget that the FURPS attributes are always part of the problem. They must be considered.

A importância relativa dos diferentes atributos de qualidade depende do projeto

Loja on-line → muitas sessões em simultâneo, transações seguras,...

App Dialer (telefonar) → uso intuitivo, comunicação clara do estado da chamada

Homebanking → segurança, disponibilidade,...

Explorar:

- Wiegers, chap. 14

Regra do negócio (“business rule”)

Business rule

→ uma política, orientação, norma, ou regulamento que define ou restringe algum dos aspectos do negócio.

Regra vs Requisito

Não um requisito de software em si (porque têm uma existência para além dos limites de qualquer aplicação de software específica), mas é a origem de vários tipos de requisitos de software.

→ Mais info...

Exemplos:

- "Um novo cliente deve pagar 30% da taxa de consultoria estimada e das despesas de viagem, em adiantamento."
- "Os clientes devem ser maiores de idade ($>=18$ anos)."
- "Há uma joia de admissão, a pagar na inscrição, no valor de 50% da mensalidade."

Estas características *não dependem do software A ou B*, mas terão de ser tidas em consideração na implementação do software.

Técnicas de Recolha de Requisitos

Processo utilizado para :

Descobrir todos os requisitos (os descobertos tarde são mais difíceis de incorporar)

Obter apoio e confiança entre os utilizadores

Qual(is) técnica(s) a utilizar?

1. Entrevistas
2. Workshops de Desenho Colaborativo (JAD)
3. Questionários Análise de documentos +
4. Observação
5. Focus Group

Credit: Dennis et al, "Systems Analysis and Design: An Object Oriented Approach with UML", 5th ed.

I Oliveira

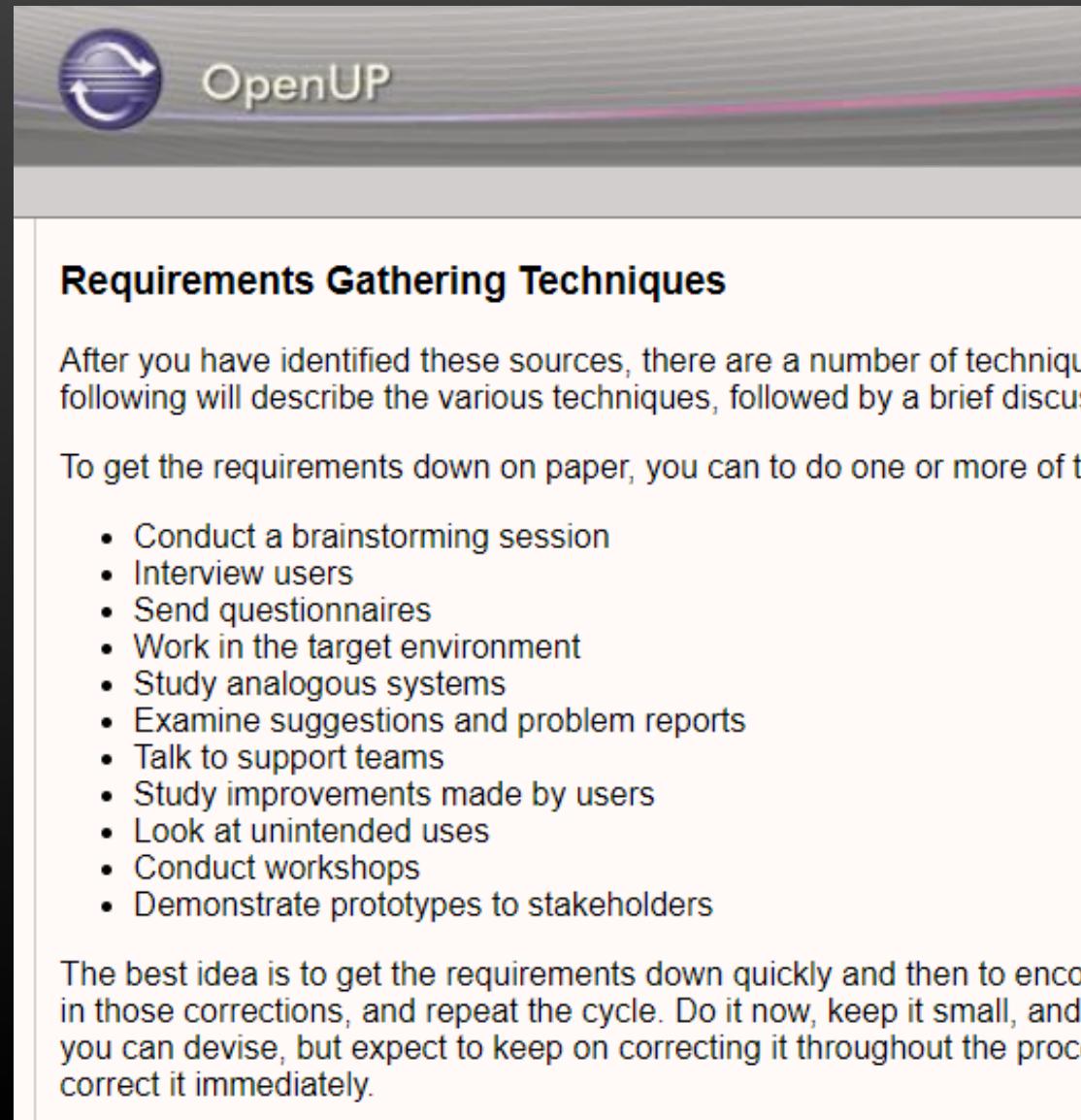
OpenUP as práticas de requisitos

OpenUP practices

[OpenUP](#) > Practices >
Technical Practices >
Shared Vision >
Requirements Gathering
Techniques

Detalhes disponíveis na
documentação do
OpenUp

I Oliveira



The screenshot shows a section titled "Requirements Gathering Techniques" from the OpenUP website. The title is bolded. Below it, a paragraph of text is partially visible. A bulleted list of ten techniques follows, starting with "Conduct a brainstorming session". The text at the bottom discusses the iterative nature of requirements gathering.

Requirements Gathering Techniques

After you have identified these sources, there are a number of techniques following will describe the various techniques, followed by a brief discussion of how to evaluate them.

To get the requirements down on paper, you can do one or more of the following:

- Conduct a brainstorming session
- Interview users
- Send questionnaires
- Work in the target environment
- Study analogous systems
- Examine suggestions and problem reports
- Talk to support teams
- Study improvements made by users
- Look at unintended uses
- Conduct workshops
- Demonstrate prototypes to stakeholders

The best idea is to get the requirements down quickly and then to encourage the users to review them, make changes, and repeat the cycle. Do it now, keep it small, and you can devise, but expect to keep on correcting it throughout the process. If you find a major error, correct it immediately.

Técnicas de recolha de requisitos comparadas

Pode ser usada uma combinação de técnicas

A análise e observação de documentos requer pouca formação; as sessões JAD podem ser muito desafiantes

	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low-Medium	Low	Low	Low to Medium

Credit: Dennis et al, "Systems Analysis and Design: An Object Oriented Approach with UML", 5th ed.

Figure 7-3 suggests the elicitation techniques that are most likely to be useful for various types of projects. Select the row or rows that represent characteristics of your project and read to the right to see which elicitation techniques are most likely to be helpful (marked with an X). For instance, if you're developing a new application, you're likely to get the best results with a combination of stakeholder interviews, workshops, and system interface analysis. Most projects can make use of interviews and workshops. Focus groups are more appropriate than workshops for mass-market software because you have a large external user base but limited access to representatives. These suggestions for elicitation techniques are just that—suggestions. For instance, you might conclude that you do want to apply user interface analysis on mass-market software projects.

	Interviews	Workshops	Focus groups	Observations	Questionnaires	System interface analysis	User interface analysis	Document analysis
Mass-market software	X		X		X			
Internal corporate software	X	X	X	X		X		X
Replacing existing system	X	X		X		X	X	X
Enhancing existing system	X	X				X	X	X
New application	X	X				X		
Packaged software implementation	X	X		X		X		X
Embedded systems	X	X				X		X
Geographically distributed stakeholders	X	X			X			

FIGURE 7-3 Suggested elicitation techniques by project characteristic.

Técnicas alternativas

Análise da área (domínio)

Estudar as características do domínio

Aprender com os outros

Aumentar a propensão para escalar para mais clientes

Mapas de conceitos

Representam relações importantes entre conceitos

Concentrar as pessoas num pequeno número de ideias-chave

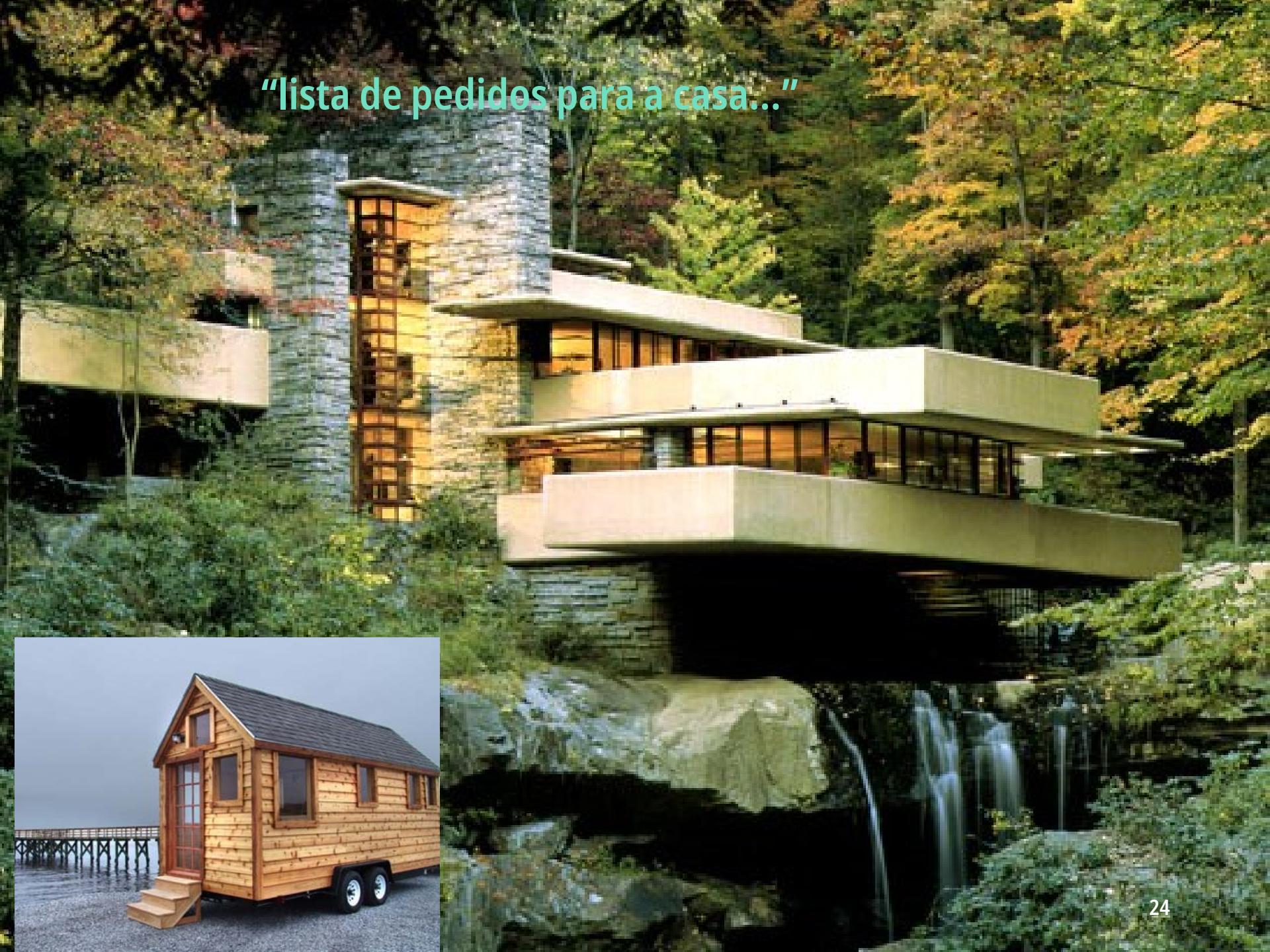
User Stories

Associado a métodos de desenvolvimento ágeis

Pouca tecnológica necessária, facilmente actualizável

→ A explorar mais à frente no semestre

“lista de pedidos para a casa...”



Requirements elicitation (Determinação dos requisitos)

Sistema a construir

Requisitos documentados na análise

Inconsistente

Inexequível
(tecnologia)

supérfluo (desnecessário)

Requisitos
em falta

Quais são as actividades de determinação dos requisitos?

O núcleo do desenvolvimento de requisitos é a elicitação

o processo de identificação das necessidades e restrições dos vários intervenientes para um sistema de software.

Elicitação não é a mesma coisa que "recolha de requisitos". Nem é uma simples questão de transcrever exactamente o que os utilizadores dizem.

Elicitação é um processo colaborativo e analítico que inclui actividades para recolher, descobrir, extrair e definir requisitos.

A elicitação é utilizada para descobrir requisitos focados no negócio + focados no utilizador + funcionais + não-funcionais.

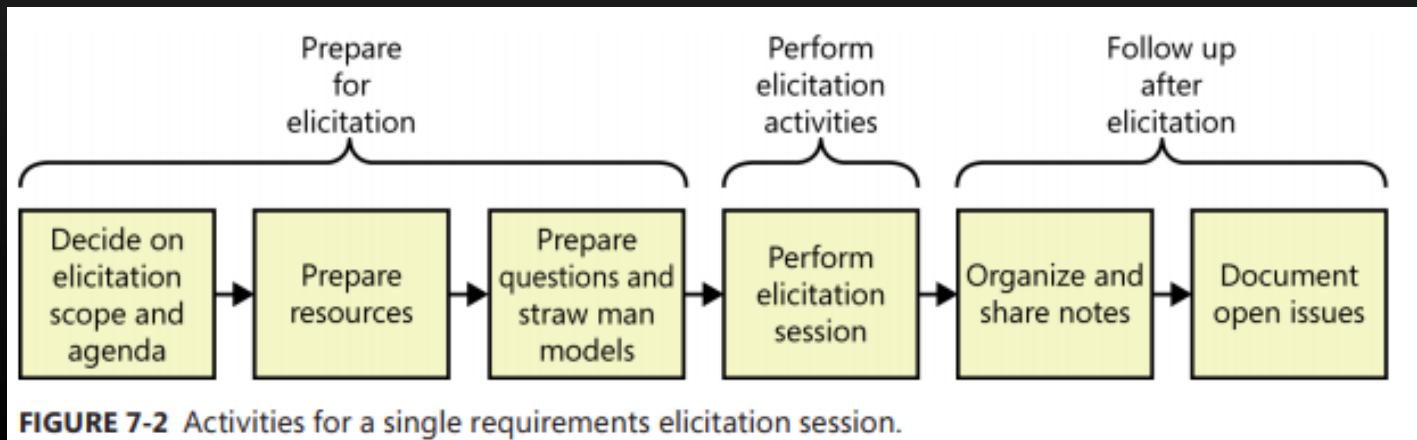


FIGURE 7-2 Activities for a single requirements elicitation session.

elicitação de requisitos como um processo

A elicitação de requisitos é talvez o aspecto mais desafiante, crítico, propenso a erros, e de comunicação intensiva no desenvolvimento de software.



Documentação dos requisitos

Requisitos são muitas vezes elencados em listas “o sistema deve...”

#	Requisito
RF.1	O sistema deve permitir a um profissional criar um novo pedido de adesão , em auto-serviço, na web.
RF.2	O sistema deve enviar credenciais temporárias para os pedidos de adesão e enviá-las, por email, aos solicitantes, até 10min depois do registo.
RF.3	O sistema deve permitir a pesquisa de cheques-dentista (emitidos) por número de utente do SNS.
RNF.1	As pesquisas de cheques-dentista têm de retornar resultados em <5 segundos ou um evento de tempo expirado.
...	

"The system shall be 100% reliable and 100% available".

"The system shall have a minimum response to a query of 1 second irrespective of system load".

AVL-1. The system shall be at least 95 percent available on weekdays between 6:00 A.M. and midnight Eastern Time, and at least 99 percent available on weekdays between 3:00 P.M. and 5:00 P.M. Eastern Time.

IOP-1. The Chemical Tracking System shall be able to import any valid chemical structure from the ChemDraw (version 13.0 or earlier) and MarvinSketch (version 5.0 or earlier) tools.

PER-1. Authorization of an ATM withdrawal request shall take no more than 2.0 seconds.

PER-2. The anti-lock braking system speed sensors shall report wheel speeds every 2 milliseconds with a variation not to exceed 0.1 millisecond.

PER-3. Webpages shall fully download in an average of 3 seconds or less over a 30 megabits/second Internet connection.

PER-4. At least 98 percent of the time, the trading system shall update the transaction status display within 1 second after the completion of each trade.

S. M. A. R. T. → Specific, Measurable, Attainable, Relevant and time-sensitive (*Especifico, Mensurável, Atingível, Relevante e Rastreável no tempo*)

Step 5: Specify well-structured quality requirements

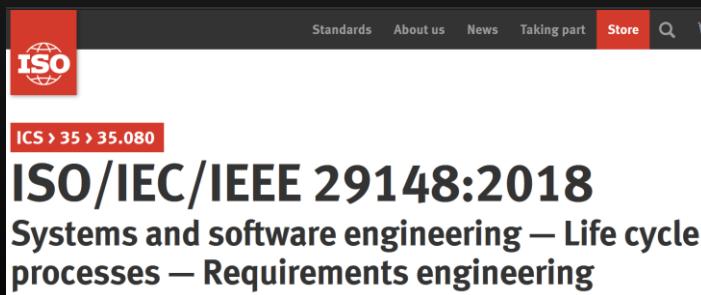
Simplistic quality requirements such as "The system shall be user-friendly" or "The system shall be available 24x7" aren't useful. The former is far **too subjective and vague**; the latter is **rarely realistic or necessary**. **Neither is measurable**. Such requirements provide little guidance to developers. So the final step is to **craft specific and verifiable requirements** from the information that was elicited regarding each quality attribute. When writing quality requirements, keep in mind the useful SMART mnemonic—make them *Specific, Measurable, Attainable, Relevant, and Time-sensitive*.

Credit: Wiegers '13

O que são requisitos bem formulados? (ISO-IEEE 29148)

Uma formulação de uma característica que:

- tem de ser satisfeita ou detida por um sistema para resolver um problema ou para atingir um objectivo de alguma "parte interessada"
- pode ser verificada,
- é qualificado por condições mensuráveis e limitado por restrições,
- e define o comportamento ou a capacidade do sistema quando usado, mas não uma capacidade do utilizador ou operador.



Elementos de estilo

Um requisito é uma declaração que expressa uma necessidade e as limitações e condições que lhe estão associadas.

Se expressa em linguagem natural, a declaração deve incluir um sujeito, um verbo e um complemento. Um requisito deve indicar o sujeito do requisito (por exemplo, o sistema, o software, etc.) e o que deve ser feito (por exemplo, operar a um nível de potência).

E.g: O Sistema de Facturação [Sujeito], deve exibir [Acção] as facturas pendentes do cliente [Complemento] por ordem crescente em que as facturas devem ser pagas.



Requisitos (e casos de utilização) no OpenUP

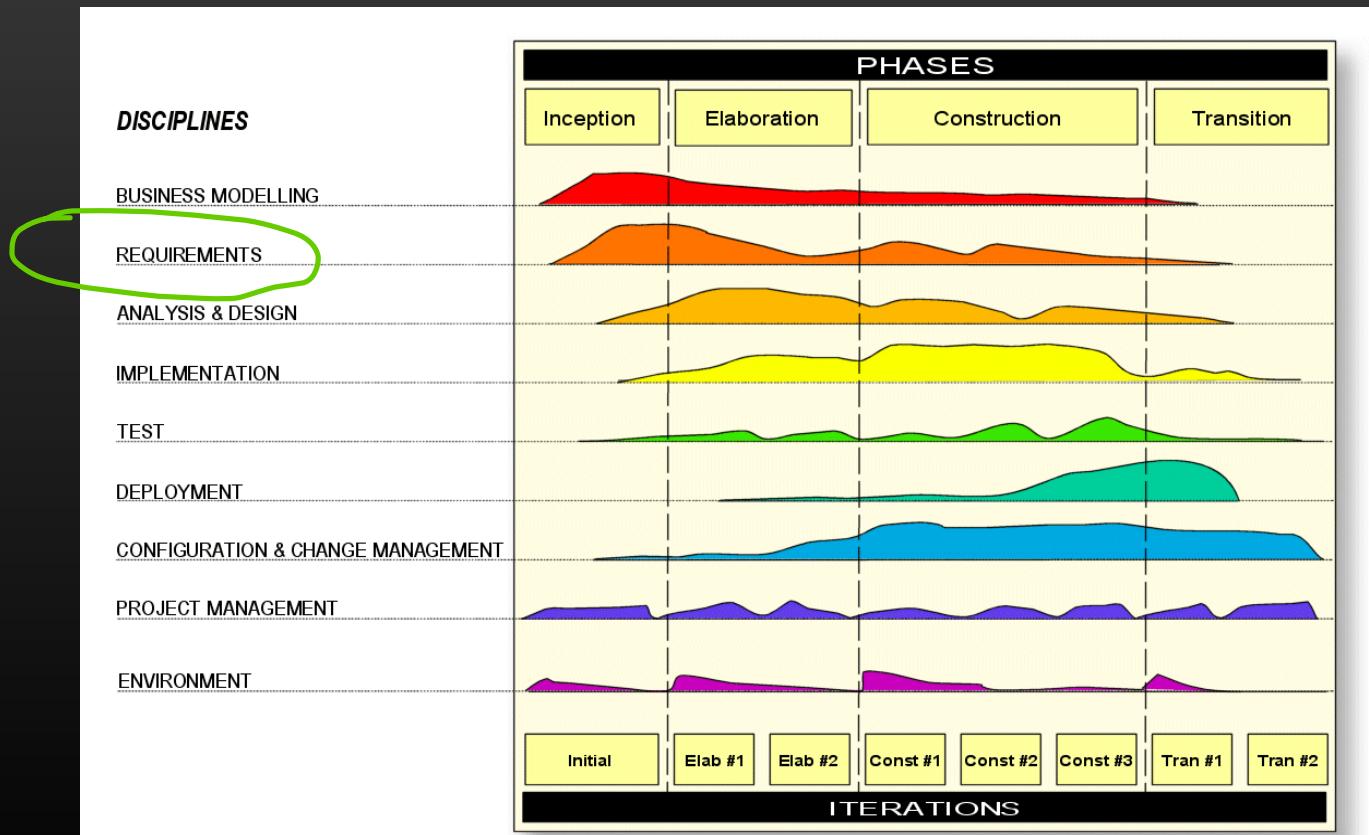
Visão geral do OpenUP/Unified Process

Os requisitos não são estanques; podem ser evoluídos → refinados

O UP oferece uma abordagem ao SDLC concebida como uma matriz, cruzando diferentes disciplinas técnicas com iterações (evoluções) no projecto.

(Nota: fases UP ≠ fases SDLC)

A análise dos requisitos é realizada principalmente no início do projeto (requisitos básicos), mas também durante as iterações (requisitos evolutivos).



Which main elicitation approaches exist?

What is the goal the **user** wants to achieve?

vs.

What capability should the **product/system** possess?



OpenUP: como identificar e descrever os requisitos, para que o âmbito fique determinado

The screenshot shows a web browser displaying the OpenUP website. The header includes the OpenUP logo, navigation links for Glossary, Feedback, and About, and a Print button. The left sidebar has a 'Where am I' link and a 'Tree Sets' link. Below these, under the 'Team' heading, is a tree structure of topics:

- Introduction to OpenUP
- Getting Started
- Delivery Processes
- Practices
 - Management Practices
 - Technical Practices
 - Concurrent Testing
 - Continuous Integration
 - Evolutionary Architecture
 - Evolutionary Design
 - Shared Vision
 - Test Driven Development
 - Use Case Driven Deve
 - How to Adopt the Us
 - Key Concepts

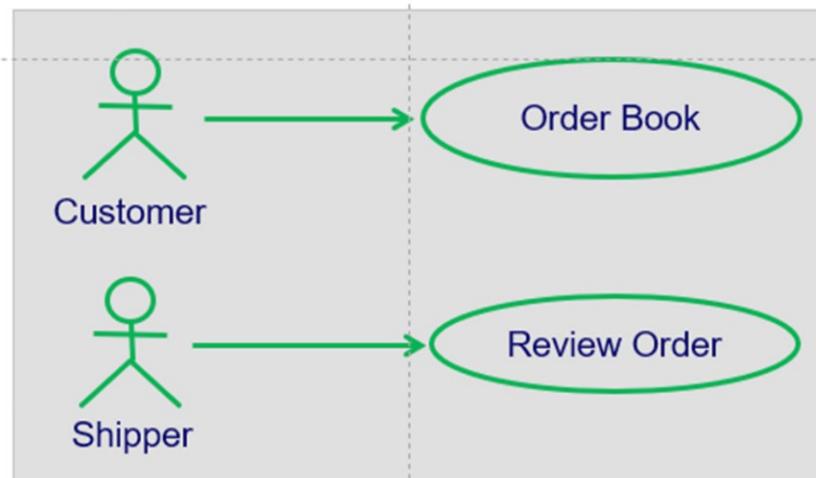
The main content area shows the breadcrumb path: Practices > Technical Practices > Use Case Driven Development > Tasks > Identify and Outline Requirements. The title of the current page is "Task: Identify and Outline Requirements". A yellow arrow icon points to the text: "This task describes how to identify and outline the requirements for the system so that the scope of work can be determined." Below this, under the heading "Disciplines: Requirements", are "Expand All Sections" and "Collapse All Sections" buttons. A section titled "Purpose" contains the text: "The purpose of this task is to identify and capture functional and non-functional requirements for the system. These requirements form the basis of communication and agreement between the stakeholders and the development team on what the system must do to satisfy stakeholder needs. The goal is to understand the requirements at a high-level so that the initial scope of work can be determined. Further analysis will be performed to detail these requirements prior to implementation." At the bottom right of the content area is a "Back to top" link.

OpenUP recommended practices

Use Case Driven Development 🏆



- This practice describes how to capture requirements with a combination of use cases and system-wide requirements, and then drive development and testing from those use cases.



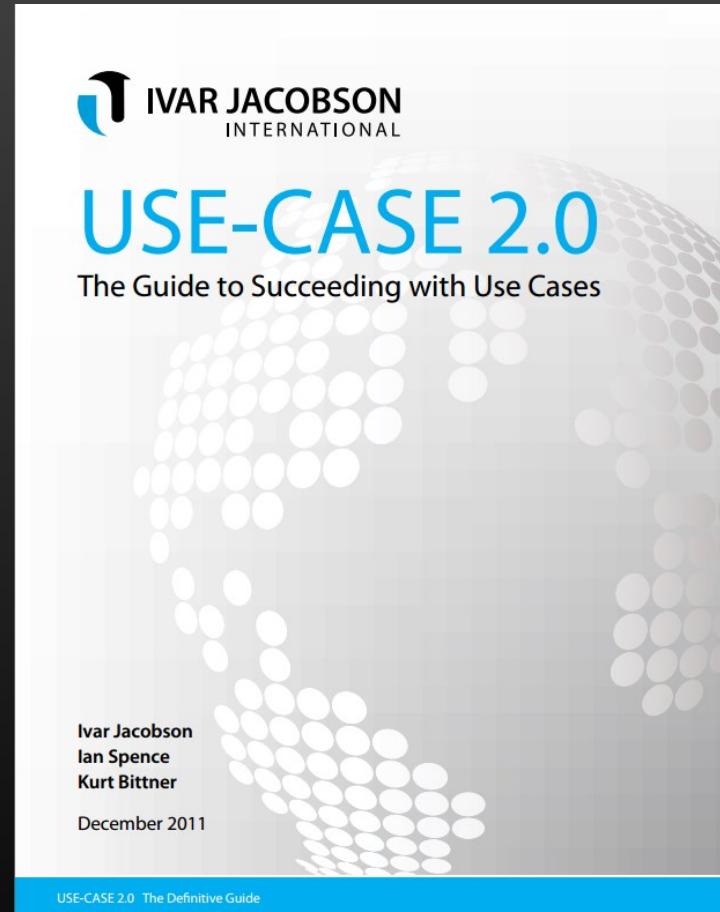
Caso de utilização (CaU)

Engloba uma sequência de ações que um sistema executa e que produzem um resultado com valor para algum ator em particular.

Implica:

- Foco no **utilizador** do sistema e nos episódios de uso
- Foco na compreensão daquilo que os atores consideram um **resultado relevante** (motivações para usar um sistema → problemas que querem resolver)

Conceito apresentado originalmente em:
Jacobson, I., 1993. *Object-oriented software engineering: a use case driven approach.*. Pearson Education.



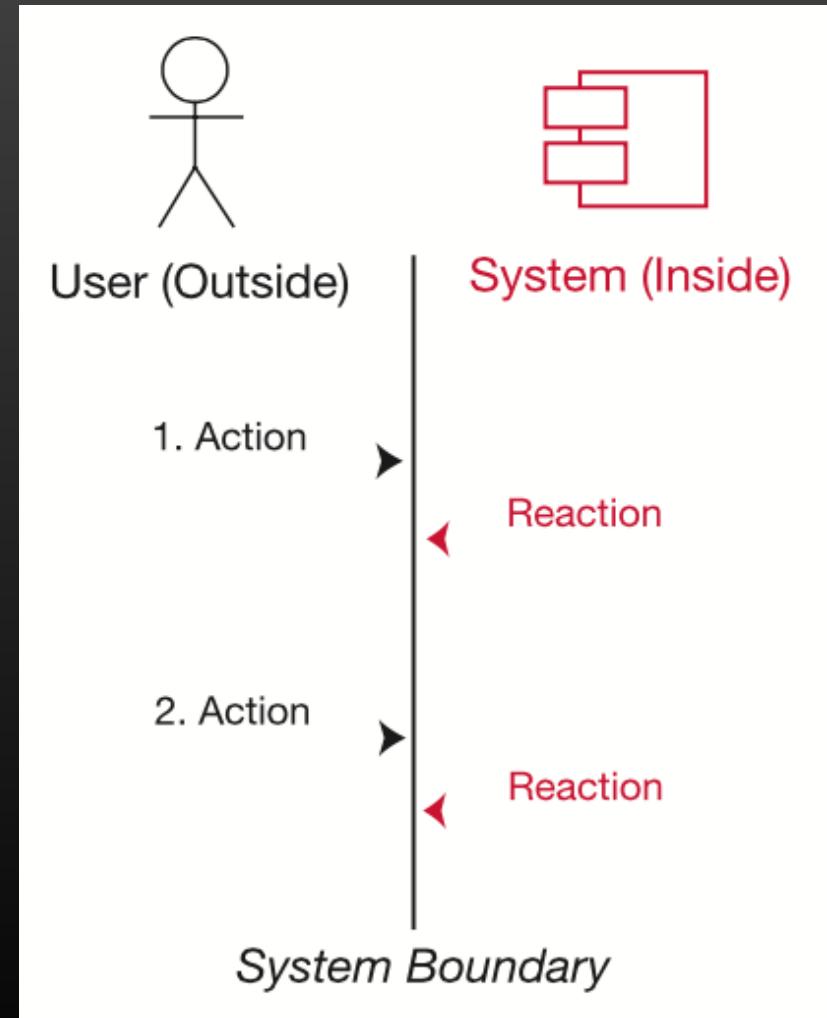
<https://www.ivarjacobson.com/publications/white-papers/use-case-ebook>

O CaU descreve **um diálogo entre o ator e o sistema**

Narrativas para contar como é que “alguém” usa um sistema

CaU: Comprar produtos (supermercado)

1. Um Cliente chega a uma caixa com artigos para comprar.
2. O Operador passa cada artigo no leitor de código de barras para registo.
3. O sistema apresenta o total provisório e a lista de artigos incluídos.
4. O Operador termina a venda e indica o tipo de pagamento.
5. O cliente introduz a informação de pagamento.
6. O sistema valida o pagamento, atualiza o stock e imprime o recibo.
7. O cliente leva o recibo e os artigos.



Use case:	Brief description:
Create new assignment	<p>The Teaching Staff creates a new Activity of type Assignment, directly inserting it in the page layout. The assignment must define a title and a time period for submissions and can be configured to work with individual or group submissions. The assignment is listed in the student view and on the specified date (or immediately, if none is given) accepts submissions from registered students.</p>
Use case: <u>Add new assignment</u> Brief description: <p>The Faculty creates assignments for students, directly inserting it in the course page. The assignment defines a time period for submissions and can be configured to work with individual or group submissions. The assignment is listed in the student view and on the specified date (or immediately, if none is given) accepts submissions from students.</p> Basic flow: <ol style="list-style-type: none"> 1. Log-in using corporate IdP. 2. Select desired course. 3. Turn editing mode on. 4. Add Assignment activity in the page layout. 5. Configure Assignment activity. 6. Commit changes. Alternative flows: <p>Step 1: IdP unavailable.</p> <p>Step 4/5: Instead of a new, empty assignment, the user may reuse an existing one.</p> Open issues: <p>Step 3/4. The course is closed. Are changes allowed to past courses?</p> <p>Step 5. The browser does not accept the rich text editor. Default to plain text?</p>	

Conclusões...

Outside the Developing Organization

Direct user	Business management	Consultant
Indirect user	Contracting officer	Compliance auditor
Acquirer	Government agency	Certifier
Procurement staff	Subject matter expert	Regulatory body
Legal staff	Program manager	Software supplier
Contractor	Beta tester	Materials supplier
Subcontractor	General public	Venture capitalist

Developing Organization

Development manager	Sales staff	Executive sponsor
Marketing	Installer	PMO
Operational support	Maintainer	Manufacturing
Legal staff	Program manager	Training staff
Information architect	Usability expert	Portfolio architect
Company owner	Shareholder	Infrastructure support

Project Team

Project manager	Tester	Product owner
Business analyst	Product manager	Data modeler
Application architect	QA staff	Process analyst
Designer	Doc writer	Hardware engineer
Developer	DBA	Infrastructure analyst

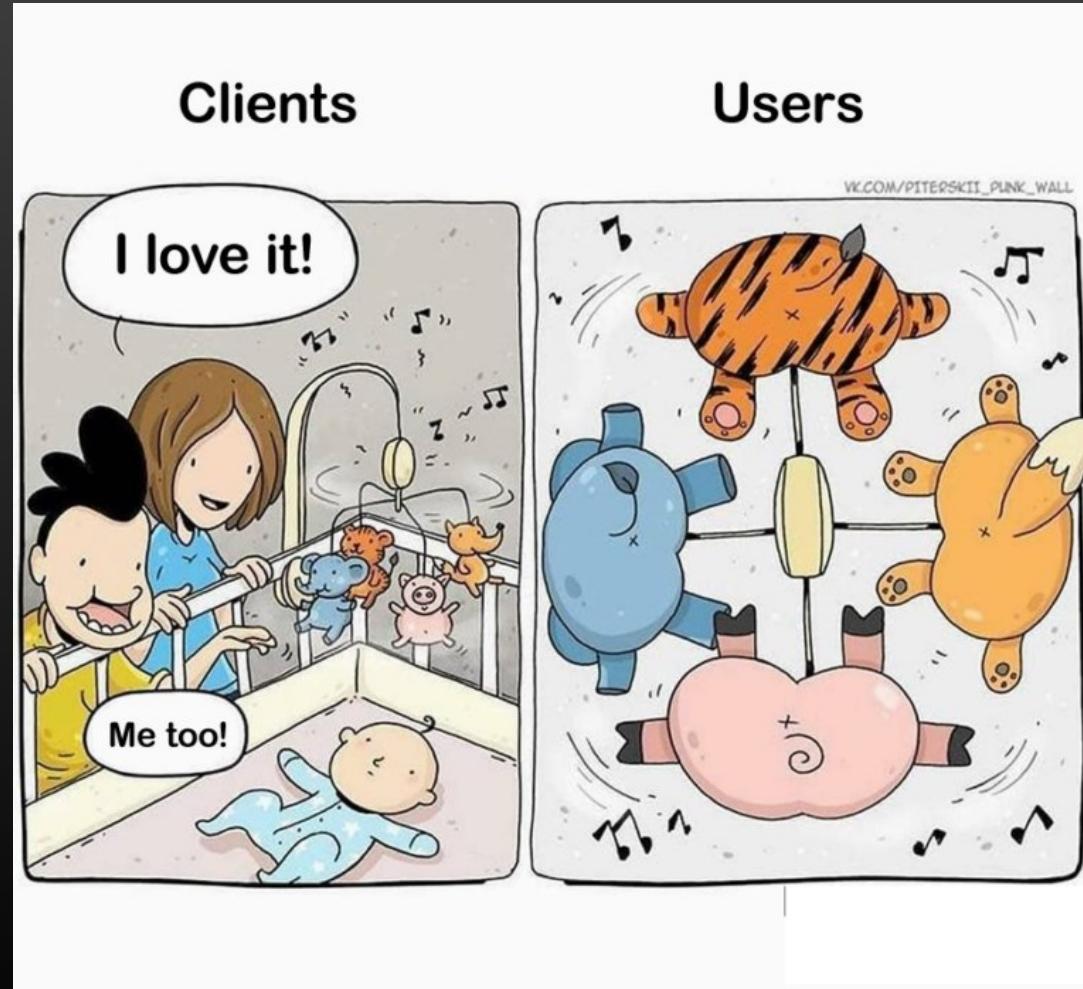
Figure 1. Some possible software project stakeholders.

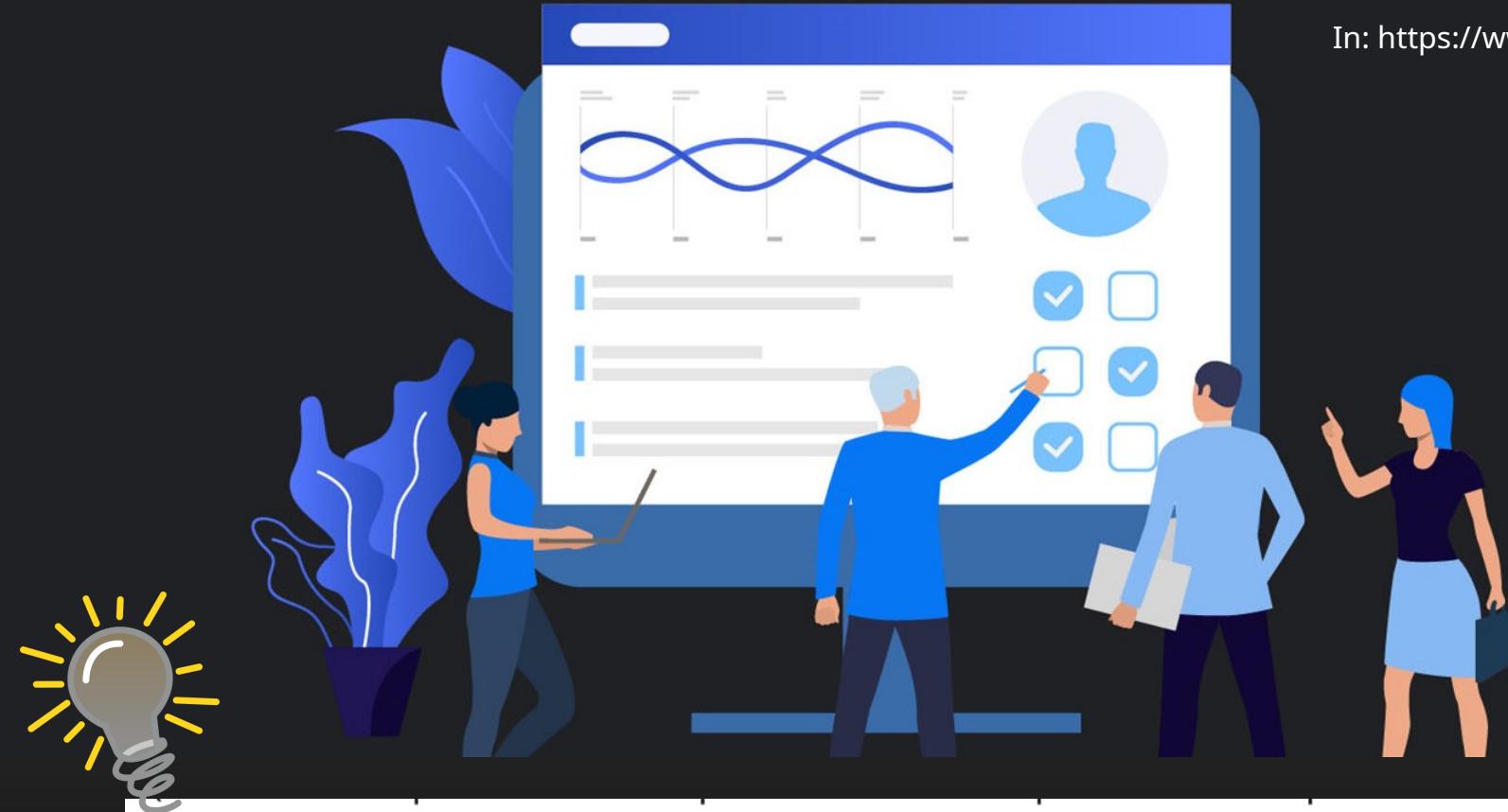
Compromisso...

Business...

Utilizadores...

Viabilidade...





And second, those of us in the software domain tend to be enamored with technical and process solutions to our challenges. We sometimes fail to appreciate that requirements elicitation—and much of software and systems project work in general—is primarily a human interaction challenge. No magical new techniques have come along to automate that, although various tools are available to help geographically separated people collaborate effectively.

In: Wiegers, "Software Requirements"

Ver também

 Karl Wiegers

Aug 12, 2019 • 10 min read ★ ·

[Twitter](#) [Facebook](#) [LinkedIn](#) [Copy](#) [+1](#) [...](#)

Ten Cosmic Truths About Software Requirements

These facts apply to nearly every project. Ignore them at your peril.



Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 3 – Requirements Determination	<ul style="list-style-type: none">• [Pressman15] – Chap. 8 – Understanding Requirements• [Wiegers13] – Chap. 1 -3

41951- ANÁLISE DE SISTEMAS

Transformação digital e o SDLC

Ilídio Oliveira | v2022/04/19

Objetivos de aprendizagem

Descrever exemplos da transformação digital dos negócios

Identificar três eixos principais na transformação digital

Relacionar o contexto das organizações com as exigências no processo de desenvolvimento

O papel do arquiteto

PRR (pós-pandemia)

Prioridades:

- Pacto Ecológico europeu
- Transformação Digital
- Resiliência/infraestruturas



"[...] outra mudança faz de mor espanto: que não se muda já como soía."

A inovação tecnológica permite - de facto, exige - que as empresas aumentem a sua agilidade e, consequentemente, a sua competitividade. É por isso que as principais prioridades dos CEOs devem ser a digitalização dos componentes essenciais do seu negócio e repensar a concepção organizacional e os processos de gestão. Apanhar esta rápida - e em crescimento rápido - "onda digital" é a única forma de evitar ficar para trás".

PROJECT ■ SYNDICATE
THE WORLD'S OPINION PAGE

 DOMINIC BARTON
Dominic Barton is the global managing director of McKinsey & Company.

JAN 15, 2016

Catching the Digital Wave

NEW YORK – Technological change has always posed a challenge for companies. But, as we saw once again in 2015, it has never occurred as rapidly, or on as large a scale, as today. As innovation sweeps across virtually every sector, from heavy industry to services, it is transforming the competitive landscape, with the most advanced companies – rather than the largest or most established players – coming out on top.

For incumbents, the threat of displacement is very real. The average tenure of a company on the S&P 500 has fallen from 90 years in 1935 to less than 18 years today. Disruptive new players like Uber, which has upended the taxi industry, are tough competitors, often staking out market share by shifting more surplus to consumers. This is part of a broader trend of intensifying competition that, according to recent research from the McKinsey Global Institute, could reduce the global after-tax profit pool from almost 10% of global GDP today to its 1980 level of about 7.9% within a decade.

<http://prosyn.org/IxXI6OW>

THE WALL STREET JOURNAL.

This copy is for your personal, non-commercial use only. To order presentation-ready copies for distribution to your colleagues, clients or visit <http://www.djreprints.com>.

<http://www.wsj.com/articles/SB10001424053111903480904576512250915629460>

ESSAY

Why Software Is Eating The World

By MARC ANDREESSEN

August 20, 2011

This week, Hewlett-Packard (where I am on the board) announced that it is exploring jettisoning its struggling PC business in favor of investing more heavily in software, where it sees better potential for growth. Meanwhile, Google plans to buy up the cellphone handset maker Motorola Mobility. Both moves surprised the tech world. But both moves are also in line with a trend I've observed, one that makes me optimistic about the future growth of the American and world economies, despite the recent turmoil in the stock market.

In short, software is eating the world.

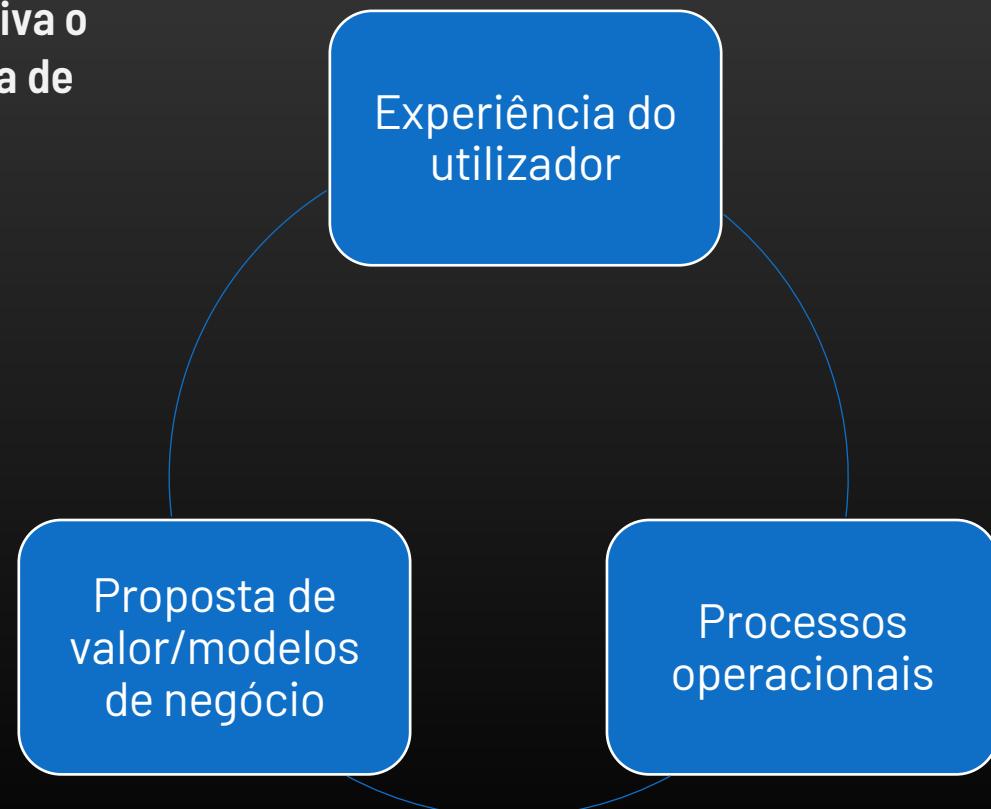
Marc Andreessen penned his famous "[Why Software Is Eating the World](#)" essay in The Wall Street Journal five years ago. Today, the idea that "every company needs to become a software company" is considered almost a cliché. No matter your industry, you're expected to be reimagining your business to make sure you're not the next local taxi company or hotel chain caught completely off guard by your equivalent of Uber or Airbnb. But while the inclination to not be "disrupted" by startups or competitors is useful, it's also not exactly practical.

Jeetu Patel
Contributor

Jeetu Patel is the chief strategy officer and head of Box Platform.

Transformação digital

A utilização de TIC para melhorar de forma decisiva o desempenho ou proposta de valor de uma empresa





O SEU BANCO SEMPRE DISPONIVEL.

Tudo o que precisa, onde estiver, como quiser.
Adesão gratuita.

[SAIBA MAIS >](#)

APP CAIXADIRECTA

**UMA EXPERIÊNCIA ÚNICA
DE ACESSO SIMPLES E CÓMODO
AO SEU BANCO.**

JÁ DISPONÍVEL PARA TABLET E SMARTPHONE IOS E ANDROID.

INOVAR NA CAIXA, COM CERTEZA.

iOS Android Windows



Serviços Disponíveis[Página Principal](#)**Alunos**[Secretaria Virtual](#)[Horários de Turmas
do 1ºAno](#)[Horários para
2014/2015](#)[Creditações OnLine](#)[Matrículas OnLine](#)[Candidaturas](#)[Candidaturas M23](#)[Candidaturas
Especiais](#)[Candidaturas CET](#)[Candidatura Cursos
Livres](#)[Candidaturas
EI/internationalstudent](#)**Docentes**[Disciplinas](#)**[Secretaria Virtual para Estudantes da UA](#)**

Sistema de apoio aos **estudantes**, servindo de extensão à Secretaria dos Serviços Académicos.

**SGQ**

A partir de 26 de Janeiro, a Universidade de Aveiro (UA) implementa o Subsistema para a Garantia da Qualidade das Unidades Curriculares relativo ao 1º semestre do ano letivo 2014/2015.

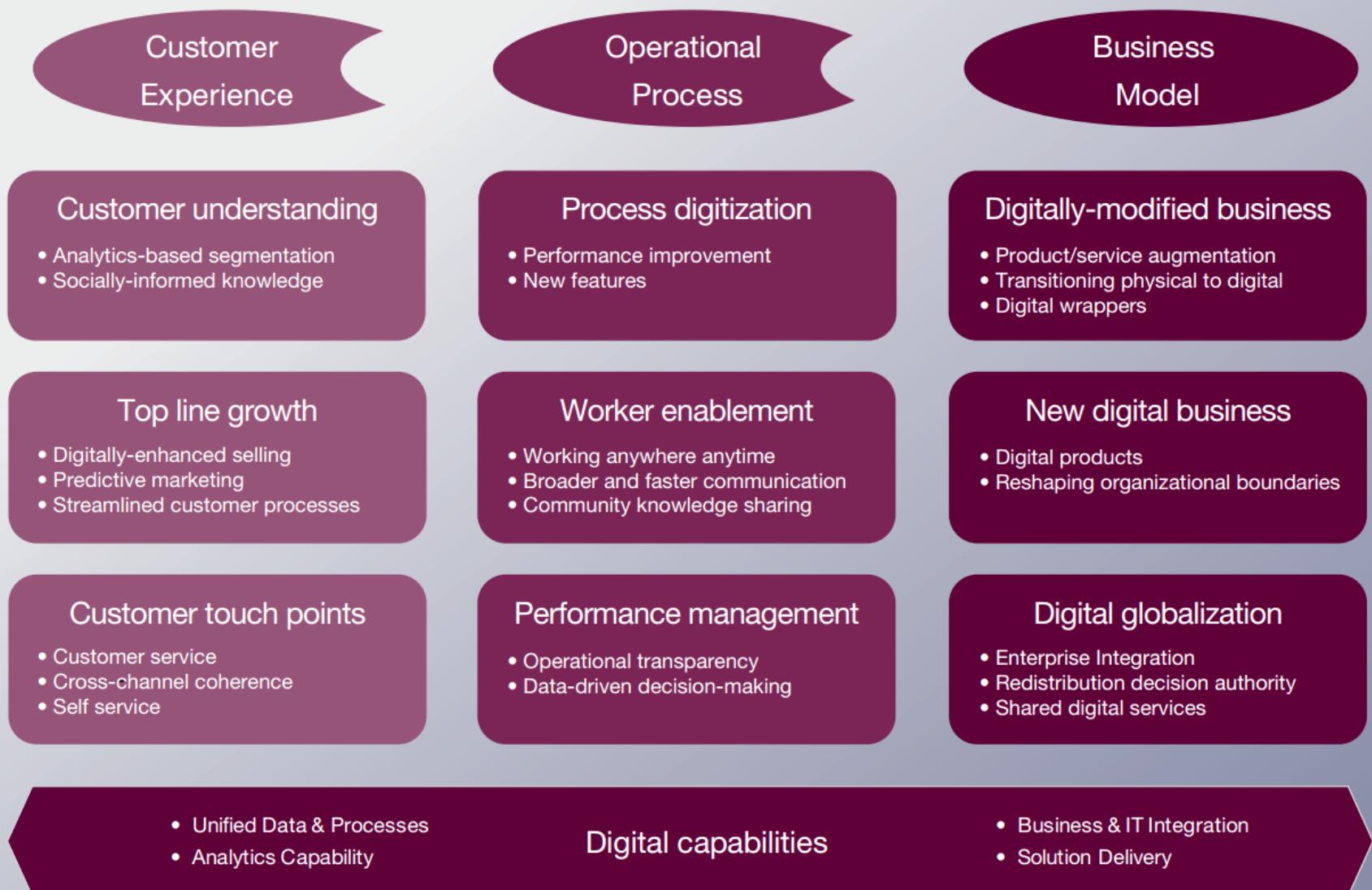
A partir dessa data e até ao dia 22 de fevereiro, a UA promoverá o lançamento dos inquéritos pedagógicos junto dos estudantes. Os inquéritos são preenchidos eletronicamente, via PACO (<http://paco.ua.pt/>) ou diretamente em <http://sgq.ua.pt>.

Participa! A tua opinião é fundamental!

U B E R

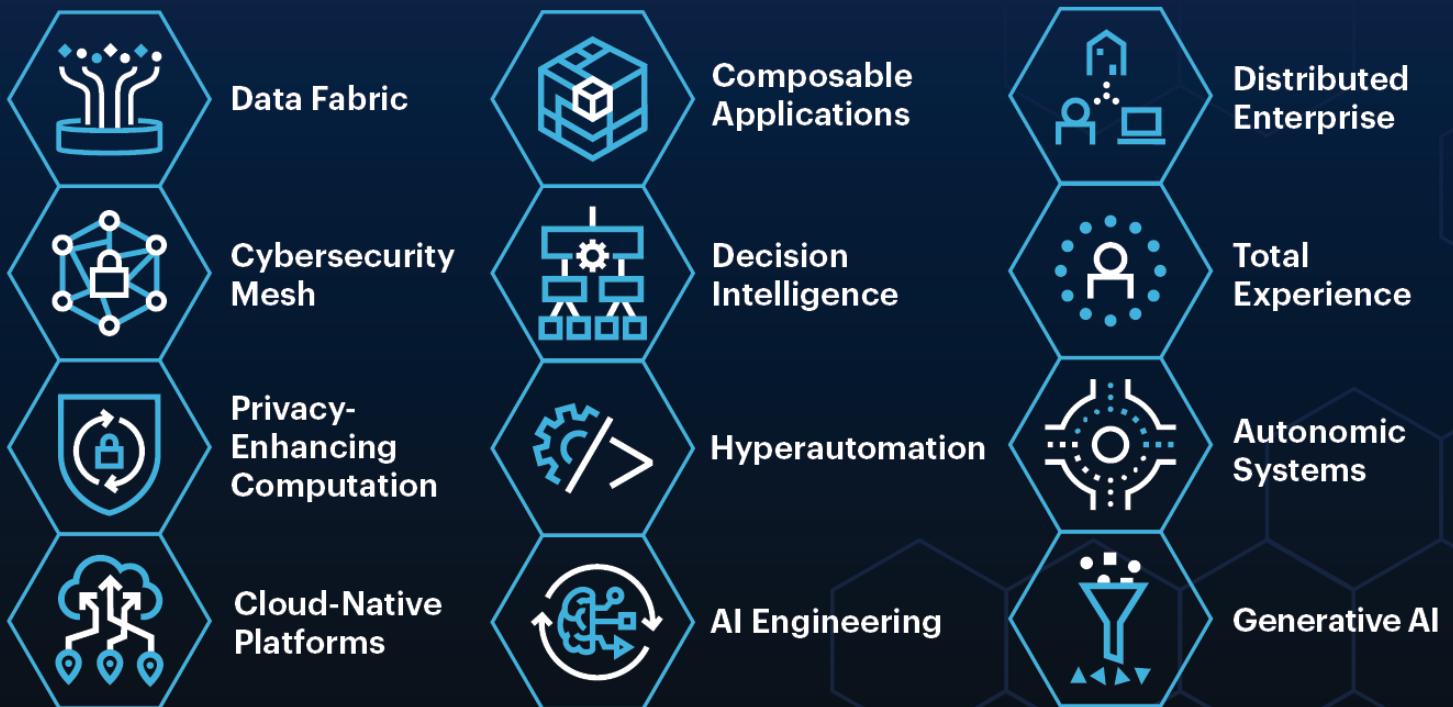


Figure 3: Building blocks of the digital transformation



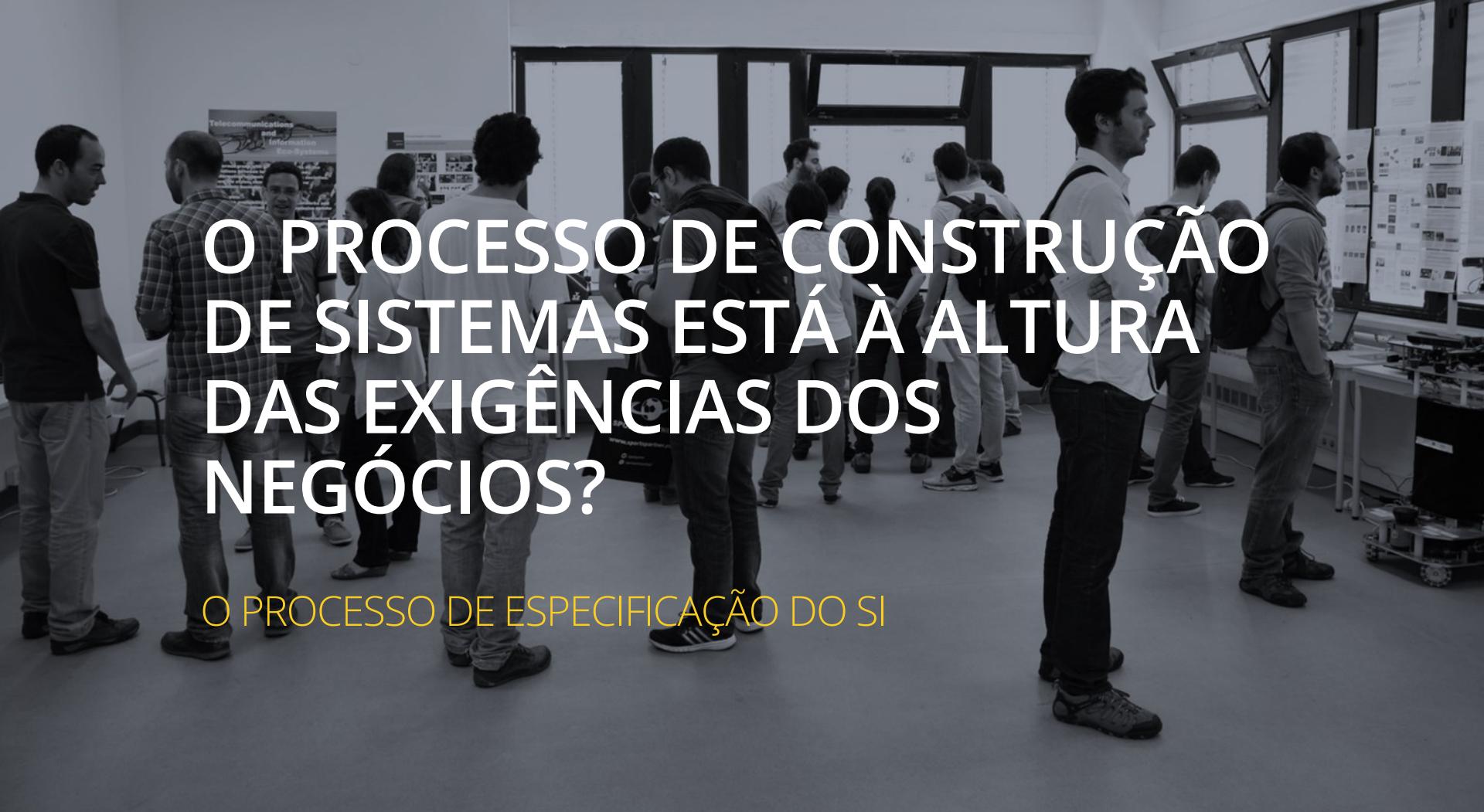
“Digital Transformation: A Roadmap for Billion-Dollar Organizations”, MIT Center for Digital Business

Top Strategic Technology Trends for 2022



© 2021 Gartner, Inc. and/or its affiliates. All rights reserved. 1397600

Gartner



O PROCESSO DE CONSTRUÇÃO DE SISTEMAS ESTÁ À ALTURA DAS EXIGÊNCIAS DOS NEGÓCIOS?

O PROCESSO DE ESPECIFICAÇÃO DO SI

In 2015

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011 - 2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Como é que compara
com outras
“engenharias”?



Integrated Requirements Engineering: A Tutorial

Ian Sommerville, Lancaster University

Before developing any system, you must understand what the system is supposed to do and how its use can support the individuals or business that will pay for that system. This involves understanding the application domain (airlines, railways, retail banking, games, and so on); the system constraints; the specific functionality required by the stakeholders who directly or indirectly use the system or the information it produces.

and essential system characteristics such as performance, security, and dependability. *Requirements engineering* is the name given to a structured set of activities that help develop this understanding and that document the system specification for the stakeholders and engineers involved in the system development.

This short tutorial introduces the fundamental activities of RE and discusses how it has evolved as part of the software engineering process. However, rather than focus on established RE techniques, I discuss how the changing nature of software engineering has led to new challenges for RE. I then introduce a number of new techniques that help meet these challenges by integrating RE more closely with other systems implementation activities.

The fundamental process

The RE process varies depending on the type of application developed, the size and culture involved, and the software used. For large military systems, there is normally a formalized systems engineering process involving a well-documented set of requirements. For developing innovative software, the process might consist of visions, and the product is simply a short vision since the software is expected to change rapidly.

Whatever the actual process, the fundamental activities are:

■ *Elicitation.* Identify sources of information about the system and discover the requirements from these.

■ *Analysis.* Understand the requirements, their overlaps, and their conflicts.

■ *Validation.* Go back to the system stakeholders to check that the requirements are correct.

■ *Design.* Create a design that satisfies the requirements.

■ *Implementation.* Implement the system.

■ *Testing.* Test the system to ensure it meets the requirements.

■ *Maintenance.* Maintain the system over its lifetime.

■ *Evolution.* Evolve the system to meet changing requirements.

■ *Retirement.* Retire the system when it is no longer needed.

This tutorial introduces the fundamental activities of requirements engineering and discusses recent developments that integrate it and system implementation.

- *Elicitation.* Identify sources of information about the system and discover the requirements from these.
- *Analysis.* Understand the requirements, their overlaps, and their conflicts.
- *Validation.* Go back to the system stakeholders to check that the requirements are correct.
- *Design.* Create a design that satisfies the requirements.
- *Implementation.* Implement the system.
- *Testing.* Test the system to ensure it meets the requirements.
- *Maintenance.* Maintain the system over its lifetime.
- *Evolution.* Evolve the system to meet changing requirements.
- *Retirement.* Retire the system when it is no longer needed.

Analista de sistemas é um papel chave do SDLC

O **analista** de sistemas analisa a situação do negócio, identifica oportunidades de melhorias e projeta um sistema de informação para implementá-las.

Ser analista de sistemas é um dos trabalhos mais desafiantes na eng.a de software.



O principal objetivo de um analista de sistemas não é criar um sistema “topo de gama” (na perspetiva da tecnologia), mas **criar valor para a organização**.

The Systems Analyst: Skills

Agents of change

Identify ways to improve the organization
Motivate & train others

Skills needed

Technical: must understand the technology

Business: must know the business processes

Analytical: must be able to solve problems

Communications: technical & non-technical audiences

Interpersonal: leadership & management

Ethics: deal fairly and protect confidential information

O SDLC é concretizado em **processos de desenvolvimento**

O que é que inclui um processo?



Core
Principles



Roles



Work Products



Disciplines



Lifecycle

<http://epf.eclipse.org/wikis/openup/>

47006- ANÁLISE E MODELAÇÃO DE SISTEMAS

Práticas da Engenharia de Requisitos

Ilídio Oliveira

V2022/05/03

Objetivos de aprendizagem

- Distinguir especificação de requisitos centrados no utilizador e no produto
- Descrever técnicas de documentação dos requisitos
- Compreender a relação entre requisitos e casos de utilização e identificar as disciplinas e actividades relacionadas com os requisitos no OpenUP
- Explicar o significado de "requisitos evolutivos"
- Descrever os tipos de informação que o Analista recolhe nas actividades de engenharia de requisitos
- Explicar como implementar actividades de rastreio de requisitos

Requisitos *vs* cenários

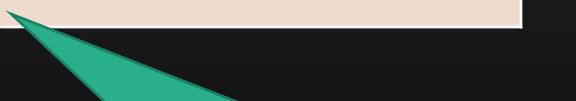
Wiegers: duas abordagens principais?

Usage-centric or product-centric?

Requirements elicitation typically takes either a usage-centric or a product-centric approach, although other strategies also are possible. The usage-centric strategy emphasizes understanding and exploring user goals to derive the necessary system functionality. The product-centric approach focuses on defining features that you expect will lead to marketplace or business success. A risk with product-centric strategies is that you might implement features that don't get used much, even if they seemed like a good idea at the time. We recommend understanding business objectives and user goals first, then using that insight to determine the appropriate product features and characteristics.

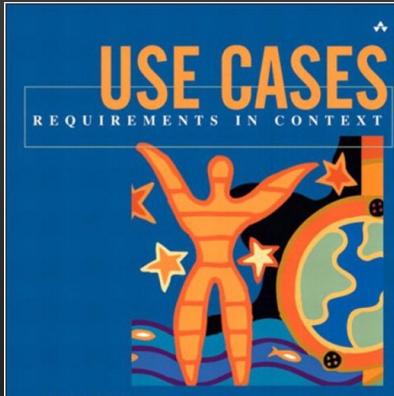
Especificação centrada no produto: “The system shall....”

#	Requisito
RF.1	O sistema deve permitir a um profissional criar um novo pedido de adesão , em auto-serviço, na web.
RF.2	O sistema deve enviar credenciais temporárias para os pedidos de adesão e enviá-las, por email, aos solicitantes.
RF.3	O sistema deve permitir a pesquisa de cheques-dentista (emitidos) por número de utente do SNS.
RNF.1	As pesquisas de cheques-dentista têm de retornar resultados em <5 segundos ou um evento de tempo expirado.
...	

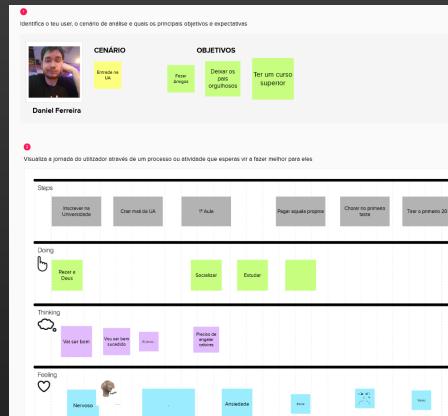


“Product centric”: pensamento focado em características técnicas do que seria um “bom” produto.

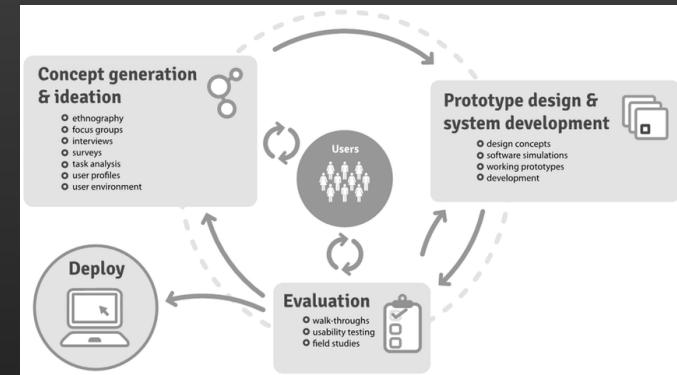
Foco na utilização → captar os requisitos ao contar histórias (narrativas)



Use cases
(Suportado na UML)



**Jornada do
utilizador (CJM)**



**Desenho centrado
no utilizador (UCD)**

...e ainda: *User Stories, Design Thinking,...*

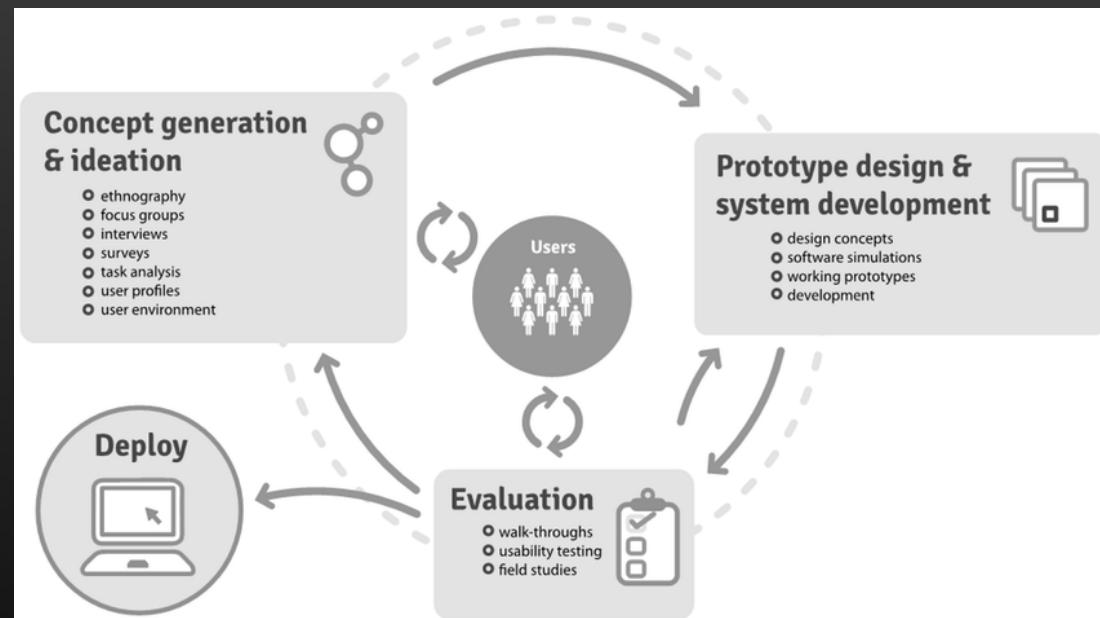
UCD: User Centered Design

Ponto de partida: *personas* & elaboração de cenários

Os requisitos são derivados dos cenários

Evolui através de ciclos de prototipagem e validação

Fortemente orientado para a satisfação do utilizador



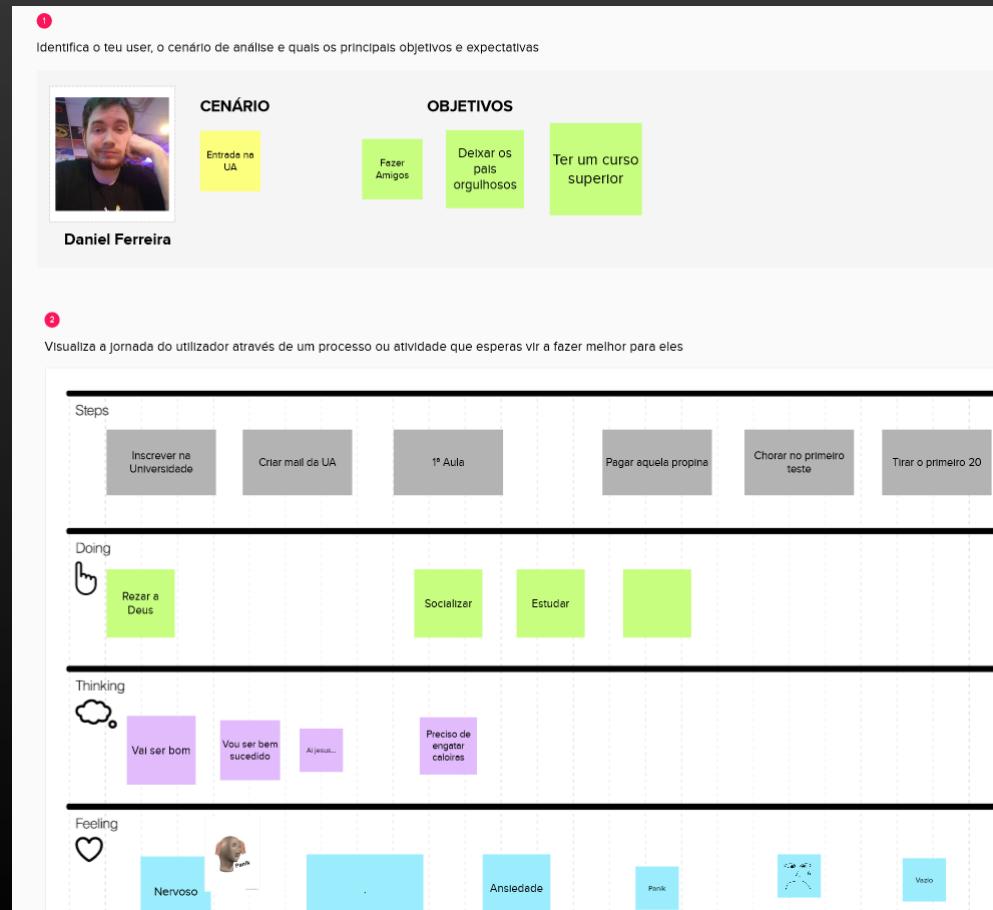
CJM: Customer Journey Maps

Foco na experiência global do utilizador

Procurar pontos fracos (“pain points”) e oportunidades

Instrumento visual: jornada do utilizador

Muito útil para gerar discussão e descoberta, em equipas pluridisciplinares



CaU: contam histórias que mostram os requisitos funcionais em contexto

Caso Utiliz.:	Consultar informação clínica (referenciação)
Motivação:	O Médico Dentista (MD) acede ao sistema para consulta informação clínica inserida pelo médico assistente.
Sequência típica:	<ol style="list-style-type: none">1. Inicia-se quando o Médico Dentista recebe um Utente portador de Cheque-Dentista para consulta.2. O MD acede à opção de pesquisa na sua página de entrada.3. O sistema apresenta o formulário de pesquisa, com as hipóteses de pesquisa por nr utente, nome, género e data de Nascimento.4. O MD insere o número de utente do SNS e confirma a pesquisa.5. O sistema pesquisa os cheques-dentista existentes para aquele utente e apresenta uma listagem ordenada do mais recente para o mais antigo.6. O MD seleciona uma entrada na lista para abrir a informação de detalhe.7. O sistema apresenta para esse CD o cabeçalho com a identificação do cheque e utente, e uma seção com a informação clínica disponível.
Sequências	...
RF.3	O sistema deve permitir a pesquisa de cheques-dentista (emitidos) por número de utente do SNS.

SRS – *Software Requirements Specification*

Documentação de requisitos: *SRS report*

Um relatório especial para documentar os requisitos.

Modelos candidatos:

- “System Proposal” by A.Dennis
- SRS by K.Wiegers
- “[Software requirements specification](#)” by ISO/IEC/IEEE 29148:2018 – International Standard – Systems and software engineering – Life cycle processes – Requirements engineering → international standard describing requirements engineering processes for development of software and hardware products.

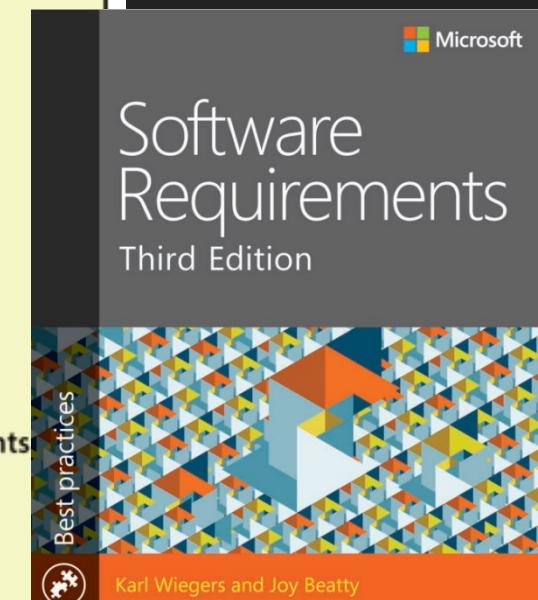


1. Introduction
1.1 Purpose
1.2 Scope
1.3 Product overview
1.3.1 Product perspective
1.3.2 Product functions
1.3.3 User characteristics
1.3.4 Limitations
1.4 Definitions
2. References
3. Specific requirements
3.1 External interfaces
3.2 Functions
3.3 Usability Requirements
3.4 Performance requirements
3.5 Logical database requirements
3.6 Design constraints
3.7 Software system attributes
3.8 Supporting information
4. Verification
(parallel to subsections in Section 3)
5. Appendices
5.1 Assumptions and dependencies
5.2 Acronyms and abbreviations

Figure 8 — Example SRS Outline

Wieger's proposal

- 1. Introduction**
 - 1.1 Purpose
 - 1.2 Document conventions
 - 1.3 Project scope
 - 1.4 References
 - 2. Overall description**
 - 2.1 Product perspective
 - 2.2 User classes and characteristics
 - 2.3 Operating environment
 - 2.4 Design and implementation constraints
 - 2.5 Assumptions and dependencies
 - 3. System features**
 - 3.x System feature X
 - 3.x.1 Description
 - 3.x.2 Functional requirements
 - 4. Data requirements**
 - 4.1 Logical data model
 - 4.2 Data dictionary
 - 4.3 Reports
 - 4.4 Data acquisition, integrity, retention, and disposal
 - 5. External interface requirements**
 - 5.1 User interfaces
 - 5.2 Software interfaces
 - 5.3 Hardware interfaces
 - 5.4 Communications interfaces
 - 6. Quality attributes**
 - 6.1 Usability
 - 6.2 Performance
 - 6.3 Security
 - 6.4 Safety
 - 6.x [others]
 - 7. Internationalization and localization requirements**
 - 8. Other requirements**
- Appendix A: Glossary**
- Appendix B: Analysis models**

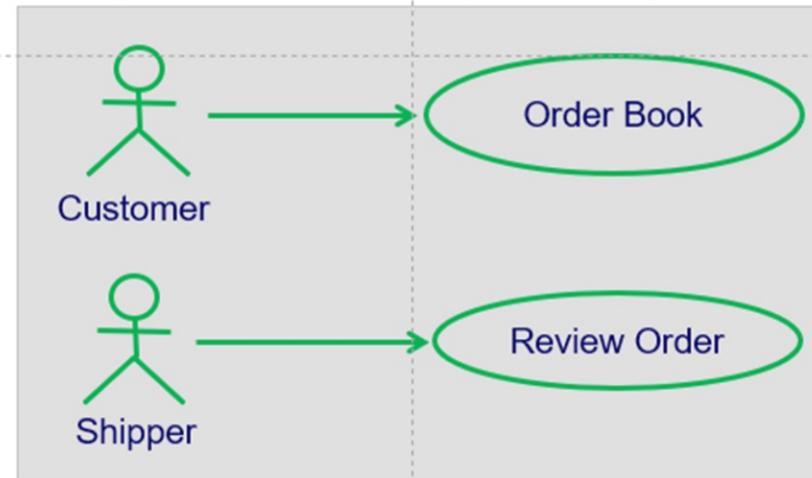


Os requisitos na OpenUP

Use Case Driven Development 🏆



- This practice describes how to capture requirements with a combination of use cases and system-wide requirements, and then drive development and testing from those use cases.



Artefactos relacionados com os requisitos no OpenUP?

Use-Case Model

Um conjunto de cenários típicos de utilização de um sistema. São sobretudo para requisitos funcionais (comportamento).

System-wide requirements

Um especificação suplementar com aquilo que não é adequado colocar nos casos de utilização. Este artefacto é sobretudo para todos os requisitos não-funcionais. É também o lugar para registar características funcionais não expressas (ou susceptíveis de ser expressas) como casos de utilização (por exemplo, uma geração de relatórios).

Glossary

O Glossário define termos dignos de nota. Abrange também o conceito de dicionário de dados, que regista requisitos relacionados com os dados, tais como regras de validação, valores aceitáveis, e assim por diante.

The screenshot shows the OpenUP web application interface. At the top, there's a navigation bar with links for 'Glossary' and 'Feedback'. Below the header, the main content area has a title 'Work Products > Requirements' and a sub-section title 'Domain: Requirements'. On the left, there's a sidebar menu under 'Team' with various categories like 'Introduction to OpenUP', 'Getting Started', 'Delivery Processes', 'Practices', 'Roles', and 'Work Products'. The 'Work Products' category is expanded, showing sub-options: 'Architecture', 'Deployment', 'Development', 'Environment', 'Project Management', 'Requirements', 'Test', 'Tasks', and 'Guidelines'. A green arrow points from the text 'Artefactos relacionados com os requisitos no OpenUP?' to the 'Requirements' link in the sidebar. To the right of the sidebar, there's a 'Relationships' section with a table. The first row of the table has a column labeled 'Work Products' containing a bulleted list: '• Glossary', '• Vision', '• System-Wide Requirements', '• Use-Case Model', and '• Use Case'. A green curly brace groups these five items. The bottom right corner of the interface has buttons for 'Expand All Sections' and 'Collapse'.

Examples of SRS reports

- SRS example from Wiegers
- System Proposal from Dennis (chap. 3)
 - * Not only requirements
- OpenUP: use cases + system-wide requirements

Types of information collected in requirements development

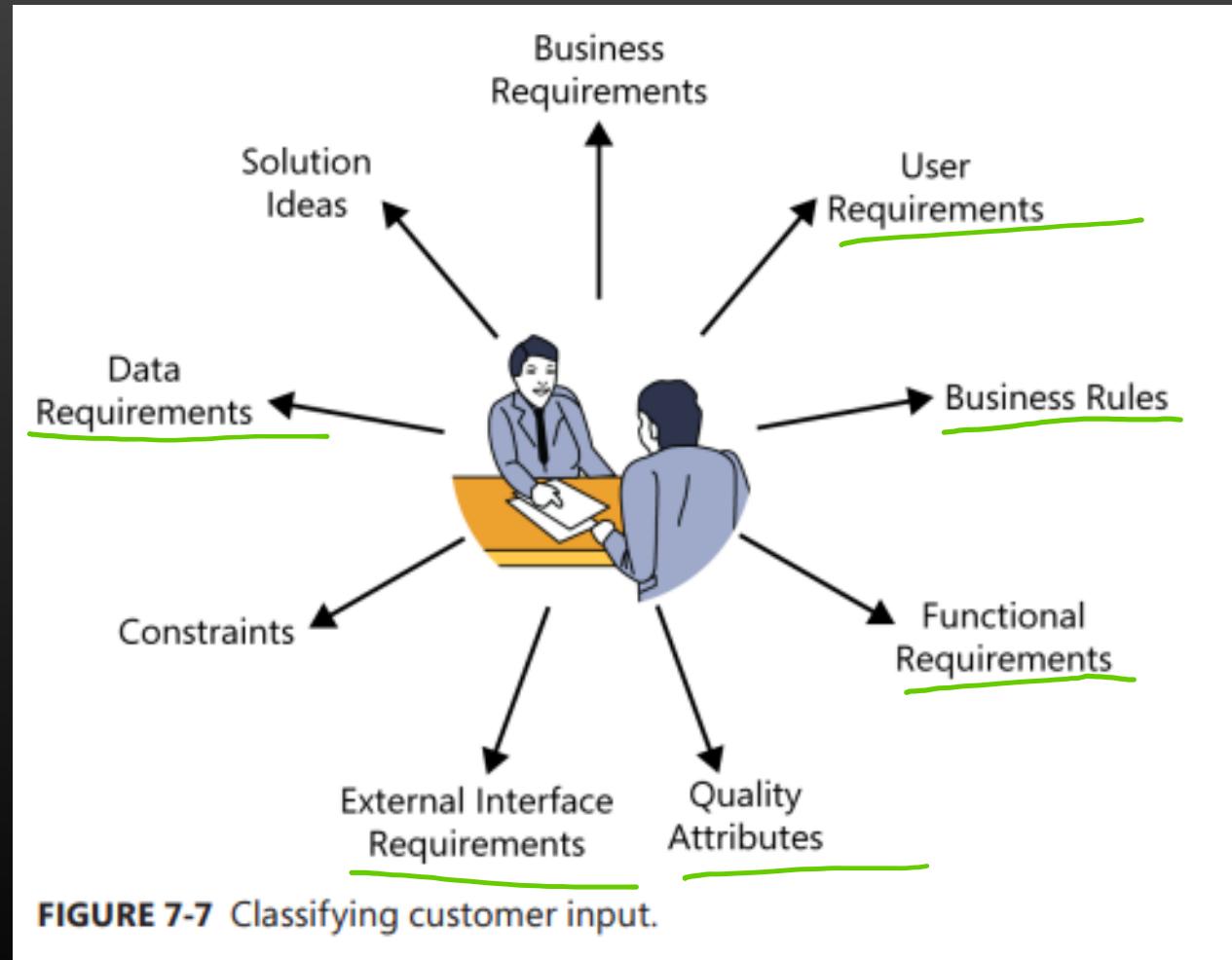


TABLE 1-1 Some types of requirements information

Term	Definition
Business requirement	A high-level business objective of the organization that builds a product or of a customer who procures it.
Business rule	A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the origin of several types of software requirements.
Constraint	A restriction that is imposed on the choices available to the developer for the design and construction of a product.
External interface requirement	A description of a connection between a software system and a user, another software system, or a hardware device.
Feature	One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements.
Functional requirement	A description of a behavior that a system will exhibit under specific conditions.
Nonfunctional requirement	A description of a property or characteristic that a system must exhibit or a constraint that it must respect.
Quality attribute	A kind of nonfunctional requirement that describes a service or performance characteristic of a product.
System requirement	A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware.
User requirement	A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute.

Abordagem para Análise de Sistemas?

The slide features the Ivar Jacobson International logo at the top left. To the right, the text "Creating winning teams." is displayed above a hexagonal grid composed of several overlapping hexagons in shades of blue, green, and yellow. Inside these hexagons are small, semi-transparent images of diverse individuals engaged in various activities, such as working on computers or discussing ideas. Below the grid, the title "Use-Case 2.0: The Hub of Modern Software Development" is prominently displayed in large, bold, black font. At the bottom of the slide, there is a navigation bar with standard video controls (play, stop, volume, etc.) and a timestamp indicating the video is 2:52 / 58:49.

<https://youtu.be/p5gDbf0je8k>

Domínio:

- Novos processos de trabalho
- Modelo dos conceitos do domínio

Requisitos:

Ponto de partida: use cases
...com a flexibilidade das user stories/use case slices

Requisitos evolutivos

Requisitos podem ser evolutivos?

Plan-driven, “waterfall” approach

Tentativa de definir e estabilizar os requisitos na primeira fase de um projeto (antes de qualquer programação)

Requisitos definidos à cabeça para o conjunto do sistema

CONTROLO

Estabilizar os requisitos à cabeça como estratégia para ↓ risco

Agile, evolutionary approach

Aceitar e preparar-se para pedidos/requisitos inevitavelmente mutáveis e/ou pouco claros dos stakeholders.

Com o suporte de algum tipo de abordagem sistemática para encontrar, documentar, organizar e acompanhar a evolução dos requisitos de um sistema.

ADAPTAÇÃO

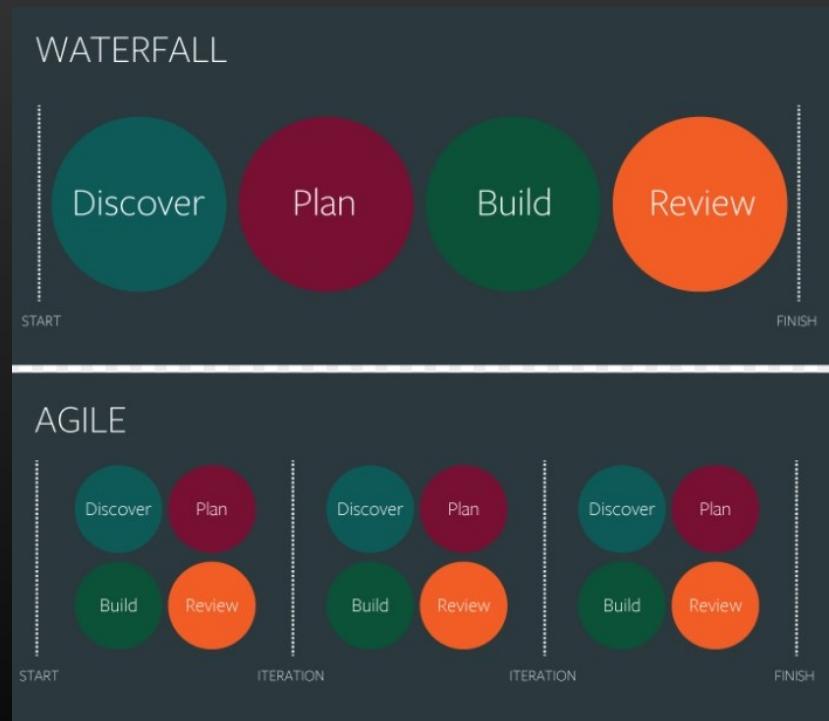
Prever formas de acomodar a mudança como estratégia para ↓ risco

Alto nível, em tempo útil; detalhado quando próximo da construção

Wiegers:

Não se tem de escrever o SRS para o produto inteiro antes de iniciar o desenvolvimento;

necessário definir os requisitos para cada incremento antes de o construir.



A OpenUP propõe requisitos evolutivos?

Practices > Technical Practices > Use Case Driven Development > Tasks > Identify and Outline Requirements

Task: Identify and Outline Requirements



This task describes how to identify and outline the requirements for the system so that the scope of work can be determined.

Disciplines: Requirements



Expand All Sections



Collapse All Sections

Purpose

The purpose of this task is to identify and capture functional and non-functional requirements for the system. These requirements form the basis of communication and agreement between the stakeholders and the development team on what the system must do to satisfy stakeholder needs. The goal is to understand the requirements at a high-level so that the initial scope of work can be determined. Further analysis will be performed to detail these requirements prior to implementation.

→ Back | Home

Práticas de eng.a de requisitos (fonte: K. Wiegers)

Requirements engineering activities

Discover/develop

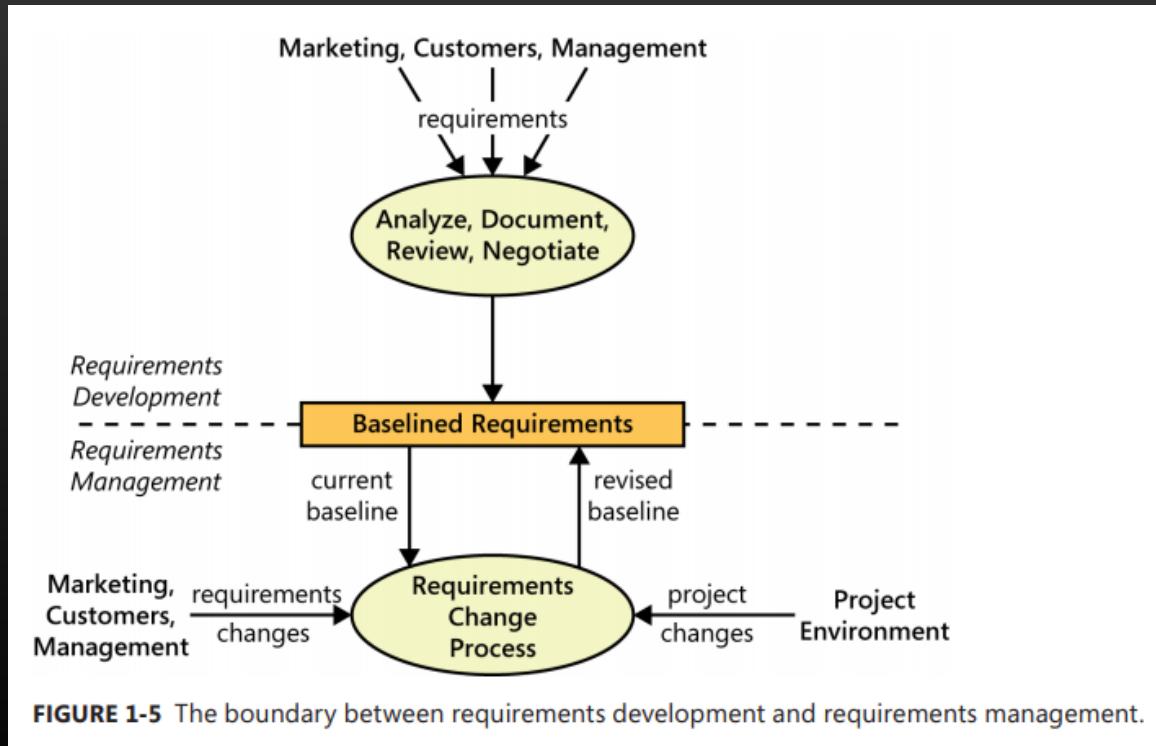
Requirements gathering and elicitation
Critical aspect in the overall lifecycle

Manage

Document, track and evolve
Apply a change process (assess the risk,...)

The approach to requirements development should be adapted to the kind of project in hands.

E.g.: how formal should be the requirements change request?



Requirements development activities

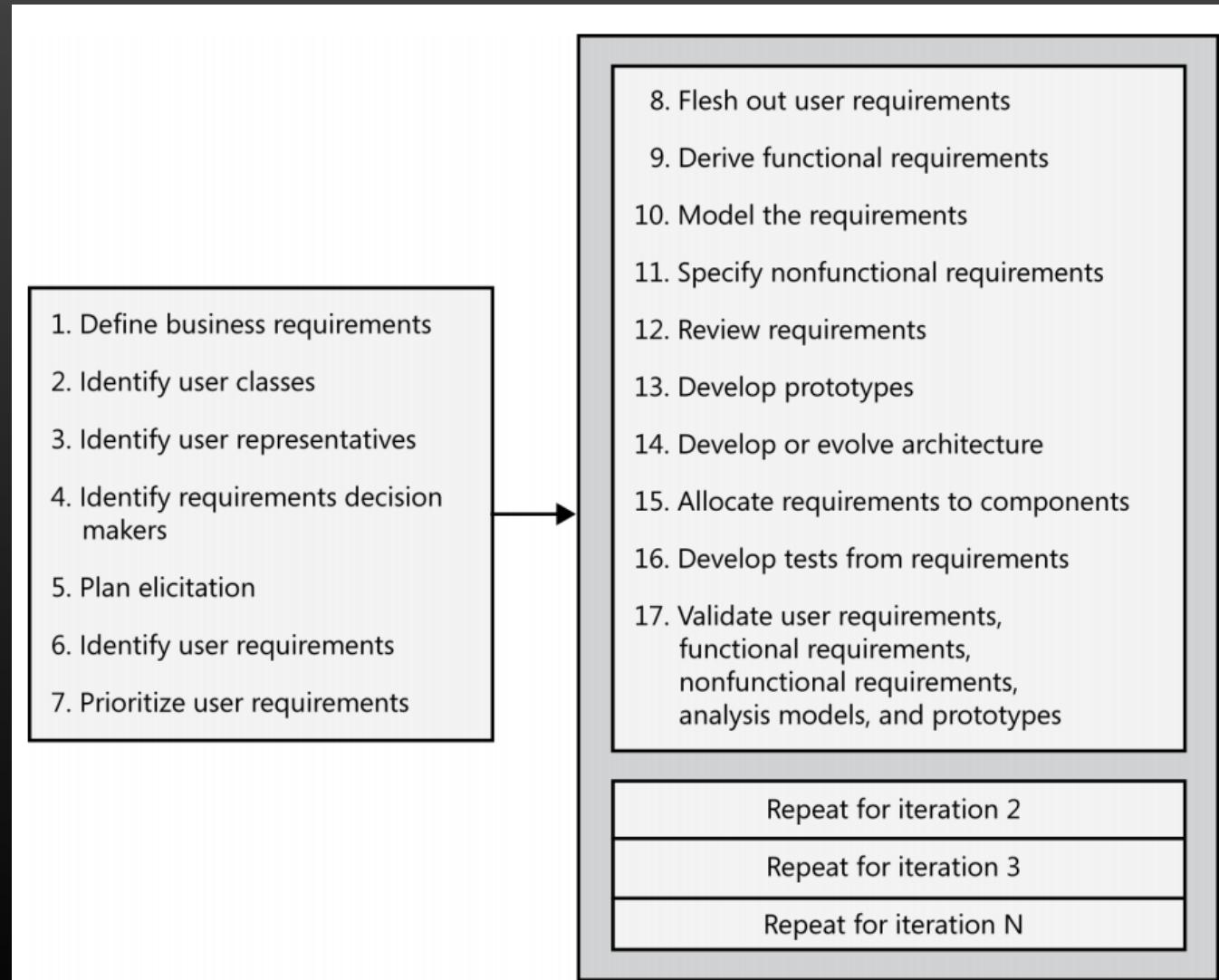


TABLE 3-1 Requirements engineering good practices

Elicitation	Analysis	Specification	Validation
<ul style="list-style-type: none">■ Define vision and scope■ Identify user classes■ Select product champions■ Conduct focus groups■ Identify user requirements■ Identify system events and responses■ Hold elicitation interviews■ Hold facilitated elicitation workshops■ Observe users performing their jobs■ Distribute questionnaires■ Perform document analysis■ Examine problem reports■ Reuse existing requirements	<ul style="list-style-type: none">■ Model the application environment■ Create prototypes■ Analyze feasibility■ Prioritize requirements■ Create a data dictionary■ Model the requirements■ Analyze interfaces■ Allocate requirements to subsystems	<ul style="list-style-type: none">■ Adopt requirement document templates■ Identify requirement origins■ Uniquely label each requirement■ Record business rules■ Specify nonfunctional requirements	<ul style="list-style-type: none">■ Review the requirements■ Test the requirements■ Define acceptance criteria■ Simulate the requirements
Requirements management		Knowledge	Project management
<ul style="list-style-type: none">■ Establish a change control process■ Perform change impact analysis■ Establish baselines and control versions of requirements sets■ Maintain change history■ Track requirements status■ Track requirements issues■ Maintain a requirements traceability matrix■ Use a requirements management tool		<ul style="list-style-type: none">■ Train business analysts■ Educate stakeholders about requirements■ Educate developers about application domain■ Define a requirements engineering process■ Create a glossary	<ul style="list-style-type: none">■ Select an appropriate life cycle■ Plan requirements approach■ Estimate requirements effort■ Base plans on requirements■ Identify requirements decision makers■ Renegotiate commitments■ Manage requirements risks■ Track requirements effort■ Review past lessons learned

Prototyping as a requirements validation practice

TABLE 15-1 Typical applications of software prototypes

	Throwaway	Evolutionary
Mock-up	<ul style="list-style-type: none">■ Clarify and refine user and functional requirements.■ Identify missing functionality.■ Explore user interface approaches.	<ul style="list-style-type: none">■ Implement core user requirements.■ Implement additional user requirements based on priority.■ Implement and refine websites.■ Adapt system to rapidly changing business needs.
Proof of concept	<ul style="list-style-type: none">■ Demonstrate technical feasibility.■ Evaluate performance.■ Acquire knowledge to improve estimates for construction.	<ul style="list-style-type: none">■ Implement and grow core multi-tier functionality and communication layers.■ Implement and optimize core algorithms.■ Test and tune performance.

Evolving requirements: life after the baseline

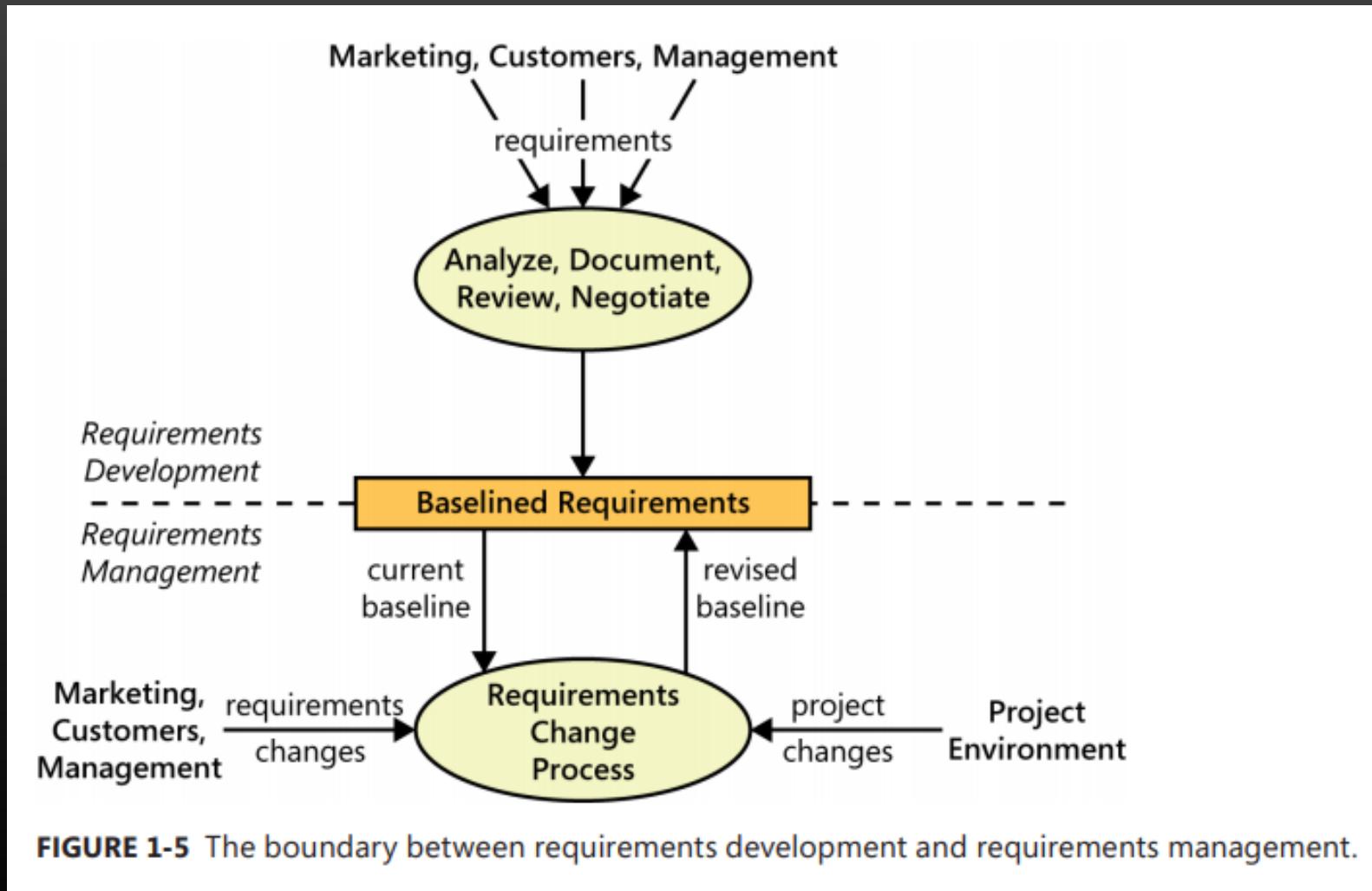
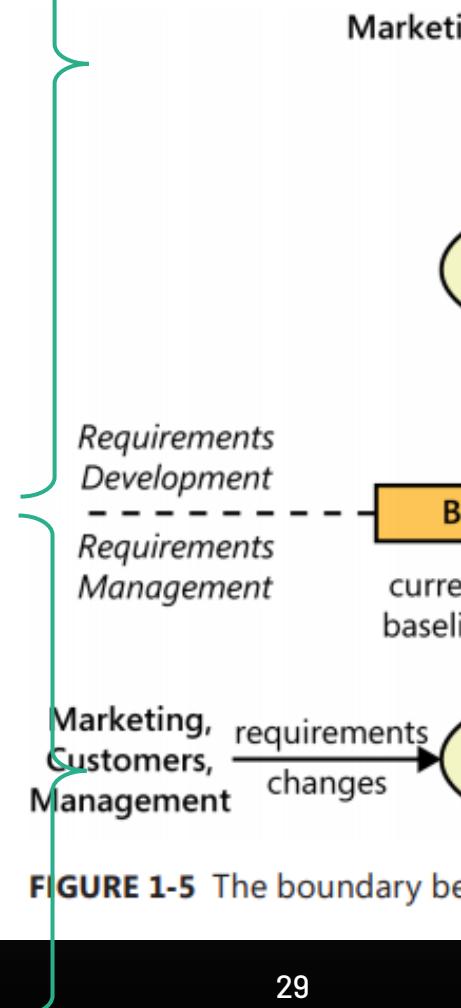


TABLE 3-1 Requirements engineering good practices

Elicitation	Analysis	Specification	Validation
<ul style="list-style-type: none">■ Define vision and scope■ Identify user classes■ Select product champions■ Conduct focus groups■ Identify user requirements■ Identify system events and responses■ Hold elicitation interviews■ Hold facilitated elicitation workshops■ Observe users performing their jobs■ Distribute questionnaires■ Perform document analysis■ Examine problem reports■ Reuse existing requirements	<ul style="list-style-type: none">■ Model the application environment■ Create prototypes■ Analyze feasibility■ Prioritize requirements■ Create a data dictionary■ Model the requirements■ Analyze interfaces■ Allocate requirements to subsystems	<ul style="list-style-type: none">■ Adopt requirement document templates■ Identify requirement origins■ Uniquely label each requirement■ Record business rules■ Specify nonfunctional requirements	<ul style="list-style-type: none">■ Review the requirements■ Test the requirements■ Define acceptance criteria■ Simulate the requirements
Requirements management <ul style="list-style-type: none">■ Establish a change control process■ Perform change impact analysis■ Establish baselines and control versions of requirements sets■ Maintain change history■ Track requirements status■ Track requirements issues■ Maintain a requirements traceability matrix■ Use a requirements management tool	Knowledge <ul style="list-style-type: none">■ Train business analysts■ Educate stakeholders about requirements■ Educate developers about application domain■ Define a requirements engineering process■ Create a glossary	Project management <ul style="list-style-type: none">■ Select an appropriate life cycle■ Plan requirements approach■ Estimate requirements effort■ Base plans on requirements■ Identify requirements decision makers■ Renegotiate commitments■ Manage requirements risks■ Track requirements effort■ Review past lessons learned	<p>The diagram illustrates the relationship between Requirements Development and Requirements Management. A vertical curly brace on the left groups the first three columns (Elicitation, Analysis, Specification) under 'Requirements Development'. A horizontal dashed line separates 'Requirements Development' from 'Requirements Management'. To the right of this line, a vertical curly brace groups 'Requirements Management' and 'Project management'. Above this, a bracket labeled 'Marketing, Customers, Management' spans both columns. An arrow points from this bracket to the word 'changes' at the bottom right, indicating that external factors influence requirements changes.</p>



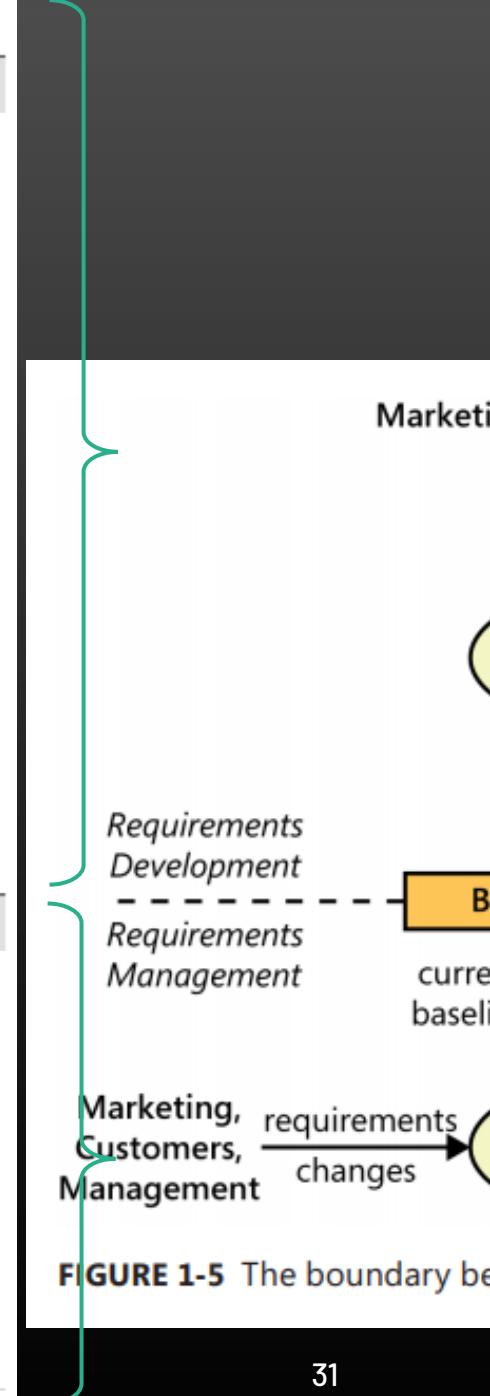
What is a requirements traceability matrix?

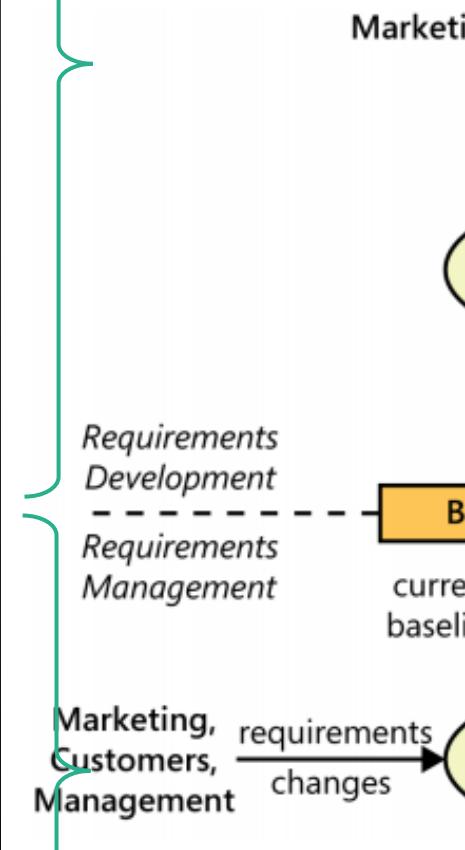
The most common way to represent the links between requirements and other elements in a system is in a requirements traceability matrix.

TABLE 29-2 Requirements traceability matrix showing links between use cases and functional requirements

Functional requirement	Use case			
	UC-1	UC-2	UC-3	UC-4
FR-1	↳			
FR-2	↳			
FR-3			↳	
FR-4			↳	
FR-5		↳		↳
FR-6			↳	

TABLE 3-1 Requirements engineering good practices

Elicitation	Analysis	Specification	Validation
<ul style="list-style-type: none">■ Define vision and scope■ Identify user classes■ Select product champions■ Conduct focus groups■ Identify user requirements■ Identify system events and responses■ Hold elicitation interviews■ Hold facilitated elicitation workshops■ Observe users performing their jobs■ Distribute questionnaires■ Perform document analysis■ Examine problem reports■ Reuse existing requirements	<ul style="list-style-type: none">■ Model the application environment■ Create prototypes■ Analyze feasibility■ Prioritize requirements■ Create a data dictionary■ Model the requirements■ Analyze interfaces■ Allocate requirements to subsystems	<ul style="list-style-type: none">■ Adopt requirement document templates■ Identify requirement origins■ Uniquely label each requirement■ Record business rules■ Specify nonfunctional requirements	<ul style="list-style-type: none">■ Review the requirements■ Test the requirements■ Define acceptance criteria■ Simulate the requirements
Requirements management	Knowledge	Project management	 <ul style="list-style-type: none">■ Establish a change control process■ Perform change impact analysis■ Establish baselines and control versions of requirements sets■ Maintain change history■ Track requirements status■ Track requirements issues■ Maintain a requirements traceability matrix■ Use a requirements management tool

**FIGURE 1-5** The boundary be-

Requirement attributes to consider:

- Date the requirement was created
- Current version number of the requirement
- Author who wrote the requirement
- Priority
- Status
- Origin or source of the requirement
- Rationale behind the requirement
- Release number or iteration to which the requirement is allocated
- Stakeholder to contact with questions or to make decisions about proposed changes
- Validation method to be used or acceptance criteria

Requirements Management Tool

- Larger project teams will benefit from letting users import requirements from source documents, define attribute values, filter and display the database contents, export requirements in various formats, define traceability links, and connect requirements to items stored in other software development tools.

E.g.: [IBM Rational DOORS](#)

Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Wiegers13] – Multiple chapters used.	<ul style="list-style-type: none">• [Dennis15] – Chap. 3 – Requirements Determination• [Pressman15] – Chap. 8 – Understanding Requirements

47006- ANÁLISE E MODELAÇÃO DE SISTEMAS

Práticas da Análise no projeto de ASis

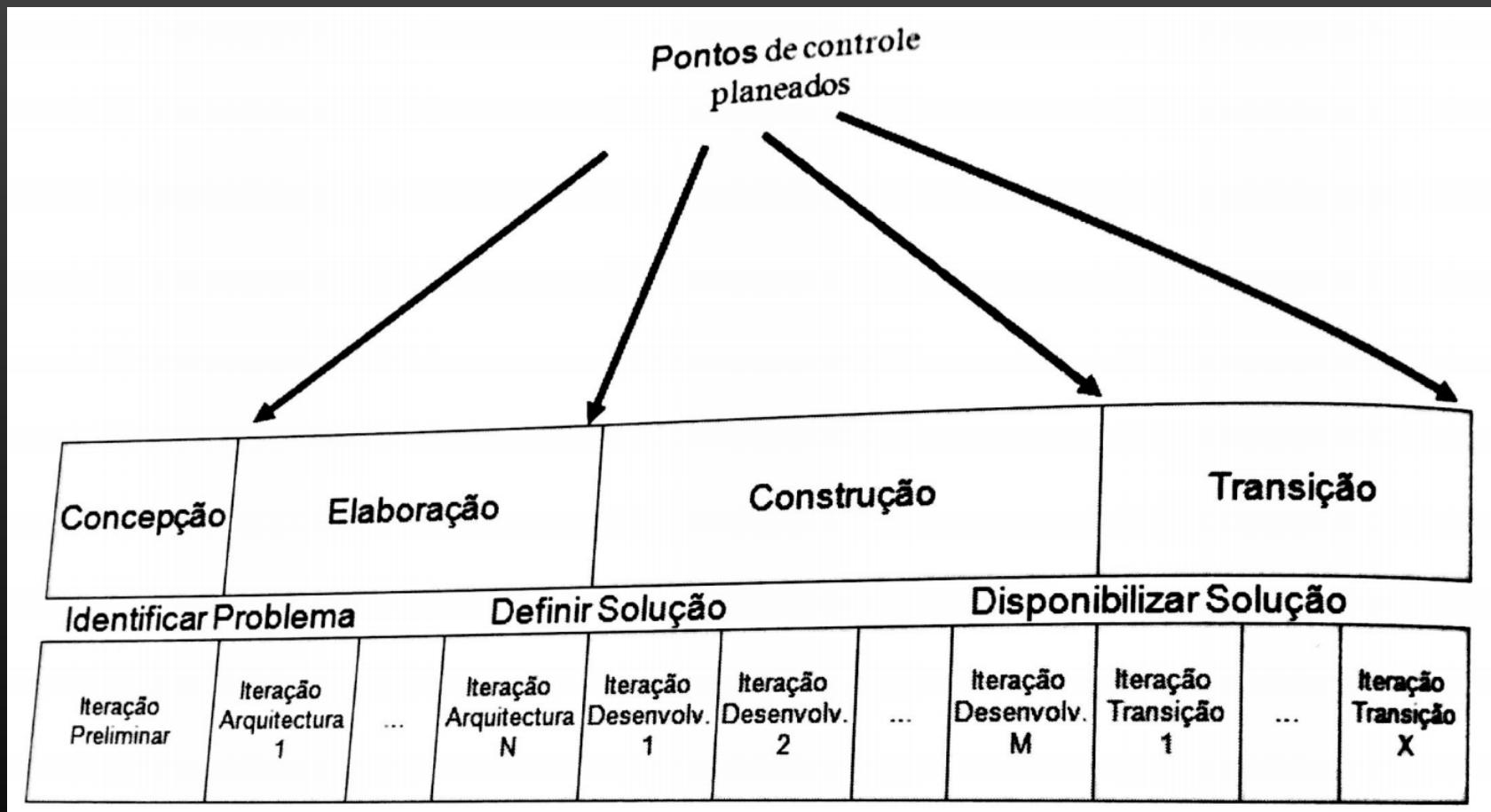
Ilídio Oliveira

V2022/05/10

Objetivos de aprendizagem

- Explicar as próximas atividades no desenvolvimento do projeto em grupo.
- Identificar os resultados produzidos pelos Analista na investigação do domínio (relacionado com o OpenUP).

OpenUP: Fases, iterações e pontos de controlo



1 2 3 4

Novos processos

Reengenharia de processos

Analizar a forma de trabalhar da organização

- Novo conceito (*green-field*)?
- Reengenharia?
- Ainda não é uma especificação de software!

Procurando oportunidades de melhoria

- “gerar valor”
- Processos mais rápidos, mais convenientes, novos serviços,...

Potenciadas pelas TI



<https://www.gartner.com/en/information-technology/insights/top-technology-trends>

Elementos de um processo

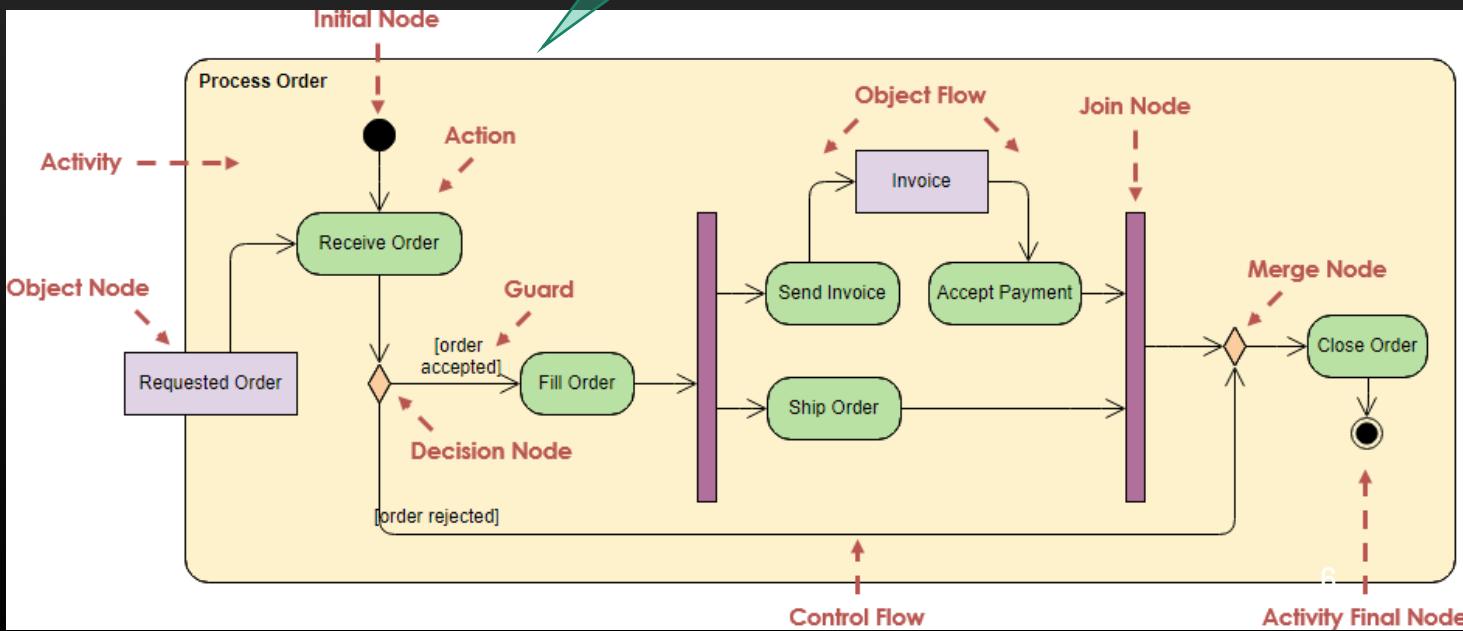
Intervenientes/papéis

Tarefas/"Ações" & Eventos

Fluxo/Decisão

Inputs/Outputs (informação)

O Analista pode usar D. Atividades para documentar os futuros processos



Problemas mais frequentes

Um (único) processo (!)

- Vs: Compra online,
Monitorização da entrega,
Gestão de parcerias,...

Detalhe excessivo para o objetivo

- clarificar as responsabilidade e
fluxo no novo “conceito”

Pouco estudo do domínio...

Focar num caso particular e não
no domínio/área

Modelo [dos conceitos] do domínio

Mapa de conceitos

Os conceitos de um problema
tendem a manter-se estáveis

Input para:

- Vocabulário coerente nas especificações
- Bases de dados
- “Domain-driven design”
- Podem revelar *Business rules*

Problemas mais frequentes

Informação a menos

Faltam entidades, faltam atributos,
não responde ao prometido nos
CaU

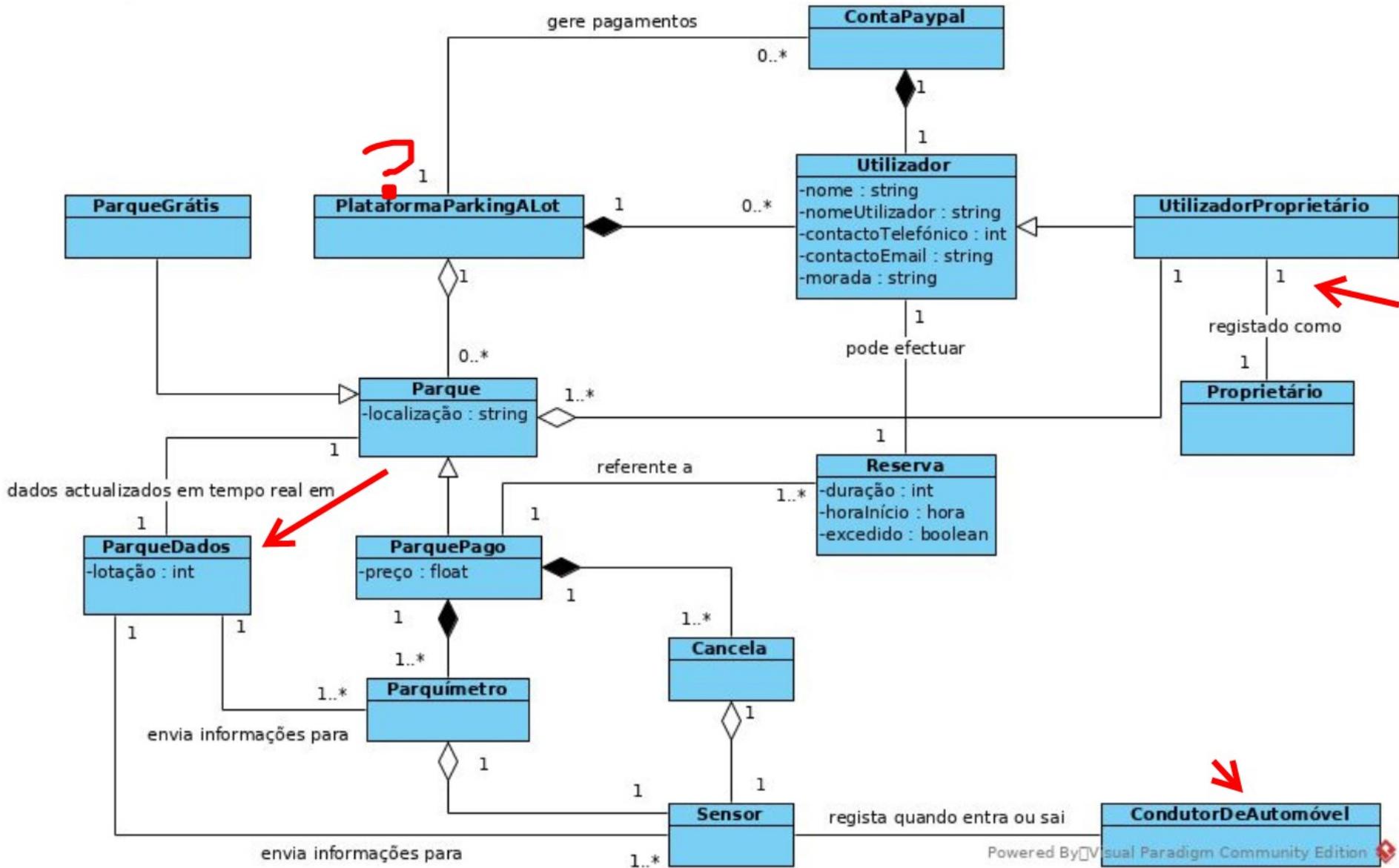
Tempo

Consigo reproduzir a informação
de há 6 meses? Consigo produzir
“mapas mensais”?

Detalhes da implementação

Login?...

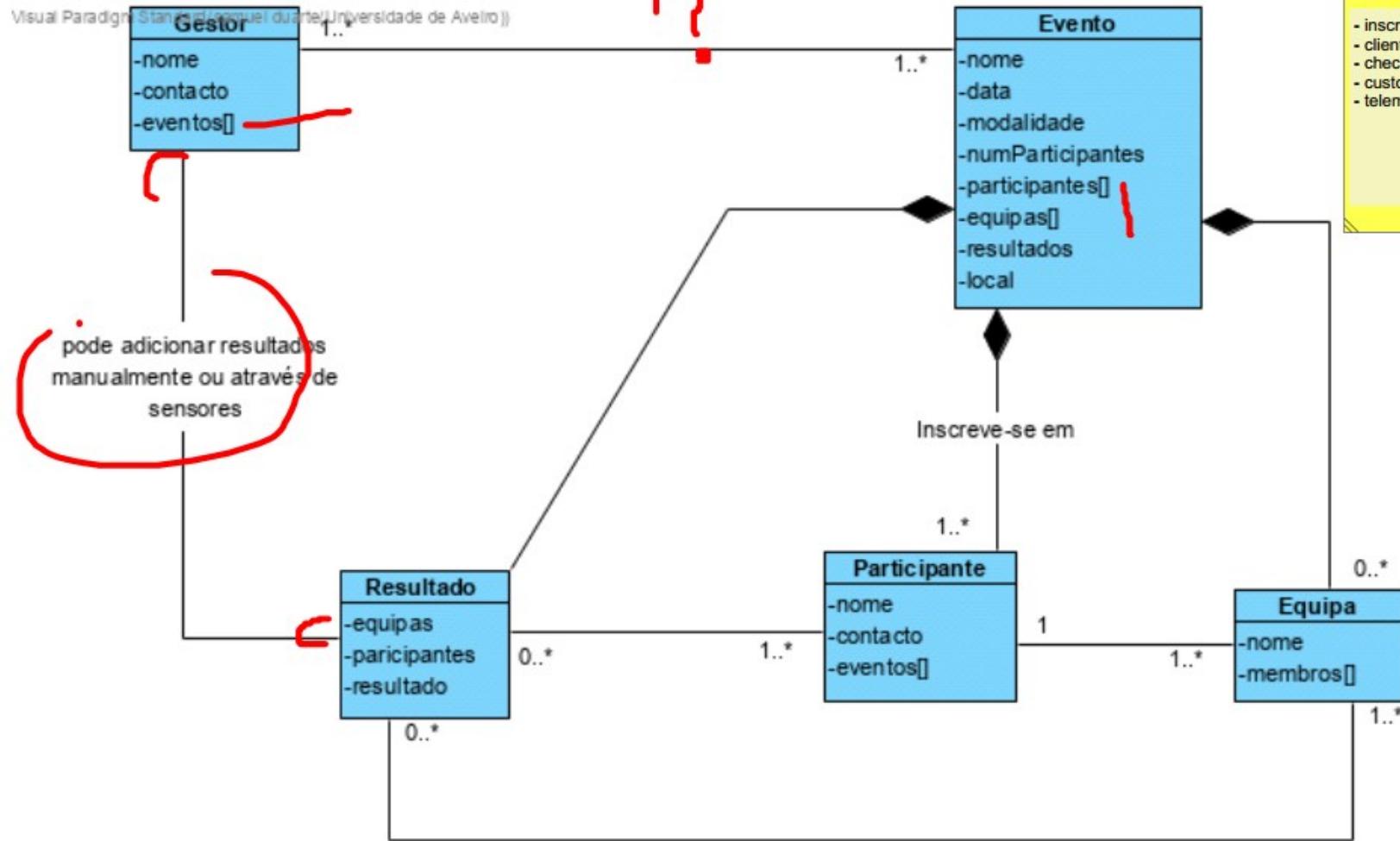
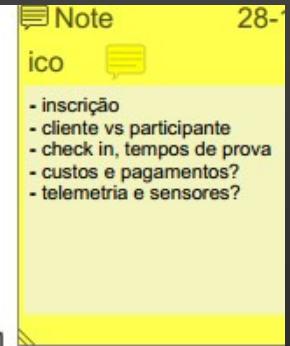
Exemplo para discussão (gestão de parques)



Exemplo p/ discussão (gestão de corridas)

Visual Paradigm Standard (manuel duarte/Universidade de Aveiro))

28-



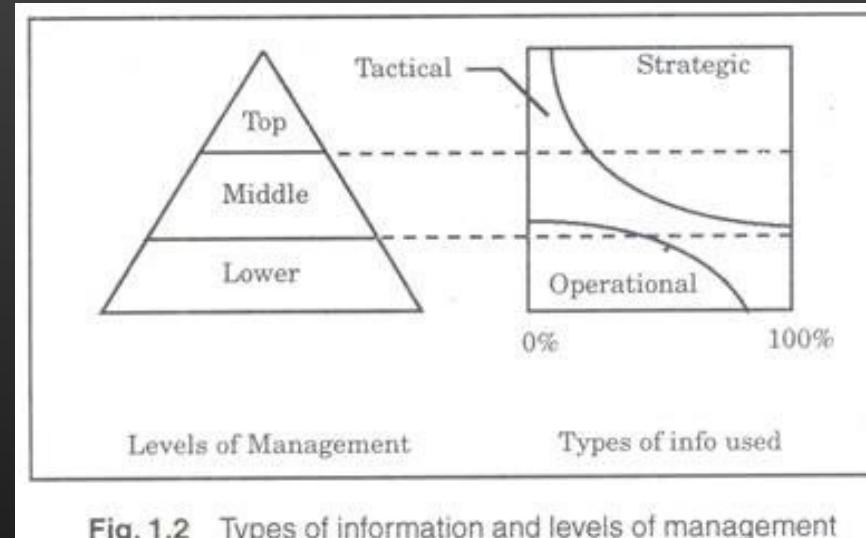
Informação a diferentes níveis

A componente operacional está assegurada?

→ Suporte às transações do dia-a-dia

A componente tática está assegurada?

→ Informação para a camada (intermedia) de gestão do negócio



O “big-boss” como um *stakeholder*...

Papel dos casos de uso

Posicionamento do UP

Segundo os autores:

“architecture centric”

“use-case driven”

“interactive and incremental”

Desenvolvimento por casos de utilização

"This practice also explains how use cases and scenarios are best developed in conjunction with (and used to drive) other development activities, including design and testing."

Practices > Technical Practices > Use Case Driven Development

Practice: Use Case Driven Development



This practice describes how to capture requirements with a combination of use cases and system-wide requirements, and then drive development and testing from those use cases.

Expand All Sections

Collapse All Sections

Relationships

Content References

- How to Adopt the Use-Case Driven Development Practice
- Key Concepts
 - Requirements
 - Use Case
 - Actor
 - Use-Case Model
- Work Products
 - Use Case
 - Use-Case Model
 - System-Wide Requirements
- Tasks
 - Identify and Outline Requirements
 - Detail Use-Case Scenarios
 - Detail System-Wide Requirements
- Guidance
 - Guidelines
 - Detail Use Cases and Scenarios
 - Identify and Outline Actors and Use Cases
 - Developing System-Wide Requirements Specification
 - Use-Cases Realizations



Inputs

- [Technical Specification]

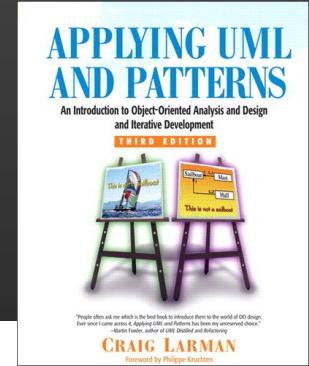
Relevância para a análise de requisitos

The Boss Test

Your boss asks, "What have you been doing all day?" You reply: "Logging in!" Is your boss happy?

If not, the use case fails the Boss Test, which implies it is not strongly related to achieving results of measurable value. It may be a use case at some low goal level, but not the desirable level of focus for requirements analysis.

That doesn't mean to always ignore boss-test-failing use cases. User authentication may fail the boss test, but may be important and difficult.



E os requisitos não funcionais

“System-wide specification”

Situações de modelação com CaU

USE CASE 24: FULLY DRESSED USE CASE TEMPLATE <NAME>

<the name should be the goal as a short active verb phrase>

Context of use: <a longer statement of the goal, if needed, its normal occurrence conditions>

Scope: <design scope, what system is being considered black-box under design>

Level: <one of: summary, user-goal, subfunction>

Primary Actor: <a role name for the primary actor, or description>

Stakeholders & Interests: <list of stakeholders and key interests in the use case>

Precondition: <what we expect is already the state of the world>

Minimal Guarantees: <how the interests are protected under all exits>

Success Guarantees: <the state of the world if goal succee

Extensions:

Trigger: <what starts the use case, may be time event>

<put here there [sic] extensions, one at a time, each referring to the step of the main scenario>

Main Success Scenario:

<put here the steps of the scenario from trigger to goal del

<step altered> <condition>: <action or sub use case>

<step #> <action description>

<step altered> <condition>: <action or sub use case>



Start simple, then refine.

Avoid "overloaded" use-case specification templates

Technology & Data Variations List:

<put here the variations that will cause eventual bifurcation in the scenario>

Atores primários e secundários

Autor primário

Solicita o sistema para resolver problemas/realizar objetivos

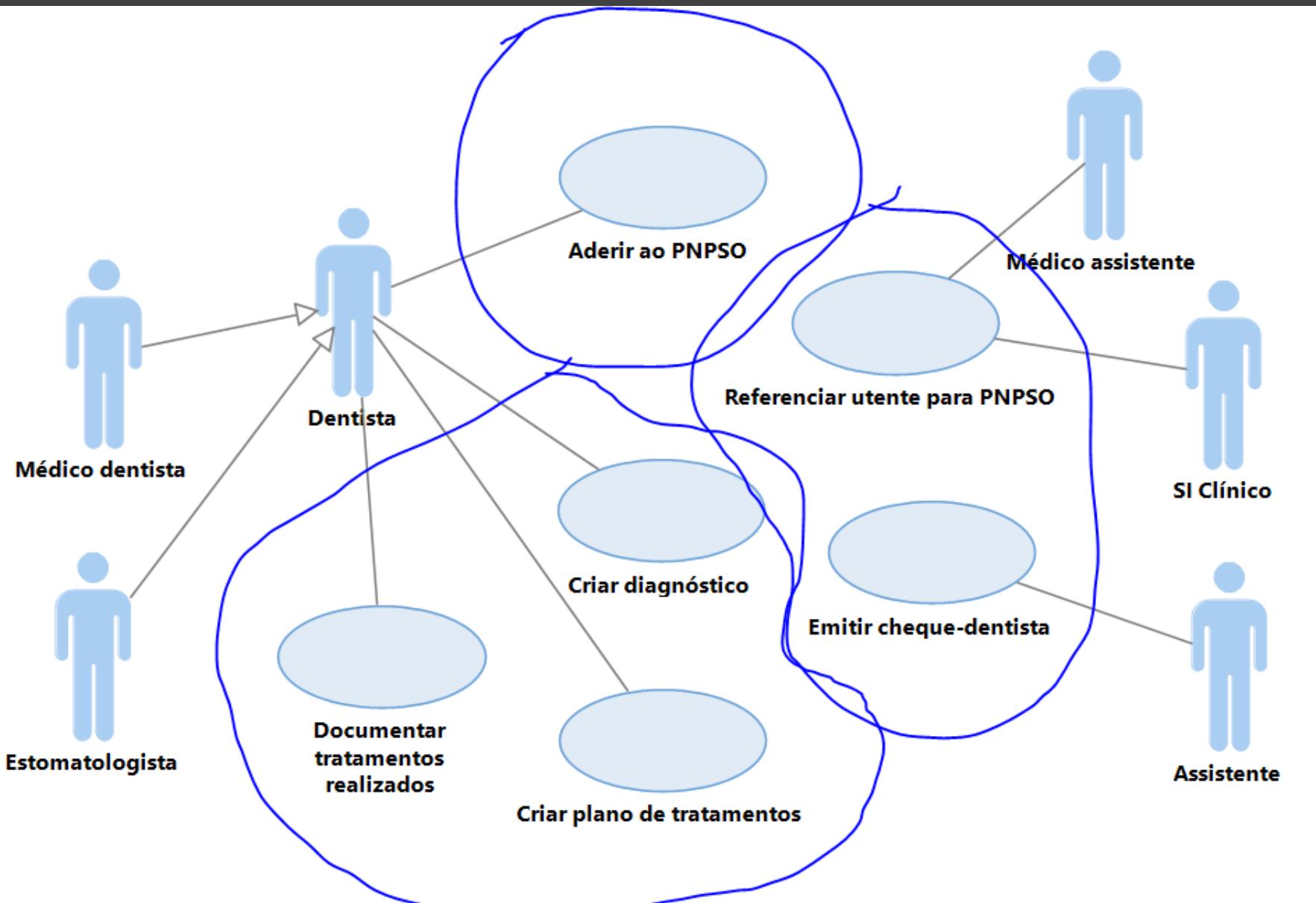
Os CaU são iniciados por um Ator primário

Autor secundário

Fornece serviços ou informação para algum cenário do CaU

Podem ser sistemas externos ou papéis de pessoas, que não são utilizadores

Podemos agrupar os casos de utilização em temas



Os casos de utilização podem ser agrupados em pacotes

Actors

Use cases

general

- + Add to Wish List
- + Cancel Order
- + Edit Shopping Cart
- + Login
- + Logout
- + Open an Account
- + Return a book
- + View Order History
- + Where's My Stuff?

admin

- + Customer Service
- + Seller
- + Shipping Clerk
- + Webmaster
- + Add Books to Catalog
- + Add Editorial Review
- + Add External Books to Catalog
- + Dispatch Order
- + Moderate Customer Reviews
- + Monitor Stock Levels
- + Order Books from Publisher
- + Process Refund
- + Remove Books from Catalog
- + Remove External Books from Catalog
- + Respond to Enquiry
- + Unlock Locked Account

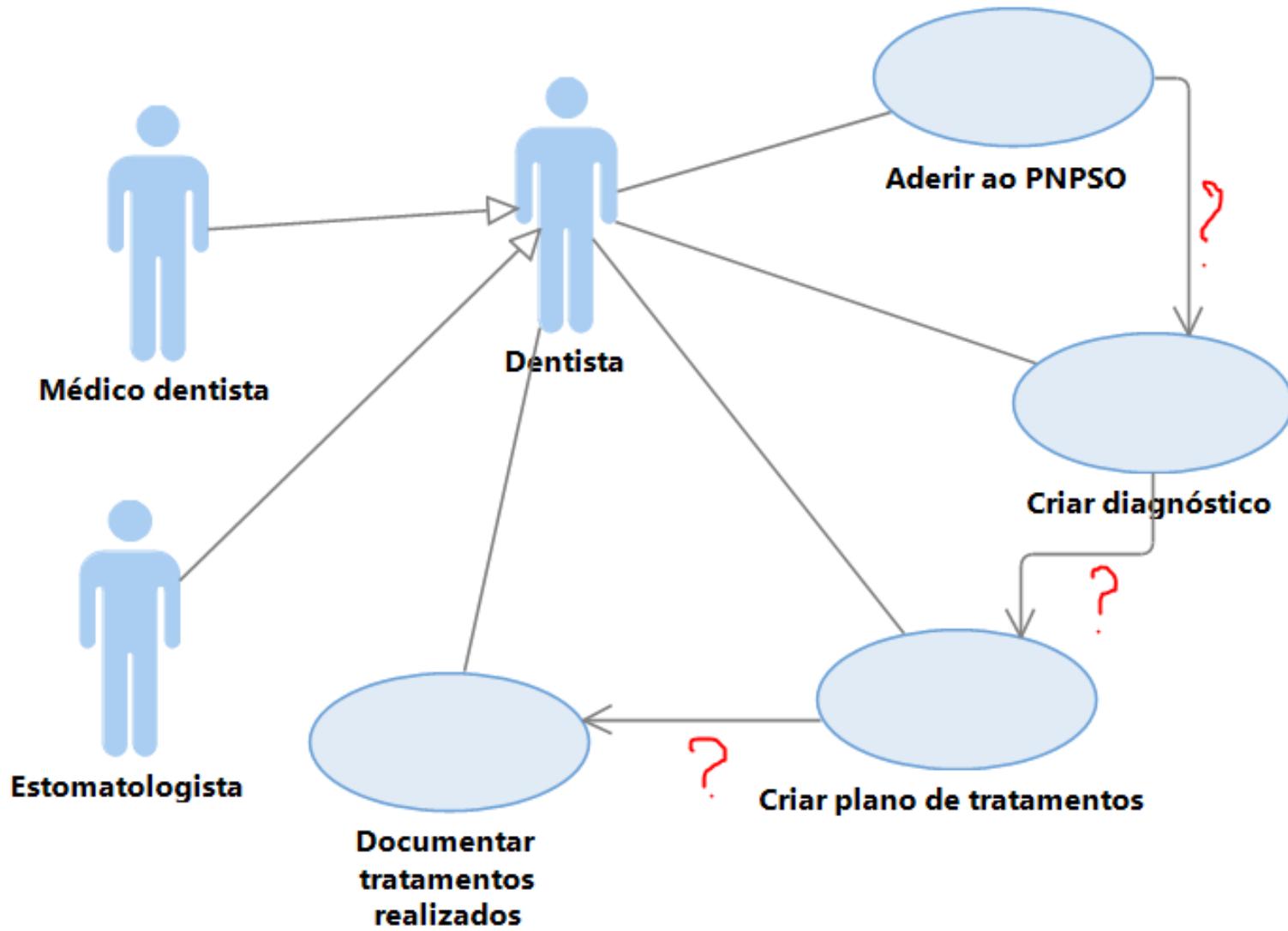
shopping

- + Customer
- + Add Item to Shopping Cart
- + Checkout
- + Edit Shopping Cart
- + Enter Address
- + Pay by Card
- + Pay by Check
- + Pay by Purchase Order
- + Remove Item From Shopping Cart
- + View Recommendations
- + View Review
- + Write Reader Review

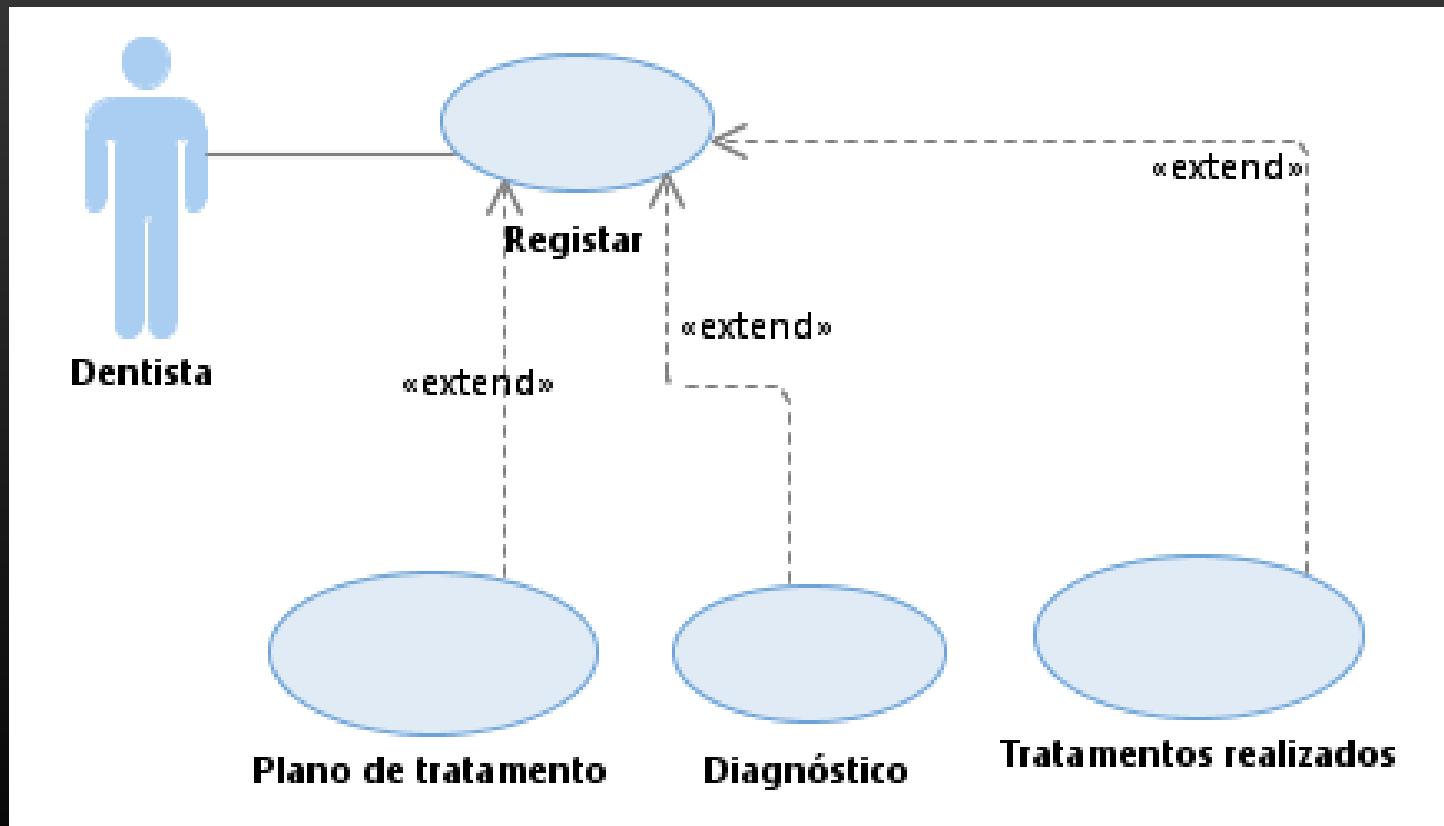
searching

- + Advanced Search
- + Search by Author
- + Search by Category
- + Search by Keyword
- + Search by Title
- + Search for Books

CaU não mostram workflow

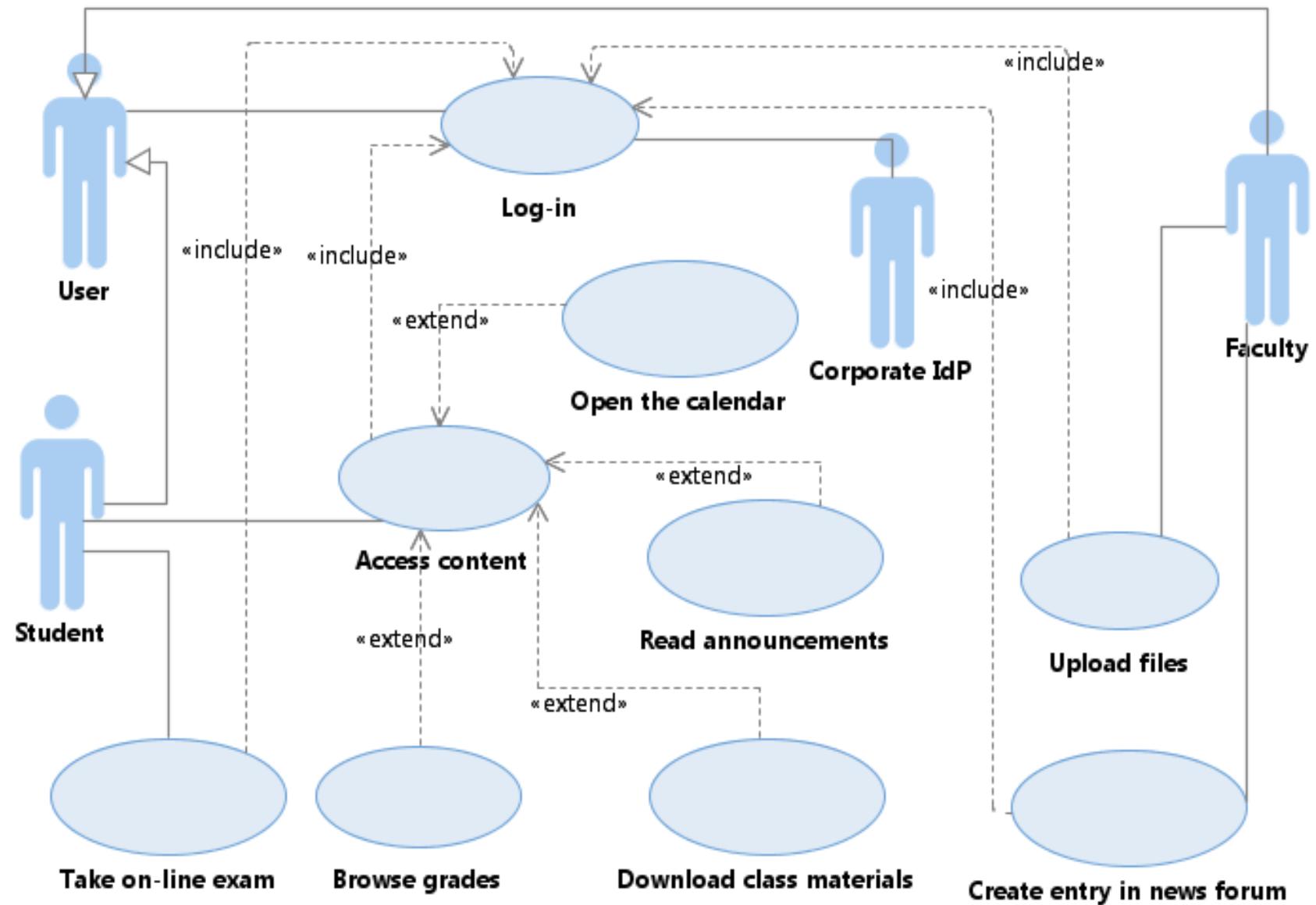


Decomposição/Agregação funcional

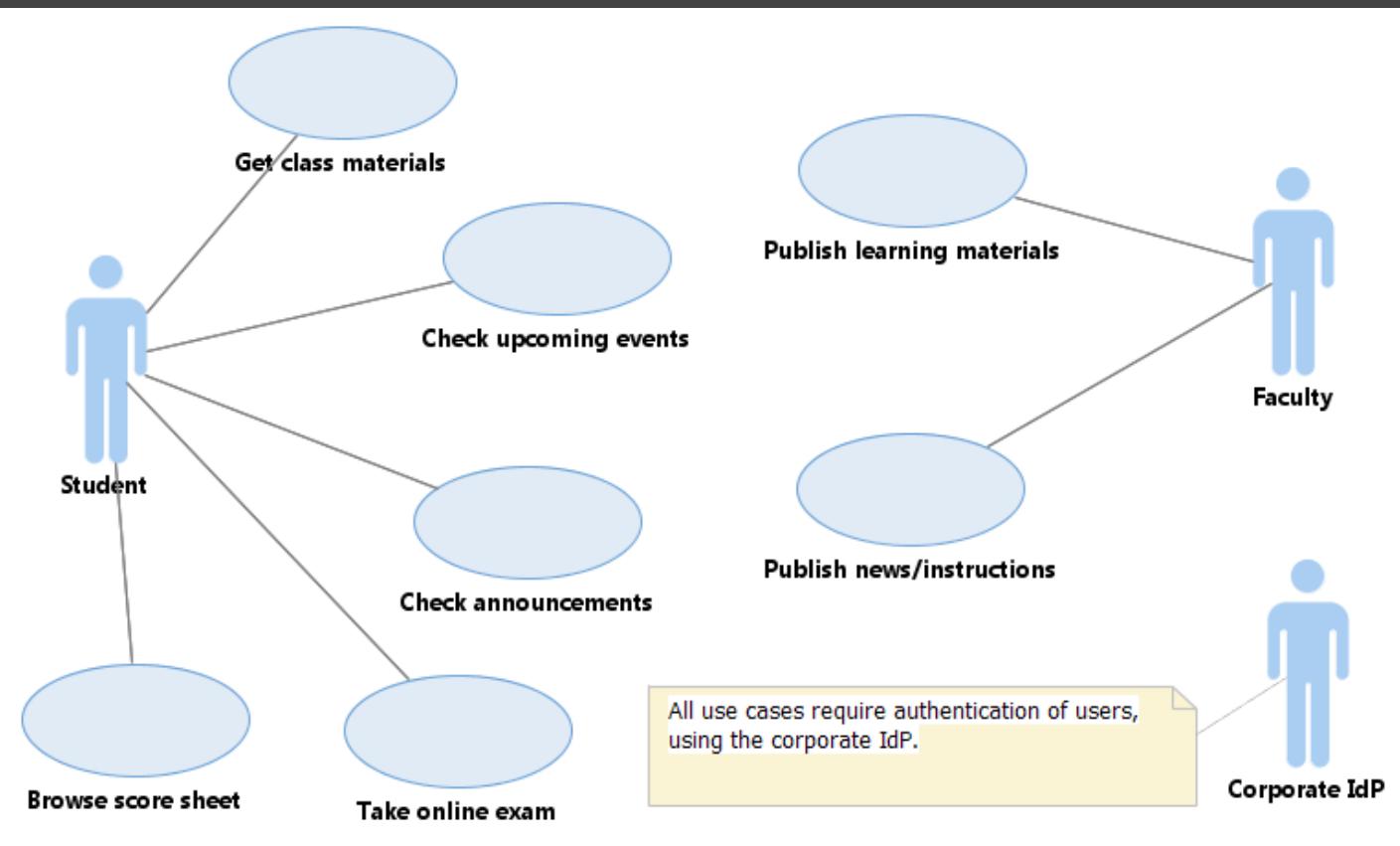


Login e Logout são casos de utilização?

Casos de utilização Moodle – Opção 1

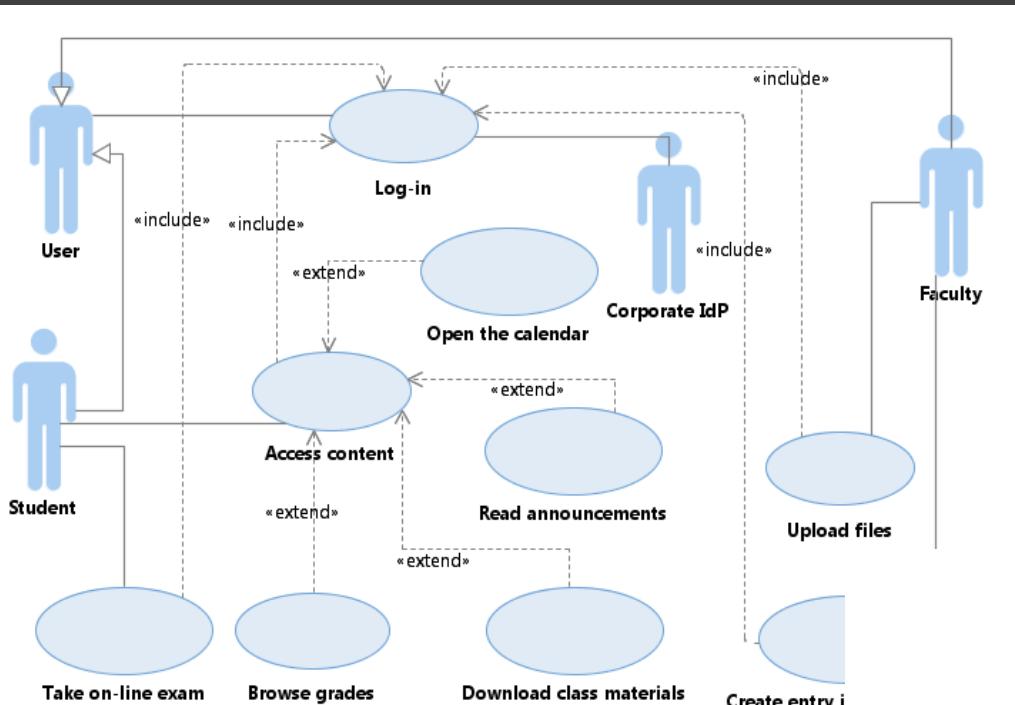


Casos de utilização Moodle – Opção 2

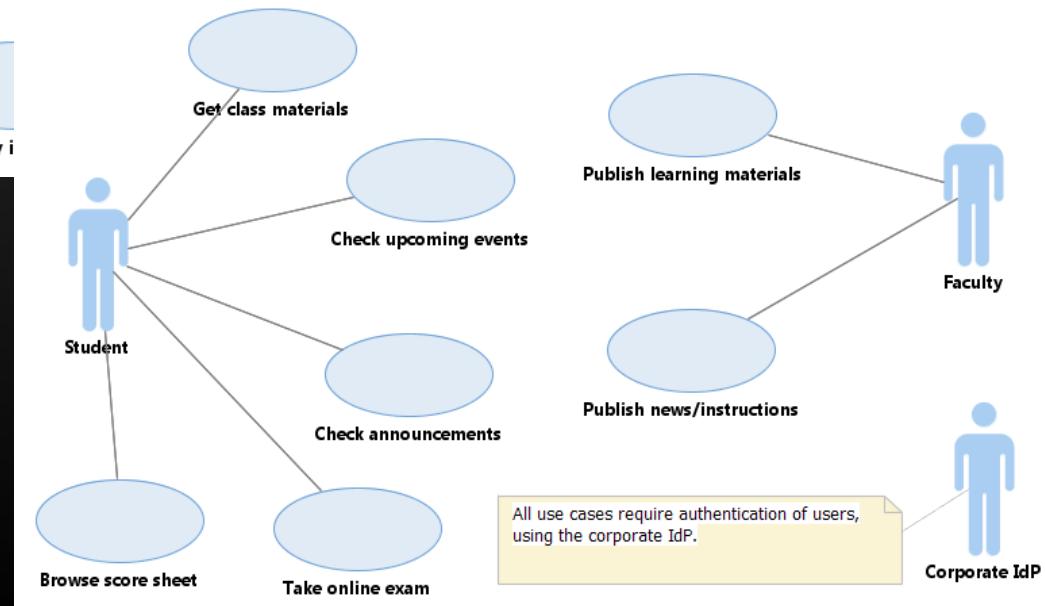


Evitar o uso de semântica avançada nos modelos, quando pode ser mais simples /direto.

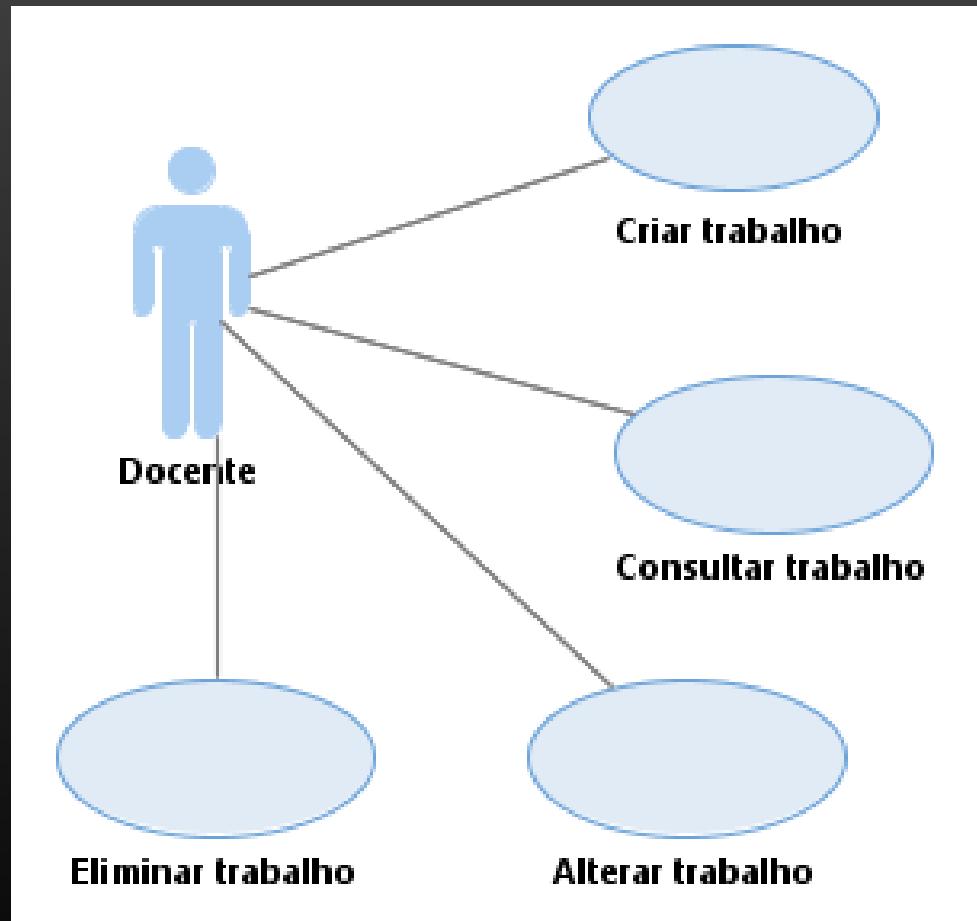
Correção? Clareza? Eficácia?



Evitar o uso de semântica avançada nos modelos, quando pode ser mais simples /direto.



Como lidar com o CRUD?



Como lidar com o CRUD?

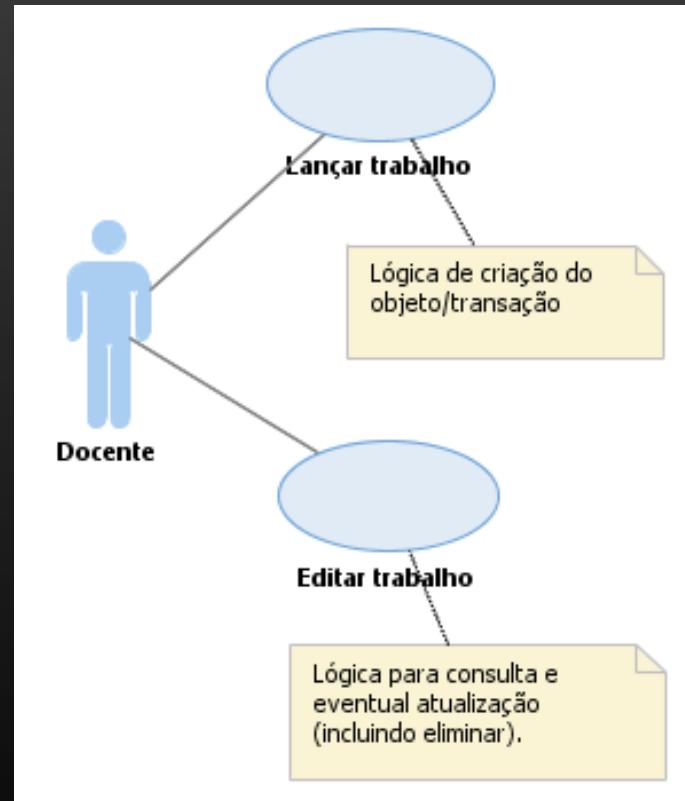
Depende do problema

Cancelar Cheque-dentista vs
Cancelar trabalho (Moodle)

Frequentemente, pode ser
resolvido com dois CaU

O fluxo que cria

O fluxo que consulta/atualiza



Desenvolvimento iterativo e incremental

THE UNIFIED SOFTWARE DEVELOPMENT PROCESS

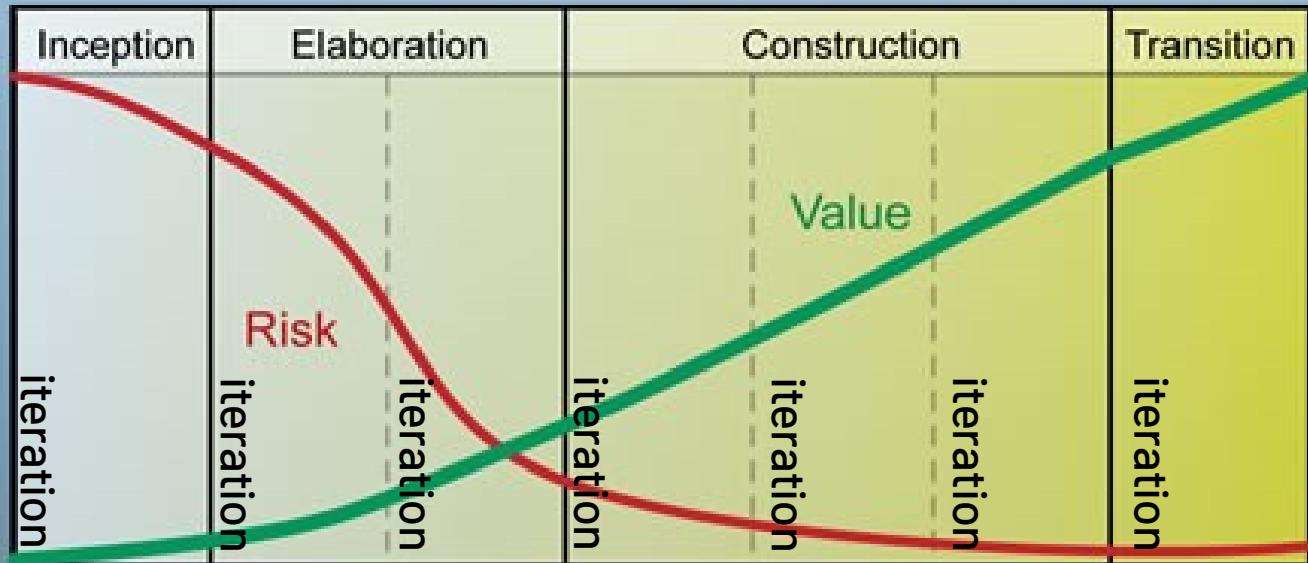
IVAR JACOBSON
GRADY BOOCHE
JAMES RUMBAUGH



The complete guide to the Unified Process from the original designers



Project Lifecycle

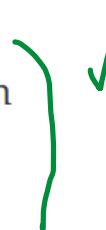


A B

Process: How to Work With Use Cases in Iterative Methods?

Use cases are central to the UP and many other iterative methods. The UP encourages **use-case driven development**. This implies:

- Functional requirements are primarily recorded in use cases (the Use-Case Model); other requirements techniques (such as functions lists) are secondary, if used at all.
- Use cases are an important part of iterative planning. The work of an iteration is—in part—defined by choosing some use case scenarios, or entire use cases. And use cases are a key input to estimation.
- **Use-case realizations** drive the design. That is, the team designs collaborating objects and subsystems in order to perform or realize the use cases.
- Use cases often influence the organization of user manuals.
- Functional or system testing corresponds to the scenarios of use cases.
- UI “wizards” or shortcuts may be created for the most common scenarios of important use cases to ease common tasks.



Priorizar

Explicitar incrementos

Característica definidora: segmentar o âmbito do projeto em incrementos

"Refinar"

Os requisitos/casos de utilização podem ser detalhados à medida do necessário.

Use cases e os métodos ágeis → Use Cases 2.0

A granularidade dos casos de uso é algo “pesada” para a gestão do dia-a-dia

Proposta Use Cases 2.0

- “Fatias” de funcionalidade
- Ponto de partida: use cases
 - ...com a flexibilidade das *user stories/use case slices*



<https://youtu.be/p5gDbf0je8k>

Unidade mais conveniente: “fatia” de funcionalidade



7. browse
and shop

priority: MUST
release: 1
size: very large
complexity: high

a use case and its properties
captured on a sticky note

7.1 select and buy
1 product

flows: BF
test: 1 product,
default payment,
valid details

5

7.2 select and buy
100 products

flows: BF
test: 100 products,
default payment,
valid details

5

7.3 support systems
unavailable

flows: BF, A9, A10,
A1, A12
test: select product,
provide information,
disconnect each
system in between 13

some slices from the
use case captured on
their own sticky notes

Gestão de equipas com o SCRUM

Ilídio Oliveira

V2022-05-17 – TP9b

Objetivos de aprendizagem

- Identificar vantagens de estruturar um projeto em iterações, produzindo incrementos frequentes.
- Caracterizar os princípios da gestão do *backlog* em abordagens ágeis.
- Identificar os papéis numa equipa de Scrum e as principais "cerimónias"
- Relacionar as atividades da Scrum com as práticas das metodologias ágeis de desenvolvimento.

Como organizar as atividades da equipa?

A metáfora do Scrum...

"restarting play in rugby football that involves players packing closely together with their heads down and attempting to gain possession of the ball"

Guia SCRUM

Incluído no Moodle

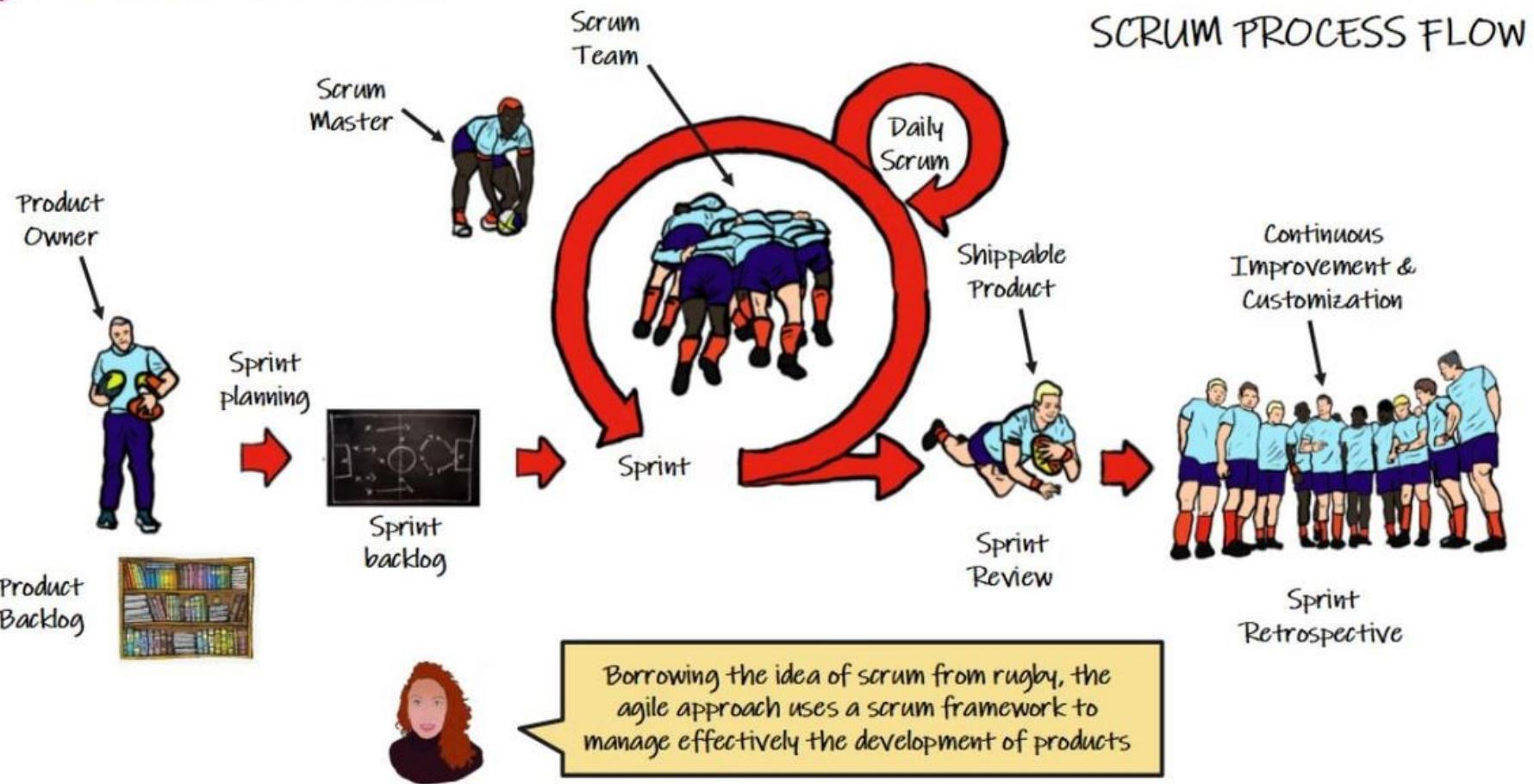
Scrum jobs?...

- [Indeed.pt](#)

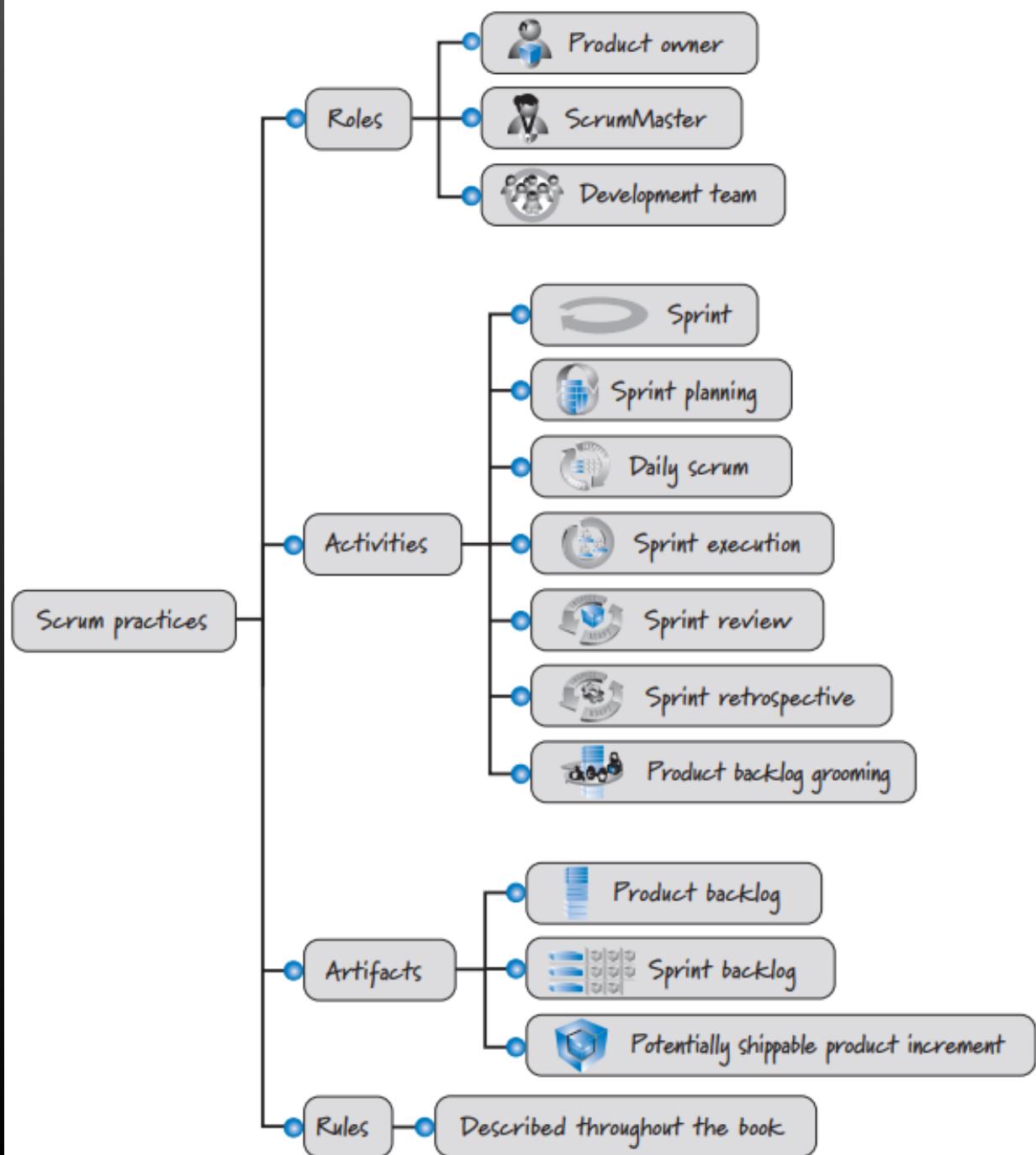


Metodologia de gestão de equipas SCRUM

SCRUM-INSTITUTE.ORG



Elementos do Scrum



"3355"

3
Roles



Product
Owner



Development
Team



Scrum Master

3
Artifacts



Product
Backlog



Sprint
Backlog



Product
Increment

5
Events



Sprint



Sprint Planning
Meeting



Daily Scrum
Meeting



Scrum Review
Meeting

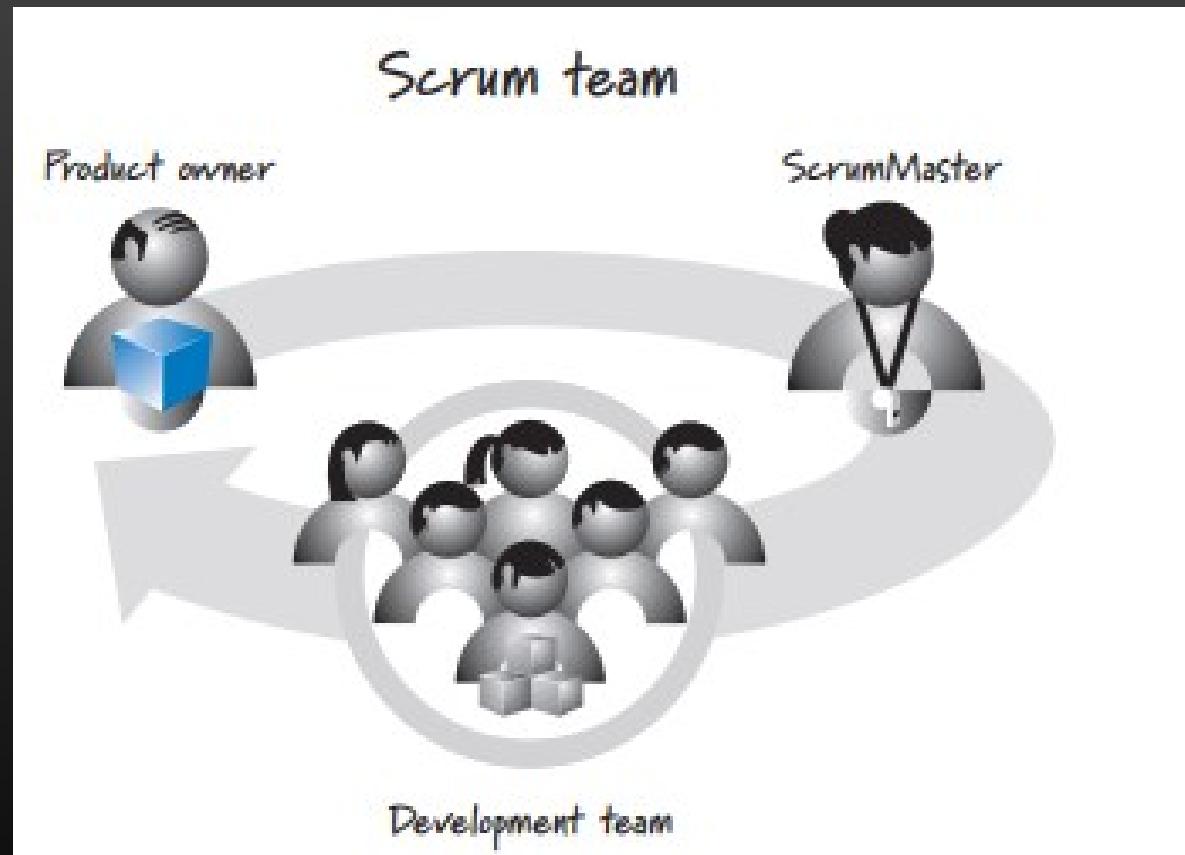


Scrum Retrospective
Meeting

5
Values



Papéis previstos no Scrum



Creating a Product: Scrum: Roles & Responsibilities



Product Owner

- Holds the vision for the product
- Determines what needs to be done
- Sets the priorities to deliver the highest value



Scrum Master

- Help the team best use Scrum to build the product
- Protecting the Scrum process
- Prevent distractions/impediments



Development Team

- Builds the product
- Self-organizing group
- Takes on and determines how to deliver chunks of work in frequent increments

Credit: Nokia Networks.

Scrum process

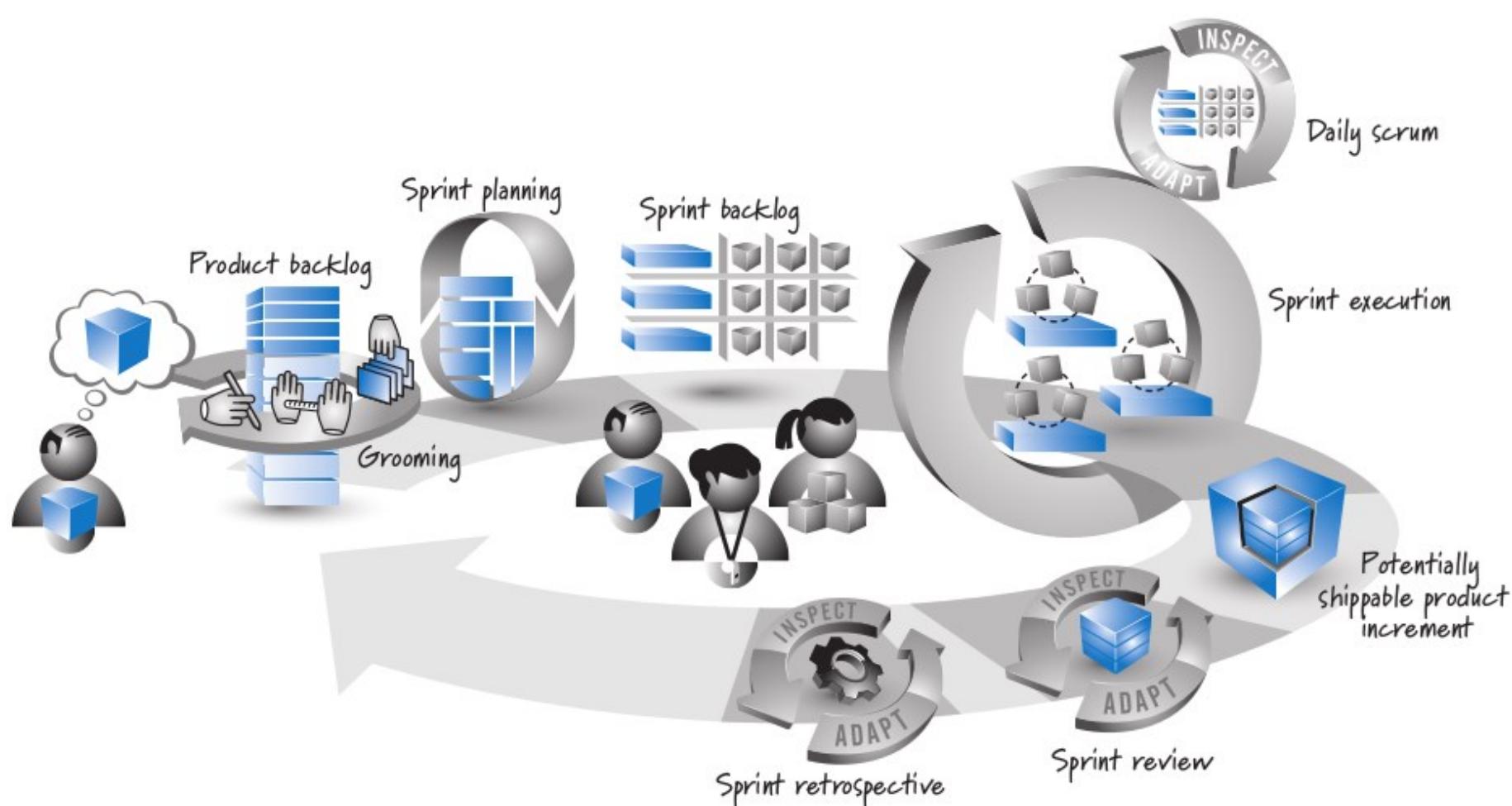
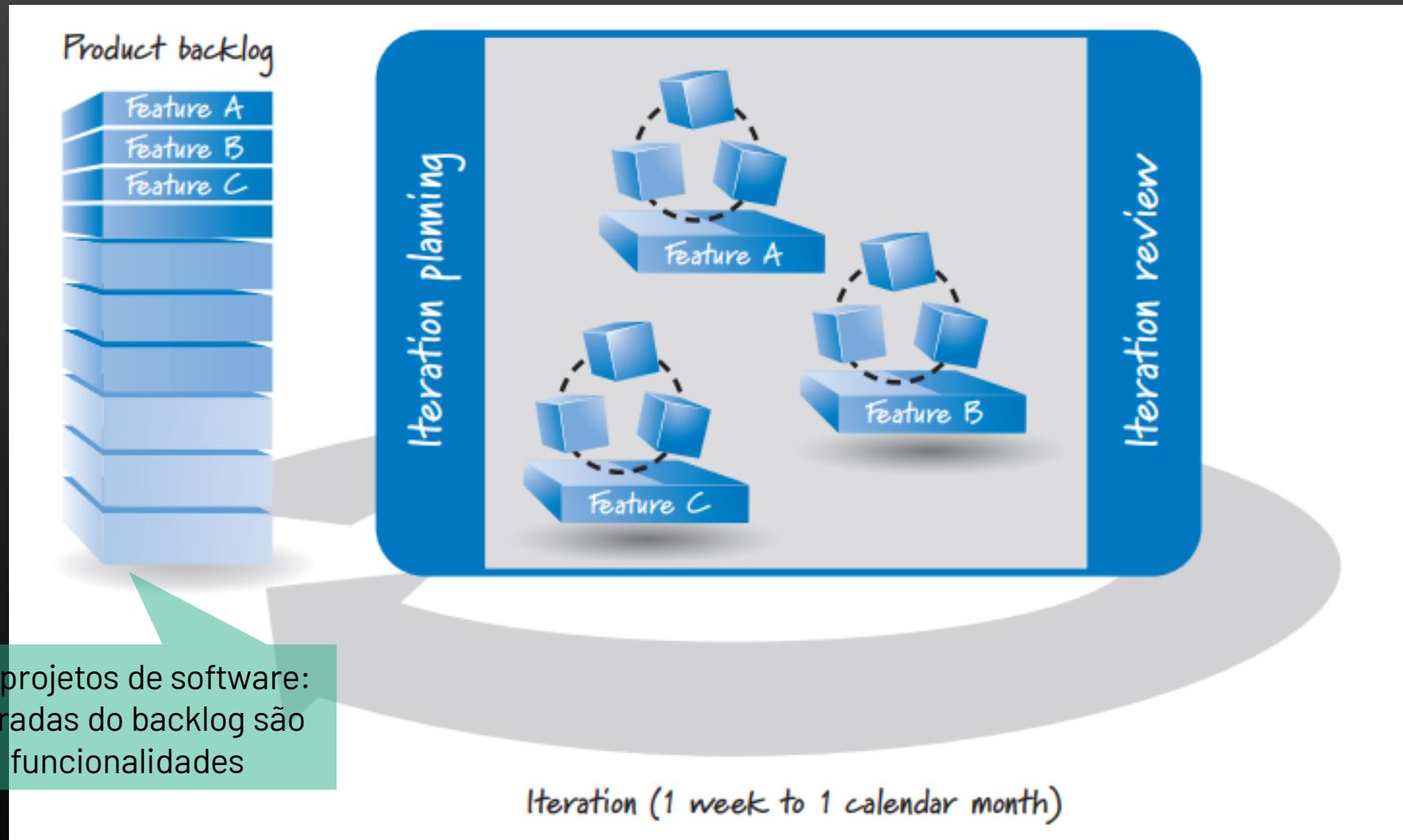
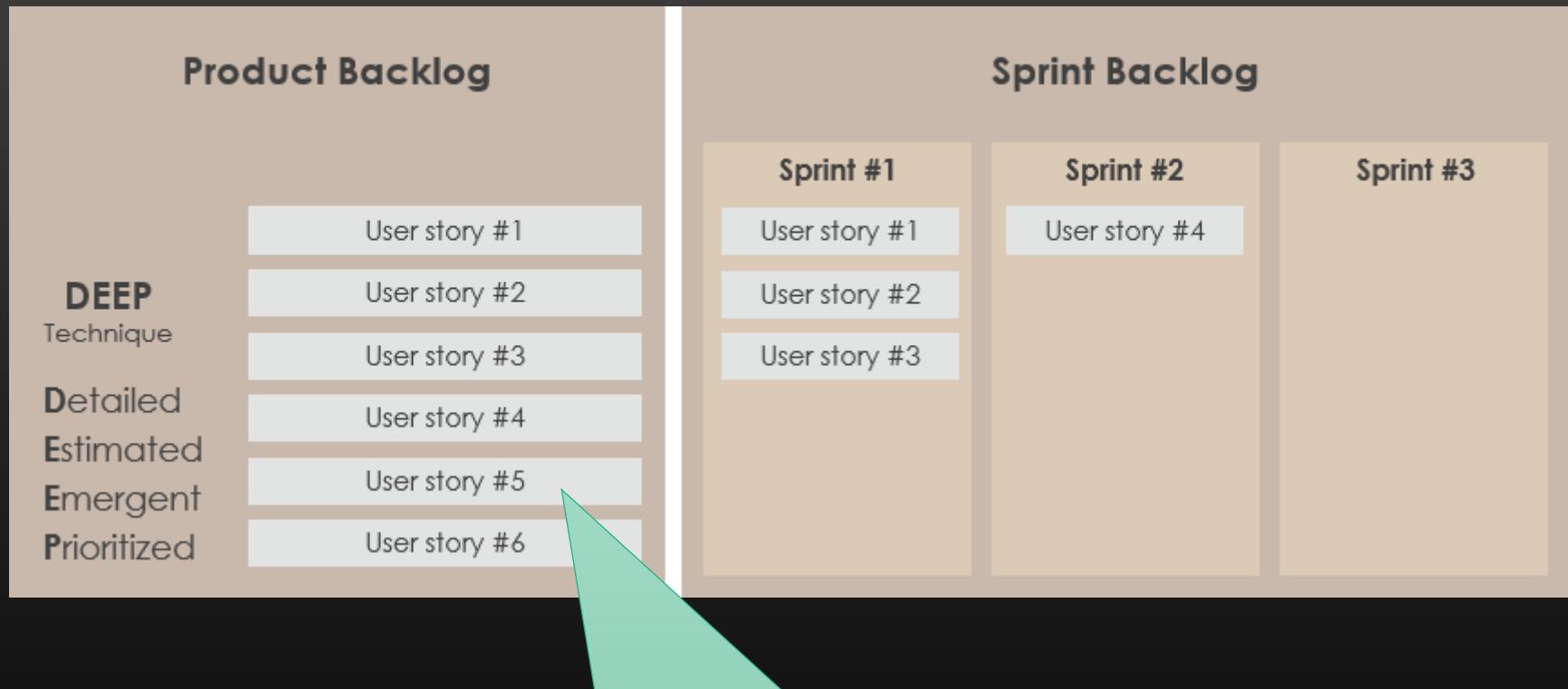


FIGURE 2.3 Scrum framework

Planeamento do trabalho e métodos ágeis

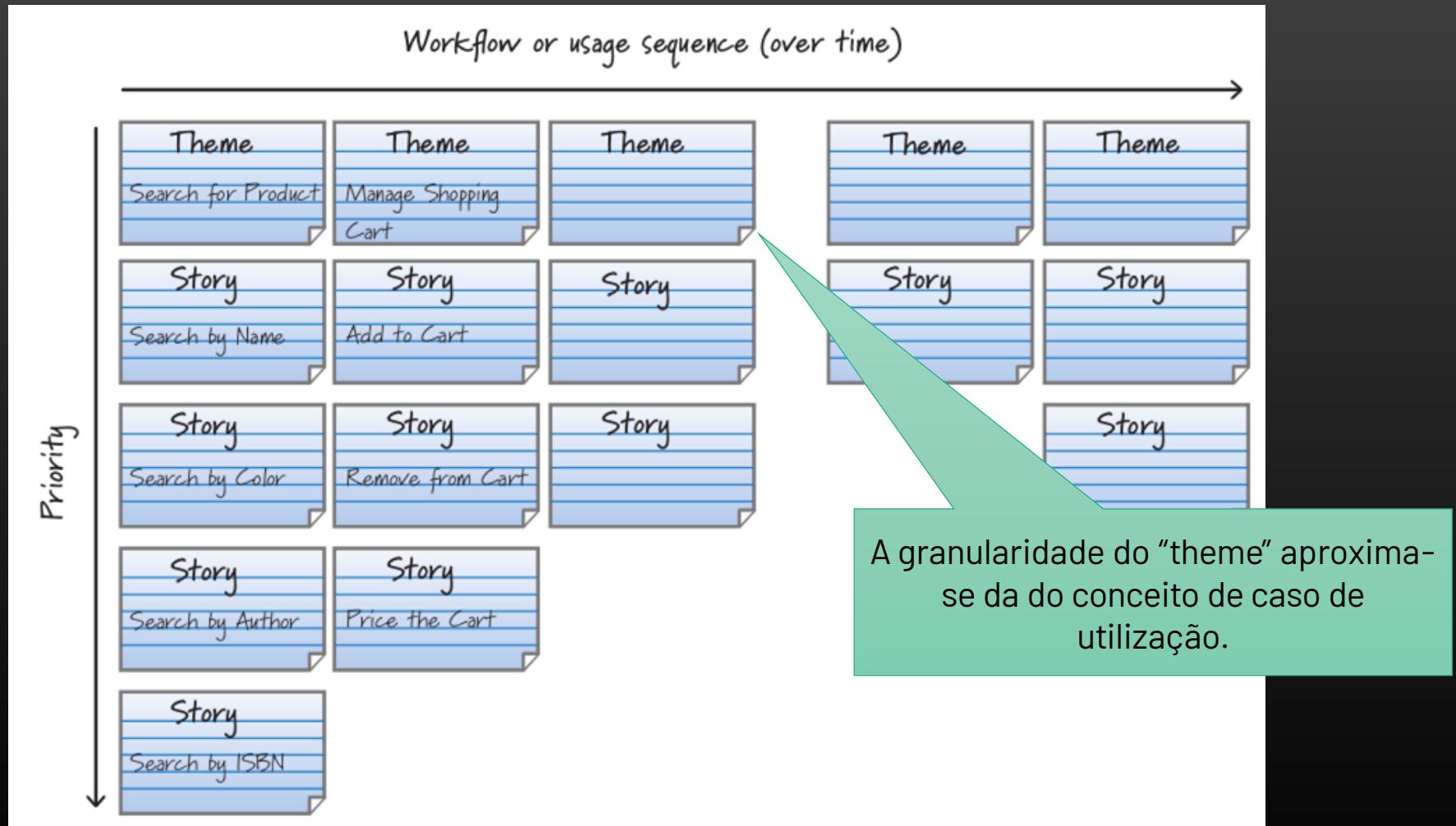


Sprint planning

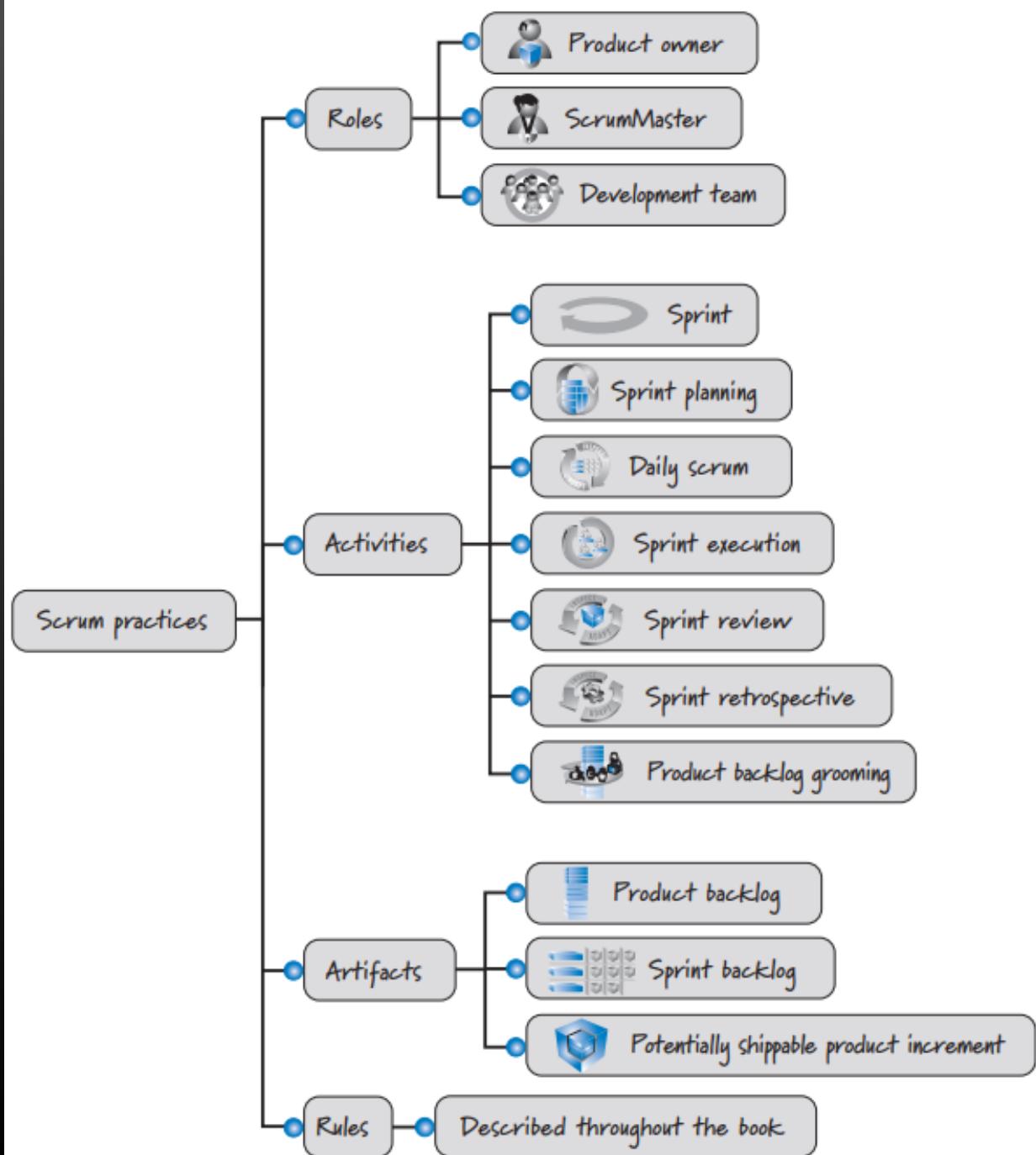


No desenvolvimento de sw, há um estilo para escrever as entradas do *backlog*, adotando o conceito de “user story”. A história é um exemplo de utilização, ~uma forma de percorrer um caso de utilização.

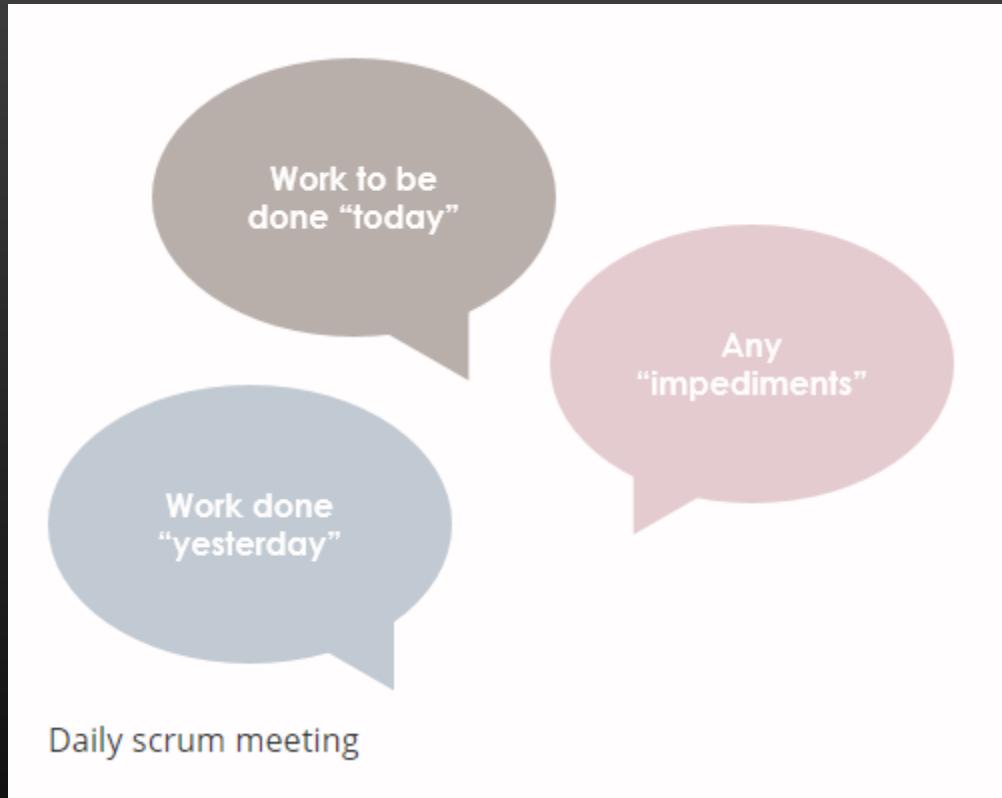
Histórias de utilização

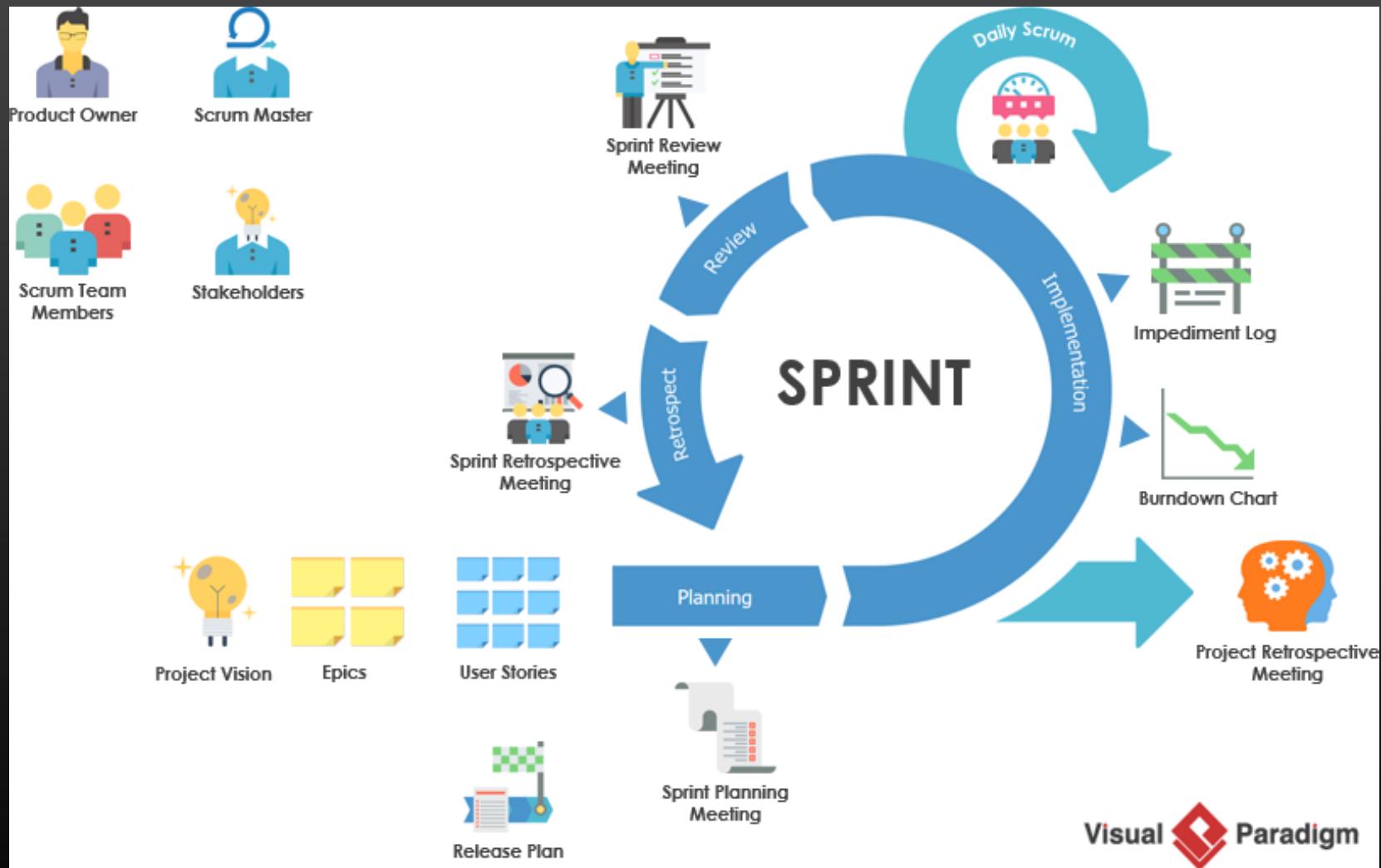


Elementos do Scrum



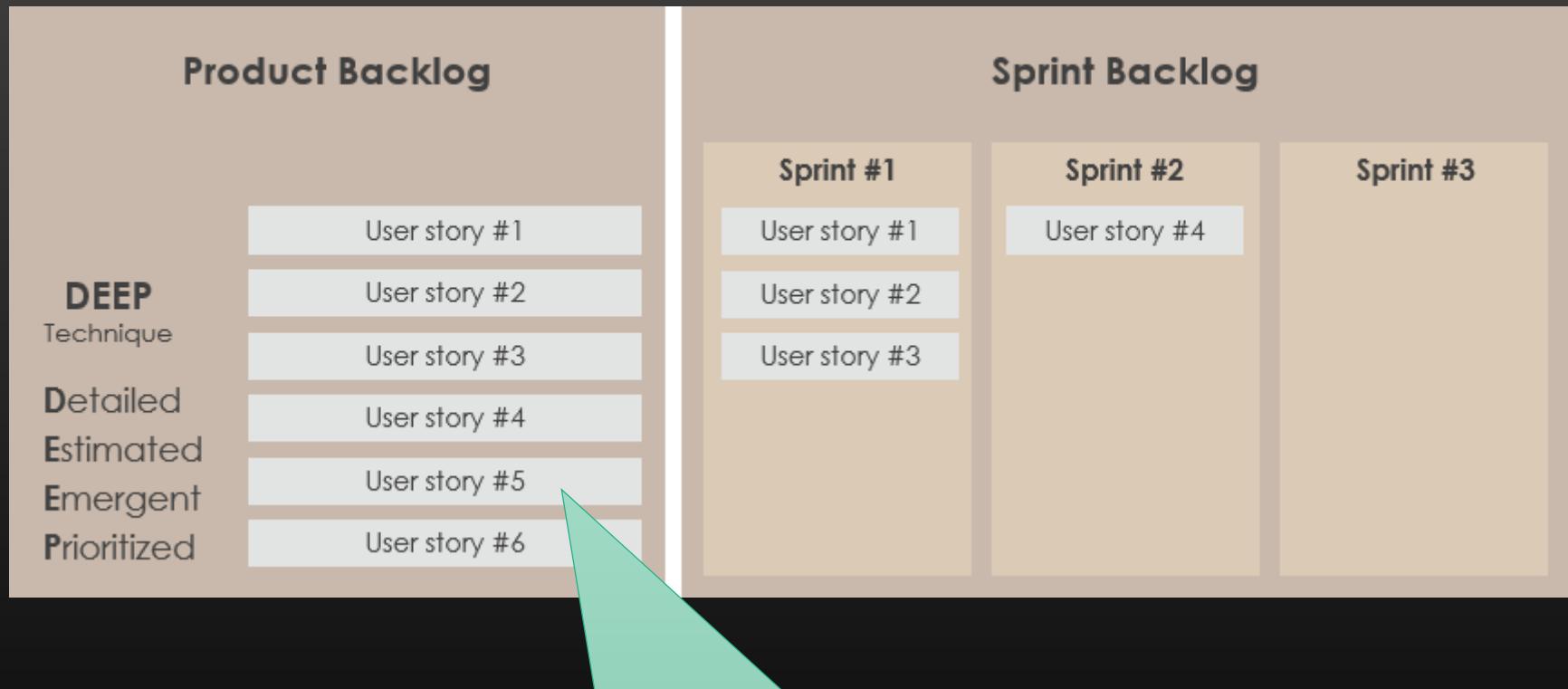
Daily Scrum





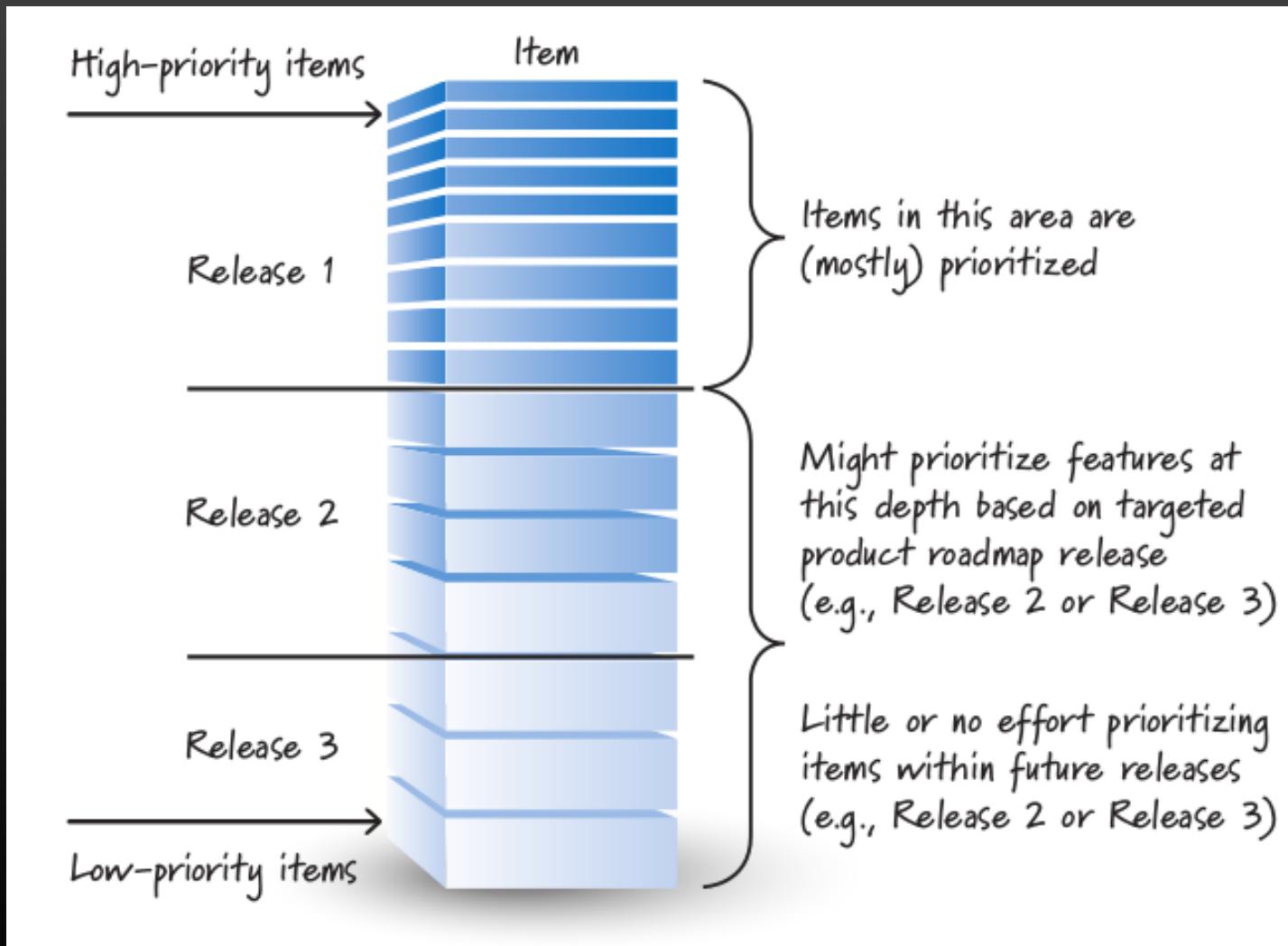
Planeamento e monitorização do progresso

Sprint planning



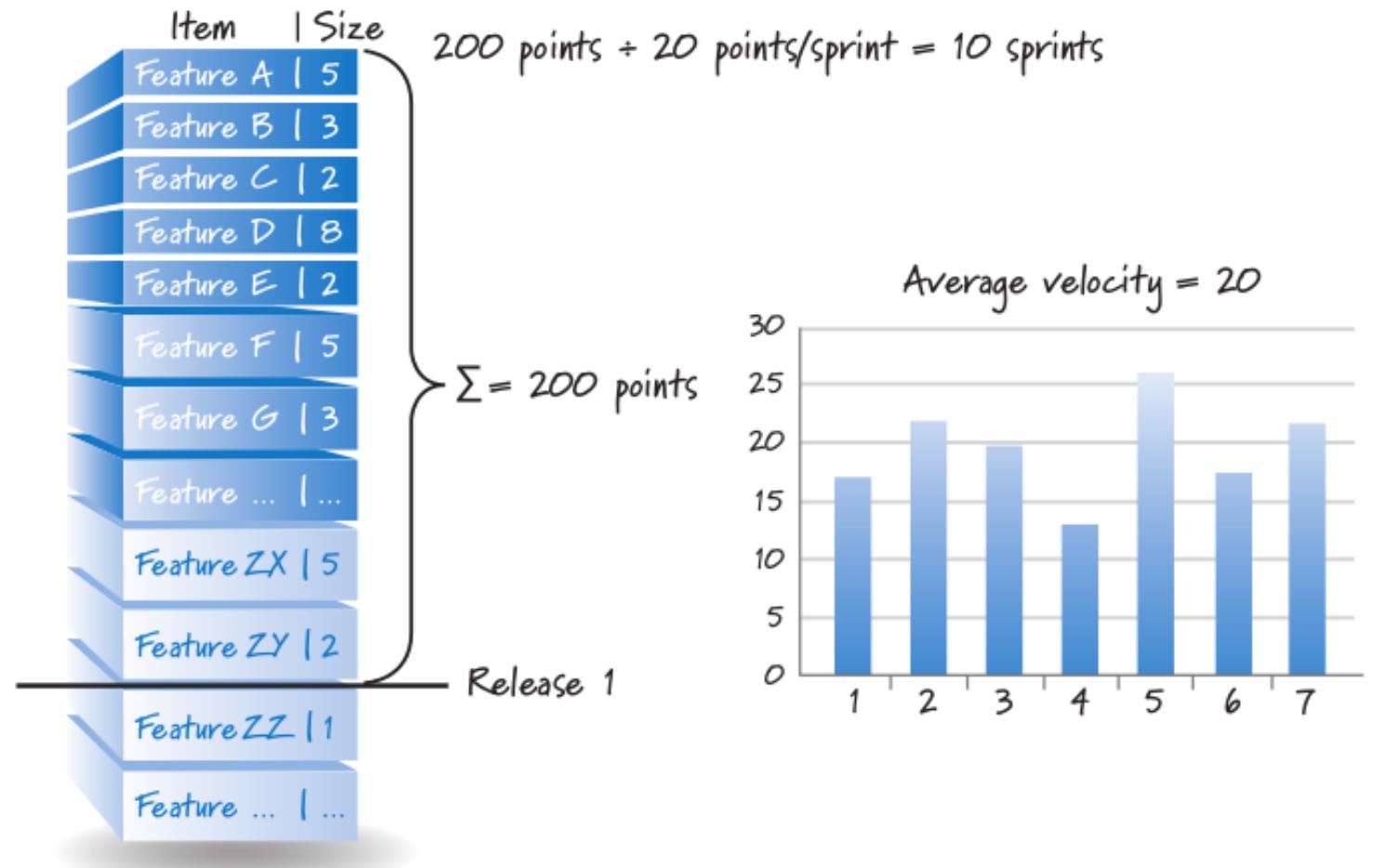
No desenvolvimento de sw, há um estilo para escrever as entradas do *backlog*, adotando o conceito de “user story”. A história é um exemplo de utilização, ~uma forma de percorrer um caso de utilização.

Scrum: backlog must be prioritized



Scrum: Velocity

Estimated size ÷ measured velocity = (number of sprints)



Pontuação

Funcionalidades (encomenda de comida online):

- F1: Inicio de sessão do utilizador (login)
- F2: Registo de novo utilizador na plataforma
- F3: Listar promoções em destaque do dia.
- F4: Colocar a encomenda (inclui pagamento)

Escala:

1pt: muito fácil. Direto de se implementar e âmbito reduzido.

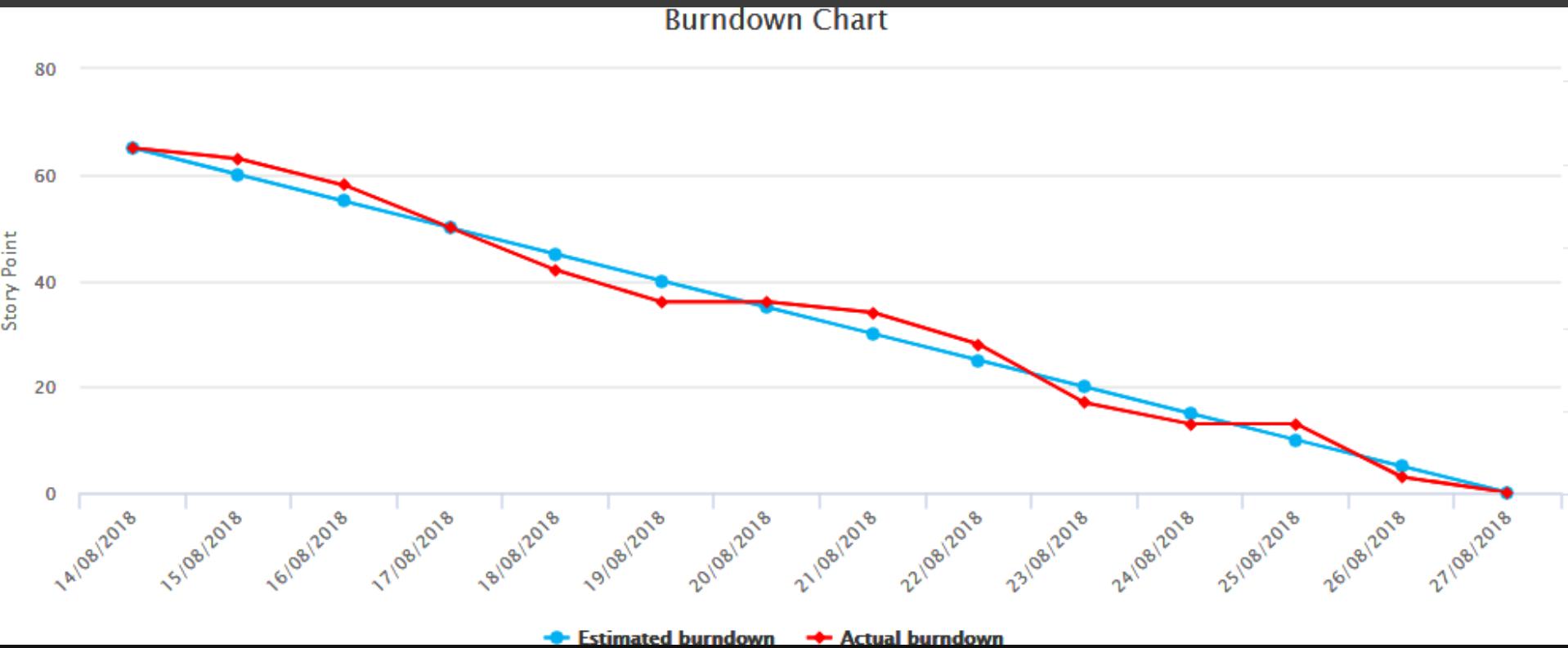
2pts: acessível; não oferece grande dificuldade.

4pts: complexo; tem várias interdependências (de outros módulos/serviços) ou um fluxo elaborado

8pts: muito complexo; requer integrações, tecnologias ou conhecimentos que não são completamente dominados

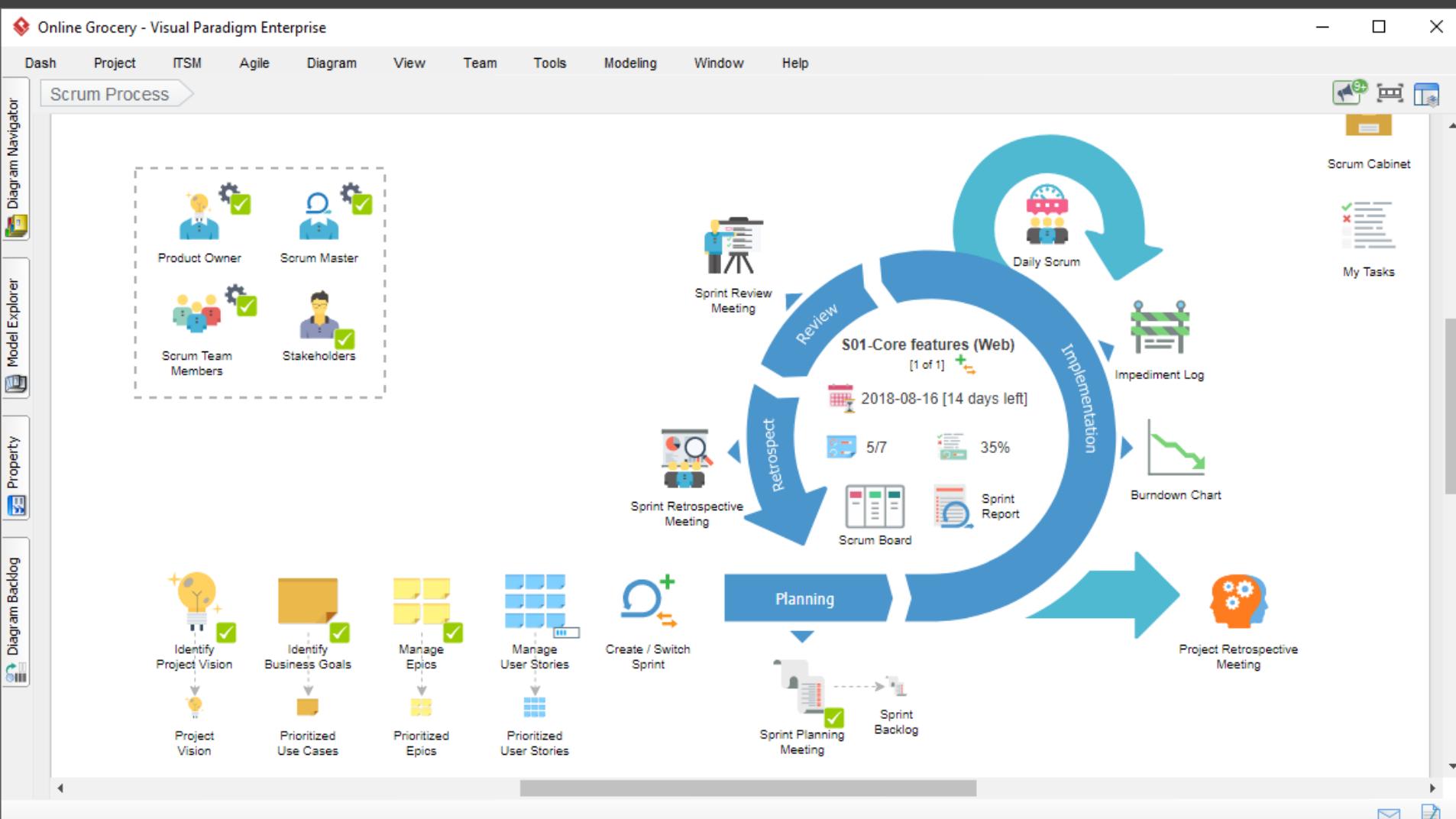
→ <http://bit.ly/2IUrNMn>

Monitorar o progresso com “burndown chart”



<https://www.visual-paradigm.com/cn/scrum/scrum-burndown-chart/>

[Optional] Scrum tutorial by VisualParadigm



<https://www.visual-paradigm.com/tutorials/agile-tutorial/>

I Oliveira

Algumas ideias a reter

- Um processo de software explica o trabalho a desenvolver para construir o produto
- O processo não explica como organizar o dia-a-dia da equipa
- A Scrum oferece uma metodologia “leve” para gestão de equipas, a construir produtos complexos
- Mas... é desafiante dominar e aplicar a Scrum!

A Scrum é especialmente adequada para métodos ágeis de desenvolvimento de software

- Sprint (iteração)
- Equipa auto-organizadas e multifuncionais (comunicação)
- Foco no incremento (entrega frequente)
- Adaptação (“*embrance change*”)

References

Core readings	Suggested readings
<ul style="list-style-type: none">Ken Schwaber, Jeff Sutherland, "Scrum Guide".	<ul style="list-style-type: none">Visual Paradigm, "What is Scrum?"[Dennis15] - Chap. 1

47006- ANÁLISE E MODELAÇÃO DE SISTEMAS

Metodologias ágeis e *user stories*

Ilídio Oliveira

V2022/05/17

Learning objectives for this lecture

Characterize the principles of backlog management in agile projects

Define and write stories for a given product.

Distinguish use story estimation and prioritization.

Write the acceptance criteria part of a user story.

Compare user stories and use cases with respect to commonalities and differences.

Describe the PivotalTracker story-based development workflow.

Requirements elicitation by exploring user-centered scenarios

- A) Use cases
- B) User stories
- C) User-centered design (UCD)
- D) Customer Journey Map (Experience maps)

Use cases way

ID and Name:	UC-4 Request a Chemical	
Created By:	Lori	Date Created: 8/22/13
Primary Actor:	Requester	Secondary Actors: Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.	
Trigger:	Requester indicates that he wants to request a chemical.	
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.	
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.	
Normal Flow:	<p>4.0 Request a Chemical from the Chemical Stockroom</p> <ol style="list-style-type: none"> Requester specifies the desired chemical. System lists containers of the desired chemical that are in the chemical stockroom, if any. System gives Requester the option to View Container History for any container. Requester selects a specific container or asks to place a vendor order (see 4.1). Requester enters other information to complete the request. System stores the request and notifies the Chemical Stockroom. 	
Alternative Flows:	<p>4.1 Request a Chemical from a Vendor</p> <ol style="list-style-type: none"> Requester searches vendor catalogs for the chemical (see 4.1.E1). System displays a list of vendors for the chemical with available container sizes, grades, and prices. Requester selects a vendor, container size, grade, and number of containers. Requester enters other information to complete the request. 	

is the request and notifies the Buyer.

4.1.E1 Is Not Commercially Available

ays message: No vendors for that chemical.

Requester if he wants to request another chemical (3a) or to exit (4a).

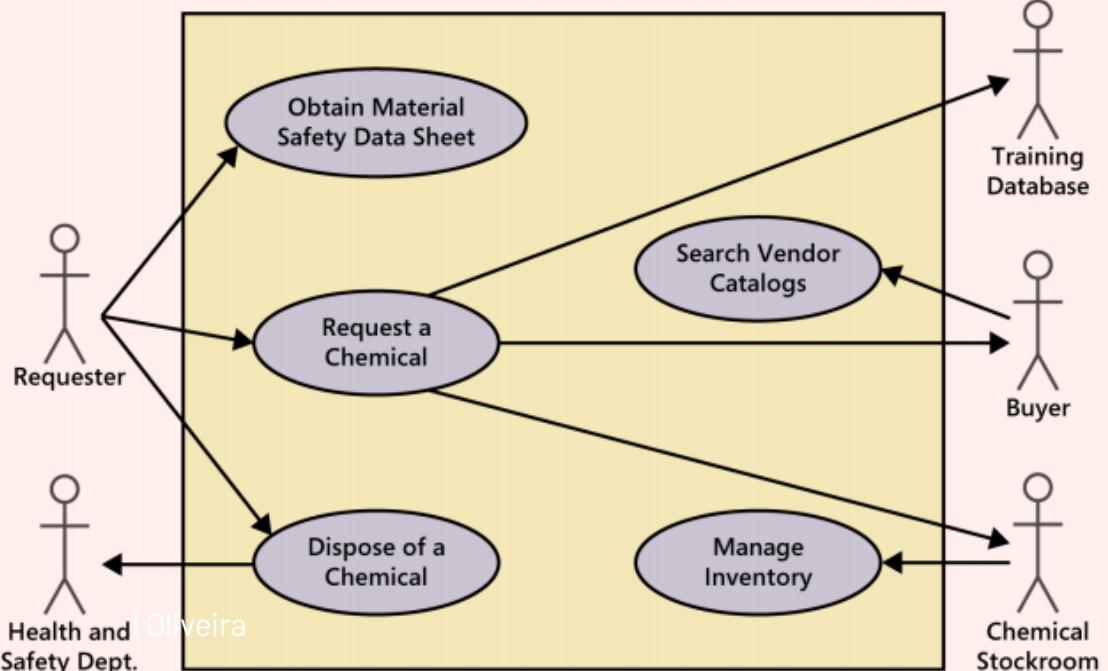
asks to request another chemical.

rts normal flow over.

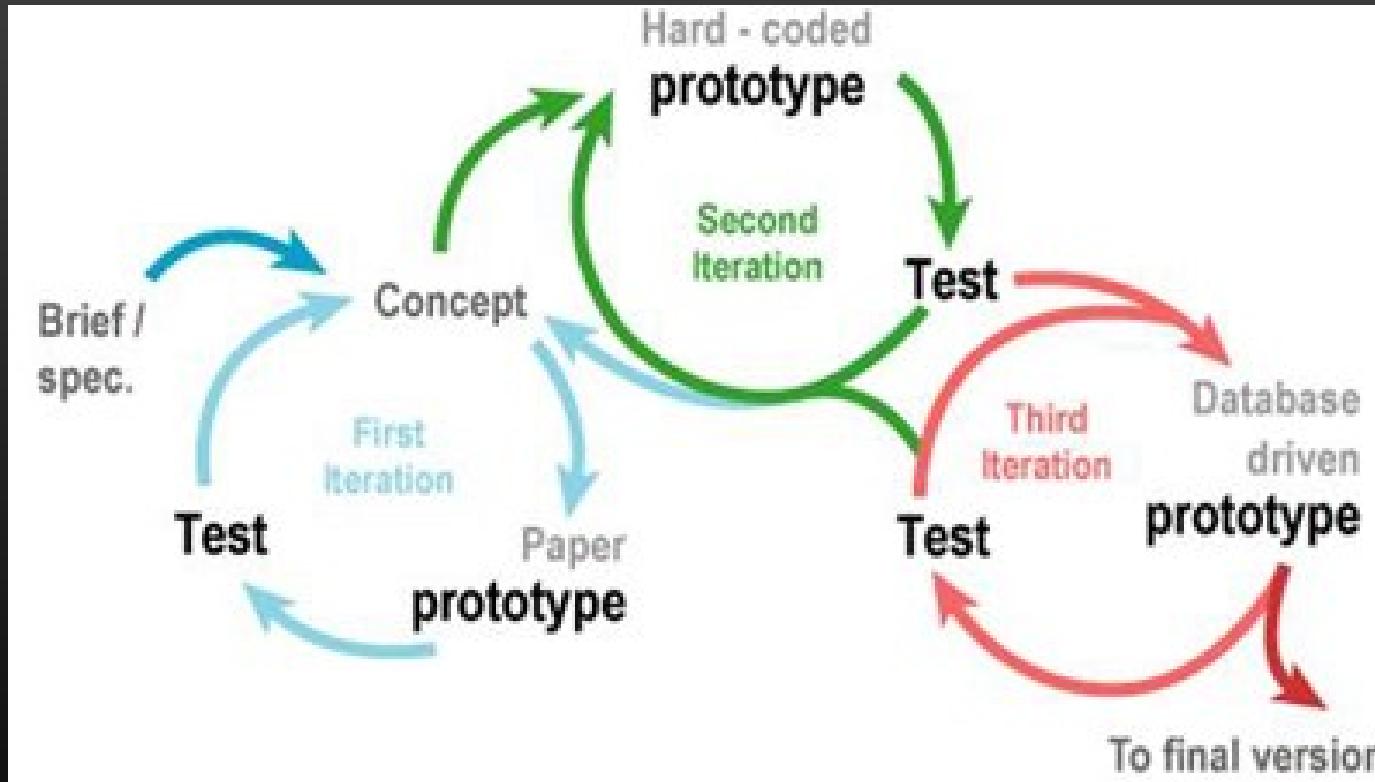
asks to exit.

minates use case.

5 times per week by each chemist, 200 times per week by chemical
f



UCD: prototyping & acceptance



<https://www.museumsandtheweb.com/mw2007/papers/brown/brown.html>

Rail Europe Experience Map

Guiding Principles

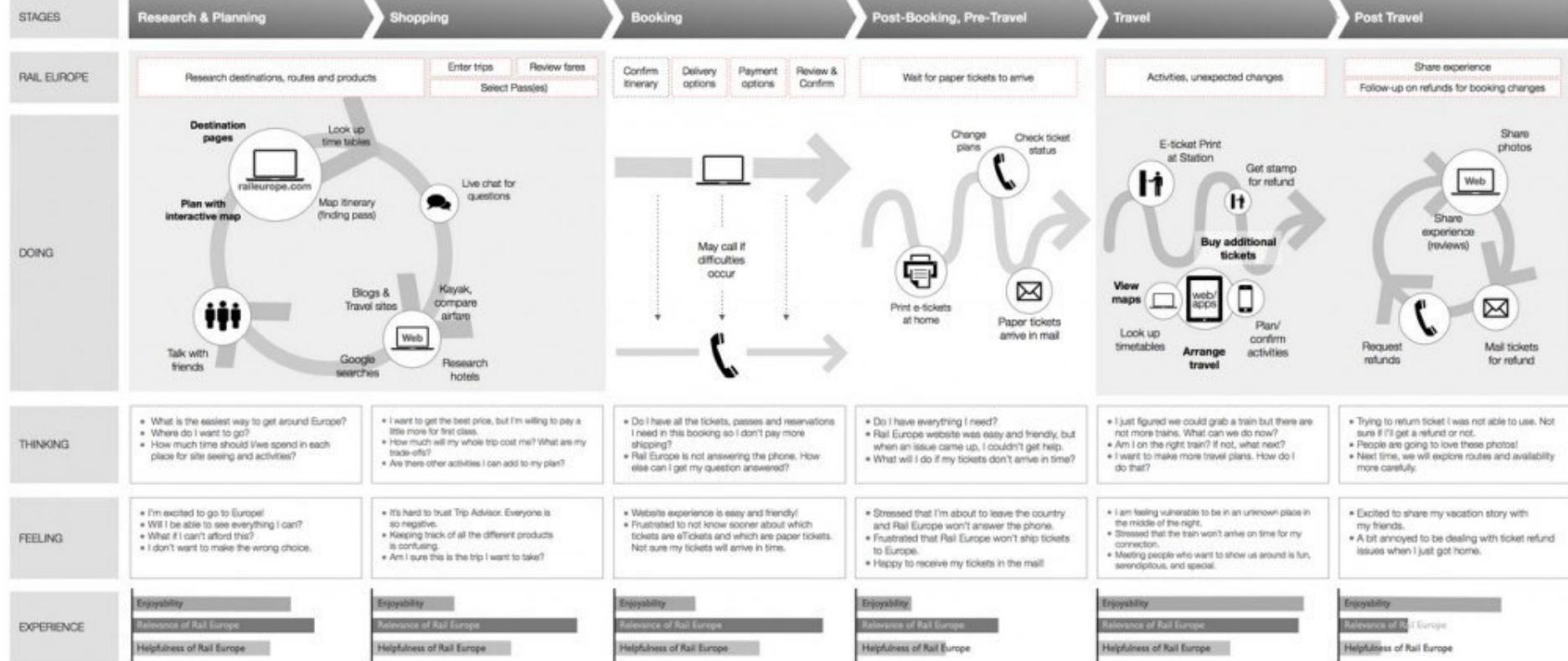
People choose rail travel because it is convenient, easy, and flexible.

Rail booking is only one part of people's larger travel process.

People build their travel plans over time.

People value service that is respectful, effective and personable.

Customer Journey



Opportunities

GLOBAL			PLANNING, SHOPPING, BOOKING			POST-BOOK, TRAVEL, POST-TRAVEL		
Communicate a clear value proposition. STAGE: Initial visit	Help people get the help they need. STAGE: Global	Support people in creating their own solutions. STAGE: Global	Enable people to plan over time. STAGES: Planning, Shopping	Visualize the trip for planning and booking. STAGES: Planning, Shopping	Arm customers with information for making decisions. STAGES: Shopping, Booking	Improve the paper ticket experience. STAGES: Post-Booking, Travel, Post-Travel	Accommodate planning and booking in Europe too. STAGE: Traveling	
Make your customers into better, more savvy travelers. STAGES: Global	Engage in social media with explicit purposes. STAGES: Global		Connect planning, shopping and booking on the web. STAGES: Planning, Shopping, Booking	Aggregate shipping with a reasonable timeline. STAGE: Booking		Proactively help people deal with change. STAGES: Post-Booking, Traveling	Communicate status clearly at all times. STAGES: Post-Booking, Post Travel	

Information sources
Stakeholder interviews
Cognitive walkthroughs

Customer Experience Survey
Existing Rail Europe Documentation



User stories

CIS board

Story Map by Easy Agile

+ Create Epic

Quick filters

Sprint swimlanes

...

?

Backlog

Navigation	CIS-1

Car Statistics	CIS-4

Phone Integration	CIS-3

Play Media	CIS-2

Fatigue Management	CIS-6

Sprint 1

The 'Young Professional' Driver / Install maps so that I can navigate to places easier	CIS-8
 2	

The 'Young Professional' Driver / Touch Screen to navigate easily	CIS-38
 1	

The 'Young Professional' Driver / Apple CarPlay Integration so that I can safely send and receive calls, texts and emails from my iOS device while driving	CIS-41
 1	

The 'Young Adult' Passenger / Allow Wifi Hotspot to support up to 5 devices	CIS-39
 1	

The 'Sunday' Driver / Enable 'Tourist Mode Assist' when travelling outside of standard travel radius	CIS-12
 2	

Sprint 2

The 'Sunday' Driver / Showcase local landmarks if travelling outside of standard travel radius	CIS-11
 3	

The 'Young Professional' Driver / Wear and Tear Report so that I can take preventative action to preserve the life of the car if needed	CIS-26
 5	

The 'Family' Driver / Microphone so that I can make phone calls safely while I'm driving	CIS-19
 4	

The 'Family' Driver / Graphical User Interface for easier use of media while driving	CIS-18
 3	

The 'Young Professional' Driver / Android Auto Integration so that I can safely send and receive calls, texts and emails while driving	CIS-42
 3	

Use cases e os métodos ágeis → Use Cases 2.0

A granularidade dos casos de uso é algo “pesada” para a gestão do dia-a-dia

Proposta Use Cases 2.0

- “Fatias” de funcionalidade
- Ponto de partida: use cases
 - ...com a flexibilidade das *user stories/use case slices*



<https://youtu.be/p5gDbf0je8k>

Unidade mais conveniente: “fatia” de funcionalidade



7. browse
and shop

priority: MUST
release: 1
size: very large
complexity: high

a use case and its properties
captured on a sticky note

7.1 select and buy
1 product

flows: BF
test: 1 product,
default payment,
valid details

5

7.2 select and buy
100 products

flows: BF
test: 100 products,
default payment,
valid details

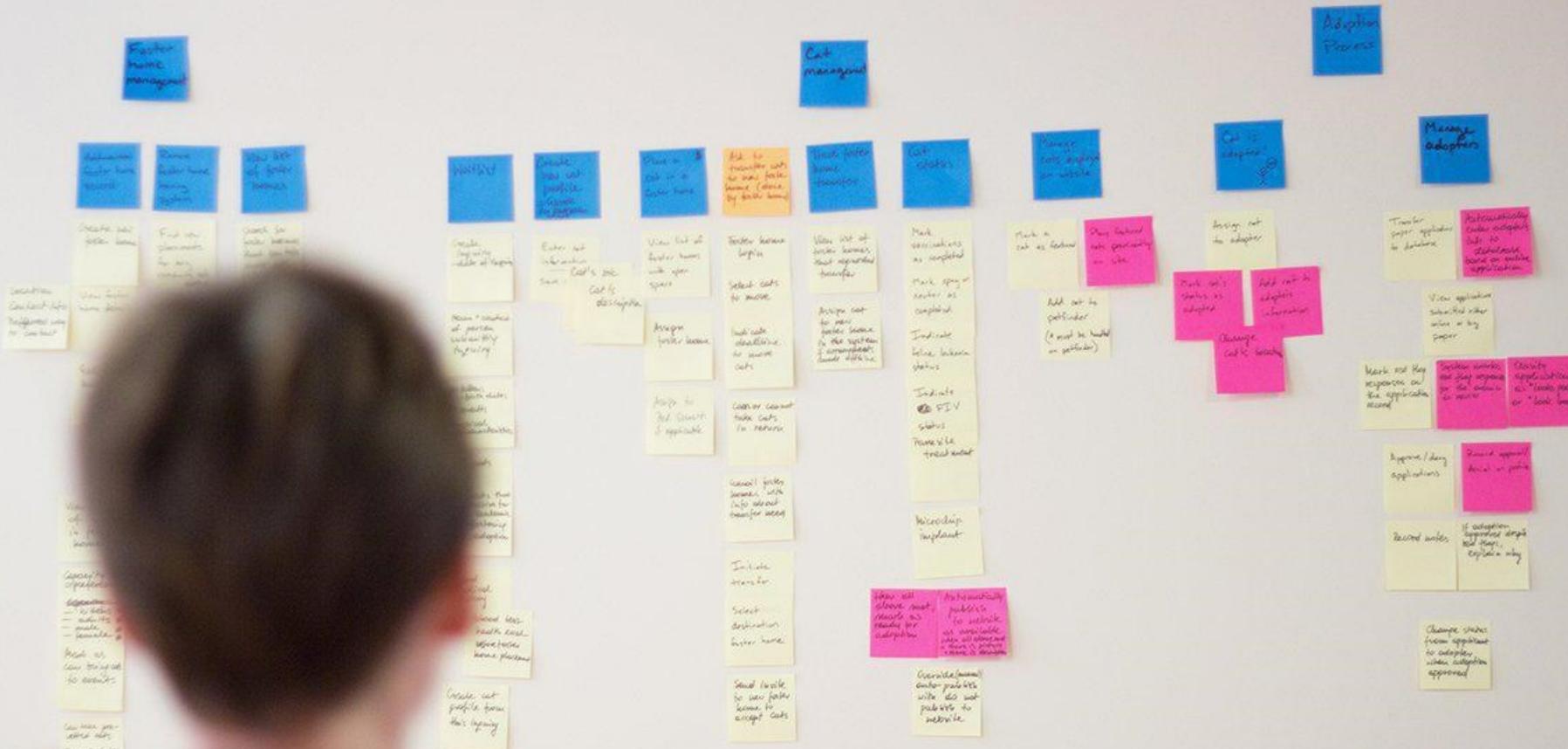
5

7.3 support systems
unavailable

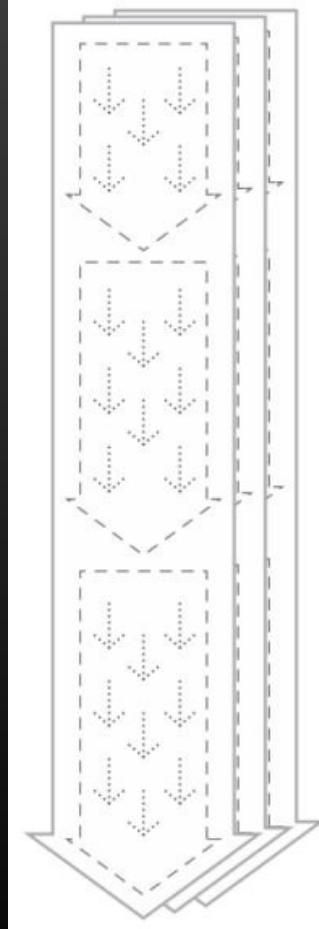
flows: BF, A9, A10,
A1, A12
test: select product,
provide information,
disconnect each
system in between 13

some slices from the
use case captured on
their own sticky notes

Sticky note metaphor



The story should clarify how to check if it is working



"As a [persona],
I want to [do something]
so that I can [realize a reward]"

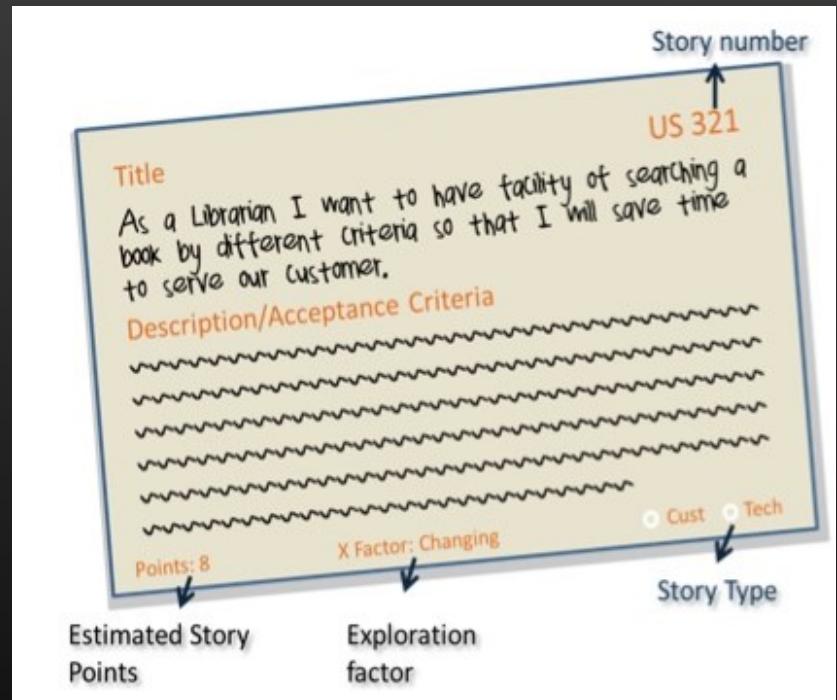
Who is this user?
What makes them tick?
Who's an example of such a person?

Why do they want to do this?
What's the benefit/reward?
How will we know of it's working?

User stories in agile methods

The *backlog* is the prioritized list of user stories –requirements– for the product and their allocation to upcoming iterations (called sprints in the agile development method called Scrum.)

User story: a “short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system” (Cohn 2010)



User story != use case

→ See [examples](#)

user stories focus on functionality

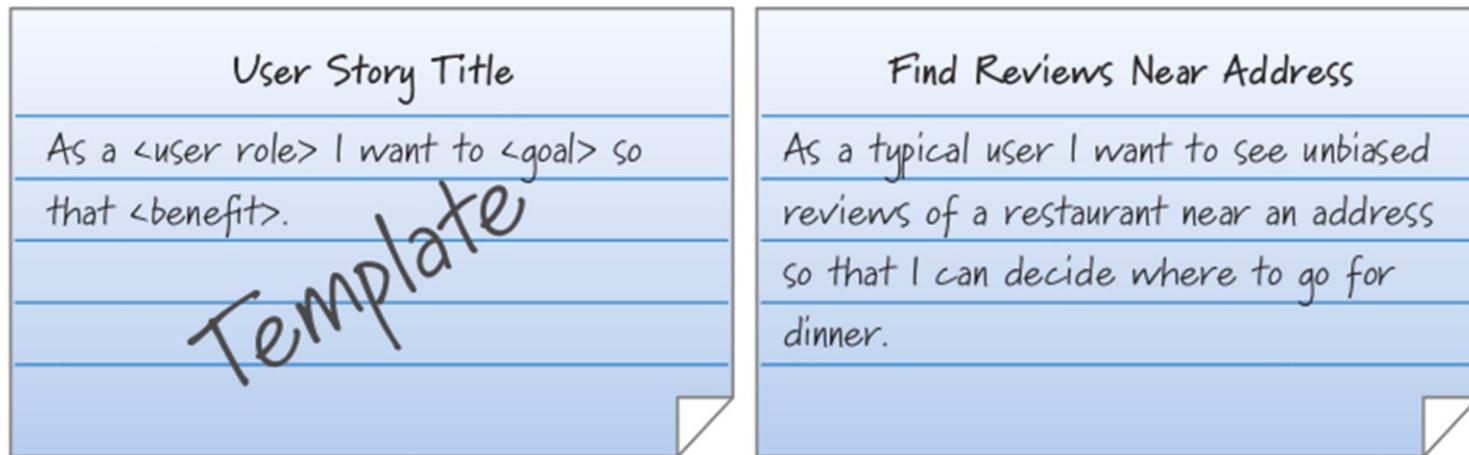
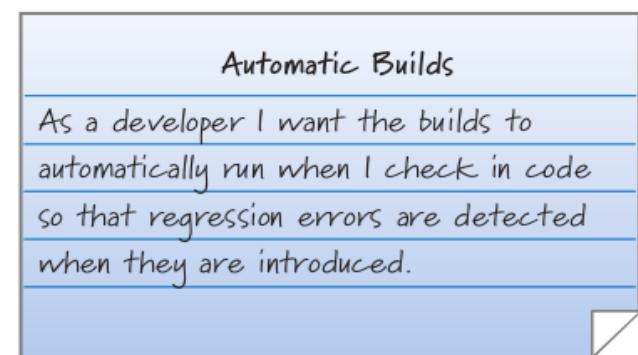


FIGURE 5.2 A user story template and card



Undesirable technical story

The user story

A “short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system” (Cohn 2010).

User stories often are written according to the following structure (other styles also are used):

As a <type of user>, I want <some goal> so that <some reason>.

→ Advantages of the “As a user, I want” user story template.

As a customer, I want to add an item into shopping cart.

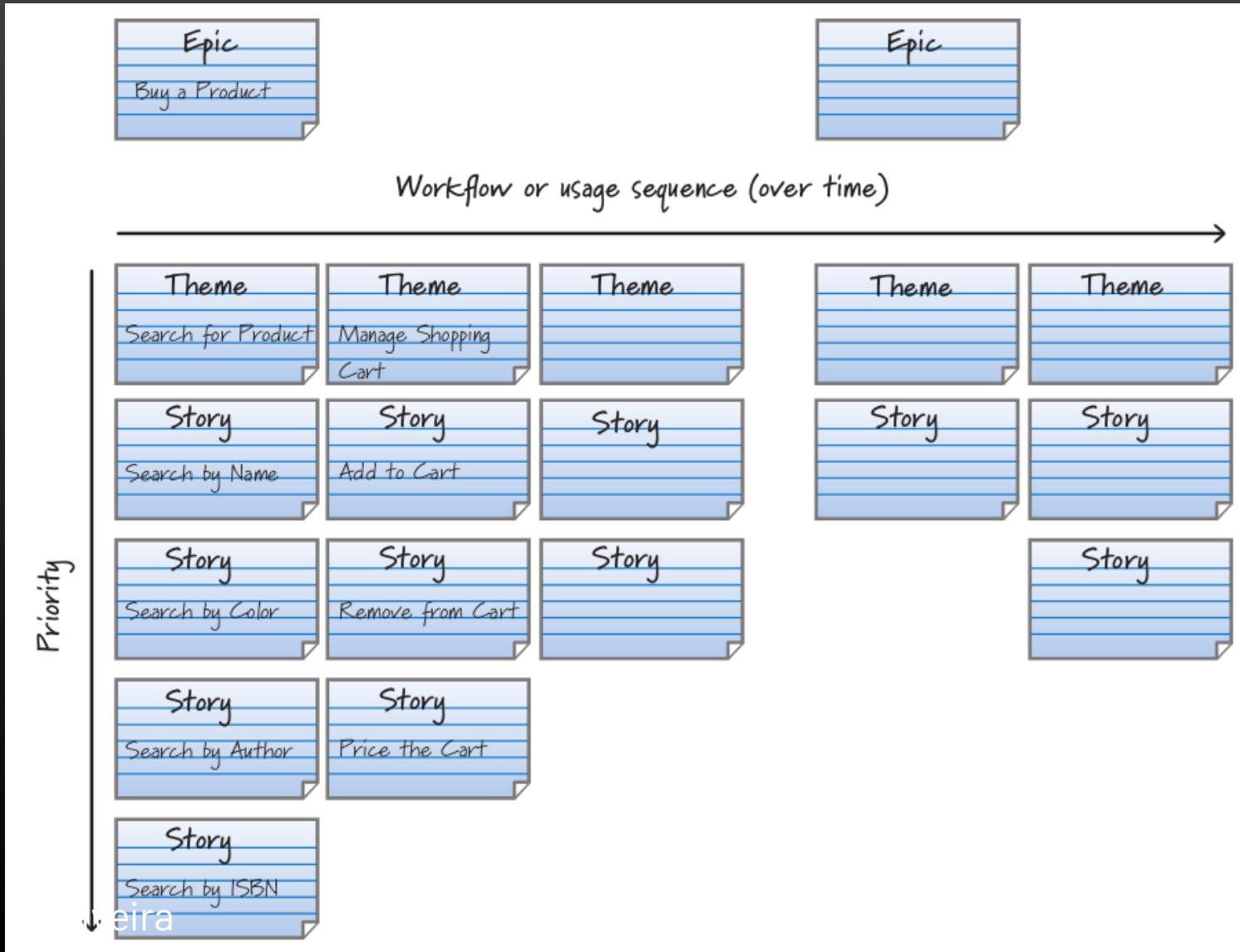
E.g. As a customer, I want to receive an SMS when the item is arrived so that I can go pick it up.

<role> represents the person, system, subsystem or any entity else who will interact with the system to be implemented to achieve a goal. He or she will gain values by interacting with the system.

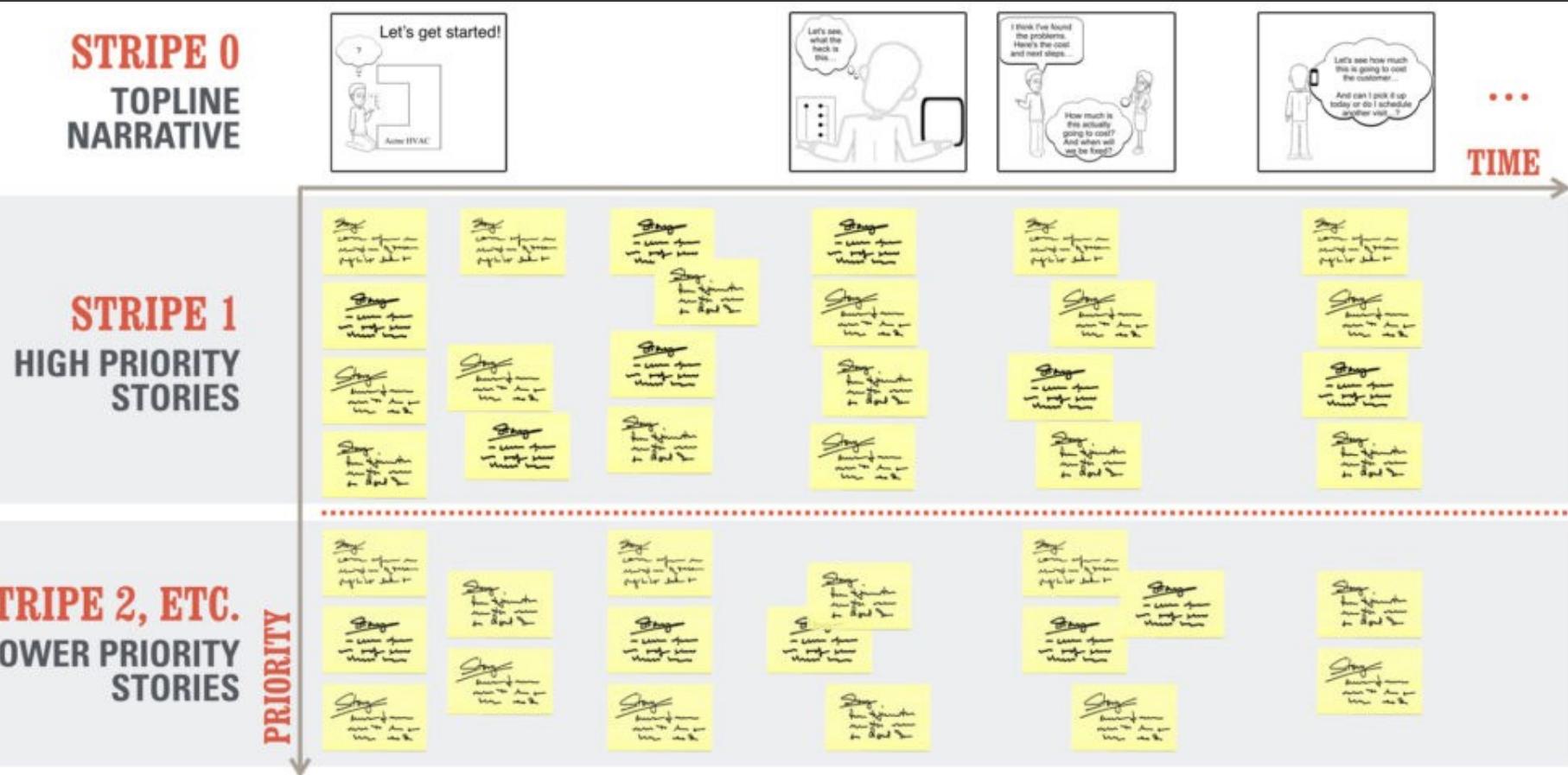
<business objective> represents a user's expectation that can be accomplished through interacting with the system.

<business value> represents the value behind the interaction with the system. May be omitted, if obvious from the business objective.

Finding good stories



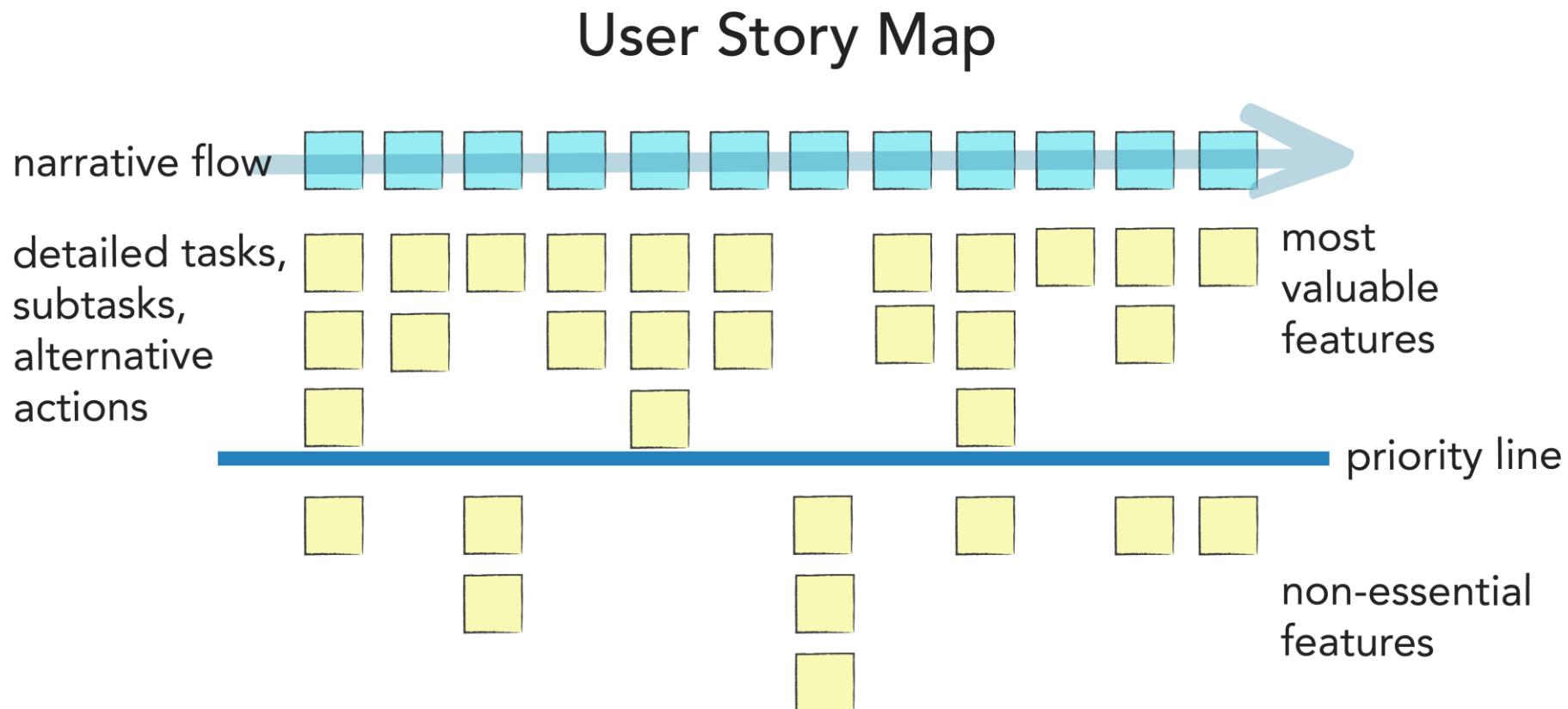
Organizing the stories in priority stripes



source: adapted from Jeff Patton's 'User Story Mapping'

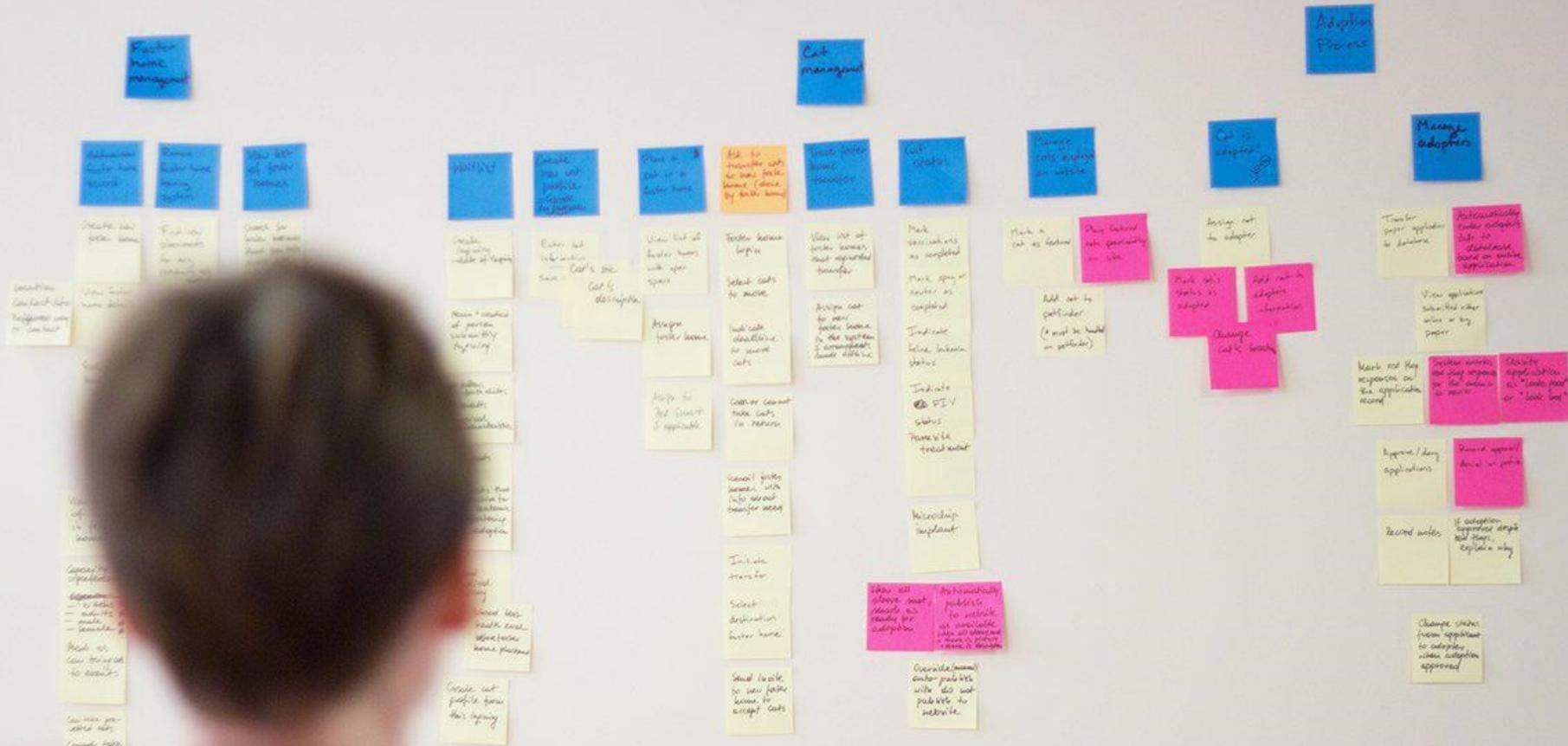
© 2015 COWAN+

The user story map



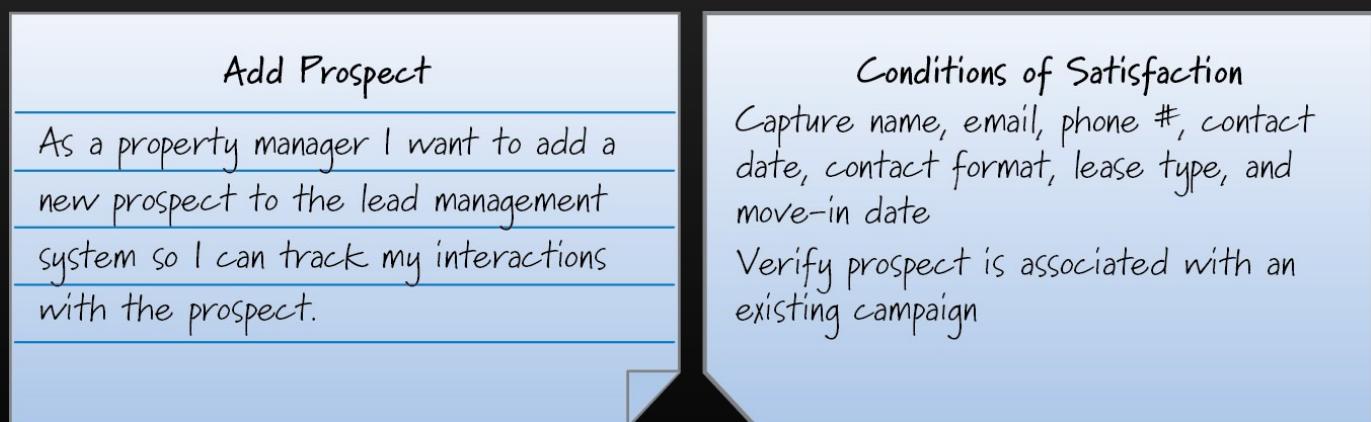
<https://www.caktusgroup.com/blog/2017/07/31/user-story-mapping-high-level-release-plan/>

Physical boards or in software tools...



Acceptance criteria

Rather than specifying functional requirements, agile teams typically elaborate a refined user story into a set of acceptance tests that collectively describe the story's "conditions of satisfaction."



→ See examples

A structured language for writing the acceptance criteria

GIVEN [necessary context] WHEN [action] THEN [reaction].

→ Gerkin DSL.

→ Used to create AC that are meant to be automatize (feed the tests)

```
Title (one line describing the story)

Narrative:
As a [role]
I want [feature]
So that [benefit]

Acceptance Criteria: (presented as Scenarios)

Scenario 1: Title
Given [context]
  And [some more context]...
When [event]
Then [outcome]
  And [another outcome]...

Scenario 2: ...
```

User story as a collaboration context

Frank Can Add Another Person as a Friend

ID #115218319 Close

STORY TYPE	★ Feature
POINTS	0 Unestimated
STATE	Start Unscheduled
REQUESTER	RJ Ryan Jones
OWNERS	<none> +
FOLLOW THIS STORY	(1 follower) <input checked="" type="checkbox"/>

Updated: less than a minute ago

DESCRIPTION (edit)

As Frank I want to add a friend I searched for to my friend network so that I can see their posts, they can see my posts and I can direct message them

GIVEN I have searched for a friend's name
WHEN I select "Add Friend" next to my friend's name
THEN my friend's name should appear in my friend list on my homepage

Dev Notes: The added friend needs to be added to the Frank's friends in database
Design Notes: Attached are mocks for the button and placement

LABELS

add friend |x| individual user |x|

scenario: Wilson posts to his own blog
Given I am logged in as Wilson
When I try to post to "Expensive Therapy"
Then I should see "Your article was published."

scenario: Wilson fails to post to somebody else's blog
Given I am logged in as Wilson
When I try to post to "Greg's anti-tax rants"
Then I should see "Hey! That's not your blog!"

scenario: Greg posts to a client's blog
Given I am logged in as Greg
When I try to post to "Expensive Therapy"
Then I should see "Your article was published."

Stories define your project

Every project starts with a story, no matter what you're building. Tracker helps your team better develop and keep track of them while they progress from start to delivered.

Start with a good story

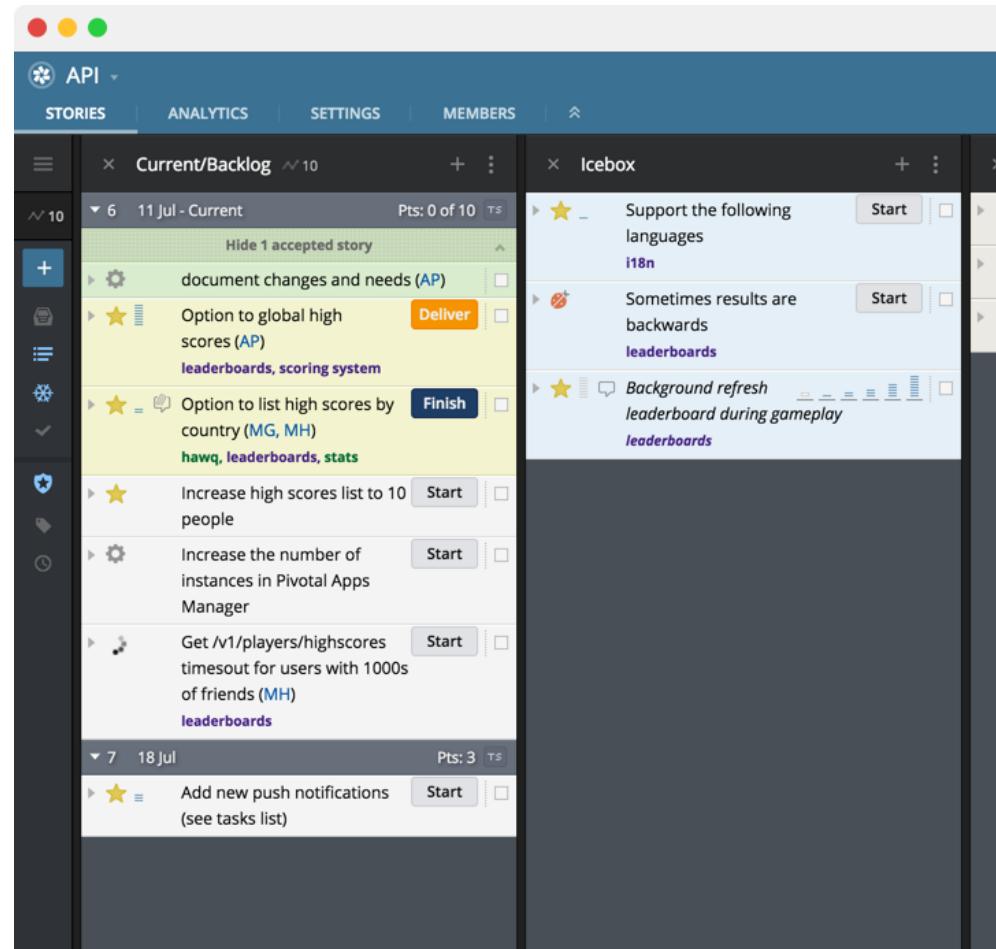
A story is a small, actionable bit of work that's either a placeholder for a future conversation or a reflection of one that already happened. Outlining what a user needs helps you focus on the what, not the how.

Define the story

Select among features, bugs, and chores to strike a healthy balance between building new features, staying ahead of technical debt, and keeping the bugs from piling up.

Estimate, then prioritize

Writing the story is just the beginning—now you get to rap about it. Estimate as a team to uncover the story's complexity. Choose among several point scales, then drag-and-drop to prioritize by iteration.



The screenshot shows the Pivotal Tracker web application. At the top, there's a navigation bar with tabs for 'STORIES', 'ANALYTICS', 'SETTINGS', and 'MEMBERS'. Below the navigation, there's a header for 'Current/Backlog' with a count of 10 stories. The main area displays a list of stories, each with a title, priority level (indicated by a star icon), and a brief description. Some stories have buttons like 'Deliver' or 'Finish'. To the right of the backlog, there's a sidebar titled 'Icebox' containing a few more stories. On the far left, there's a vertical sidebar with various icons and a search bar.

Story ID	Date	Priority	Description	Status
6	11 Jul - Current	High	document changes and needs (AP)	Pts: 0 of 10
6	11 Jul - Current	Medium	Option to global high scores (AP) leaderboards, scoring system	Pts: 0 of 10
6	11 Jul - Current	Low	Option to list high scores by country (MG, MH) hawq, leaderboards, stats	Pts: 0 of 10
7	18 Jul	Medium	Increase high scores list to 10 people	Pts: 3
7	18 Jul	Medium	Increase the number of instances in Pivotal Apps Manager	Pts: 3
7	18 Jul	Medium	Get /v1/players/highscores timeout for users with 1000s of friends (MH) leaderboards	Pts: 3
7	18 Jul	Low	Add new push notifications (see tasks list)	Pts: 3

Pivotal Tracker style

Tracker lists stories in a project's Current and Backlog [panels](#) in priority order.

Dragging a story to the top of your Backlog makes it the top priority

Tracker is a "pull" scheduling system – it pulls in stories to fill the team's WIP limit, based on [velocity](#).

Tracker automatically moves stories from the top of your project's Backlog into the current iteration according to the current [velocity](#). (depending on the individual [story estimates](#)).

Tracker projects use a linear point scale (i.e., 0, 1, 2, 3). You can change

The screenshot shows the Pivotal Tracker interface with two main panels: 'Current' and 'Backlog'.
Current Panel: Shows stories for the '20 Jun - Current' iteration. Stories include: 'Initial demo to investors' (Priority 3), 'Admin can review all order questions and send responses to shoppers (MR)' (Priority 2, blocked by 'admin'), 'Signed in shopper should be able to post product reviews (DST)' (Priority 2), 'Product browsing pagination not working in IE6 (MR)' (Priority 2, blocked by 'admin'), 'Some product photos not scaled properly when browsing products (DST)' (Priority 2, blocked by 'shopping'), 'Request higher number of production slices, for scaling (DST)' (Priority 2, deployment), 'We may experience some slight turbulence and then...explode. (DST)' (Priority 2), 'Integrate with payment gateway (MR)' (Priority 1, blocked by 'checkout, shopping'), 'Shopper should be able to reset forgotten password (MR)' (Priority 1, blocked by 'shopping, signup / signin'), 'Cart manipulation should be AJAXy (MR)' (Priority 1, blocked by 'cart, shopping'), 'Set up Engine Yard production environment' (Priority 1, deployment, shopping), 'Beta launch' (Priority 1, blocked by 'Finish').
Backlog Panel: Shows stories for future iterations:

- 27 Jun:** 'When shopper is browsing products, show average product rating and number of reviews next to each product (DST)' (Priority 3, user generated content)
- 4 Jul:** 'Shopper should be able to read reviews for a product' (Priority 3, user generated content), 'Admin should be able to mark a product as featured' (Priority 3, admin, featured products, needs design, shopping), 'Featured products should appear on the site landing page' (Priority 3, featured products, needs design)
- 11 Jul:** 'When checking out, shopper should have the option to sign in to their account' (Priority 3, shopping, signup / signin), 'If authorization is successful, show order number and confirmation message to shopper' (Priority 3, checkout, needs discussion, shopping), 'Signed in shopper should be able to review order history' (Priority 3, shopper accounts), 'Signed in shopper should be able to save product to favorites' (Priority 3, shopper accounts)
- 18 Jul:** 'Signed in shopper should be able to review and remove product from favorites' (Priority 3, shopper accounts), 'Apply styling to all shopper facing parts of the site, based on assets from designer' (Priority 3, design), 'Admin should be able to create and edit blog articles' (Priority 3, admin, blog, needs design, shopping), 'Admin should be able to save blog articles in draft mode' (Priority 3, admin, blog), 'Signed in shopper should be able to save credit card and address information used in checkout (MR)' (Priority 3, shopper accounts), 'Published blog articles should appear on the site' (Priority 3, blog, needs design)

PivotalTracker panels

The image shows a screenshot of the PivotalTracker application interface. It features five main panels:

- Done:** Shows completed stories. A story from April 27 is highlighted, showing a rejection dialog with the reason "Integration fails using Alliance credits". Number 8 is circled around the Done tab.
- Current:** Shows stories in progress. A story from May 2 is highlighted, showing a rejection dialog with the reason "Integration fails using Alliance credits". Number 7 is circled around the Current tab.
- Backlog:** Shows stories in the backlog. A story from May 30 is highlighted, showing a rejection dialog with the reason "Integration fails using Alliance credits". Number 4 is circled around the Backlog tab.
- Icebox:** Shows a single story: "Signed in shopper should be able to review order details". Number 1 is circled around the Icebox tab.
- Detailed Story View:** A modal window for a story from May 2. The story title is "When checking out, shopper should have the option to sign in to their account (MR)". The status is "Rejected". The reason for rejection is "Integration fails using Alliance credits". Buttons for "Accept" and "Reject" are visible. Number 5 is circled around the "Accept" button.

The detailed story view also includes fields for ID, Story Type (Feature), Points (Unestimated), Requester (MR), Owners (None), Follow This Story (1 follower), Description (edit), Labels, Tasks (Add a task), and Activity.

Story Points

Funcionalidades (encomenda de comida online):

- F1: Inicio de sessão do utilizador (login)
- F2: Registo de novo utilizador na plataforma
- F3: Listar promoções em destaque do dia.
- F4: Colocar a encomenda (inclui pagamento)

Escala:

1pt: muito fácil. Direto de se implementar e âmbito reduzido.

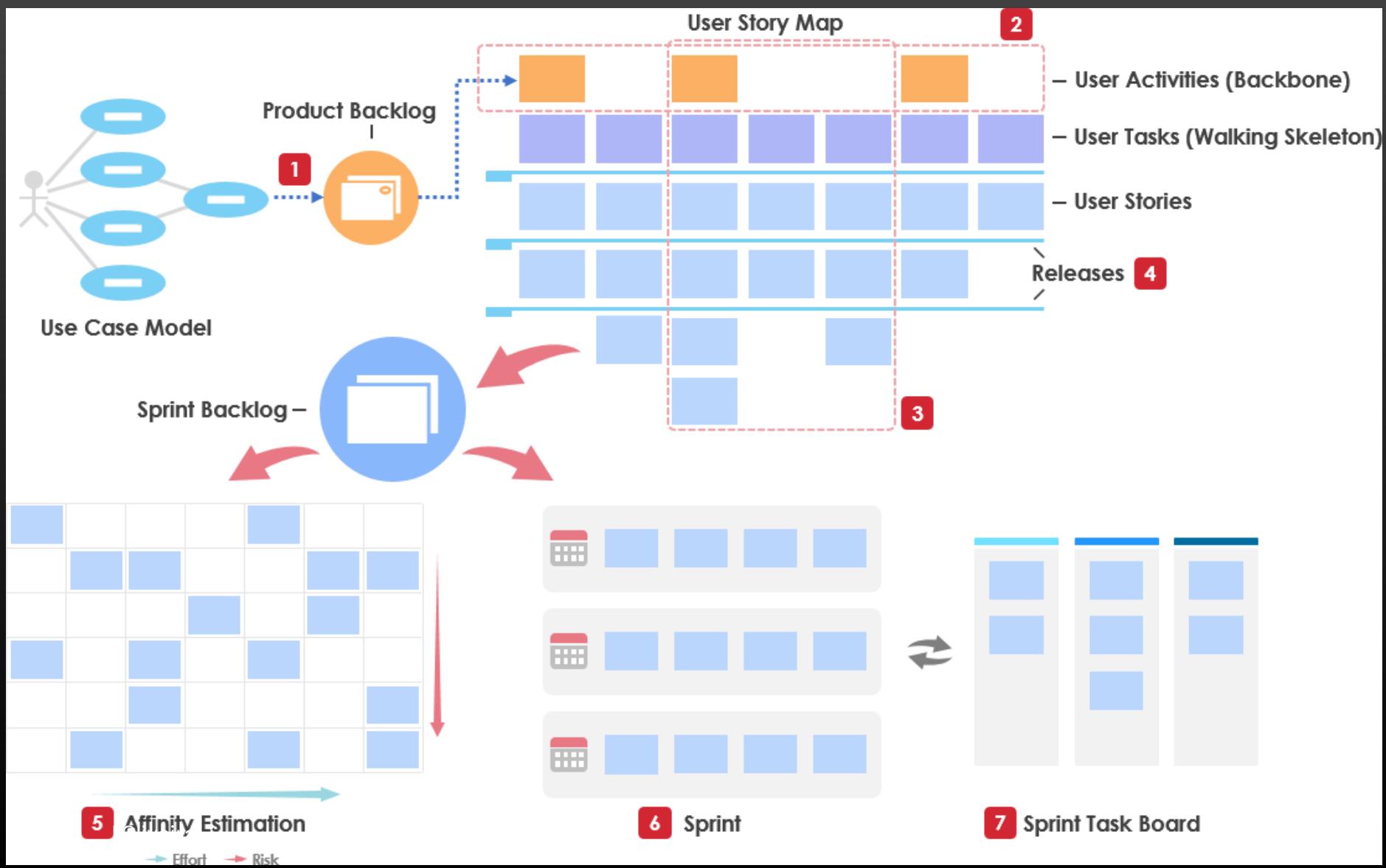
2pts: acessível; não oferece grande dificuldade.

4pts: complexo; tem várias interdependências (de outros módulos/serviços) ou um fluxo elaborado

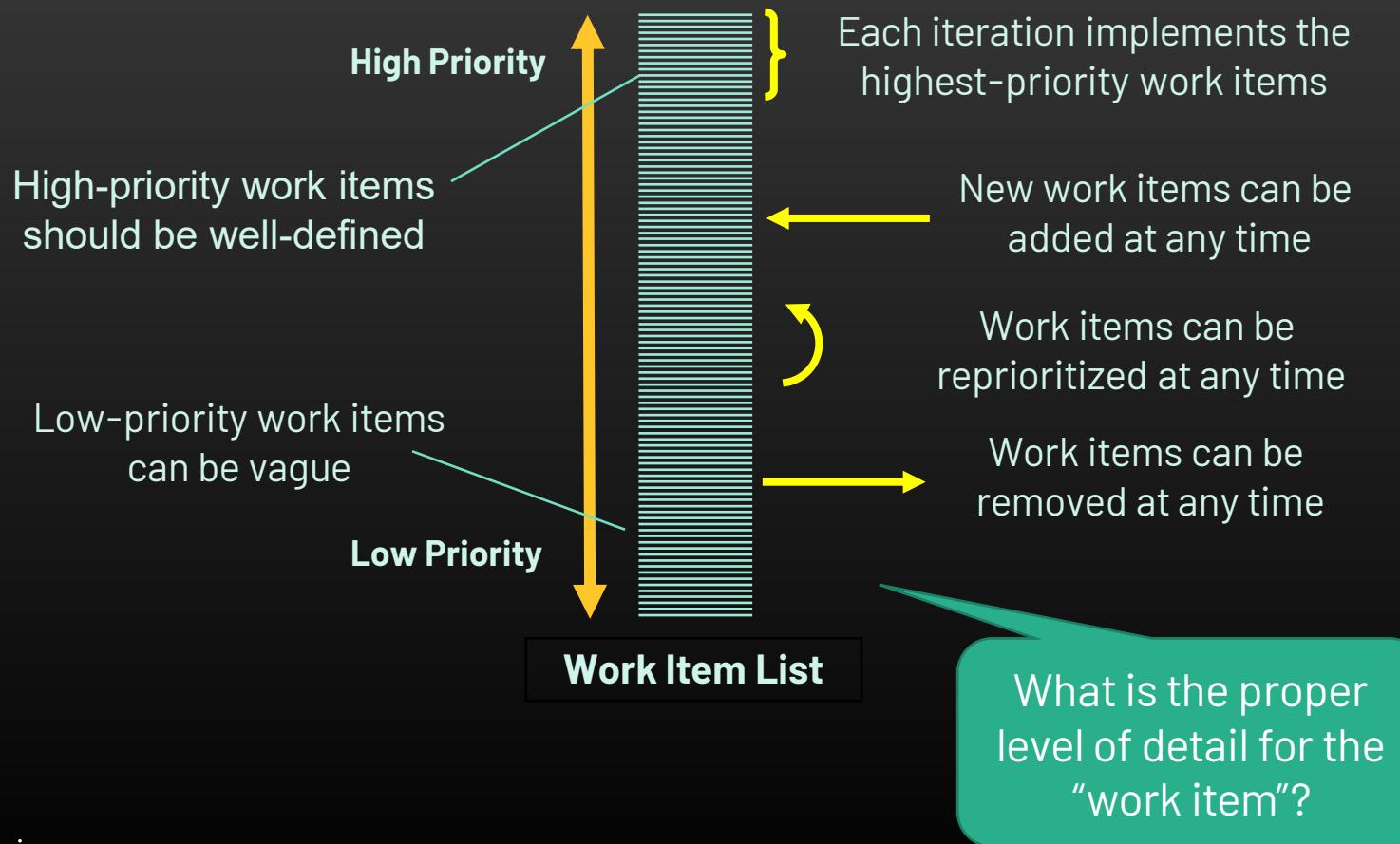
8pts: muito complexo; requer integrações, tecnologias ou conhecimentos que não são completamente dominados

→ <http://bit.ly/2IUrMn>

Agile in Visual Paradigm



Managing the work items list (*backlog*)



Scrum framework activities

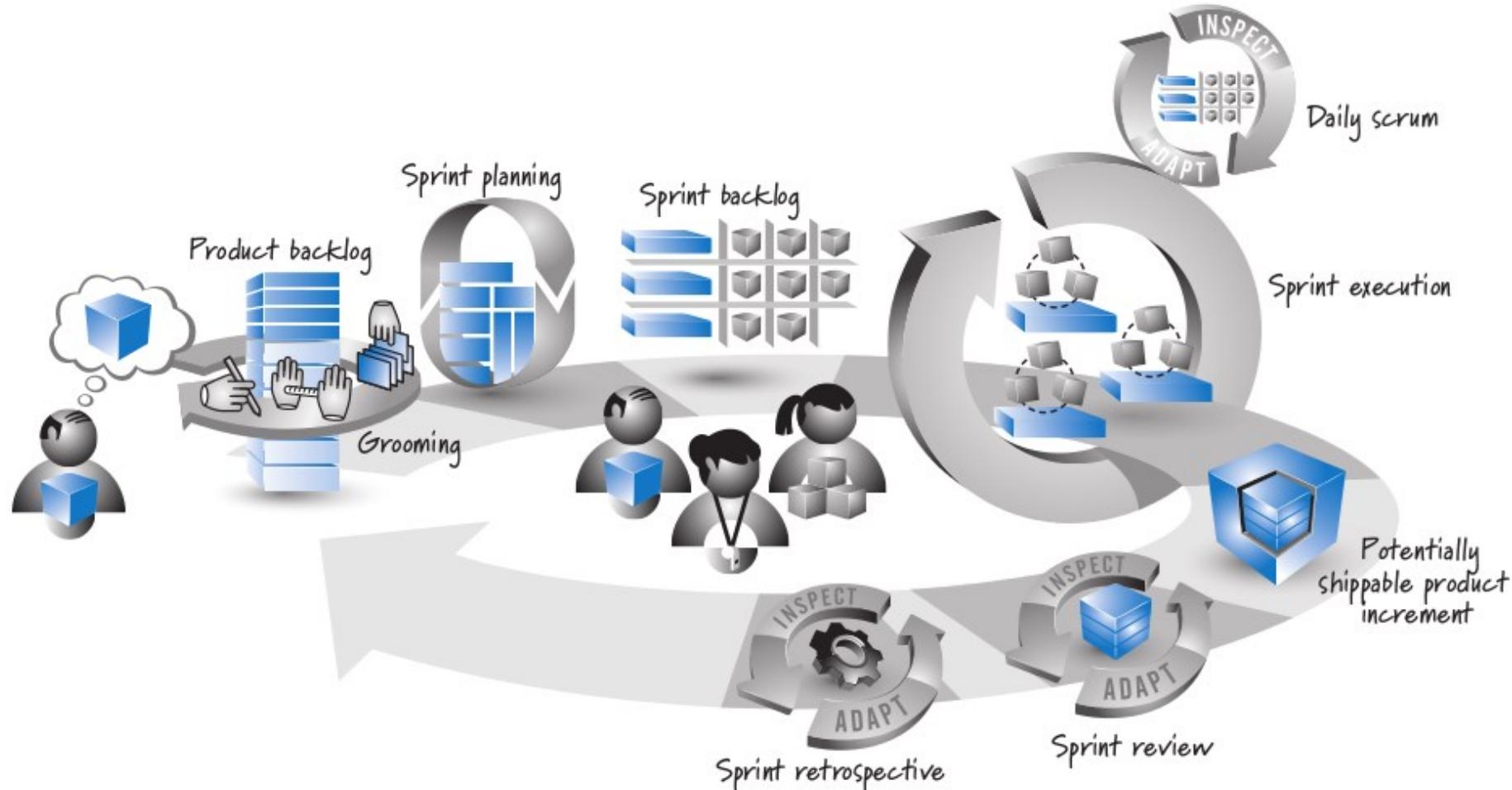


FIGURE 2.3 Scrum framework

Requirements elicitation by exploring user-centered scenarios

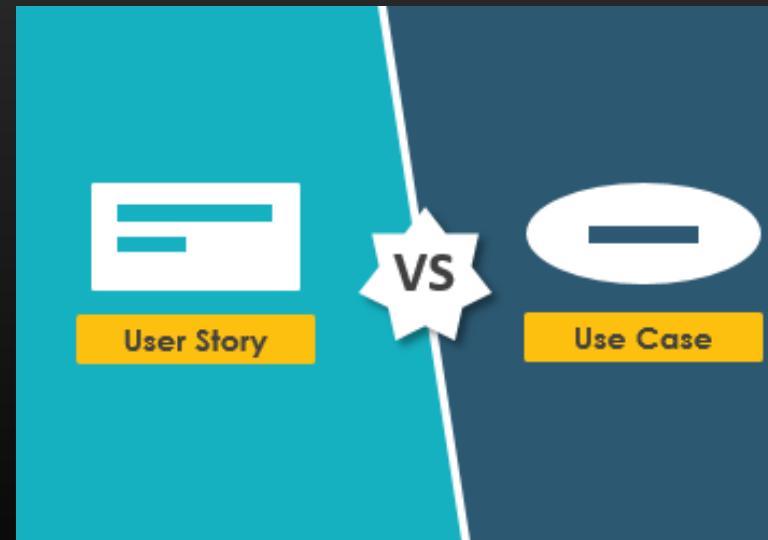
- A) Use cases
- B) User stories
- C) User-centered design (UCD)
- D) Customer Journey Map (Experience maps)

Recall: use cases and related assets

A *use case* describes a sequence of interactions between a system and an external actor that results in the actor being able to achieve some outcome of value.

The names of use cases are always written in the form of a verb followed by an object.

The use case is supplemented with a detailed description (following a template)



ID and Name:	UC-4 Request a Chemical	
Created By:	Lori	Date Created: 8/22/13
Primary Actor:	Requester	Secondary Actors: Buyer, Chemical Stockroom, Training Database
Description:	The Requester specifies the desired chemical to request by entering its name or chemical ID number or by importing its structure from a chemical drawing tool. The system either offers the Requester a container of the chemical from the chemical stockroom or lets the Requester order one from a vendor.	
Trigger:	Requester indicates that he wants to request a chemical.	
Preconditions:	PRE-1. User's identity has been authenticated. PRE-2. User is authorized to request chemicals. PRE-3. Chemical inventory database is online.	
Postconditions:	POST-1. Request is stored in the CTS. POST-2. Request was sent to the Chemical Stockroom or to a Buyer.	
Normal Flow:	4.0 Request a Chemical from the Chemical Stockroom <ol style="list-style-type: none"> Requester specifies the desired chemical. System lists containers of the desired chemical that are in the chemical stockroom, if any. System gives Requester the option to View Container History for any container. Requester selects a specific container or asks to place a vendor order (see 4.1). Requester enters other information to complete the request. System stores the request and notifies the Chemical Stockroom. 	
Alternative Flows:	4.1 Request a Chemical from a Vendor <ol style="list-style-type: none"> Requester searches vendor catalogs for the chemical (see 4.1.E1). System displays a list of vendors for the chemical with available container sizes, grades, and prices. Requester selects a vendor, container size, grade, and number of containers. Requester enters other information to complete the request. System stores the request and notifies the Buyer. 	
Exceptions:	4.1.E1 Chemical Is Not Commercially Available <ol style="list-style-type: none"> System displays message: No vendors for that chemical. System asks Requester if he wants to request another chemical (3a) or to exit (4a). a. Requester asks to request another chemical. b. System starts normal flow over. a. Requester asks to exit. b. System terminates use case. 	
Priority:	High	
Frequency of Use:	Approximately 5 times per week by each chemist, 200 times per week by chemical stockroom staff	

Use cases and similar user stories

TABLE 8-2 Some sample use cases and corresponding user stories

Application	Sample use case	Corresponding user story
Chemical tracking system	Request a Chemical	As a chemist, I want to request a chemical so that I can perform experiments.
Airport check-in kiosk	Check in for a Flight	As a traveler, I want to check in for a flight so that I can fly to my destination.
Accounting system	Create an Invoice	As a small business owner, I want to create an invoice so that I can bill a customer.
Online bookstore	Update Customer Profile	As a customer, I want to update my customer profile so that future purchases are billed to a new credit card number.

More often: user story ⊂ use case

Recall that user stories are concise statements of user needs, in contrast to the richer description that a use case provides. In the agile world, a user story sometimes covers the same scope as an entire use case, but in other cases a user story represents just a single scenario or alternative flow. If an agile development team were discussing requirements for the CTS, they might come up with user stories such as the following:

As a chemist, I want to request a chemical so that I can perform experiments.

As a chemist, I want to request a chemical from the Chemical Stockroom so that I can use it immediately.

As a chemist, I want to request a chemical from a vendor because I don't trust the purity of any of the samples available in the Chemical Stockroom.

The first of these three stories corresponds to the use case as a whole. The second and third user stories represent the normal flow of the use case and the first alternative flow, from Figure 8-3.

At this level, use cases look much like user stories. Both are focused on understanding what different types of users need to accomplish through interactions with a software system. However, the two processes move in different directions from these similar starting points, as illustrated in Figure 8-1. Both approaches can also produce other deliverables, such as visual analysis models, but Figure 8-1 illustrates the core distinction.

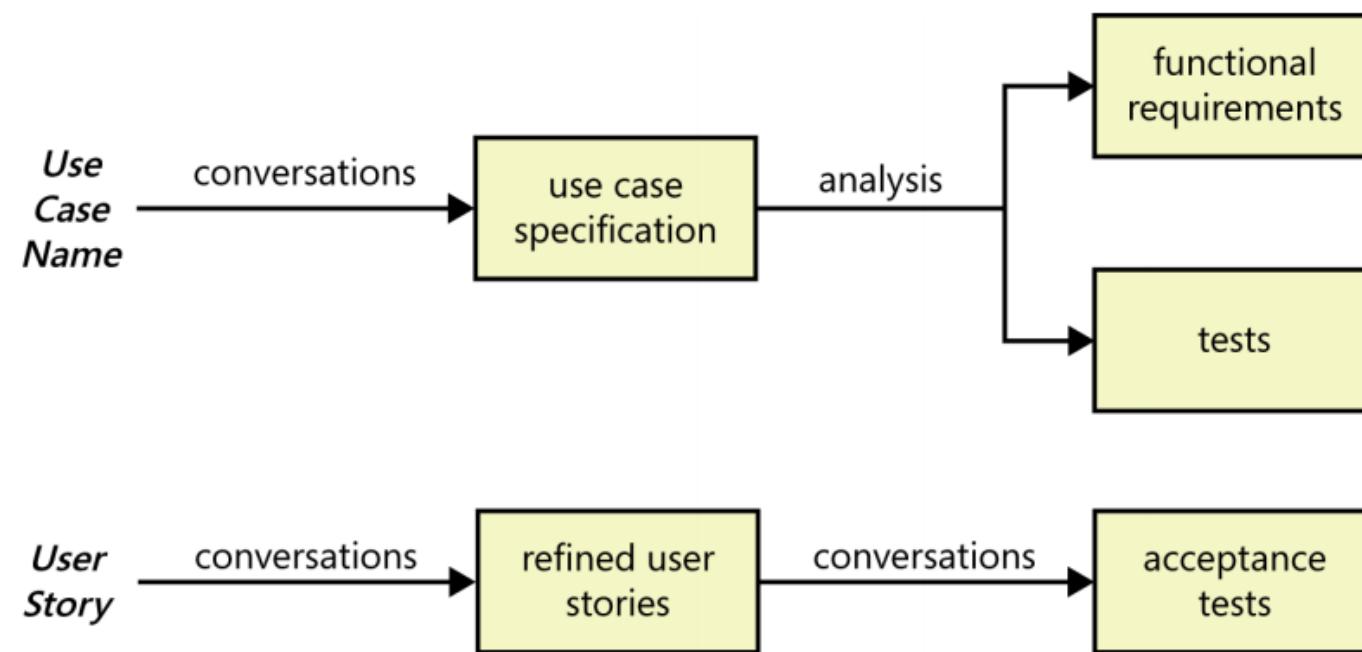


FIGURE 8-1 How user requirements lead to functional requirements and tests with the use case approach and the user story approach.

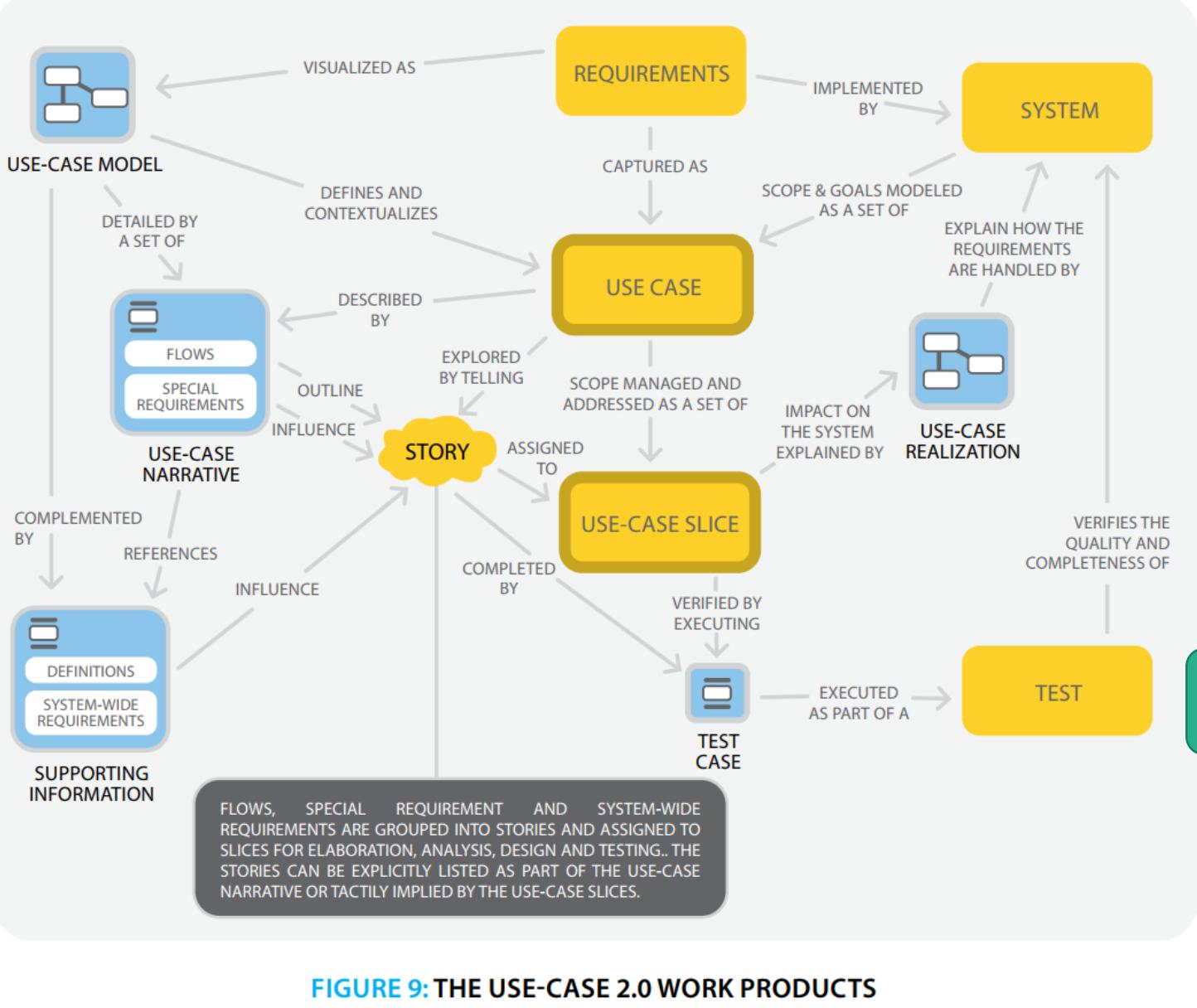
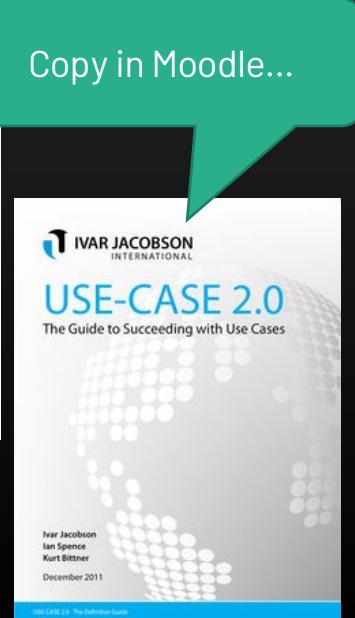


FIGURE 9: THE USE-CASE 2.0 WORK PRODUCTS



Jacobson: flows in a use case match stories

A story is described by part of the use-case narrative, one or more flows and special requirements, and one or more test cases. The key to finding effective stories is to understand the structure of the use-case narrative. The network of flows can be thought of as a map that summarizes all the stories needed to describe the use case. **Figure 8** illustrates the relationship between the flows of a use-case narrative and the stories it describes.

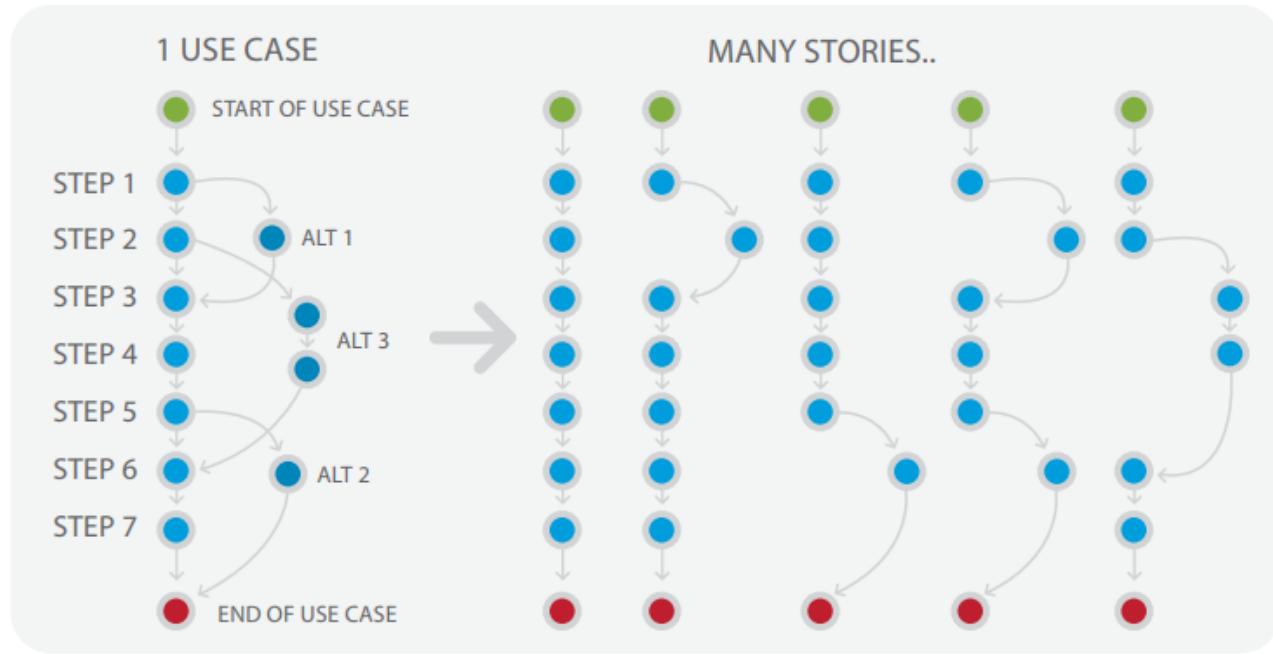
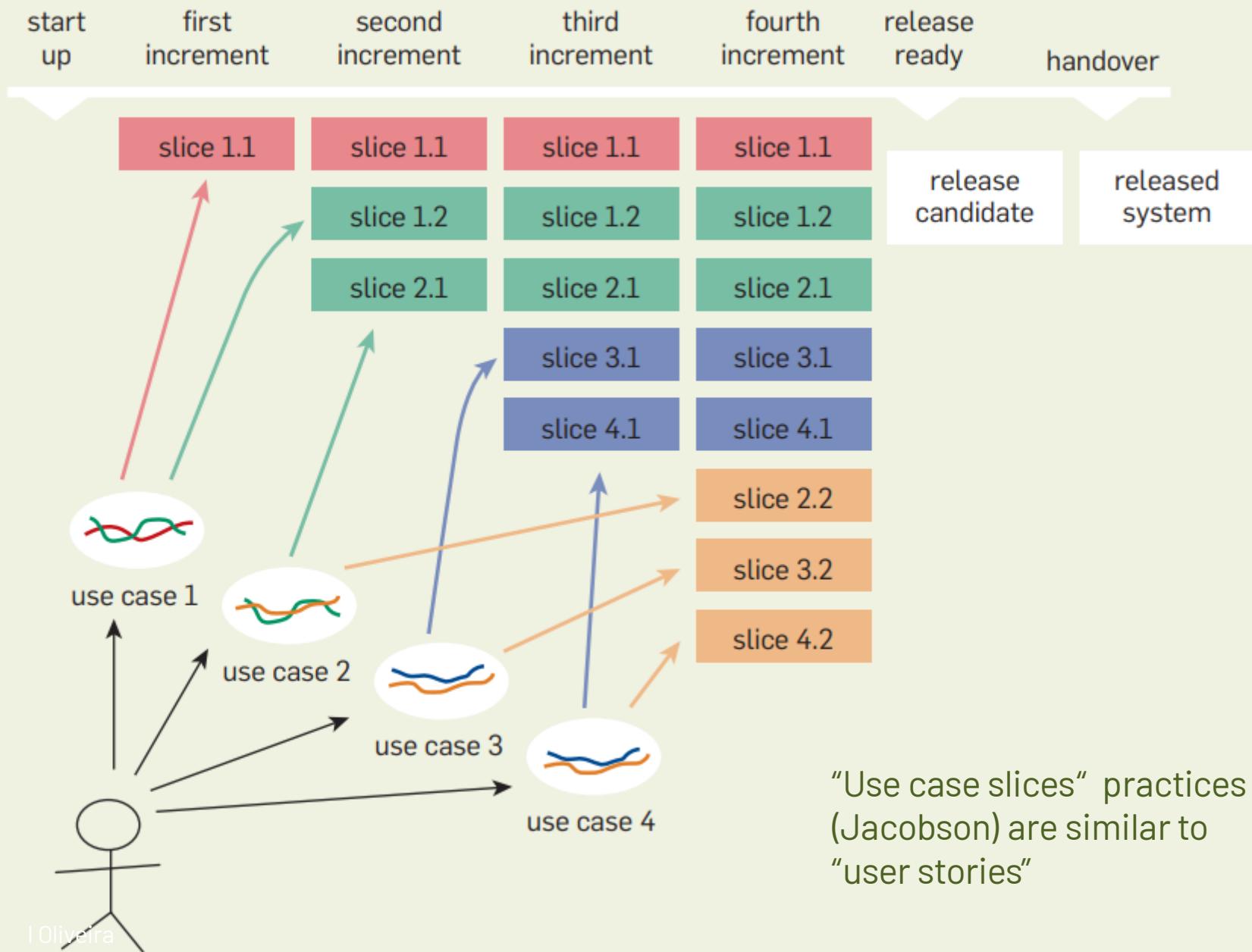


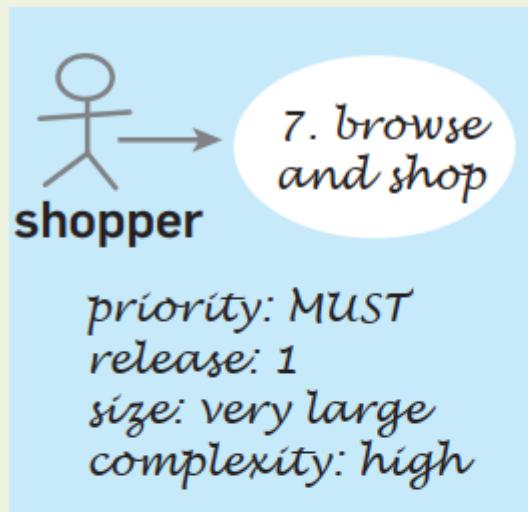
FIGURE 8:
THE RELATIONSHIP BETWEEN THE FLOWS AND THE STORIES

Figure 4. Use cases, use-case slices, increments, and releases.



"Use case slices" practices
(Jacobson) are similar to
"user stories"

Figure 5. Capturing the properties of a use case and its slices using sticky notes.



a use case and its properties captured on a sticky note

7.1 select and buy 1 product

flows: BF
test: 1 product, default payment, valid details

5

7.2 select and buy 100 products

flows: BF
test: 100 products, default payment, valid details

5

7.3 support systems unavailable

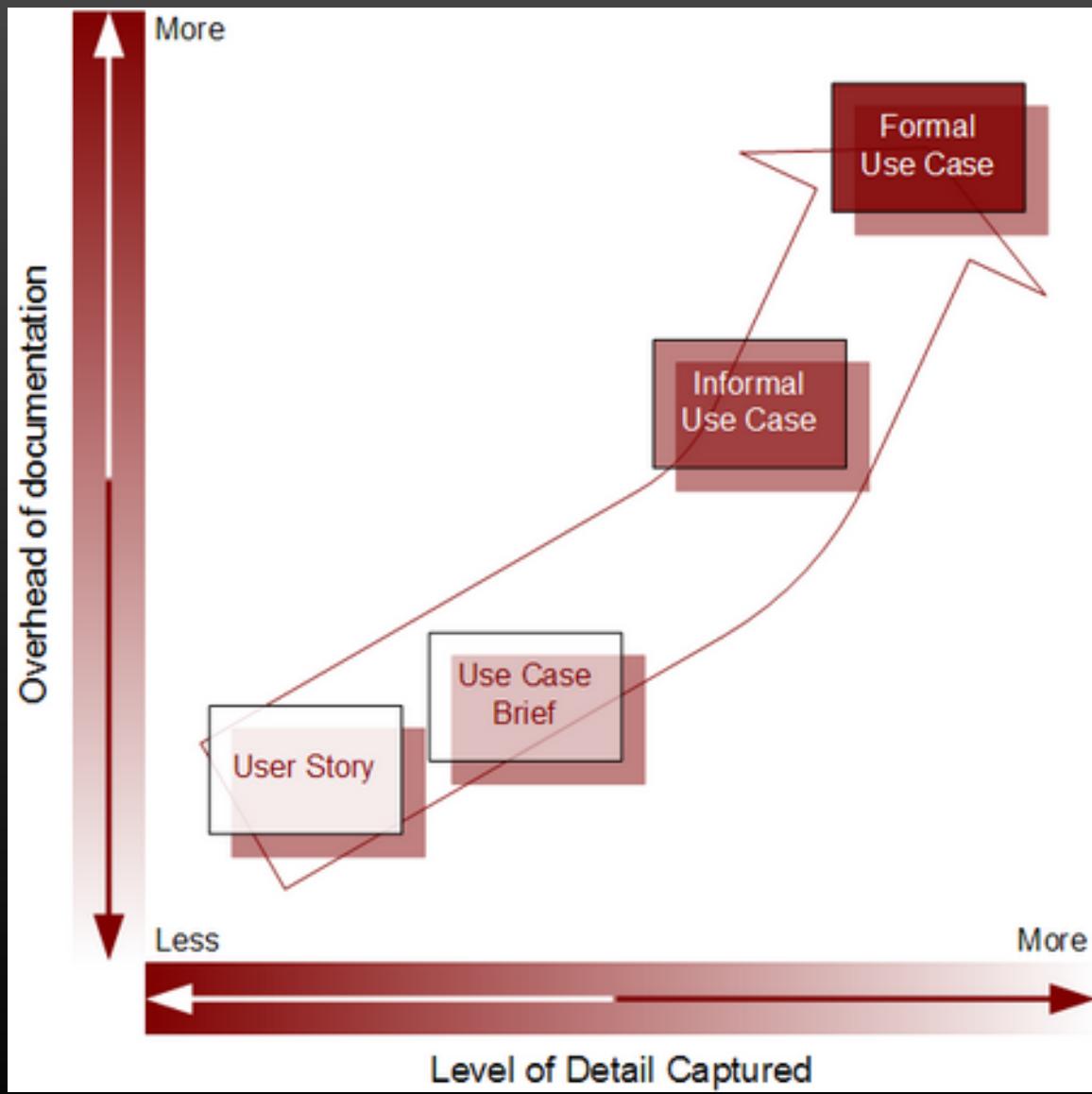
flows: BF, A9, A10, A1, A12
test: select product, provide information, disconnect each system in between

13

some slices from the use case captured on their own sticky notes

User stories and use cases: what is the difference?

Common	Use-Cases	User-story
<ul style="list-style-type: none">• Both take a usage-centric approach• Both are placeholders for a conversation• Both result in test cases that represent the acceptance criteria• Both can be estimated	<ul style="list-style-type: none">• Big picture to help people understand the extent of the system and its value• Dive further into describing how the user imagines interacting with the system to accomplish his objective.• Provide project participants with a structure and context that a collection of user stories lacks• You can examine each element of a use case (flows, preconditions, postconditions, and so on) to look for pertinent functional and nonfunctional requirements and to derive tests. This helps you avoid overlooking any requirements• Active scope management	<ul style="list-style-type: none">• Concise statement of a user's needs• Easy access to domain experts available (refine the story as needed)• More suited to act as a backlog item for daily activities (Scrum, Kanban, specification by example)• Explicit acceptance criteria



Usage-centric approaches to requirements: benefits and limitations

Both use cases and user stories shift from the product-centric perspective of requirements elicitation to discussing what users need to accomplish, in contrast to asking users what they want the system to do.

The intent of this approach is to describe tasks that users will need to perform with the system, or user-system interactions that will result in a valuable outcome for some stakeholder.

Use cases and user stories work well for exploring the requirements for business applications, websites, kiosks, and systems that let a user control a piece of hardware.

However: they are inadequate for understanding the requirements of certain types of applications. Applications such as batch processes, computationally intensive systems, business analytics, and data warehousing might have just a few use cases. The complexity of these applications lies in the computations performed, the data found and compiled, or the reports generated, not in the user-system interactions.

Nor are use cases and user stories sufficient for specifying many embedded and other real-time systems.

Take away messages

- Agile projects (especially Scrum ones) use a product backlog, which is a prioritized list of the functionality to be developed.
- Product backlog items can be whatever the team desires, but user stories have emerged as the best and most popular form of product backlog items.
- Both use cases and user stories focus on conversations and system usage by people.
- Use cases provide more structure and a way to document details collected in analysis.
- User stories are refined as needed; focus on giving an example for a small feature.

References

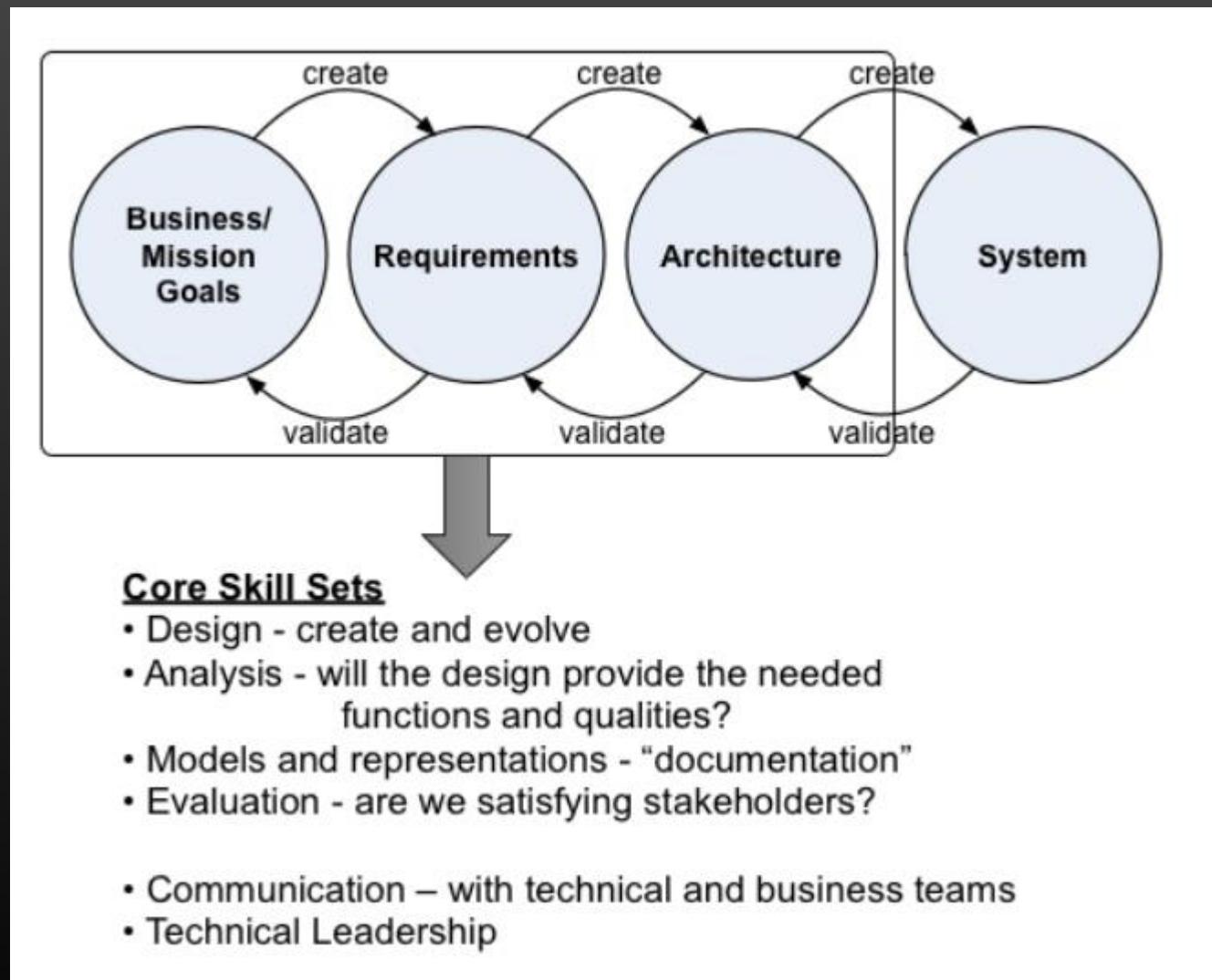
Core readings	Suggested readings
<ul style="list-style-type: none">• Jacobson, I., Spence, I., & Kerr, B. (2016). <u>Use-case 2.0</u>. <i>Communications of the ACM</i>, 59(5), 61–69.• “<u>User Story vs Use Case for Agile Software Development</u>”, Visual Paradigm	<ul style="list-style-type: none">• Jacobson, I., Spence, I., & Bittner, K. (2011). <u>Use-Case 2.0</u> <i>The Guide to Succeeding with Use Cases</i>. [e-Book]• <u>User story</u> (VisualParadigm handbook)

Vistas de Arquitetura - complemento

Ilídio Oliveira

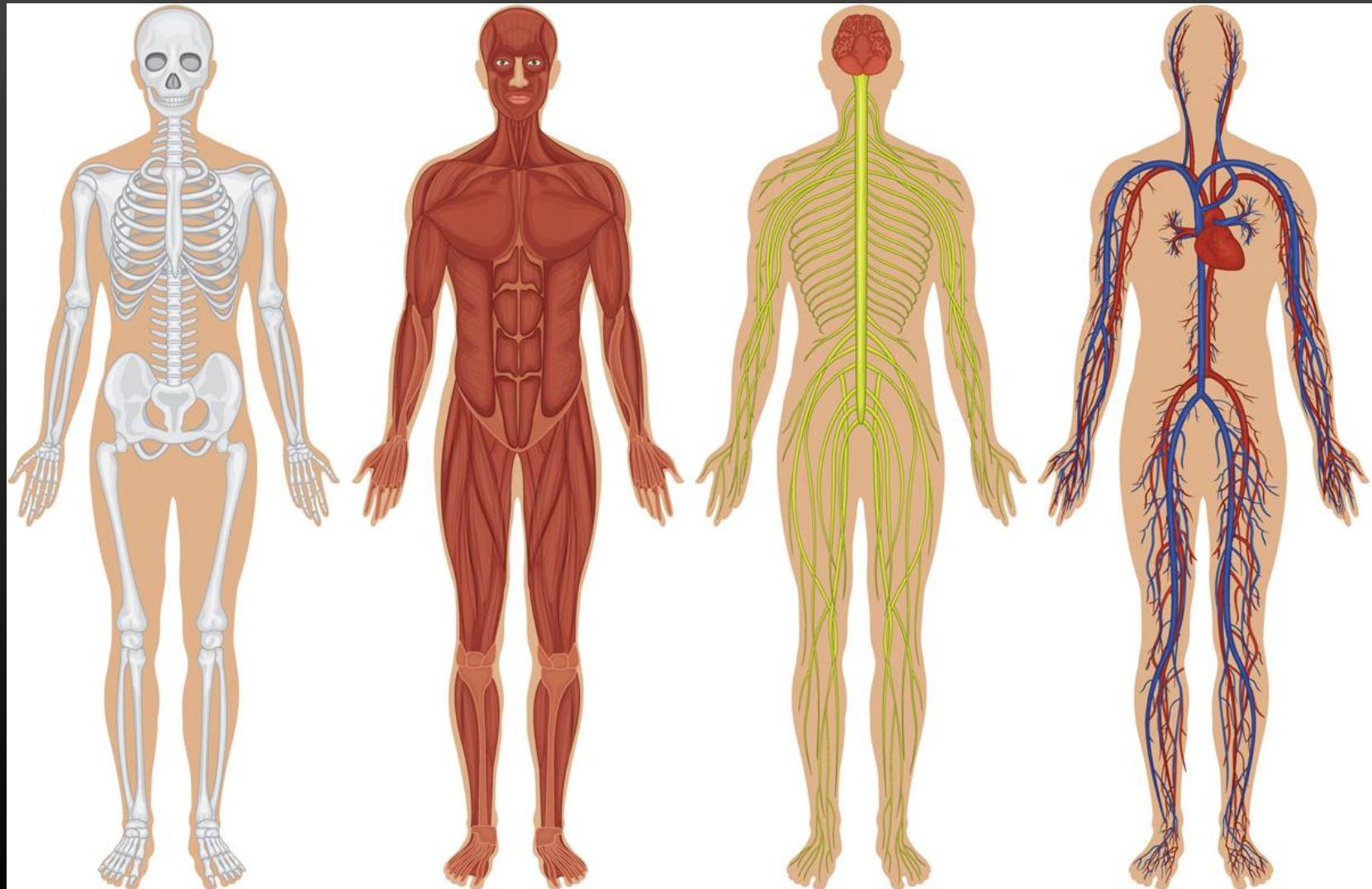
v2022-05-25

Papel do arquiteto (de software)



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

Sistemas feitos de estruturas... partes, relações, comportamento.



O que é a arquitetura de software?

A Arquitectura é um Conjunto de Estruturas (de Software)

Partes/elementos relevantes ligados por alguma relação.

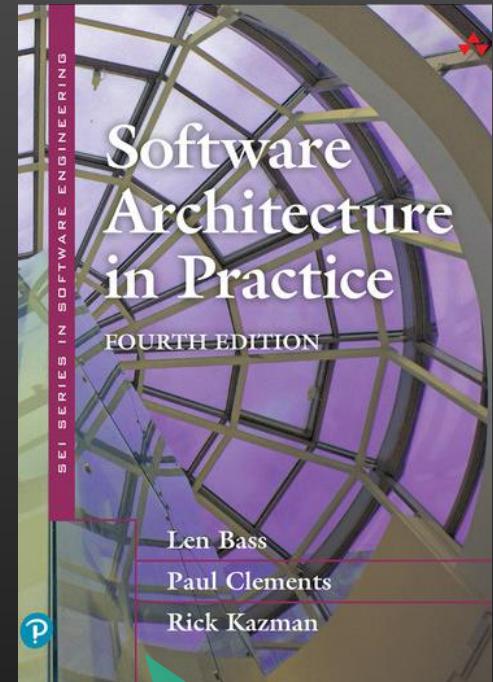
Arquitectura é uma Abstracção

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.



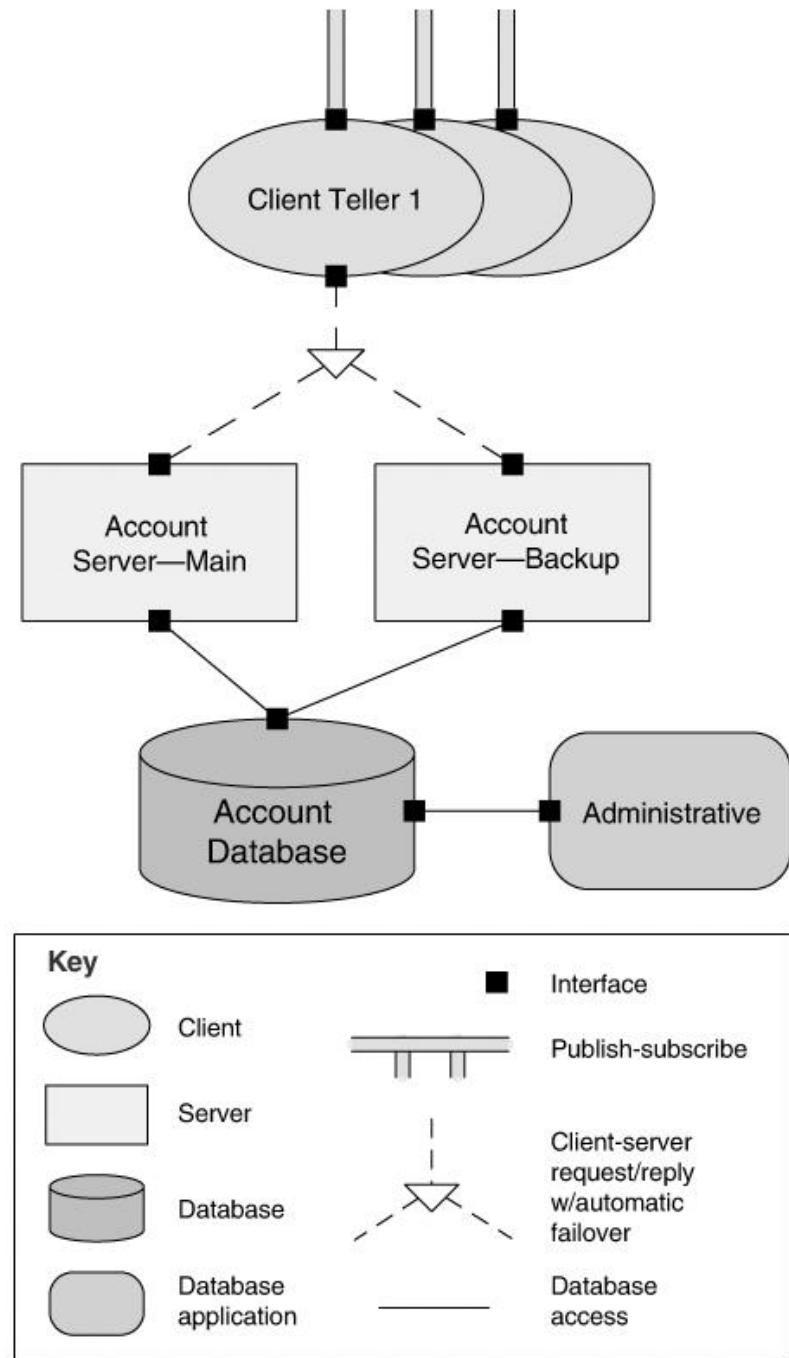
The software architecture of a system is the set of structures needed to reason about the system. These structures comprise software elements, relations among them, and properties of both.

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).

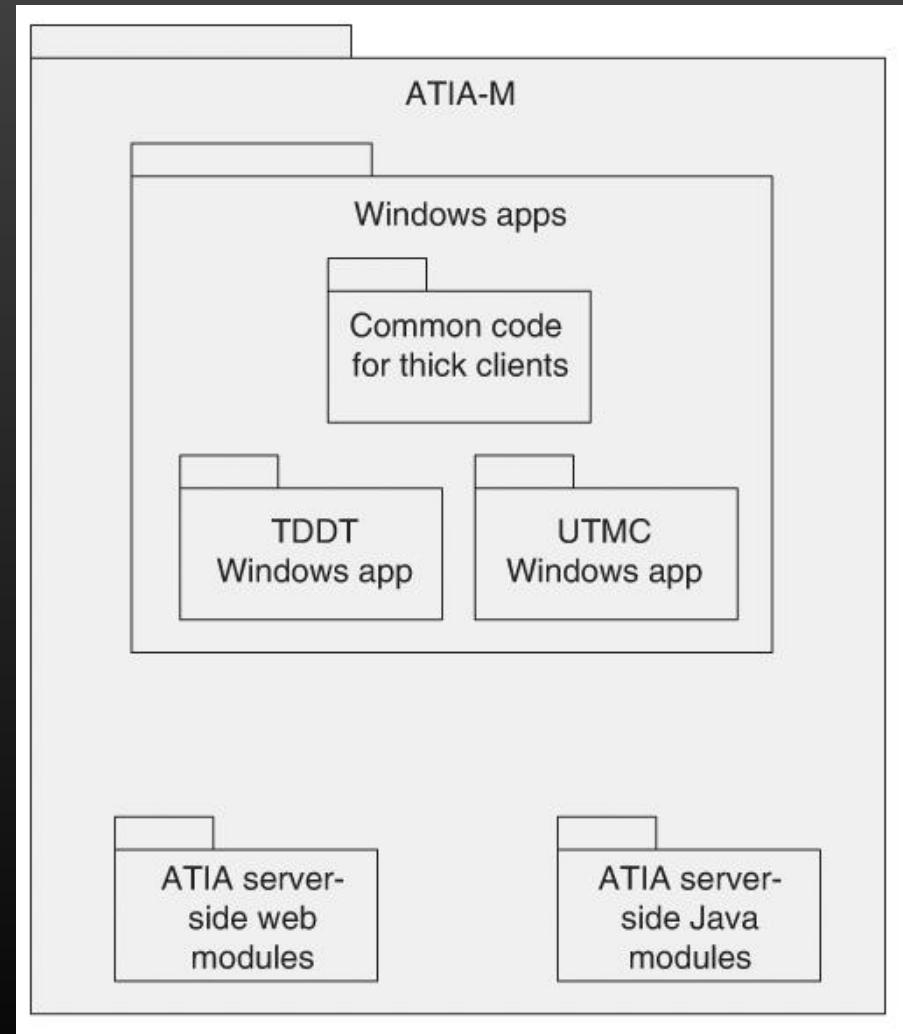


Tipos de estruturas #2

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.



Tipos de estruturas #3

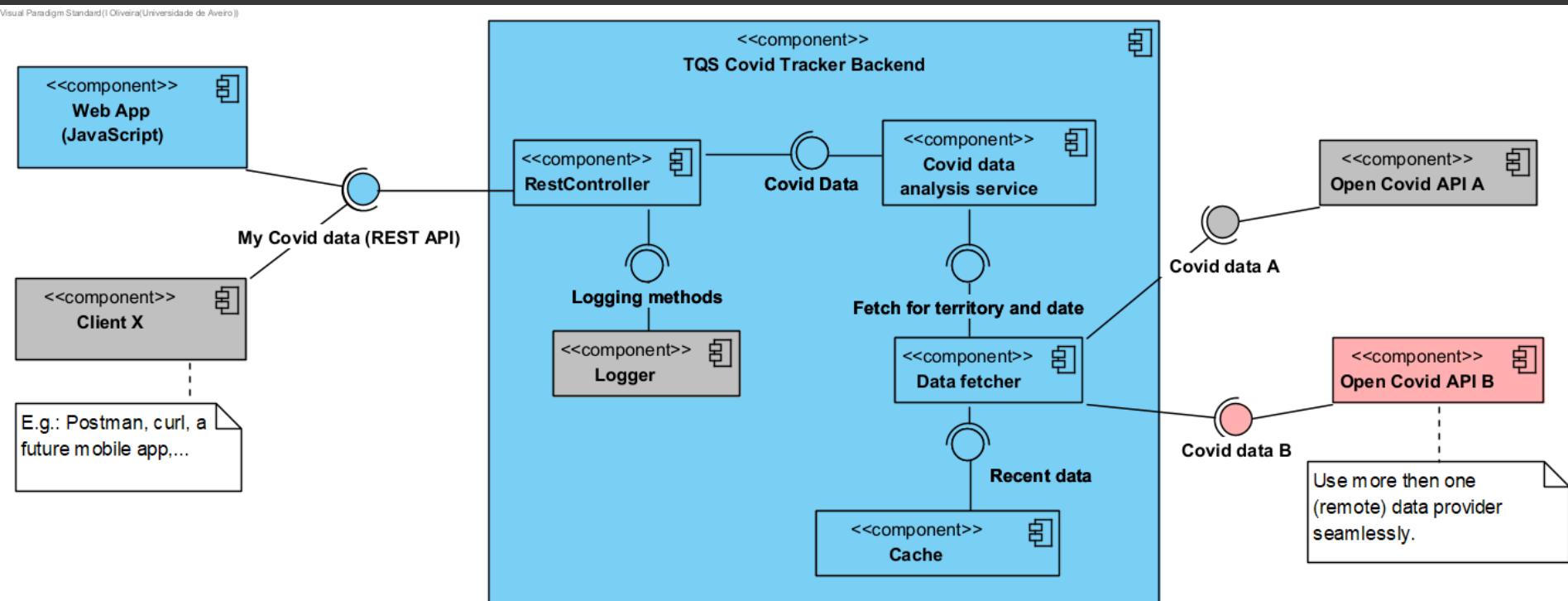
3/ As estruturas de alocação (*Allocation structures*) estabelecem o mapeamento das estruturas de software nas estruturas de não-software do sistema, tais como os seus ambientes de desenvolvimento, teste, e execução.

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?

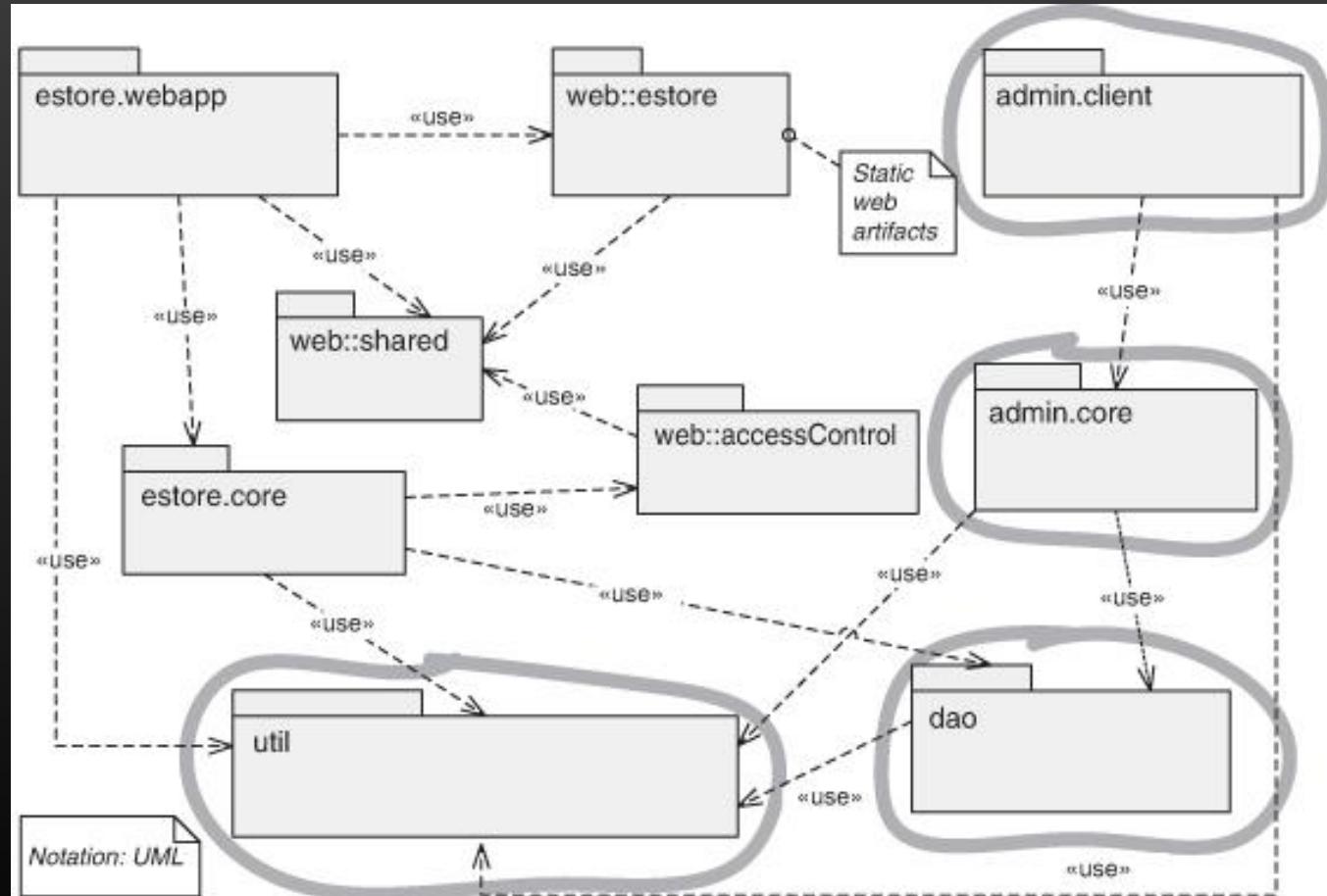
Caso de exemplo: componentes-conetores (C&C)

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



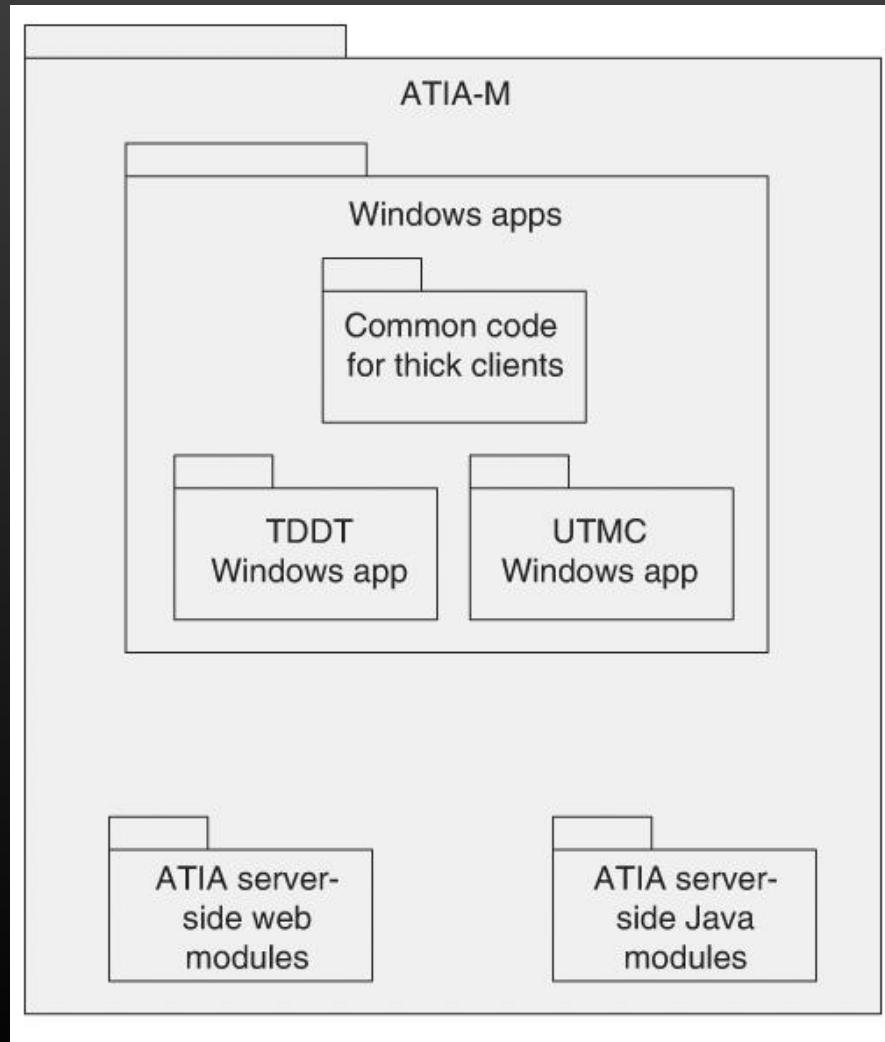
Exemplo de estrutura: módulos

Relação <<uses>>: as unidades estão relacionadas pela relação de *uses*, uma forma especializada de dependência.



Exemplo de estrutura: módulos

Decomposição (relação “is-a-submodule-of”)



Exemplo de estrutura: módulos

Layers: uma camada "virtualiza" os subsistemas subjacentes e fornece um conjunto coeso de serviços através de uma interface controlada.

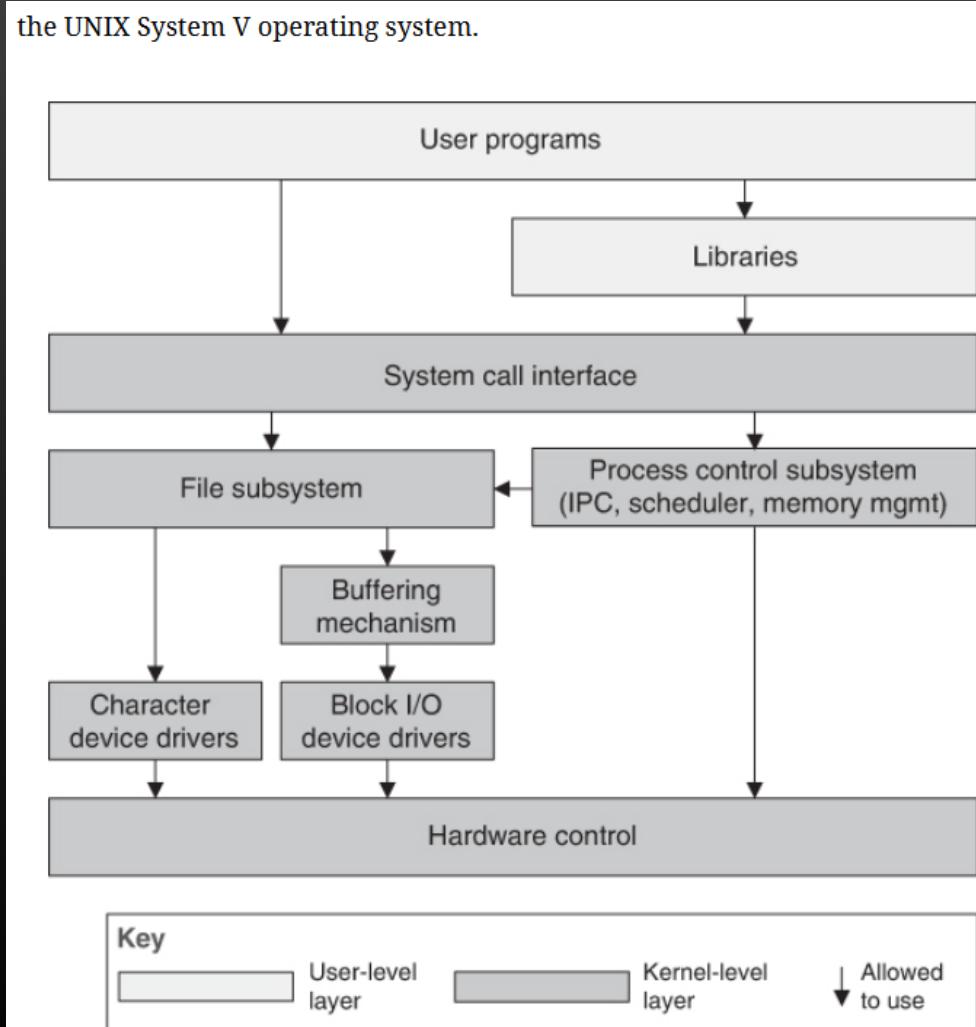


Figure 1.7 Layer structure

Casos de exemplo: alocação/atribuição

O diagrama de instalação mostra a alocação de unidades a nós computacionais.

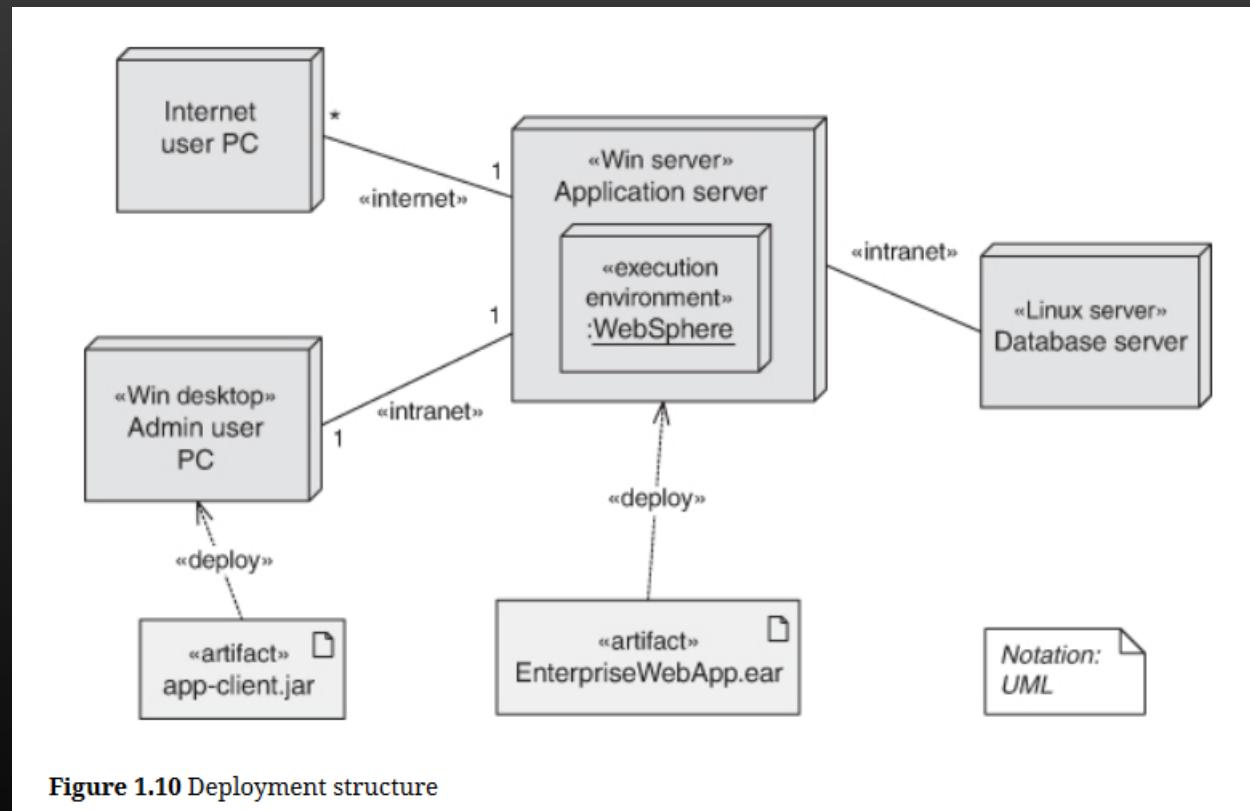
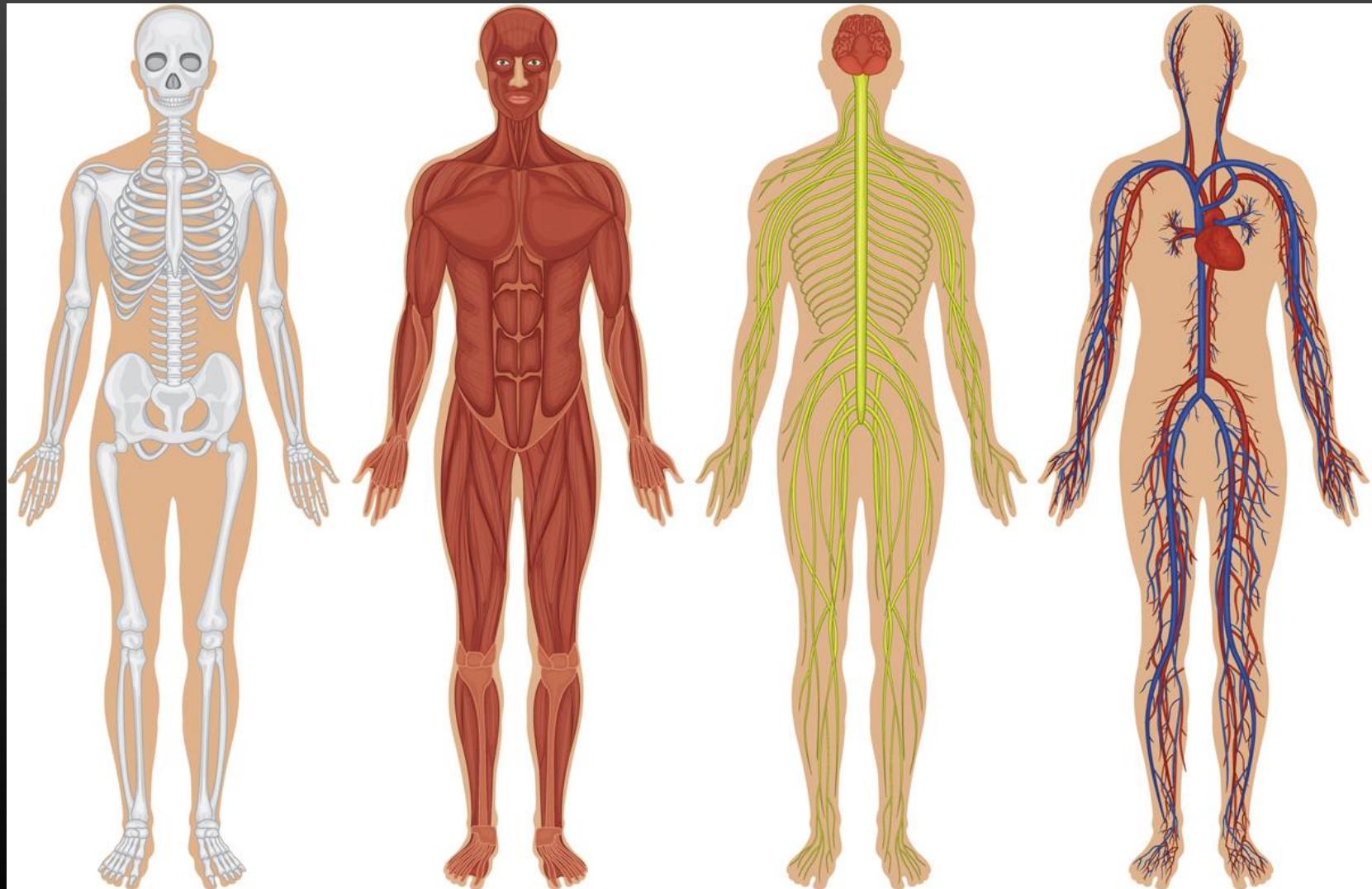


Figure 1.10 Deployment structure

Requisitos com impacto nas decisões de arquitetura

O que terá influenciado o “desenho”?



As decisões de arquitectura 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 (ASR) é um requisito que terá um efeito profundo na arquitectura - ou seja, a arquitectura pode muito bem ser substancialmente diferente na ausência de tal requisito.

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.*

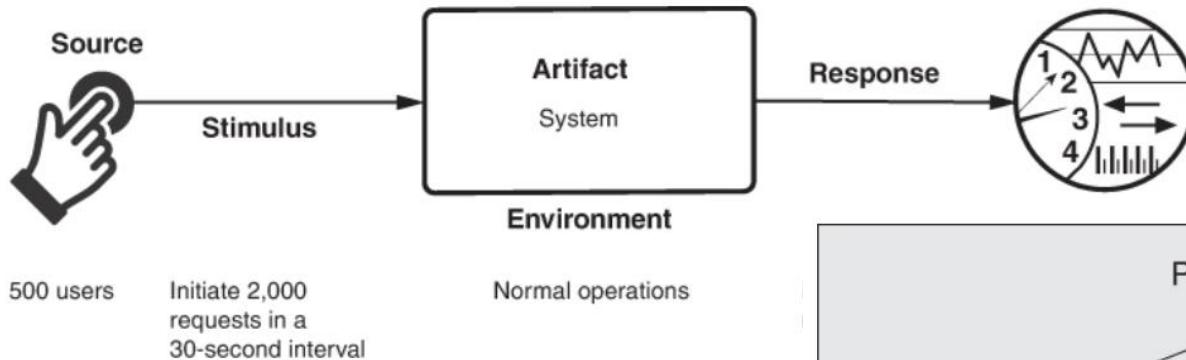
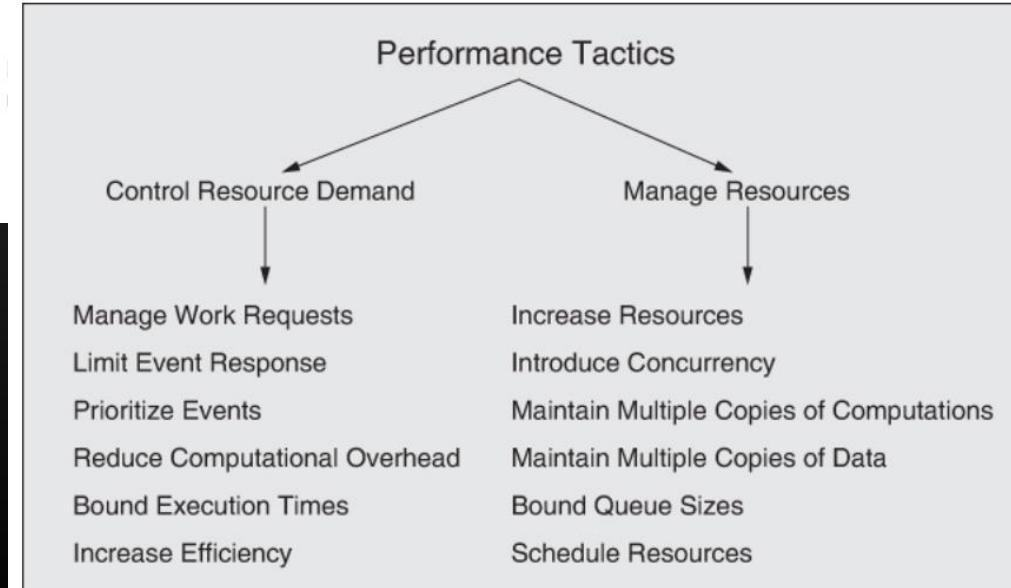
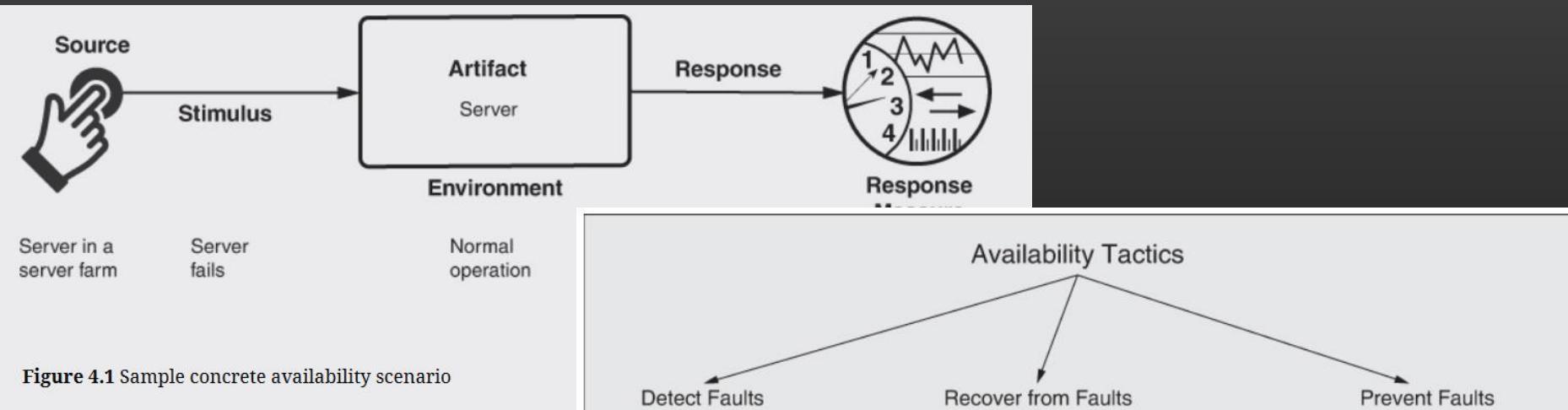


Figure 9.1 Sample performance scenario

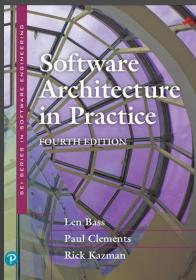


Os requisitos conduzem a arquitetura

E.g.: disponibilidade como atributo de qualidade



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 permitem 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.

Documentar a arquitetura

Documentar a arquitetura

Uma arquitectura de software é uma entidade complexa que não pode ser descrita de uma forma unidimensional simples.

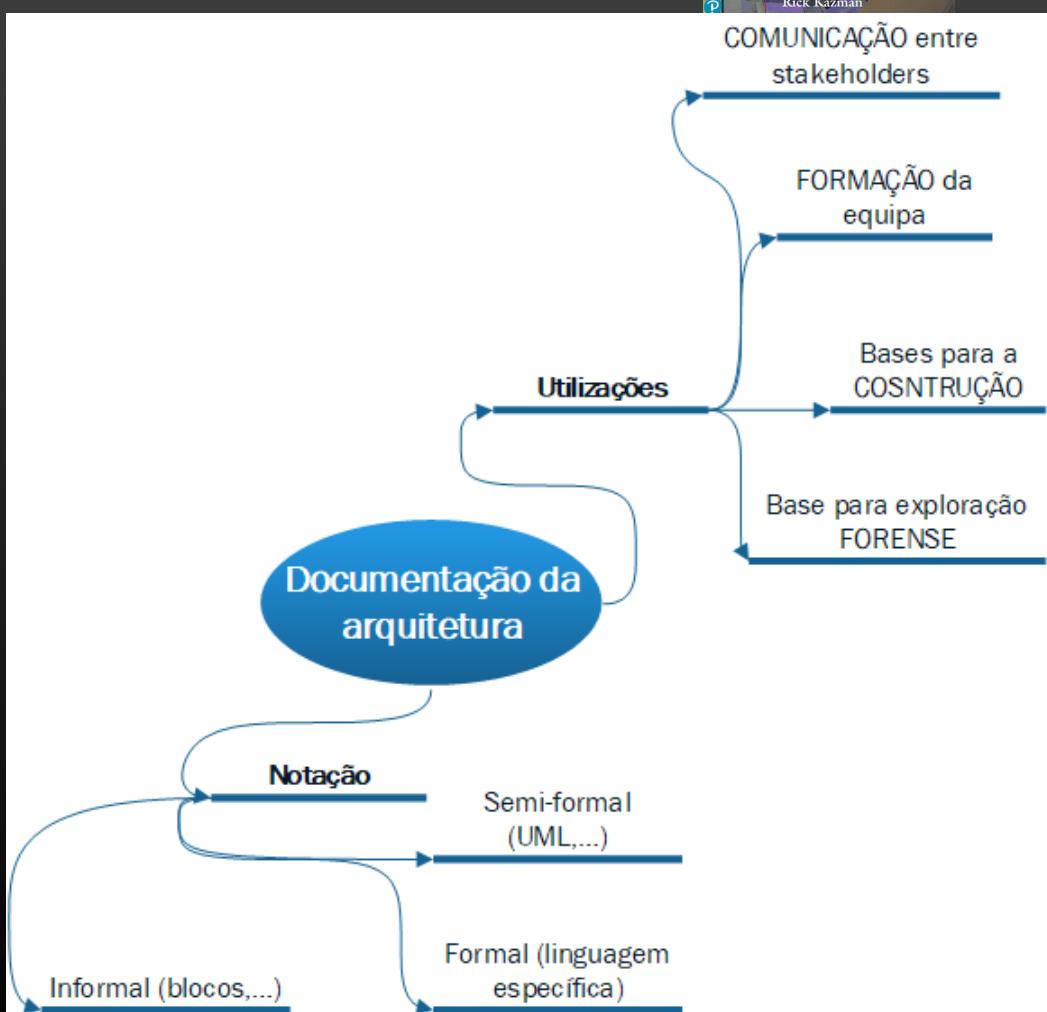
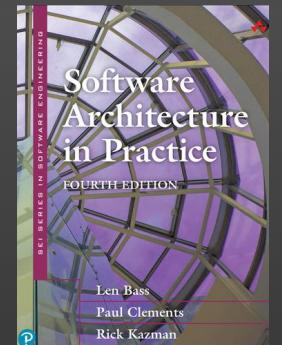
Uma vista é uma representação de um conjunto de elementos e relações entre eles - não todos os elementos do sistema, mas aqueles de um determinado tipo.

E.g.: uma visão em camadas de um sistema mostraria elementos do tipo "camada"; ou seja, mostraria a decomposição do sistema em camadas, juntamente com as relações entre essas camadas. Uma visão em camadas pura não mostraria, contudo, os serviços do sistema, ou clientes e servidores, ou modelo de dados, ou qualquer outro tipo de elemento.

As vistas permitem-nos dividir a entidade multidimensional (a arquitectura do software) em várias representações parciais e inteligíveis do sistema.

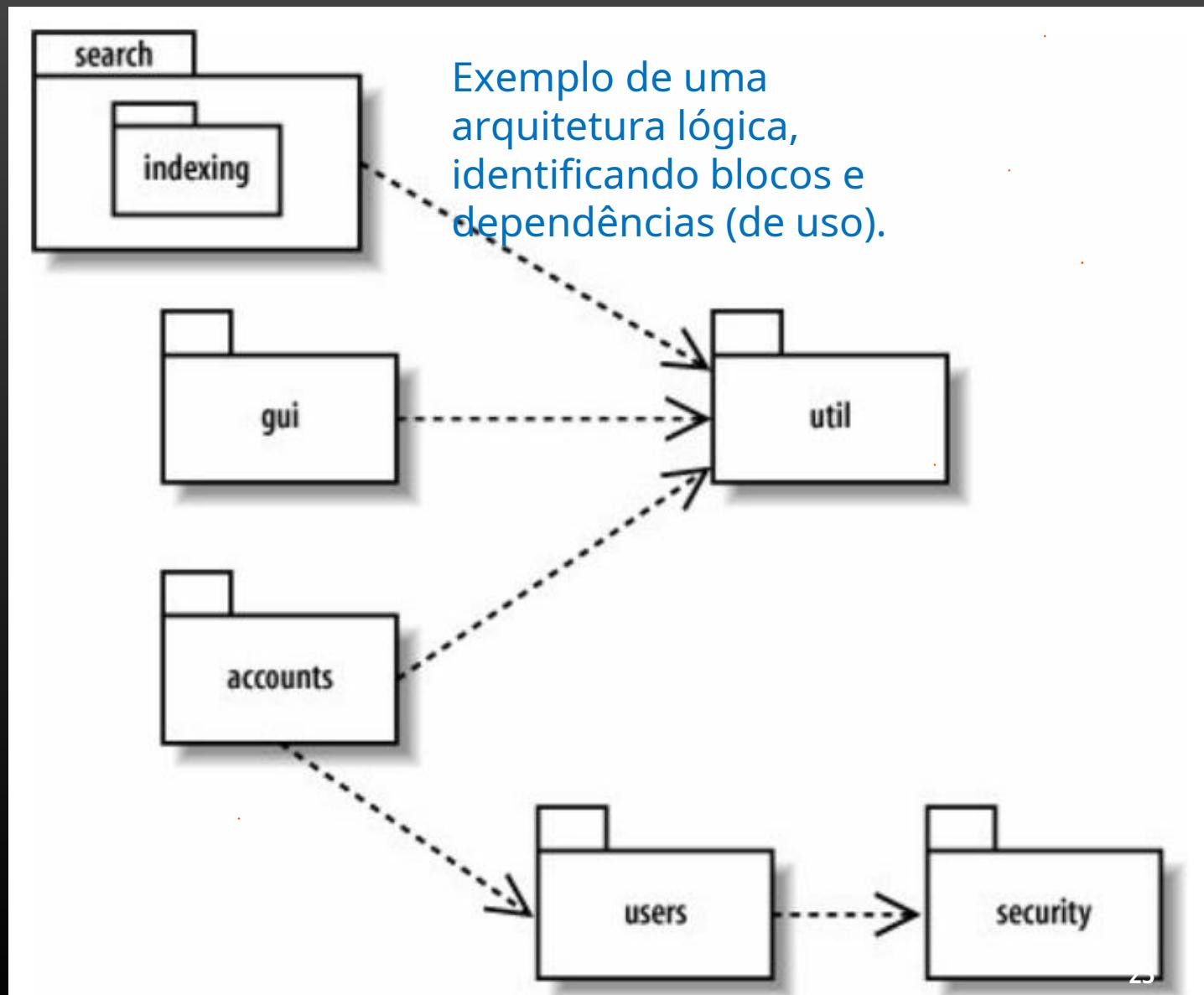
Documentar uma arquitectura é uma questão de documentar as vistas relevantes e depois acrescentar documentação que se aplica a mais do que uma visão.

Sec. 22.1

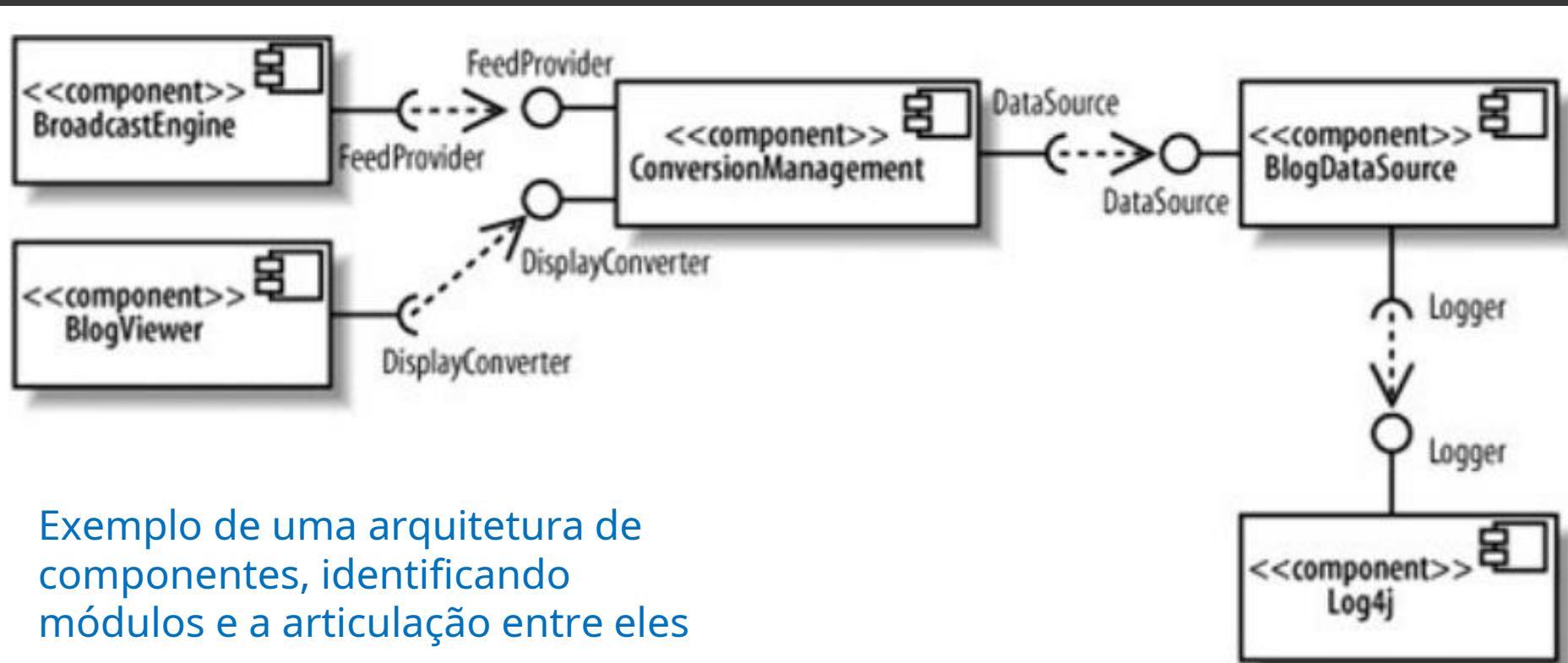


Views	Modules	C&C	Allocation
Key element	Modules , which are implementation units of software that provide a coherent set of responsibilities	Components : principal processing units and data stores. Connectors : pathways of interaction between components.	Software elements and environmental elements.
Relations	Is-part-of , which defines a part/whole relationship between the submodule (the part) and the aggregate module (the whole) Depends-on , defines a dependency between two modules Is-a , which defines a generalization/specialization relationship between a more specific module (the child) and a more general module (the parent)	Attachments : Components are associated with connectors to yield a graph.	Allocated-to : A software element is mapped (allocated to) an environmental element.
Sample usages	Blueprint for construction of the code Analysis of the impact of changes Planning incremental development Communicating the functionality of a system and the structure of its code base Supporting the definition of work assignments, implementation schedules, and budget information	Guide development by specifying the structure and behavior of runtime elements. Help reason about runtime system quality attributes, such as performance and availability.	For reasoning about performance, availability, security, and safety. For reasoning about the form and mechanisms of system installation.

Vista de módulos na UML (d. de pacotes)

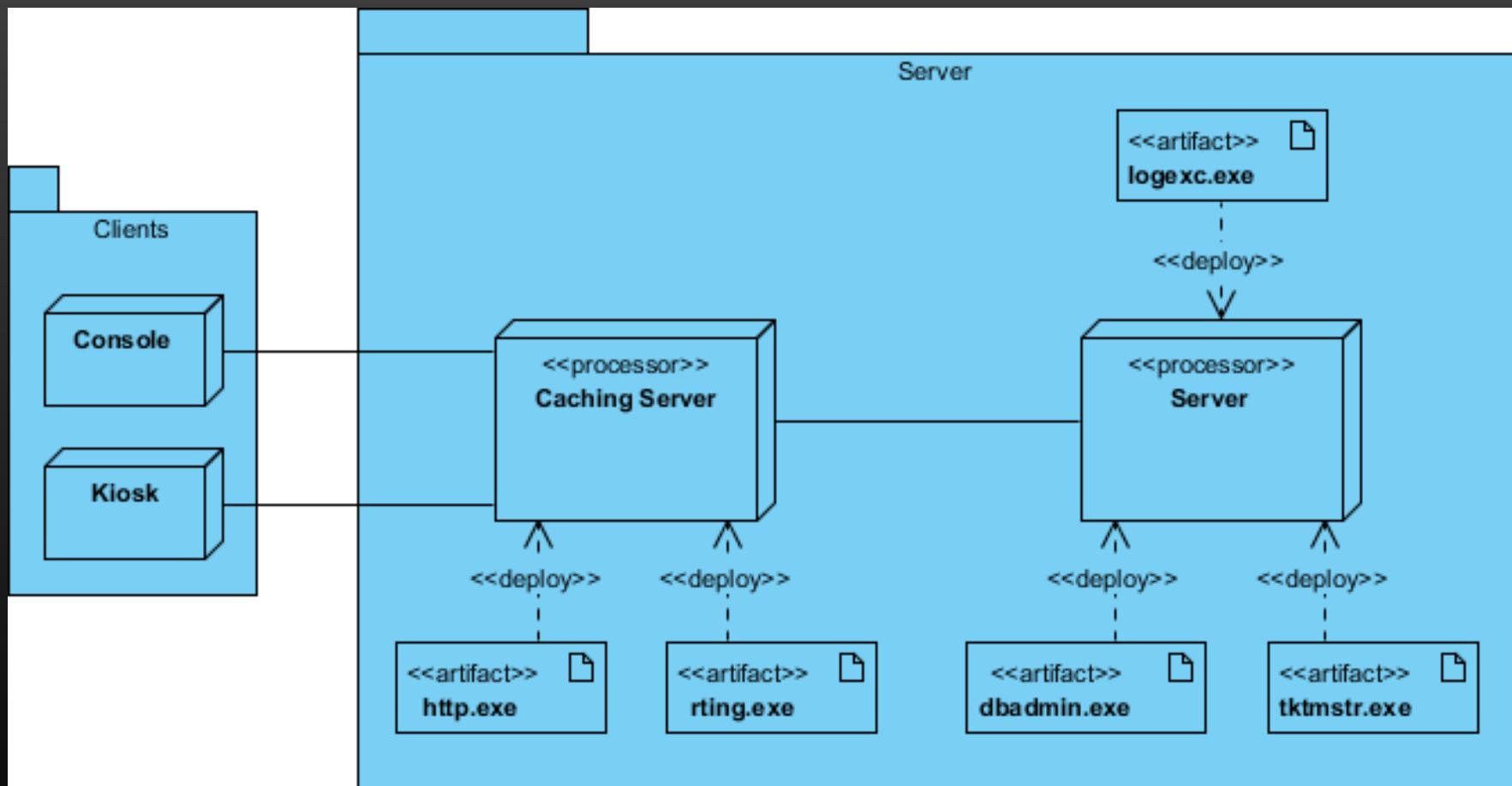


Vista C&C na UML (d. componentes)

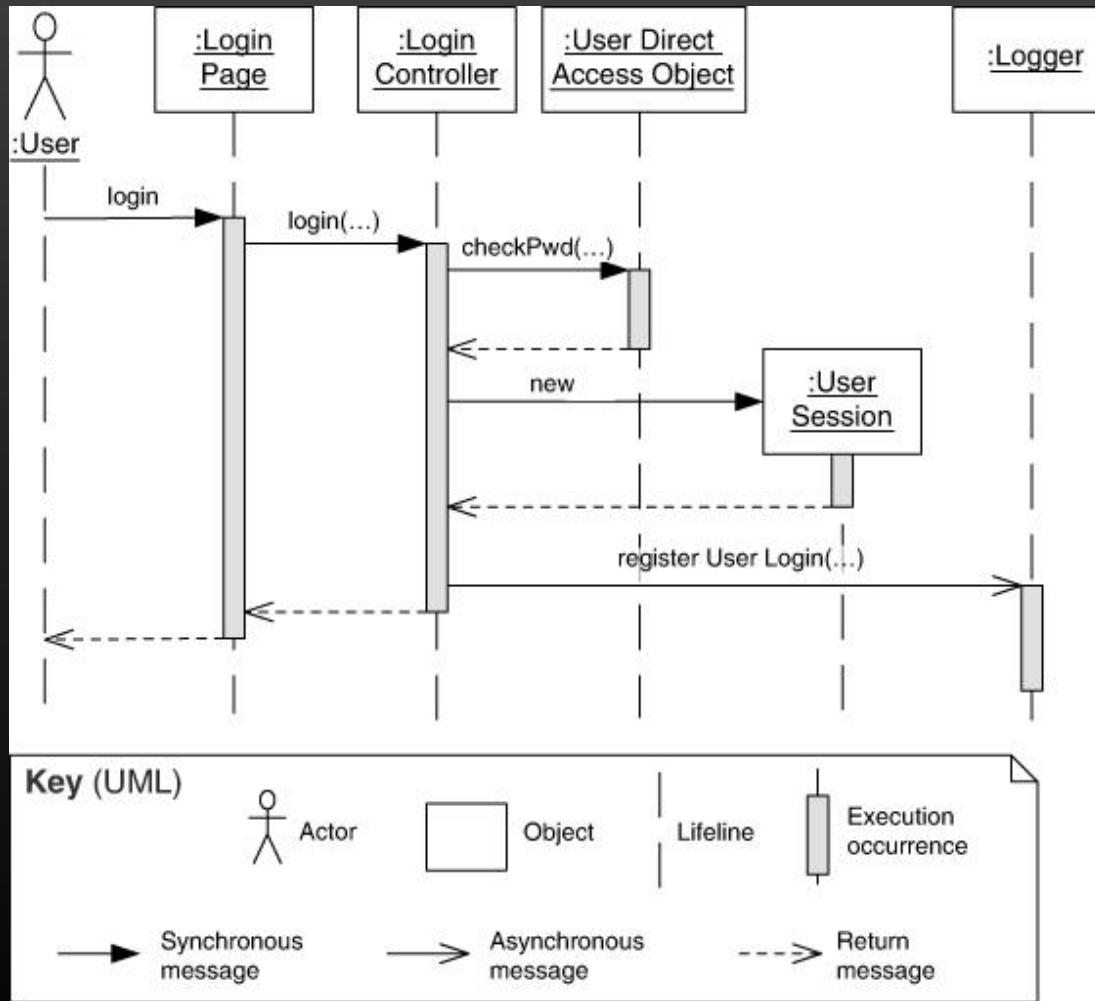


Exemplo de uma arquitetura de componentes, identificando módulos e a articulação entre eles

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



Vista comportamental (d. sequência)



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.

In some cases, architectural elements are composed in ways that solve particular problems. These compositions have been found to be useful over time and over many different domains, so they have been documented and disseminated. These compositions of architectural elements, which provide packaged strategies for solving some of the problems facing a system, are called patterns.

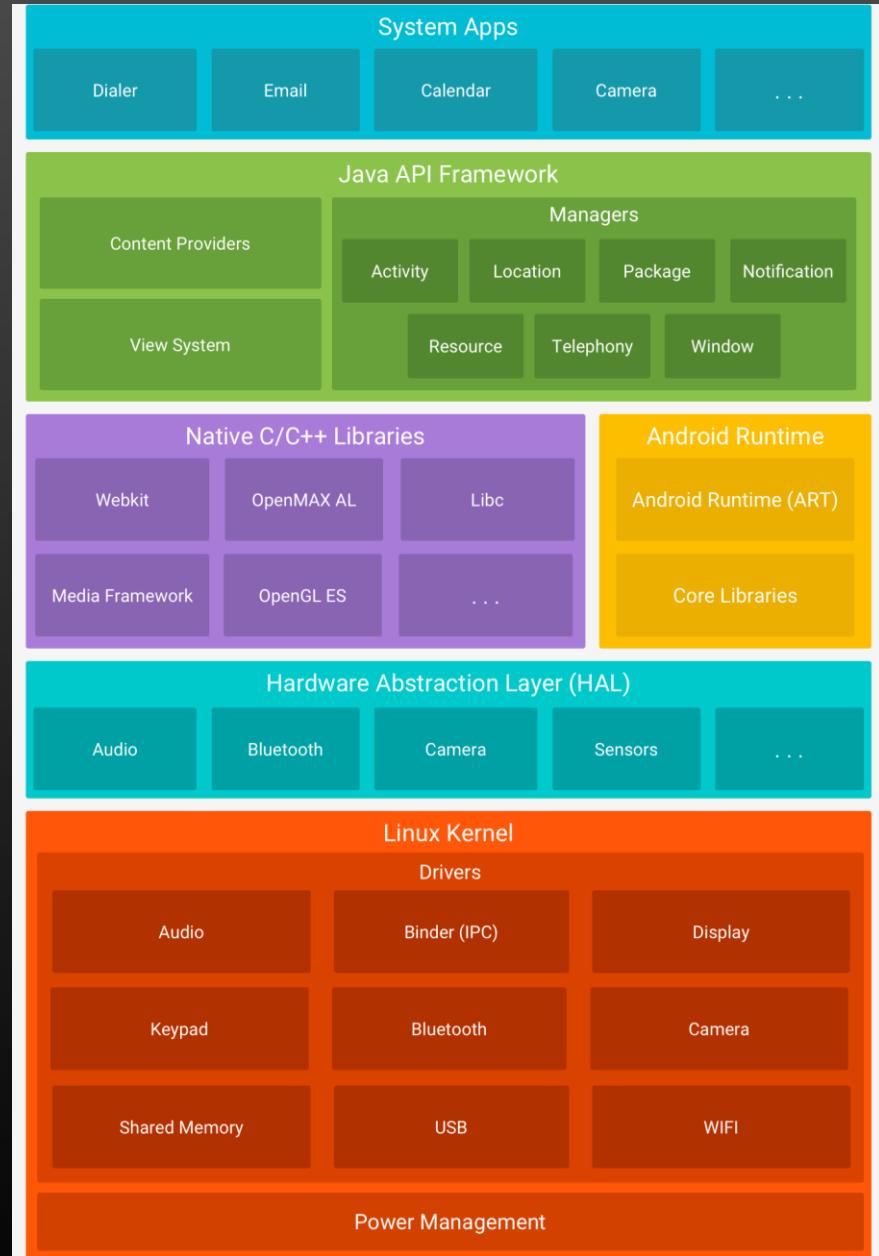
Exemplos:

- 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

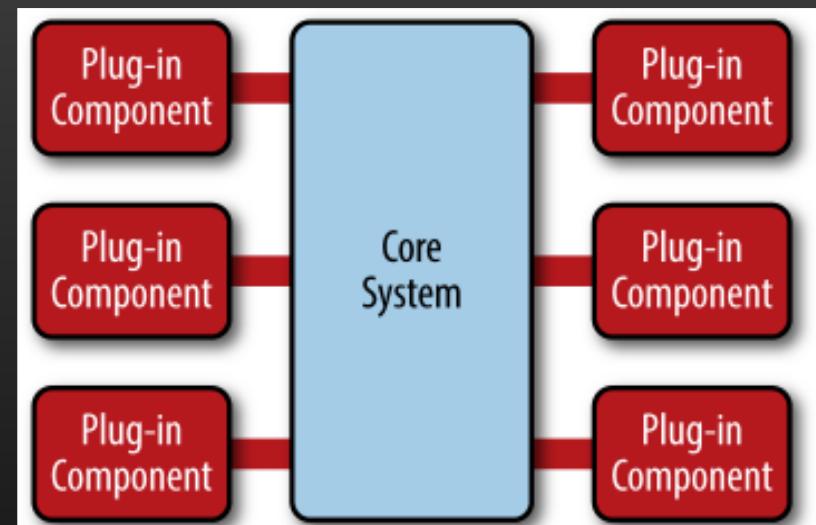


Arquitetura *Microkernel*

**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.

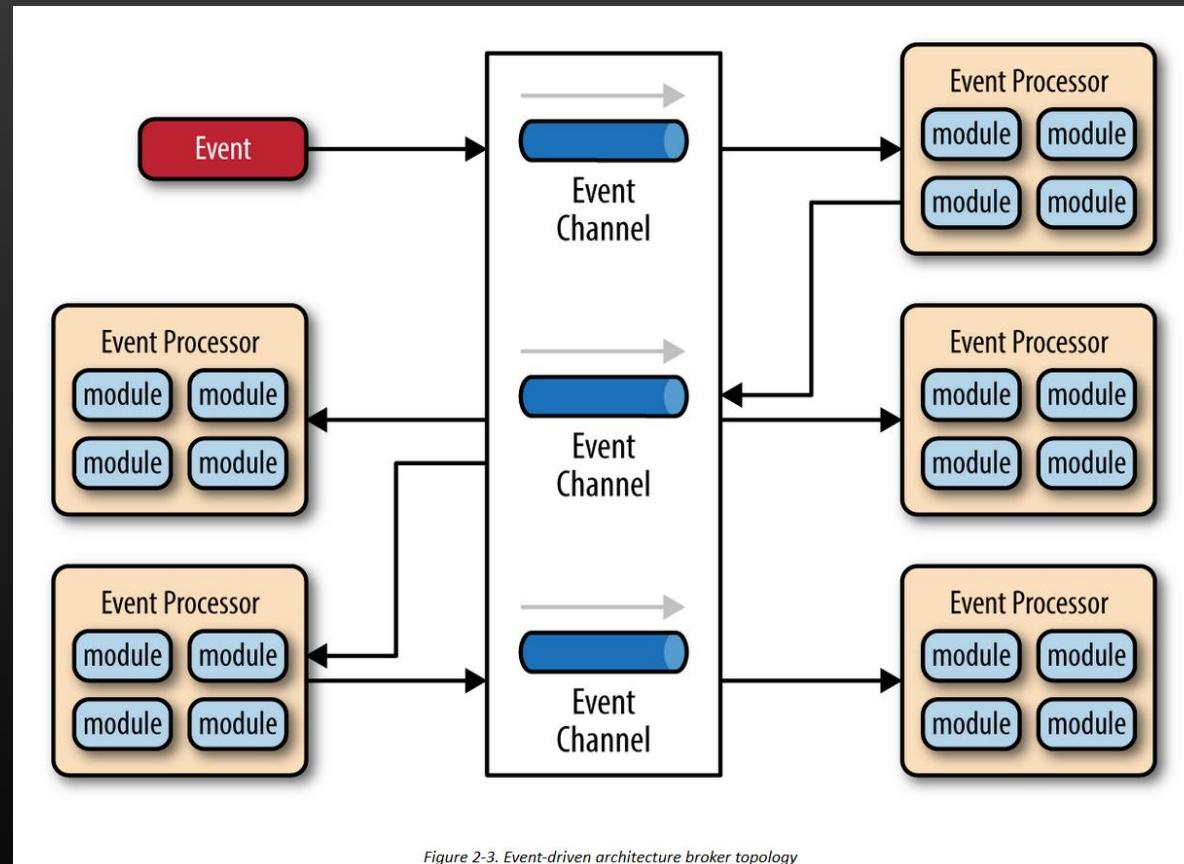


Figure 2-3. Event-driven architecture broker topology

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

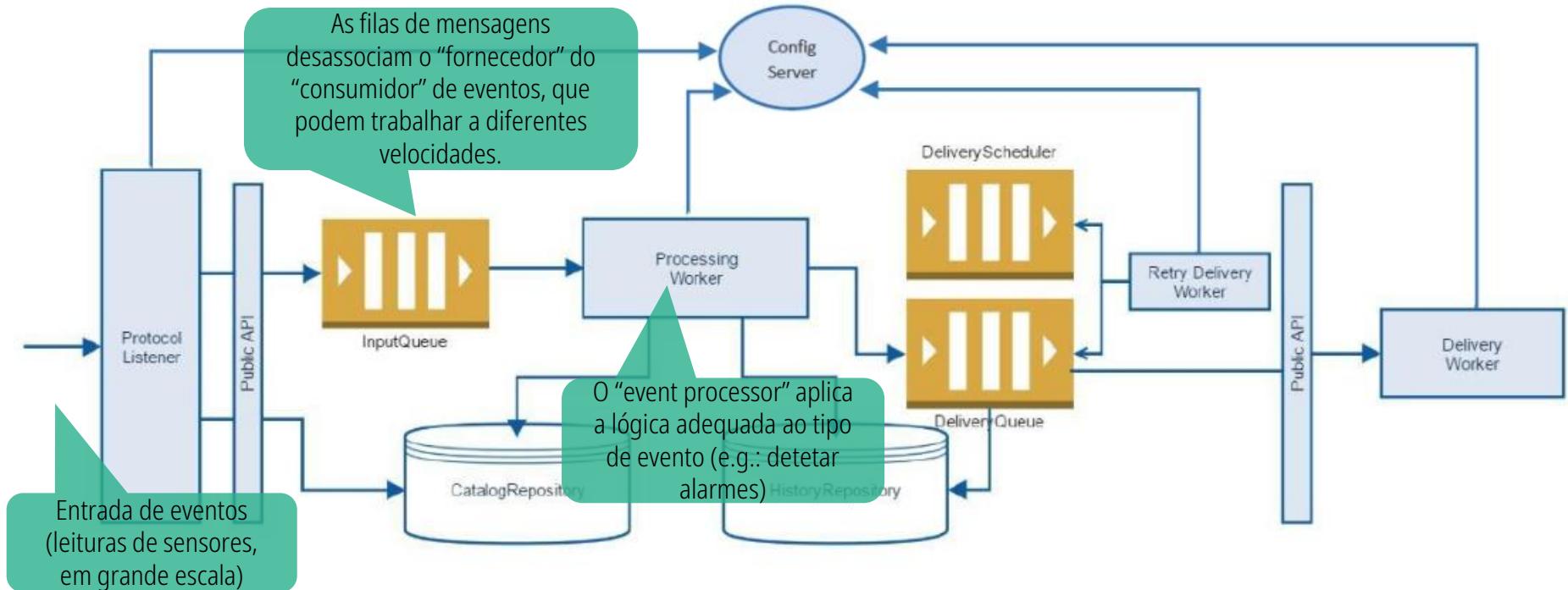


Figura 5.2: Arquitetura e componentes da SmartIOT.

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

Arquitetura por camadas

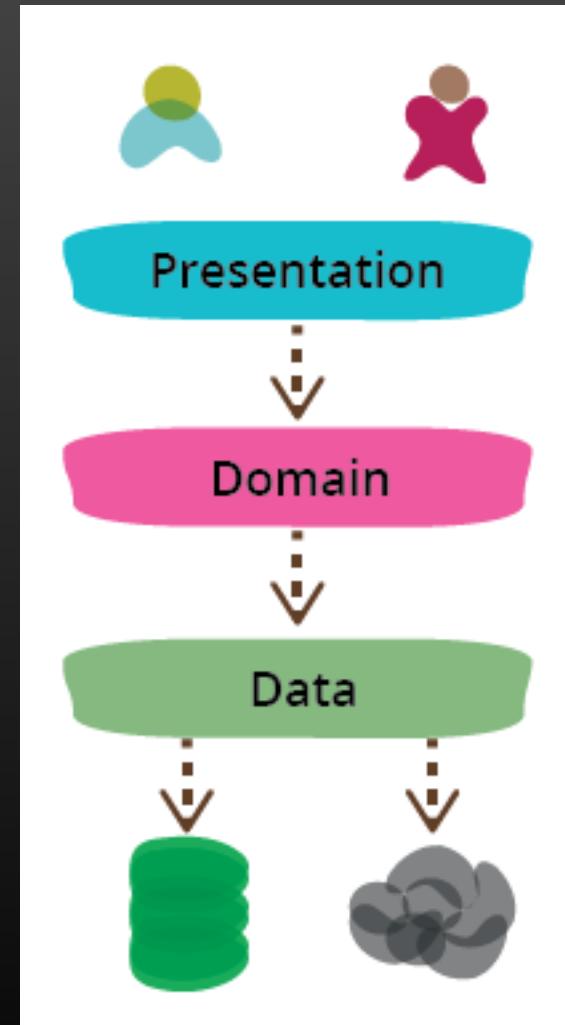
Divisão modular da solução de software em camadas/níveis de abstração.

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>

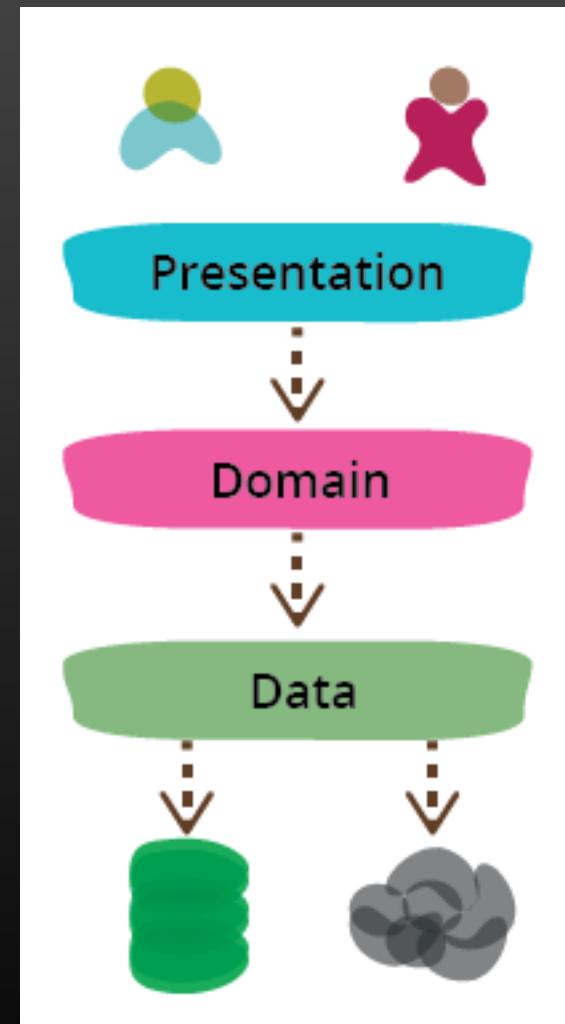
A arquitetura de 3 camadas

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)

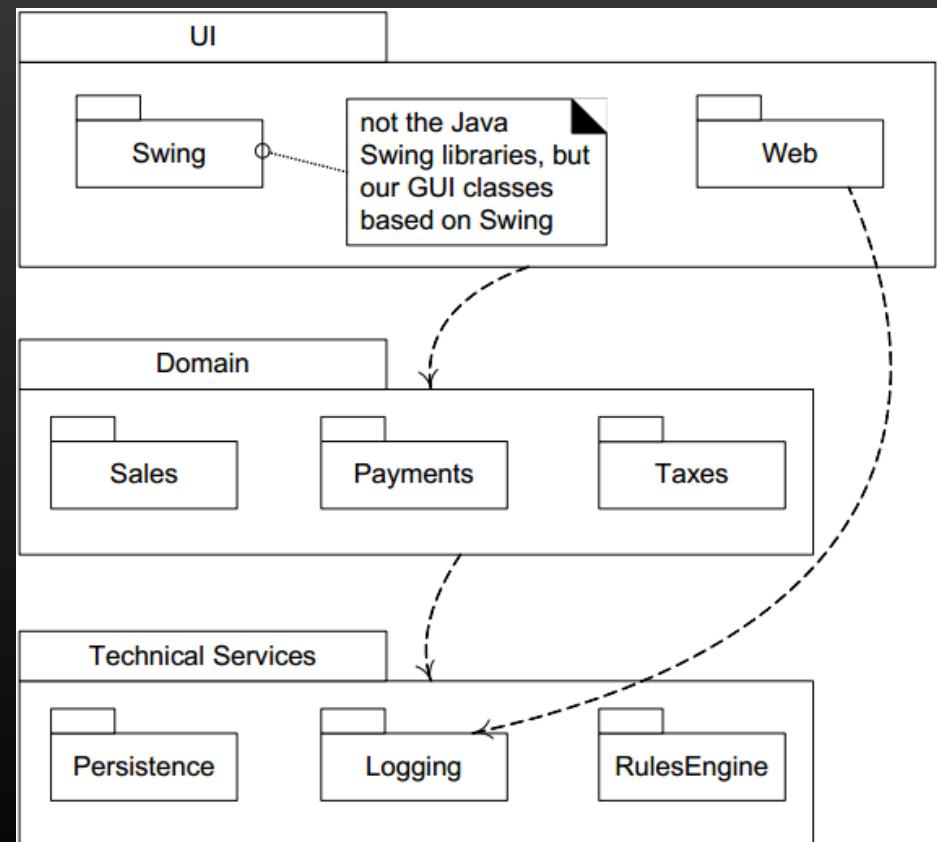
Camadas verticais:

- 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

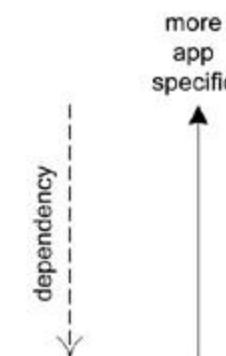
Business Infrastructure
(AKA Low-level Business Services)

(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

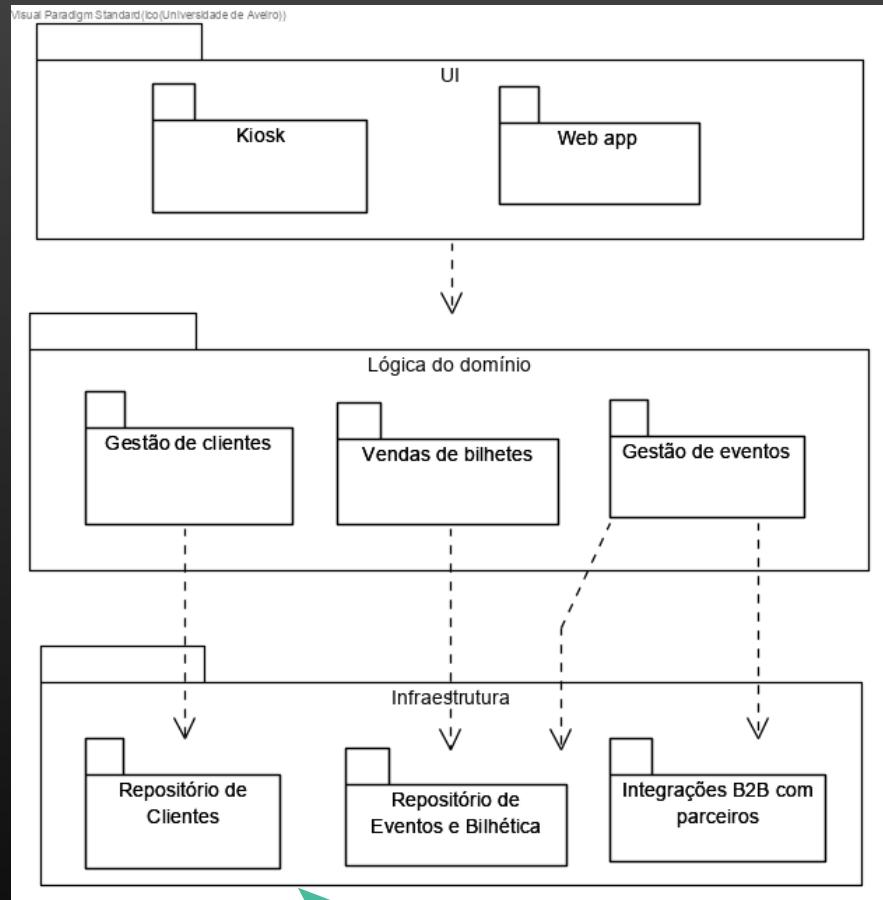
Foundation
(AKA Core Services, Base Services, Low-level Technical Services/Infrastructure)



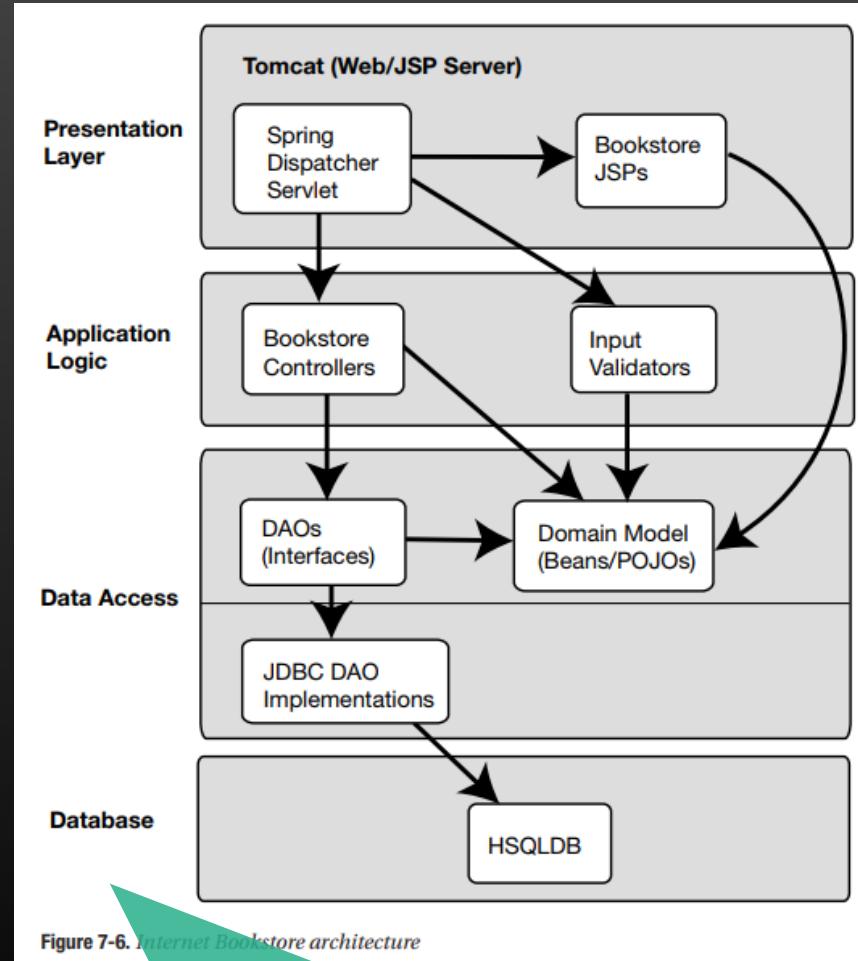
Algumas arquiteturas baseiam-se no modelo três camadas que ampliam para mostrar uma maior especialização de responsabilidades

width implies range of applicability →

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.
I Oliveira



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".

Referências

Core readings	Suggested readings
<ul style="list-style-type: none">• [Larman04] – Chap. 13	<ul style="list-style-type: none">• Bass, Clements, Kazman, "<u>Software Architecture in Practice</u>", 4th ed.• Fowler, "<u>Software Architecture Guide</u>"

ANÁLISE DE SISTEMAS

Testes e *Quality Assurance* na Construção

Ilídio Oliveira

v2022/05/21

Objetivos de aprendizagem

Identificar as actividades de validação e verificação no SDLC

Descrever as camadas da pirâmide de teste

Descrever o objecto dos testes de unidade, integração, sistema e aceitação

Explicar o ciclo de vida do TDD

Explicar como as actividades de QA são inseridas no processo de desenvolvimento numa abordagem clássica e em métodos ágeis

Relacionar os critérios de aceitação da história com testes ágeis

Explicar o conceito de especificações executáveis (e a relação com BDD).

Algumas ideias do desenvolvimento ágil

QUICK LOOK

What is it? Agile software engineering **combines a philosophy and a set of development guidelines.**

The philosophy encourages customer satisfaction and early incremental delivery of software; small, highly motivated project teams; informal methods; minimal software engineering work products; and overall development simplicity. The development guidelines stress **delivery over analysis and design** (although these activities are not discouraged), and active and continuous communication between developers and customers.

Who does it? Software engineers and other project stakeholders (managers, customers, end users) work together on an agile team—a team that is self-organizing and in control of its own destiny. An agile team fosters communication and collaboration among all who serve on it.

Why is it important? The modern business environment that spawns computer-based systems and software products is fast-paced and ever-changing. Agile software engineering represents a reasonable alternative to

conventional software engineering for certain classes of software and certain types of software projects. It has been demonstrated to deliver successful systems quickly.

What are the steps? Agile development might best be termed “software engineering lite.” The basic framework activities—communication, planning, modeling, construction, and deployment—remain. But they morph into a minimal task set that pushes the project team toward construction and delivery (some would argue that this is done at the expense of problem analysis and solution design).

What is the work product? Both the customer and the software engineer have the same view—the only really important work product is an operational “software increment” that is delivered to the customer on the appropriate commitment date.

How do I ensure that I’ve done it right? If the agile team agrees that the process works, and the team produces deliverable software increments that satisfy the customer, you’ve done it right.

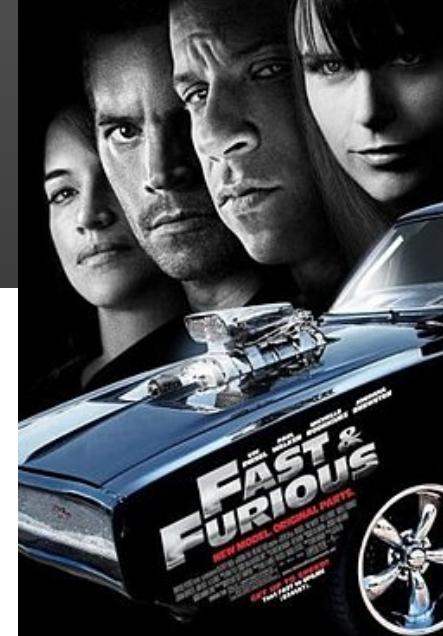
O dinamismo do mercado obriga a igual dinamismo do desenvolvimento. Especialmente quando os produtos de software assumem um papel fundamental na criação das vantagens competitivas.

A transformação digital (competitiva) obriga a uma eng.a de software competitiva.

Velocidade “furiosa”?

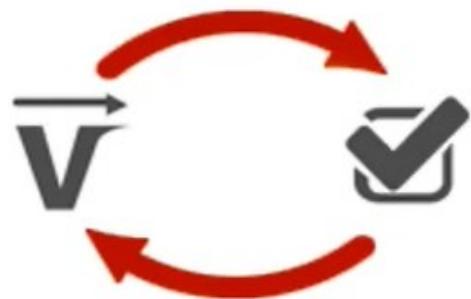


Greater speed may generate more risk and less quality...



... but

Velocity = Direction + Speed



quick feedback improves direction which improves quality which improves speed which improves feedback

Para avançar depressa e com segurança, é preciso preparar a “máquina”: mexer no próprio processo de engenharia de sw.

Crédito: Rui Gonçalves, Winning Consulting

É indispensável considerar as práticas que podem induzir ou medir a qualidade do produto

GARANTIA DE QUALIDADE DE SOFTWARE

conjunto de atividades (práticas) para controlar e monitorizar o processo de desenvolvimento de software para atingir os objetivos do projeto com um certo nível de confiança em termos de qualidade

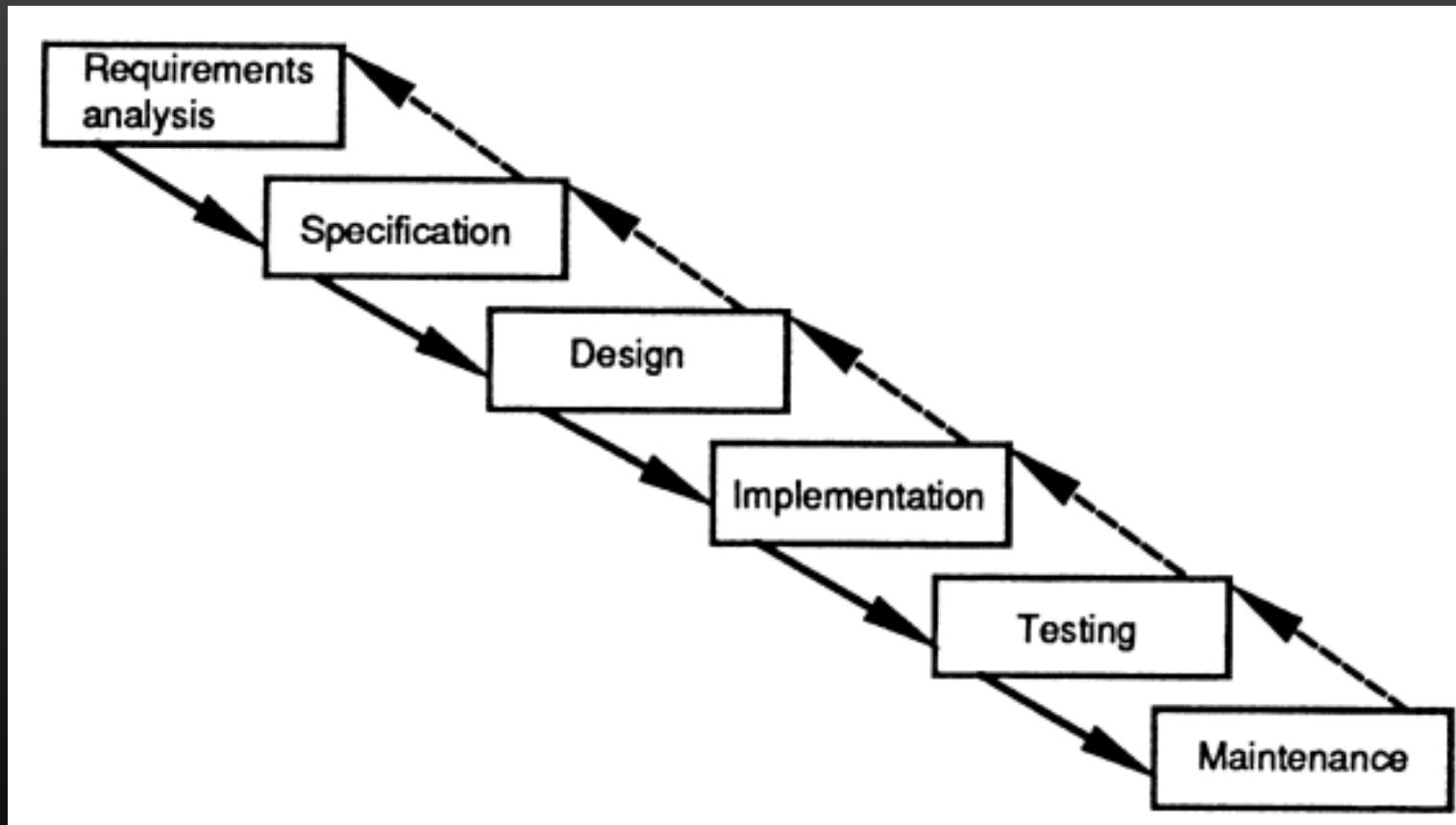
CONTROLO DE QUALIDADE DE SOFTWARE

Avalia se os produtos de software estão dentro dos padrões de qualidade definidos recorrendo a inspeções e diferentes tipos de testes

SQA != SQC

**A garantia de qualidade é parte integrante
de um processo de engenharia**

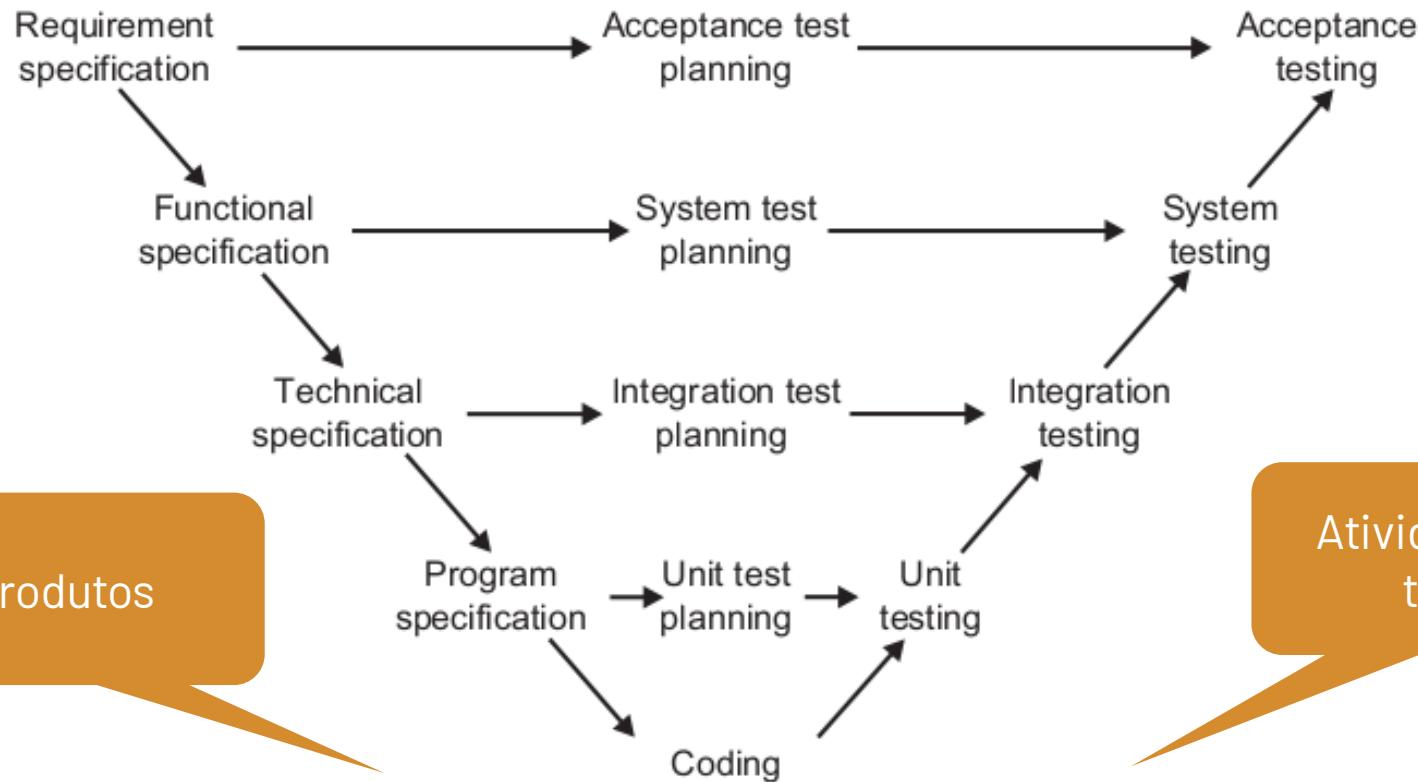
Abordagem de engenharia "clássica": Modelo waterfall



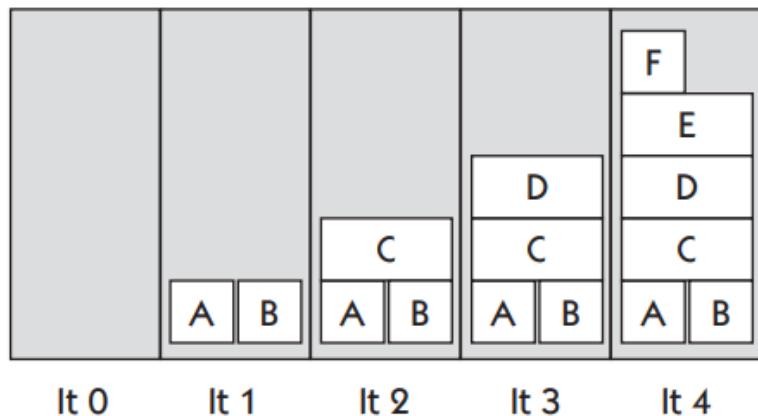
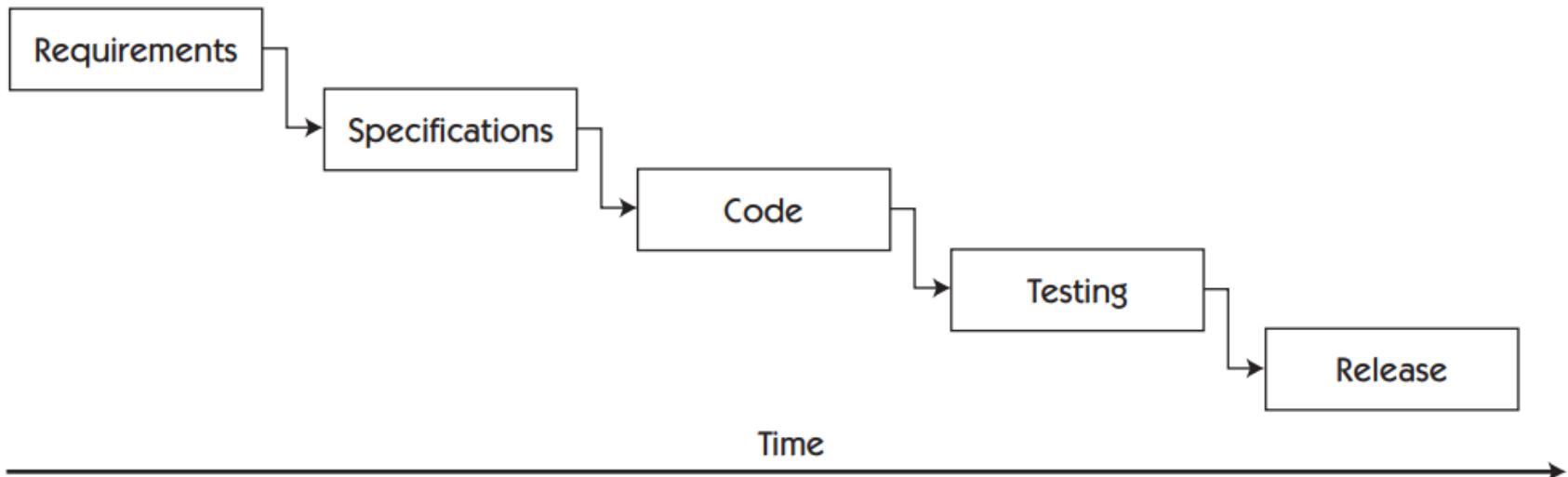
W. Royce, “Managing the Development of Large Software Systems,” *Proc. Westcon*, IEEE CS Press, 1970, pp. 328-339.

Ciclo de vida dos testes e o ciclo de vida do desenvolvimento do sw na abordagem sequencial

Figure 2.2 V-model for software development



Phased or gated—for example, Waterfall

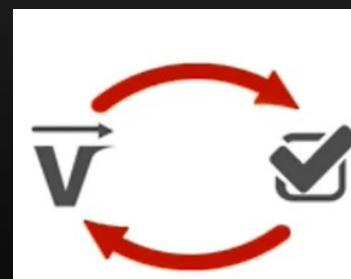
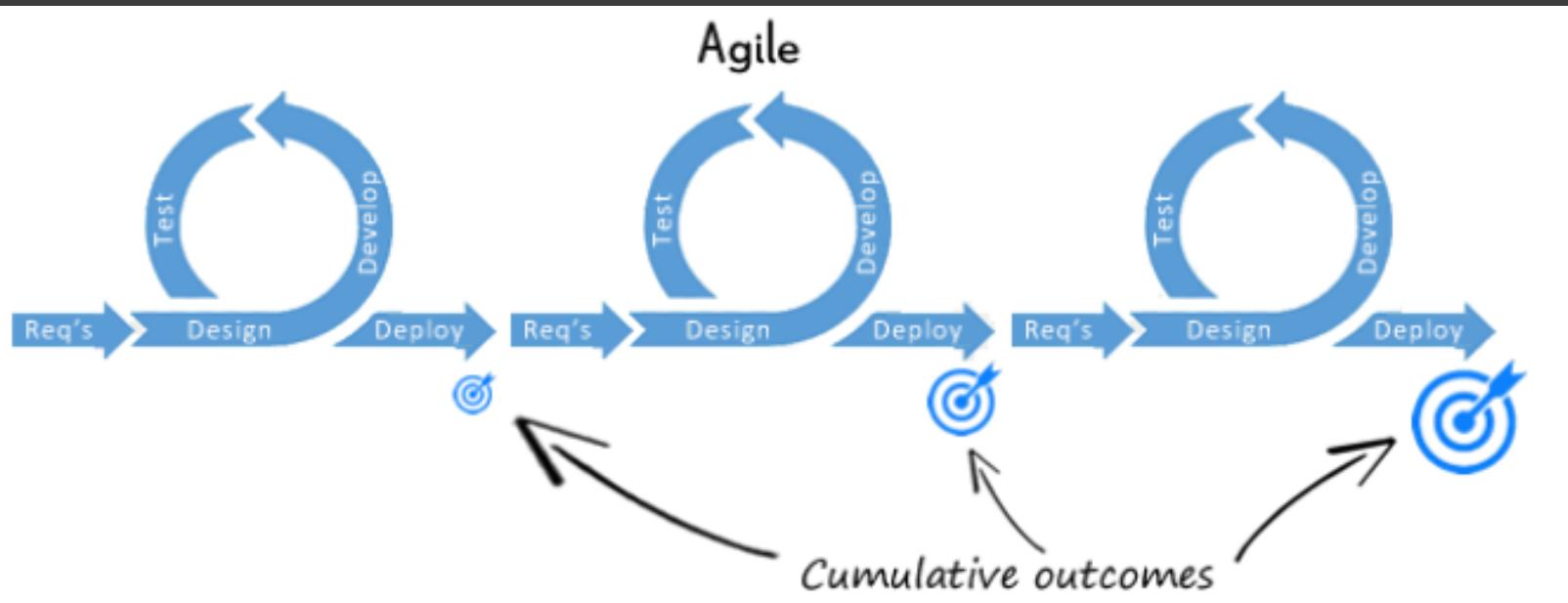


Agile:
Iterative & incremental

- Each story is expanded, coded, and tested
- Possible release after each iteration

Figure 1-4 Traditional testing vs. agile testing

Na abordagem ágil, a garantia de qualidade tem de ser aplicada em cada ciclo



quick feedback
improves direction
which improves quality which improves speed
which improves feedback

O papel dos testes de software

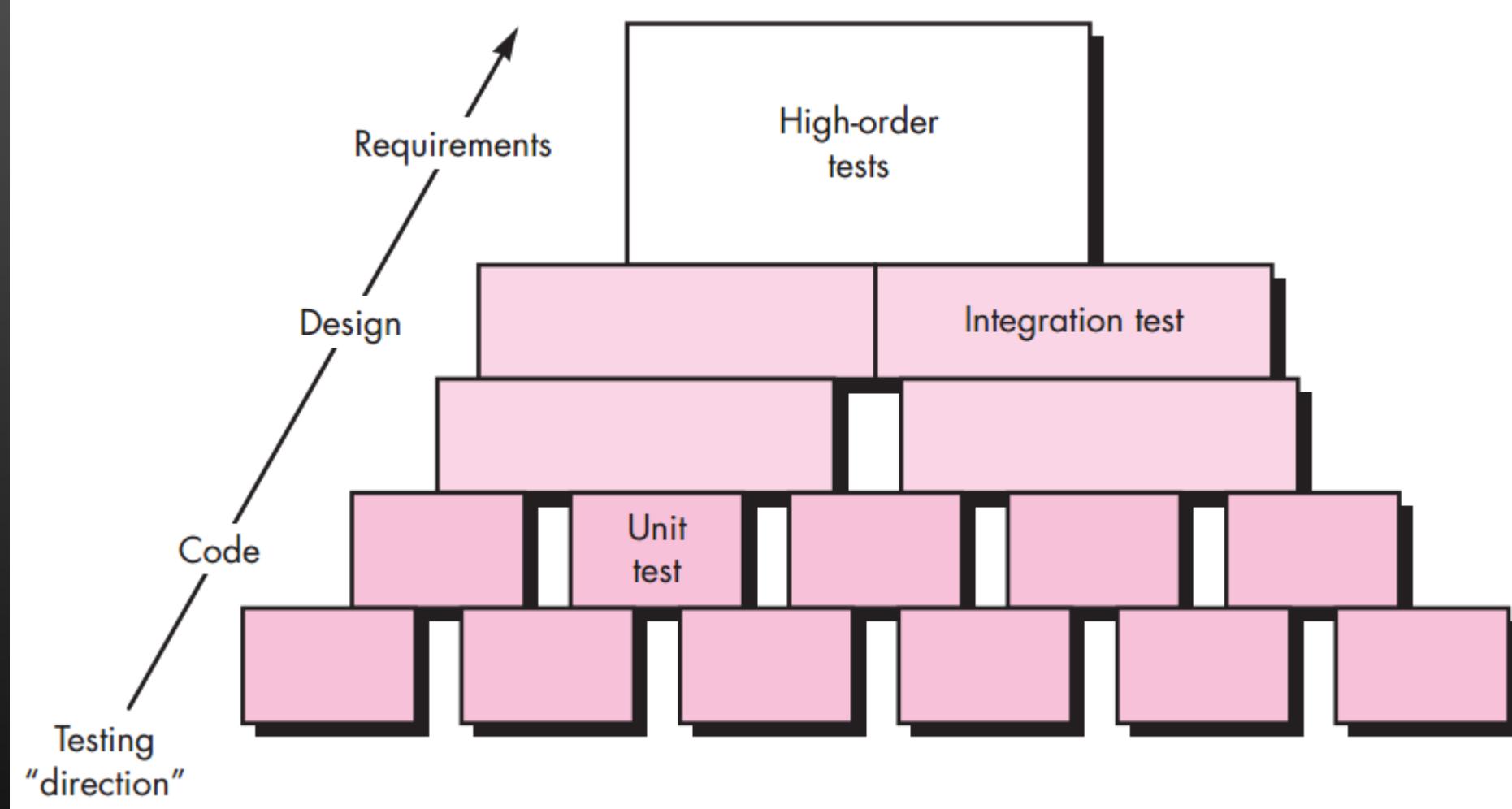
Verificação vs Validação

VERIFICAÇÃO: ESTAMOS A FAZER O SISTEMA DA FORMA CORRETA?

Verificar os produtos em relação às especificações
Verificar a consistência dos módulos
Comparar com as melhores práticas da indústria...

VALIDAÇÃO: ESTAMOS A FAZER O SISTEMA ADEQUADO?

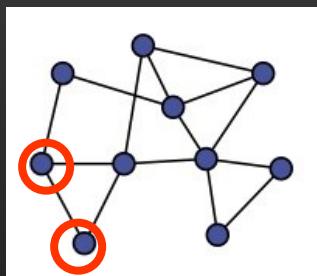
Avaliar os produtos em função das necessidades e expectativas dos utilizadores



Os testes começam ao nível dos componentes e progridem para “fora”

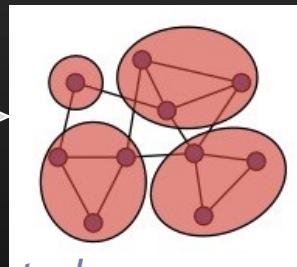
Different testing techniques are appropriate at different moments/software

Unit testing

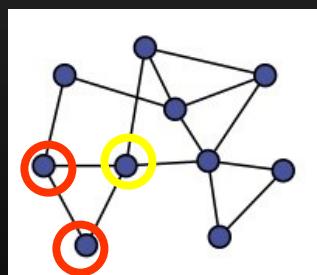


Each module does what it is supposed to do?

integration testing

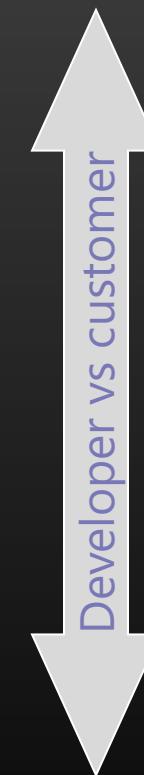


managing complexity



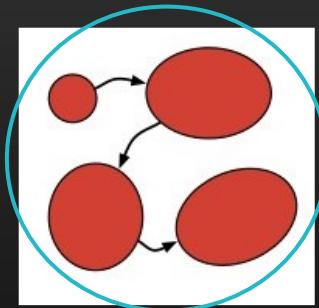
Do you get the expected results when the parts are put together?

Integration testing



Does the program satisfy the requirements?

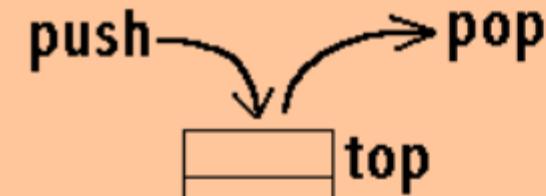
Acceptance / Functional Testing



System testing

The whole system functions as expected, in the target config?

Teste unitário: o contrato de uma *stack*



Operations

- `push(x)`: add an item on the top
- `pop`: remove the item at the top
- `peek`: return the item at the top (without removing it)
- `size`: return the number of items in the stack
- `isEmpty`: return whether the stack has no items

Unit test example: Verifying the unit contract

A stack is empty on construction

A stack has size 0 on construction

After n pushes to an empty stack, $n > 0$, the stack is not empty $\&\&$ its size is n

If one pushes x then pops, the value popped is x, the size is decreased by one.

If one pushes x then peeks, the value returned is x, but the size stays the same

If the size is n, then after n pops, the stack is empty and has a size 0

Popping from an empty stack does throw a
NoSuchElementException

Peeking into an empty stack does throw a
NoSuchElementException

For bounded stacks only, pushing onto a full stack does throw an
IllegalStateException

→ See also: Ray Toal's notes.

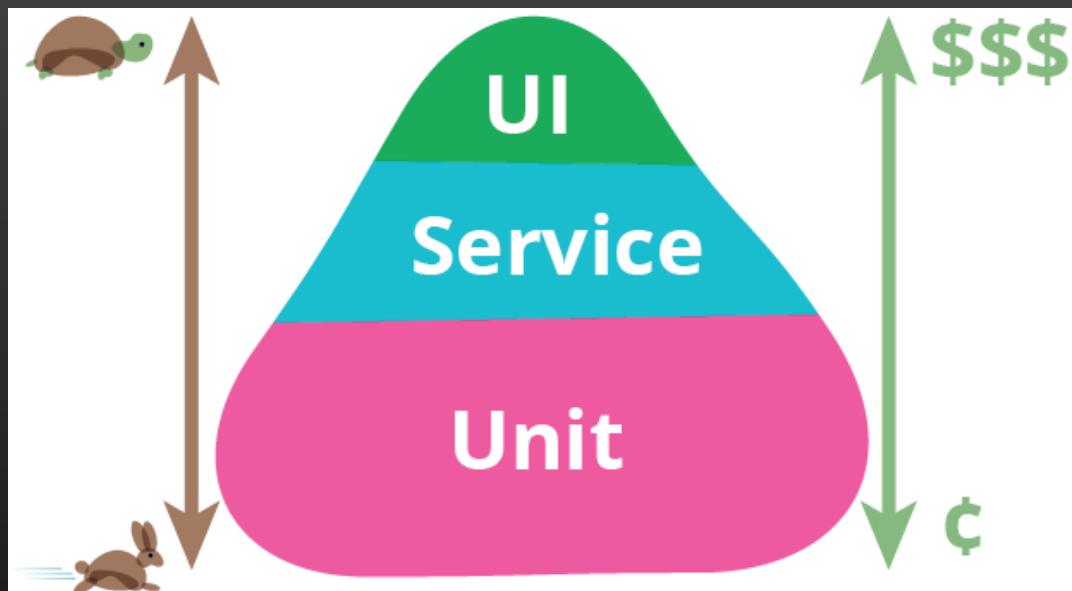
Exemplo de teste UI: Automatização de testes de aplicações Web com Selenium IDE

The screenshot shows a Firefox browser window with the Selenium IDE extension open. The extension's interface is visible on the left, displaying a project named "as1" and a list of test steps for a test case titled "comprar-livro". The test steps include commands like "open", "click", "type", and "assert title", targeting specific elements on the Bertrand Livreiros website.

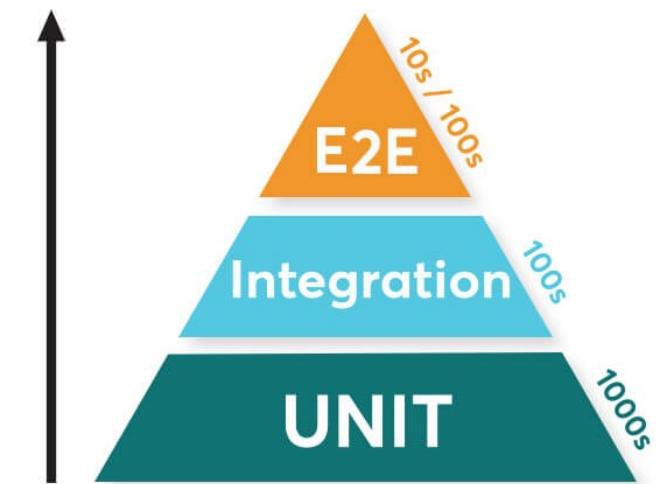
The main content area of the browser shows a product page for Isaac Asimov's book "Eu, Robô". The page includes a large image of Isaac Asimov, his name, and a brief biography mentioning he was born in 1920 and emigrated to the United States.

Command	Target	Value
1 ✓ open	https://www.bertrand.pt/	
2 ✓ click	id=form-searchform-palavra	
3 ✓ type	id=form-searchform-palavra	isaac asimov
4 ✓ click	css=.item-auto-complete-item	
5 ✓ click	css=.product-title	
6 ✓ assert title	Isaac Asimov - Bertrand Livreiros - livraria Online	
7 ✓ assert text	css=.product-title	Eu, Robô

Pirâmide dos testes



<https://martinfowler.com/bliki/TestPyramid.html>



<https://www.blazemeter.com/blog/agile-development-and-testing-an-introduction> 18

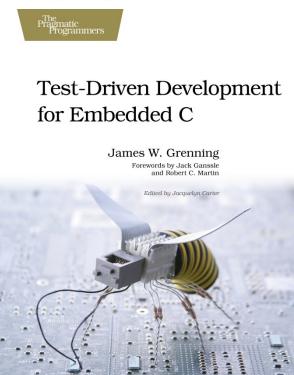
Test-driven development

Desenvolvimento conduzido pelos testes.

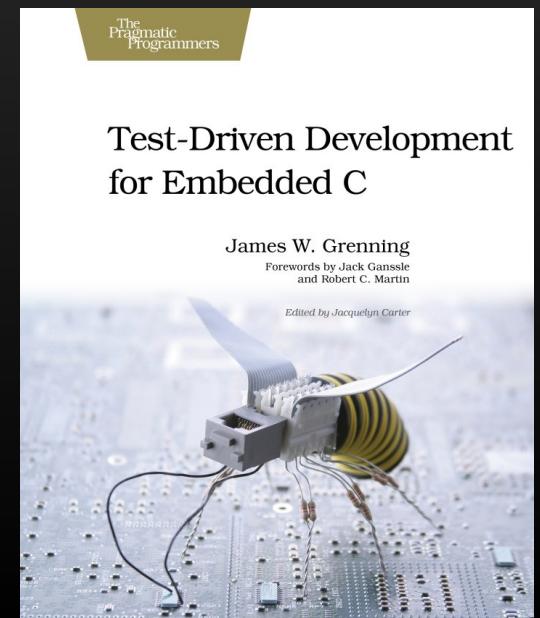
Debug Later Programming

We've all done it—written a bunch of code and then toiled to make it work. **Build it and then fix it.** Testing was something we did after the code was done. It was always an afterthought, but it was the only way we knew.

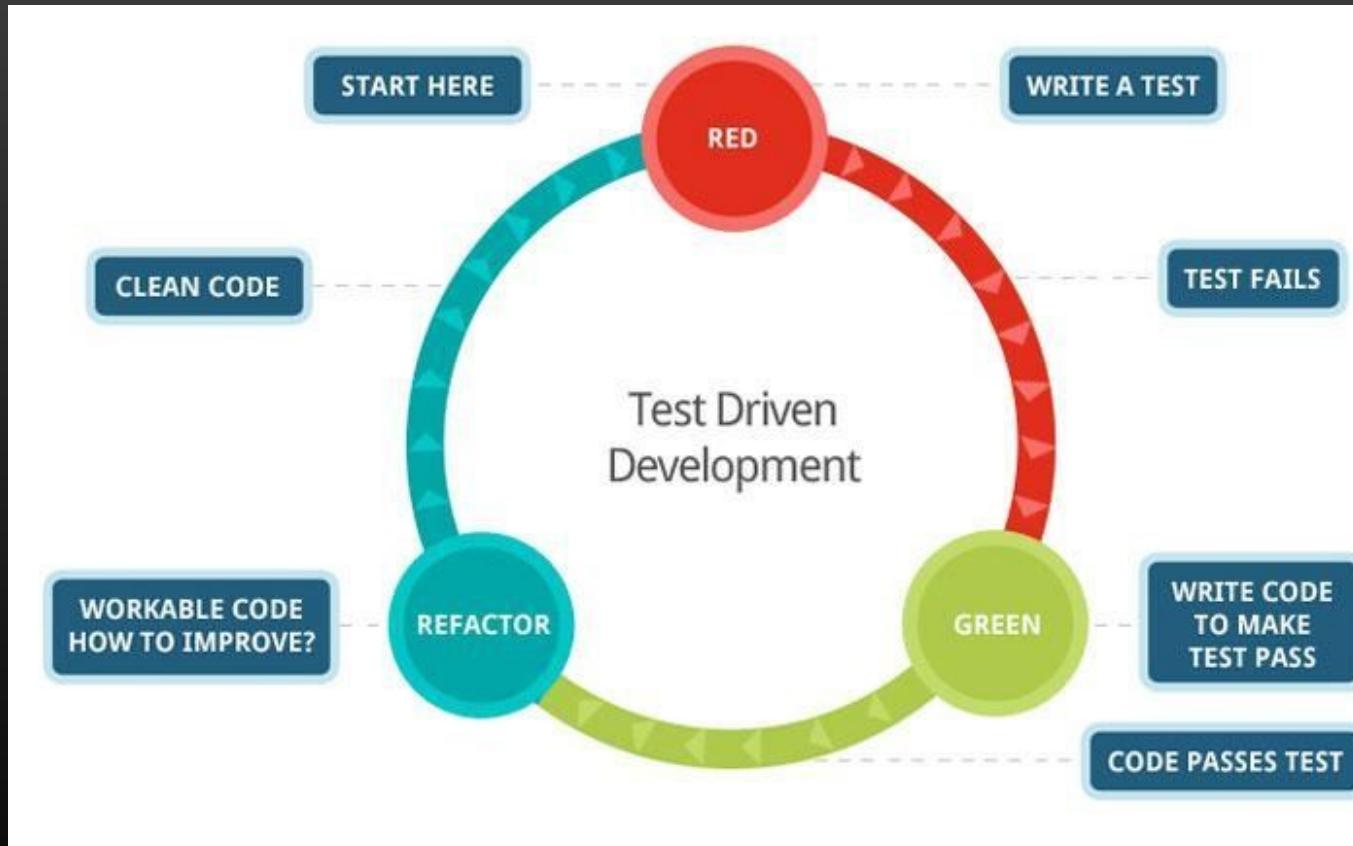
We would spend about half our time in the unpredictable activity affectionately called *debugging*. Debugging would show up in our schedules under the guise of test and integration. It was always a source of risk and uncertainty. Fixing one bug might lead to another and sometimes to a cascade of other bugs. We'd keep statistics to help predict



Test-Driven Development is a technique for building software incrementally. Simply put, no production code is written without first writing a failing unit test. Tests are small. Tests are automated. Test-driving is logical. Instead of diving into the production code, leaving testing for later, the TDD practitioner expresses the desired behavior of the code in a test. The test fails. Only then do they write the code, making the test pass.



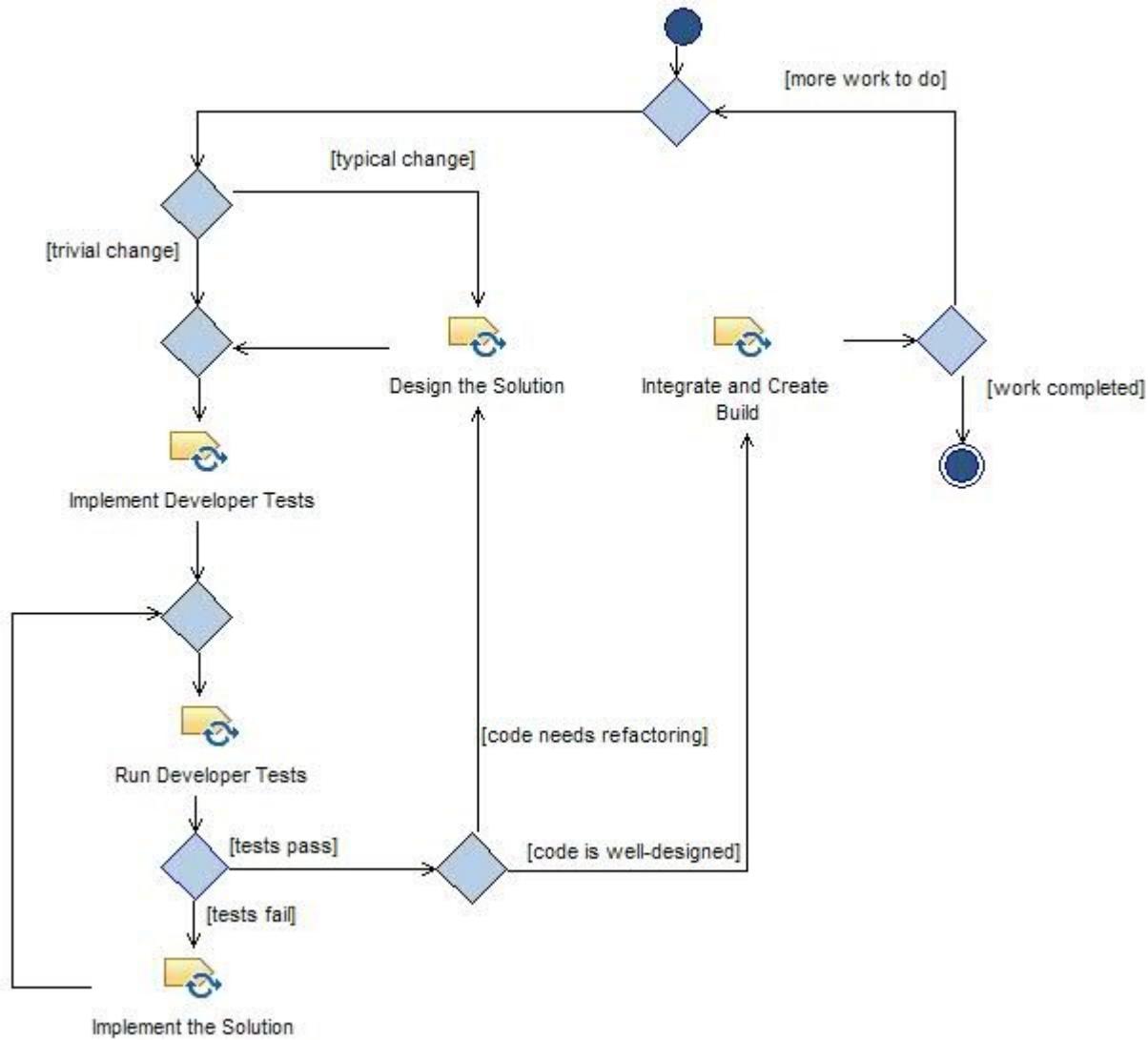
TDD: Test Driven Development



At the core of TDD is a repeating cycle of small steps known as the TDD microcycle. Each pass through the cycle provides feedback answering the question, does the new and old code behave as expected? The feedback feels good. Progress is concrete. Progress is measurable. Mistakes are obvious.

The steps of the TDD cycle in the following list are based on Kent Beck's description in his book *Test-Driven Development* [Bec02]:

1. Add a small test.
2. Run all the tests and see the new one fail, maybe not even compile.
3. Make the small changes needed to pass the test.
4. Run all the tests and see the new one pass.
5. Refactor to remove duplication and improve expressiveness.



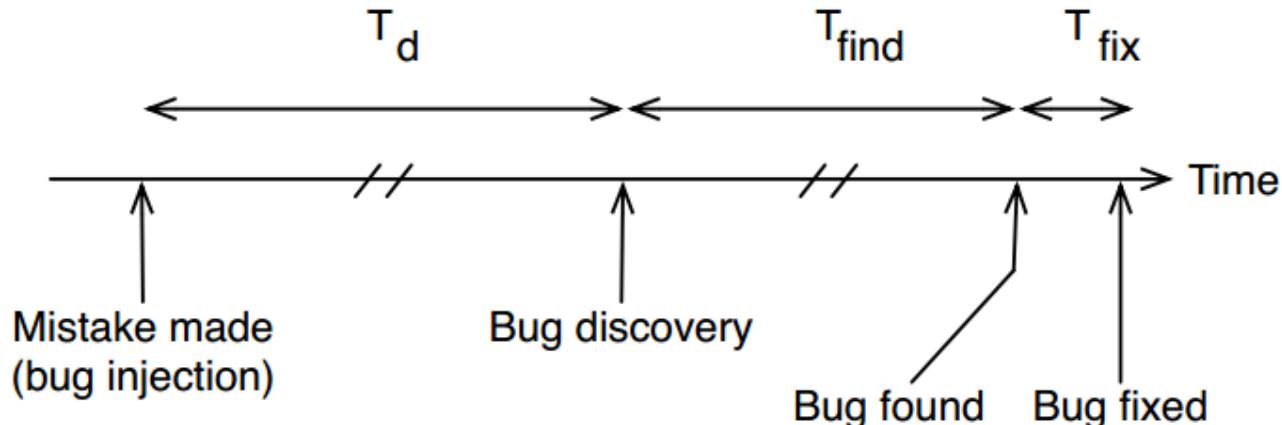


Figure 1.1: Physics of Debug-Later Programming

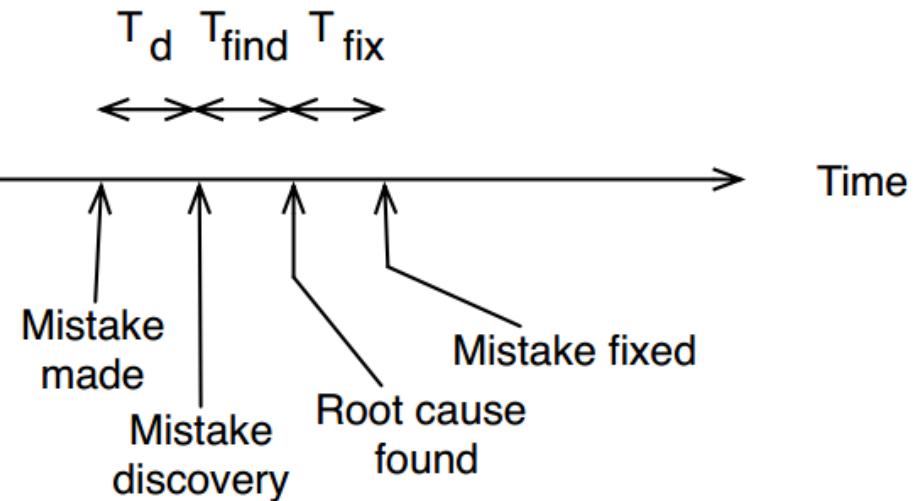


Figure 1.2: Physics of Test-Driven Development

The benefits of TDD: an investigation

A summary of selected empirical studies of test-driven development: industry participants*

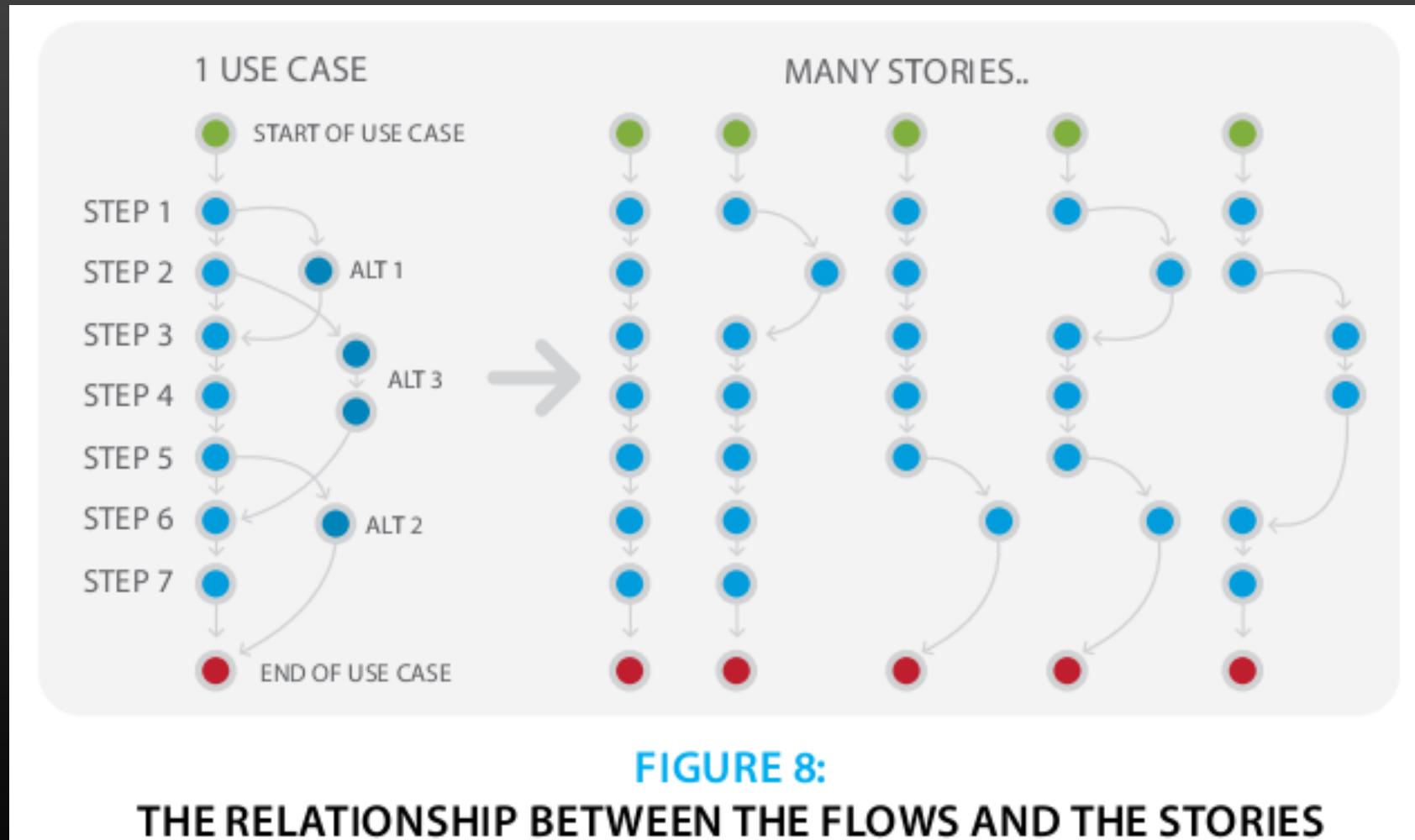
Family of studies	Type	Development time analyzed	Legacy project?	Organization studied	Software built	Software size	No. of participants	Language	Productivity effect
Sanchez et al. ⁶	Case study	5 years	Yes	IBM	Point-of-sale device driver	Medium	9–17	Java	Increased effort 19%
Bhat and Nagappan ⁷	Case study	4 months	No	Microsoft	Windows networking common library	Small	6	C/C++	Increased effort 25–35%
	Case study	≈7 months	No	Microsoft	MSN Web services	Medium	5–8	C++/C#	Increased effort 15%
Canfora et al. ⁸	Controlled experiment	5 hours	No	Soluziona Software Factory	Text analyzer	Very small	28	Java	Increased effort by 65%
Damm and Lundberg ⁹	Multi-case study	1–1.5 years	Yes	Ericsson	Components for a mobile network operator application	Medium	100	C++/Java	Total project cost increased by 5–6%



Story Testing

Executable Use Cases

Stories, use cases, scenarios



A story and tests

Title (one line describing the user story)

Narrative:

As a [role]

I want [feature]

So that [benefit]

Acceptance Criteria: (present in user stories)

Scenario 1: Title

Given [context]

And [some more context].

When [event]

Then [outcome]

And [another outcome]...

Scenario 2:

Frank Can Add Another Person as a Friend

ID #115218319 Close

STORY TYPE Feature

POINTS Unestimated

STATE Start Unscheduled

REQUESTER RJ Ryan Jones

OWNERS <none> +

FOLLOW THIS STORY (1 follower)

Updated: less than a minute ago

DESCRIPTION (edit)

As Frank I want to add a friend I searched for to my friend network so that I can see their posts, they can see my posts and I can direct message them

GIVEN I have searched for a friend's name
WHEN I select "Add Friend" next to my friend's name
THEN my friend's name should appear in my friend list on my homepage

Dev Notes: The added friend needs to be added to the Frank's friends in database
Design Notes: Attached are mocks for the button and placement

LABELS

add friend |x| individual user |x|

→ Principles for user stories content

Acceptance criteria: Given, When, Then style

Structured syntax
(Gherkin) to describe a feature (for testing):

Feature: what

Scenario: some determinable business situation

Given: preparation/setup (e.g.: required data)

- And...

When: the set of actions (execute).

- And...

Then: specifies the expected resulting state (assert).

- And...

Feature: Check Covid-19 stats

Scenario: Obtain world data

Given I am in the Home page

When I check to obtain world data

And I choose the date 2021-01-01

And I click 'Submit' under the covid section

Then I should receive covid stats from the 'world'

And the date 'at' field should be 2021-01-01

And no other date field should appear

Scenario: Obtain country data

Given I am in the Home page

When I uncheck to obtain world data

And I choose stats after 2021-12-12

And I choose the country 'Portugal'

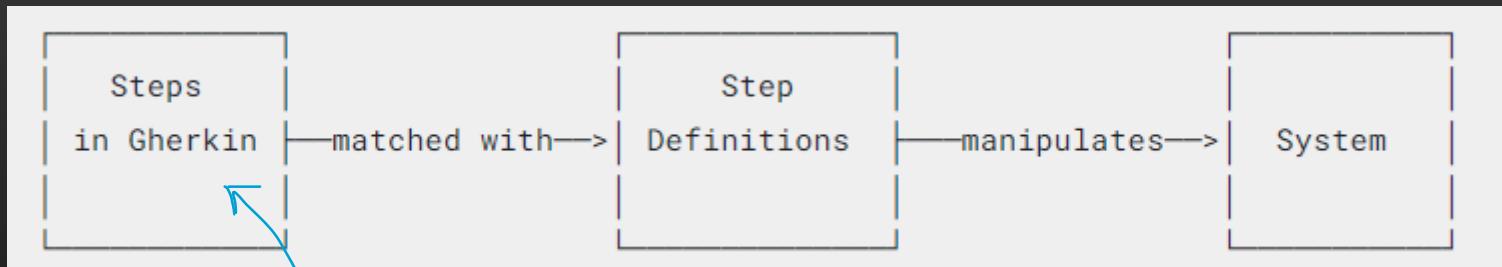
And I click 'Submit' under the covid section

Then I should receive covid stats from 'Portugal'

And the date 'after' field should be 2021-12-12

And no other date field should appear

Cucumber framework



```
Feature: Check Covid-19 stats

  Scenario: Obtain world data
    Given I am in the Home page
    When I check to obtain world data
      And I choose the date 2021-01-01
      And I click 'Submit' under the covid section
    Then I should receive covid stats from the 'world'
      And the date 'at' field should be 2021-01-01
      And no other date field should appear

  Scenario: Obtain country data
    Given I am in the Home page
    When I uncheck to obtain world data
      And I choose stats after 2021-12-12
      And I choose the country 'Portugal'
      And I click 'Submit' under the covid section
    Then I should receive covid stats from 'Portugal'
      And the date 'after' field should be 2021-12-12
      And no other date field should appear
```



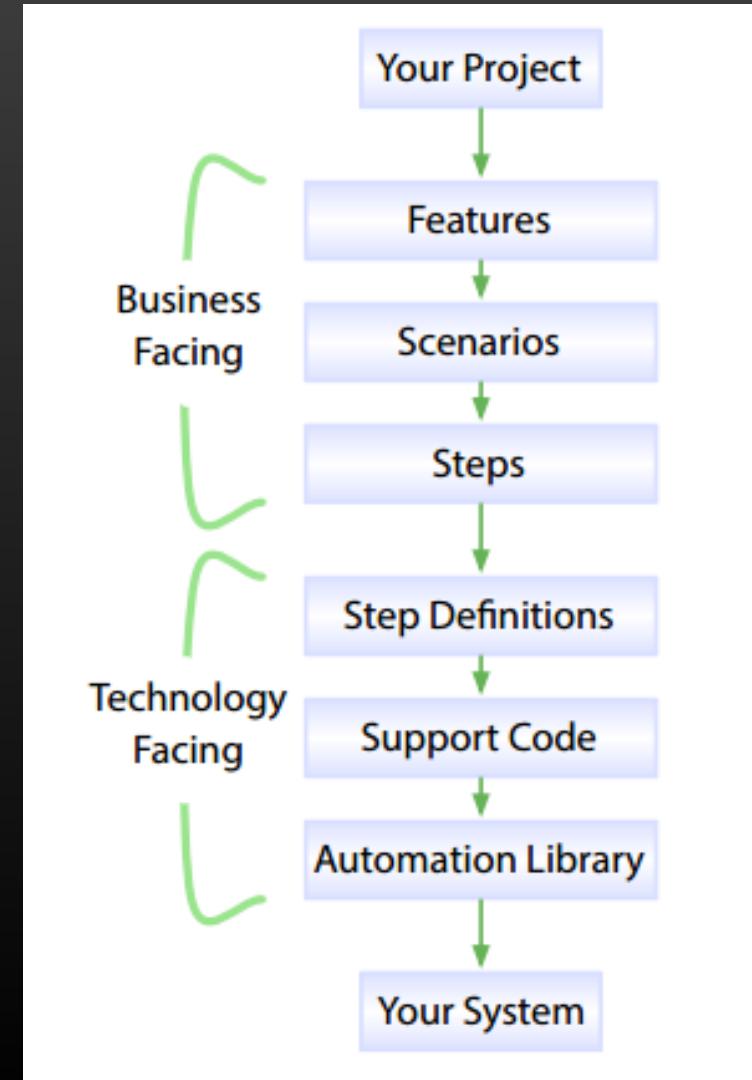
Cucumber reads specifications from plain-language text files called **features**, examines them for **scenarios** to test.

Each scenario is a list of **steps** for Cucumber to work through.

Along with the features, you give Cucumber a set of **step definitions**, which map the business-readable language of each step into code to carry out whatever action is being described by the step.

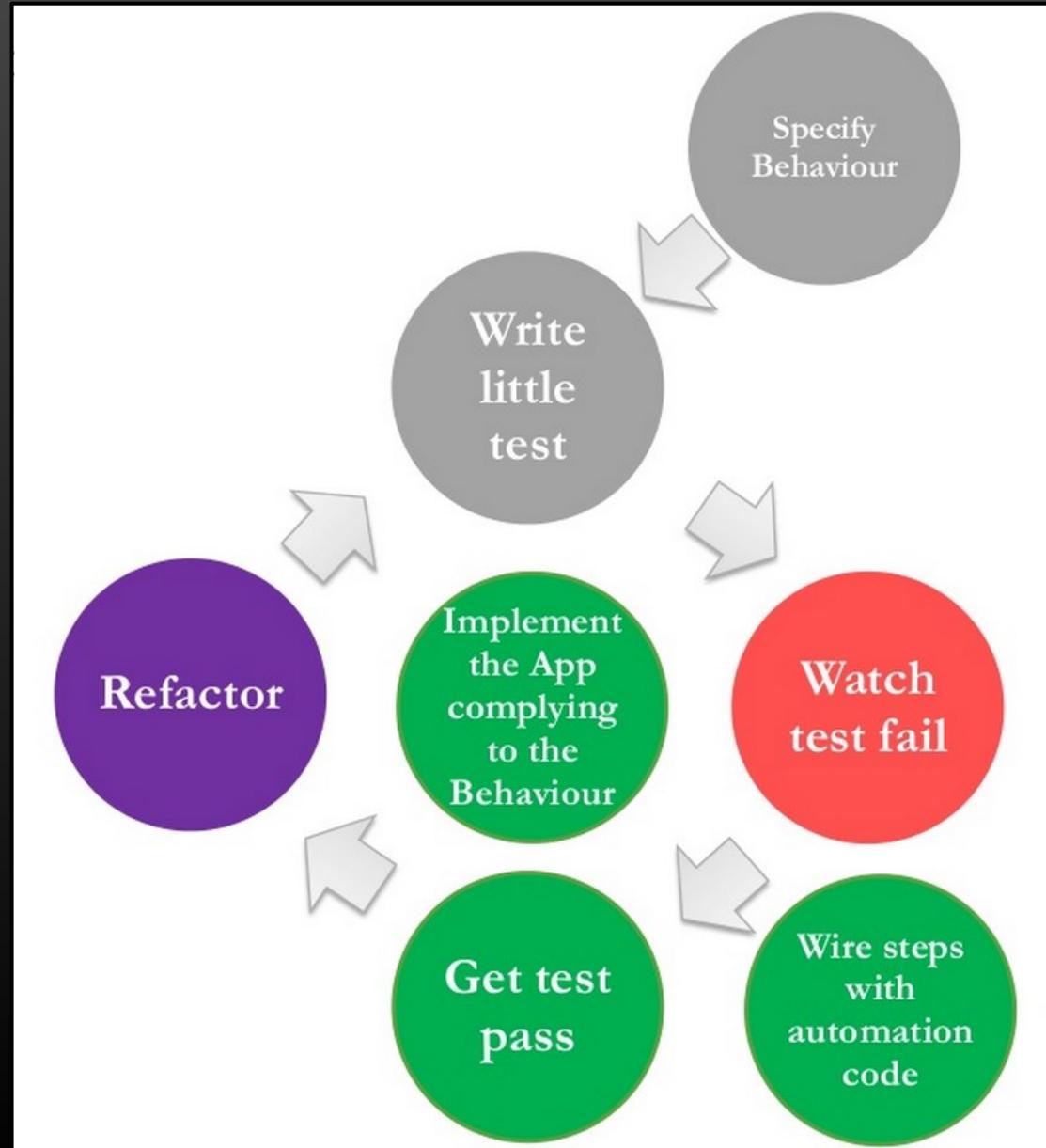
The **step definition** itself will probably just be one or two **lines of code**, specific to the domain of your application.

Sometimes that may involve using an **automation library**, like the browser automation library Selenium.



(cucumber demo)

BDD: Behaviour-driven development



Credit: Nalin Goonawardana

Uncomplicate TDD and BDD

by JEFF NYMAN posted on 17 SEPTEMBER 2017

BDD IS AN ABSTRACTION OF TDD

So here's how I see it. The key value of TDD is that at each step of the way, you have demonstrably relevant working software as well as an itemized set of what we can call "executable specifications" that illustrate aspects of behavior. And we do this at the appropriate level of abstraction. Which takes us to BDD.

BDD is really just the addition of business concerns to the technical concerns that we deal with in TDD. BDD wasn't a reaction to TDD, as is often stated. BDD was simply an approach that let us move up the abstraction chain *as people became more comfortable with TDD*.

Simply put, BDD is about writing those conditions we talked about in the context of scenarios such that they will tell us the kind of behavior change they affect. A good barometer for adding a scenario might be asking if the scenario you are writing would be worth explaining to a business stakeholder. And you can frame that by asking what value it provides them. What aspect of the overall user experience is being captured in that scenario?

And not just "failure" in a singular sense, but failure modes based on the likely sensitivities of

<http://testerstories.com/2017/09/uncomplicate-tdd-and-bdd/>

context of your immediate work.

References

Core readings	Suggested readings
<ul style="list-style-type: none">• “<u>Story testing</u> - executable use cases - for embedded systems”, J. Grenning	<ul style="list-style-type: none">• [Dennis]- Chap. 12.• [Pressman] – Chap. 17 (“Software Testing Strategies)

We create the technology to connect the world

NOKIA



June 2022

Agenda

- Nokia Software @ Portugal
- Agile Software Development
- Scrum
- How We Work
- Wellbeing & Social Responsibility

Nokia Software @ Portugal

Highly engaged engineers

Startup company can-do spirit

High level of OSS domain knowledge and experience

Proven experience in customer cooperation for High Value Applications

R&D collaboration with local Nokia Services and Care Centers

Active cooperation with top universities



Nokia

- 15+ years of SW
- Software Engineering
- Telecommunications
 - 5G, Network Slicing
 - Radio, Core, IP, Transport
 - NW and Service Automation
- Agile, Lean, Scrum, Kanban, SAFe
- DevOps
- Telco Cloud
- Big Data, Analytics
- Artificial Intelligence & Machine Learning
- WebApps
- CSF
- UX and Interaction Design
- Technical Writing
- OpenSource, ONAP

Providing a full range of services on a global scale



Sample of supported customers



Key Operation data (Yearly)

- 80.000 sites managed remotely
- 174.000 sites operated 24/7
- 2,3 M of working orders executed
- 12.800 support tickets
- 400.000 SW upgrades

CNS @ Nokia Portugal

A team of 750 and counting...



Core Networks

- CN Global Care (#130)
- CN Global Services (#100)



Business Applications

- BA Digital Operations (#190)
- BA Solutions & Services (#60)
- BA Analytics (#40)
- BA Security (#<25)



Enterprise Solutions

- EnSo SIS (#< 50)
- SIS SI & Delivery
- SIS Solution Support
- SIS Solution Management



Cloud & Cognitive Networks

- C&CS OD GDC EU (#80)
- C&CS SP (#12)
- C&CS BOA
- C&CS NFaaS
- C%CS MPS

Total HC **230**

Total HC **320**

Total HC **<50**

Total HC **106**

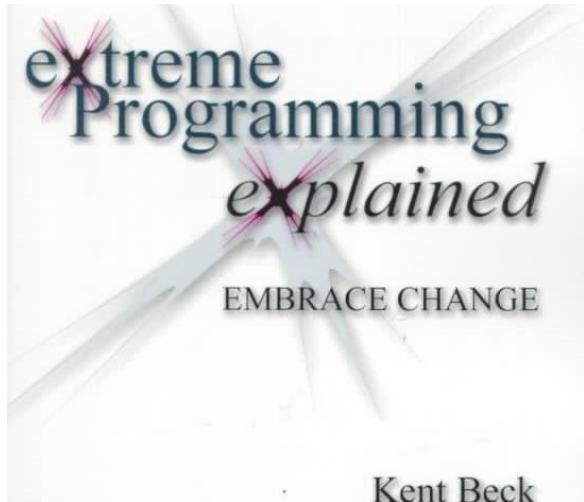
Global Business Center Total HC **<50**

Nokia Software in Portugal –NAC



Agile Software Development

XP Programming and the case of an adjective that became a noun





How the customer explained it



How the project leader understood it



How the engineer designed it



How the programmer wrote it



How the sales executive described it



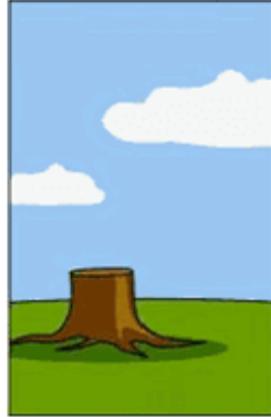
How the project was documented



What operations installed



How the customer was billed



How the helpdesk supported it

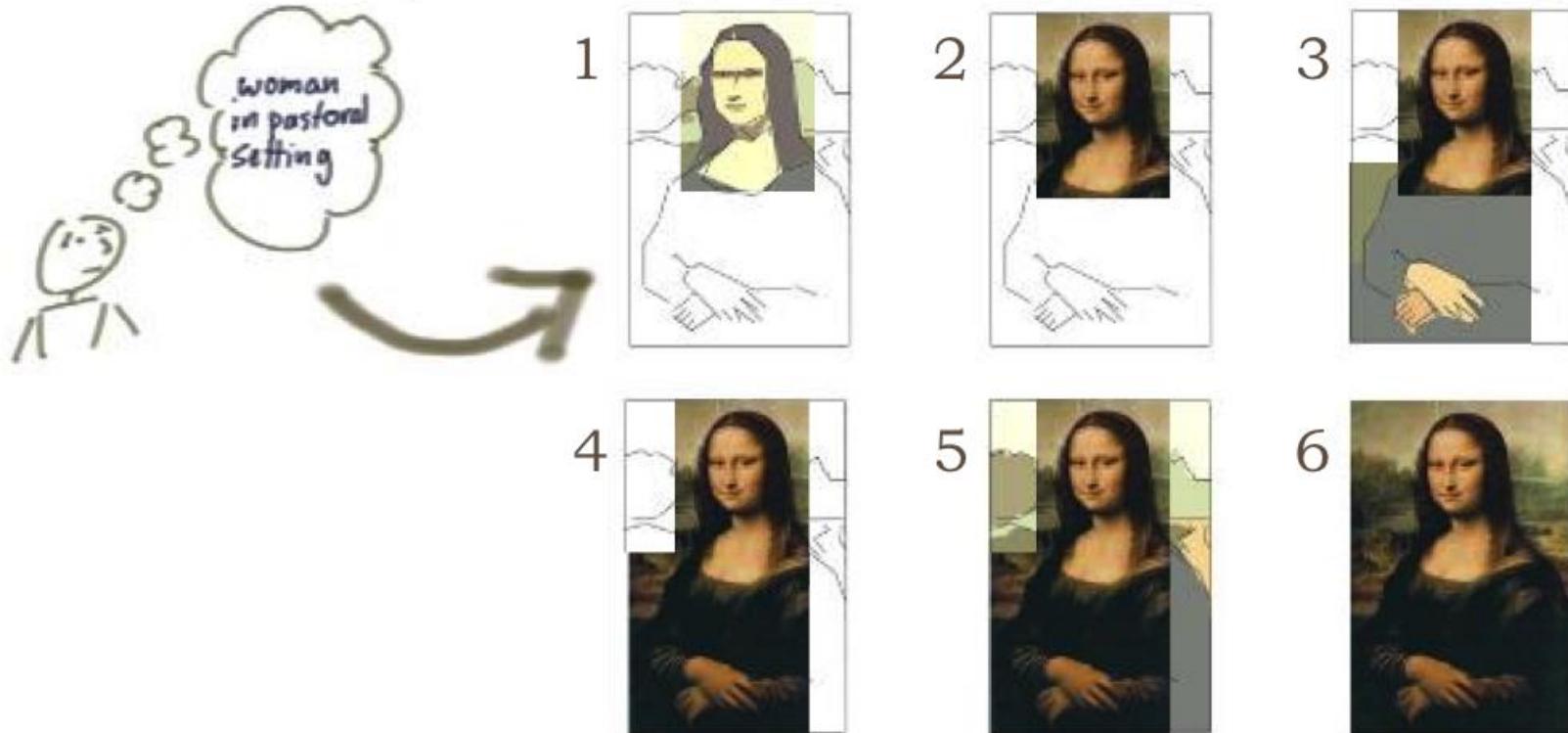


What the customer really needed

The Incremental vs. Iterative Conundrum



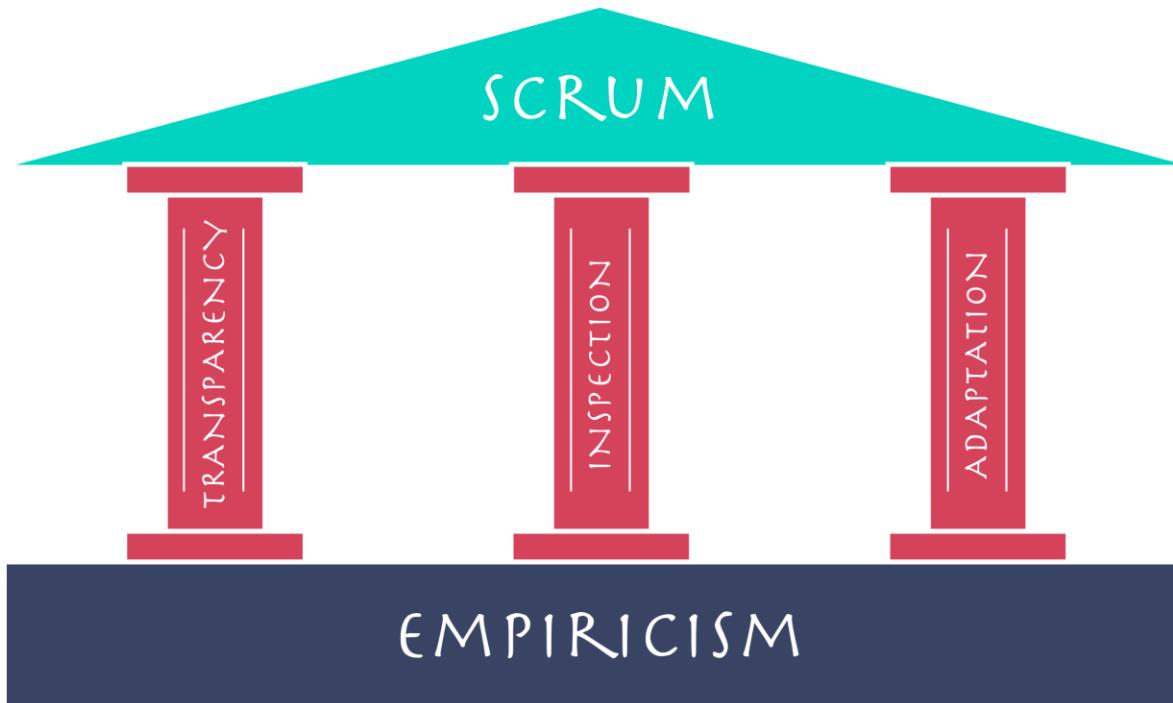
Iterative Incremental Approach



Scrum



Pillars of Scrum



Transparency

Giving visibility to the significant aspects of the process to those responsible for the outcome.

Inspection

Timely checks on the progress toward a sprint goal to detect undesirable variances.

Adaptation

Adjusting a process as soon as possible to minimize any further deviation or issues.

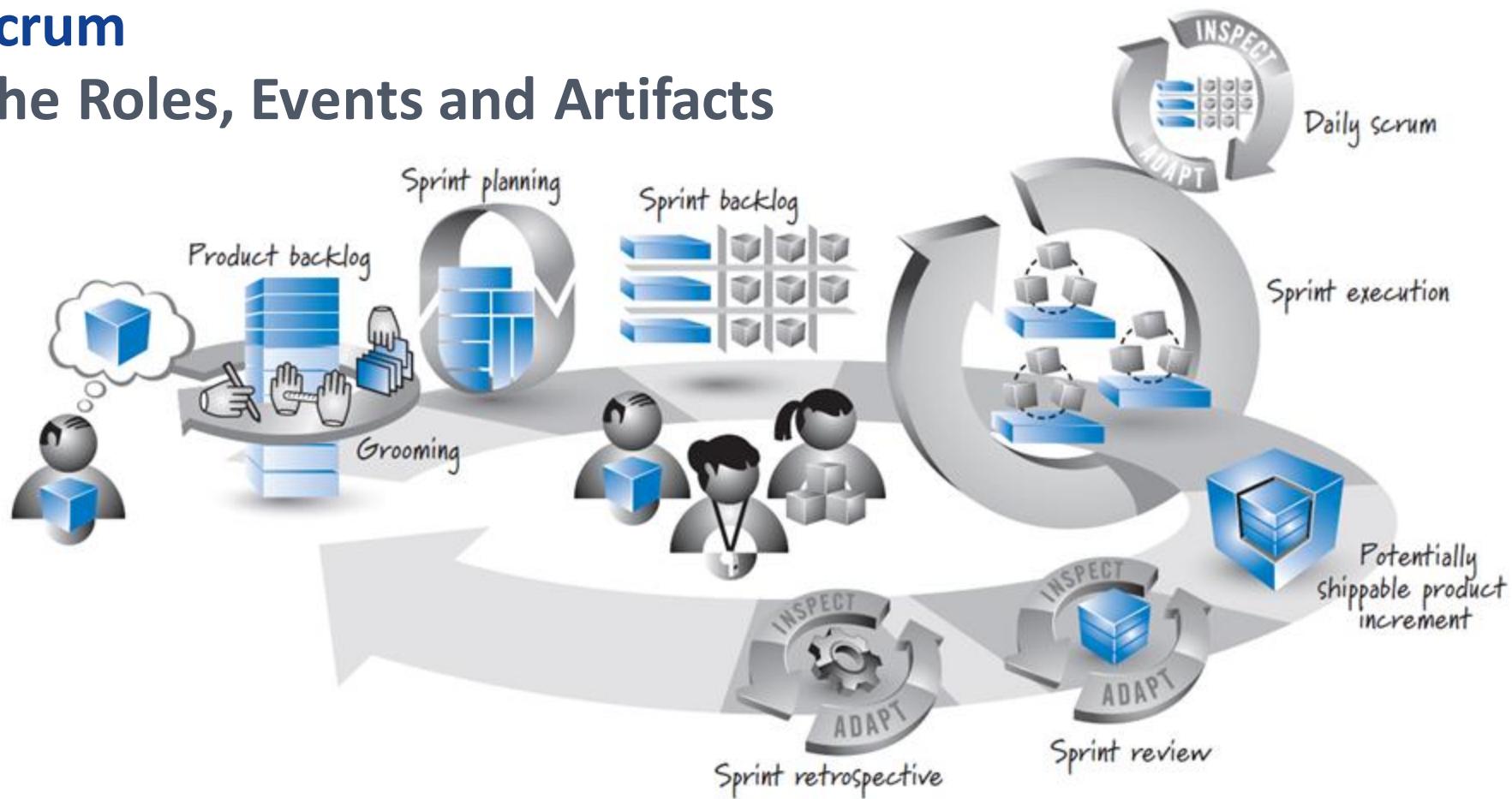
Scrum is founded on empiricism and lean thinking. Empiricism asserts that knowledge comes from experience and making decisions based on what is observed. If we apply Scientific method to software development, we must use empiricism and let reality speak for itself and correct any previous errors and integrate any new insights into previous knowledge.

Values of Scrum



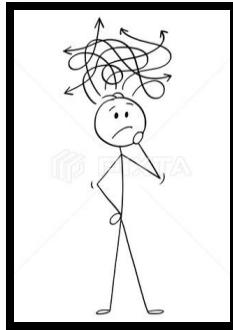
Scrum

The Roles, Events and Artifacts



Scaling Frameworks

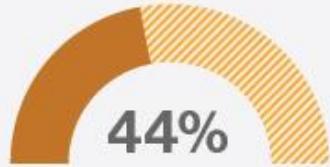
Do we really need them? Business model and reality



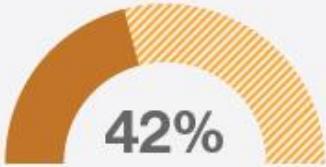
Scaling Agile @ Spotify
with Tribes, Squads, Chapters & Guilds

Is Scrum the Silver Bullet of Software Development?

Causes of failure – State of Agile Report 2015



Lack of experience
with Agile methods



Company philosophy
or culture at odds with
core Agile values



Lack of management
support



External pressure
to follow traditional
waterfall processes



Lack of support
for cultural transition



A broader organizational
or communications
problem

Is Scrum the Silver Bullet of Software Development?

Top Agile adoption barriers – State of Agile Report 2022

- Inconsistencies in processes and practices 46%
- Cultural clashes 43%
- General organizational resistance to change 42%
- Lack of skills and experience 42%
- Absence of leadership participation 41%
- Inadequate management support and sponsorship 40%

The key challenges organizations face when adopting Agile have remained largely unchanged for the past several years.

Challenges with **organizational culture, resistance to change, and lack of support and skills** continue to be problems.

How we work...



Scrum Artifacts – Product & Sprint Backlogs

Assure Program Kanban

Backlog

Board ↗

QUICK FILTERS: [PI candidate] Features Enablers [PI-5 (inc stretched)] [Not in PI-5] [PI5 Releases] [NAC 20] [NAC] [NPM Std] [NetAct] [NPM Big Data] [DO Suite] Only My Issues Recently Updated

Candidates (FS/F0) 62 issues

NOKIA Dashboards Projects Issues Boards Structure Tests Plans Requirements PIA Create

All sprints Switch sprint

PI NPM-21428 Improve PM adaptations support and content indexer performance 3 Versions NAC 20

PI NPM-25022 NAC user session management (advanced) NAC 20

PI NPM-23341 NPM-4101 HardCamele 20

PI NPM-4101 All sprints 20

PI NPM-20254 QUICK FILTERS: In PI Not in PI Only My Issues Recently Updated Zero SP NAC21 FP2101 not NAC21 FP2101 Hide Resolved

PI NPM-24748 BACKLOG 2 TO DO 17 ON HOLD 0 IN PROGRESS 17 IN REVIEW 1 IN TESTING 0 READY FOR DEMO 0 DONE / OBSOLETE 62

STT & Faults 2 issues

PI NPM-20208

PI NPM-25446

PI NPM-22882

PI NPM-14138 Objectives 1 issue

PI NPM-25668

PI NPM-24519

PI NPM-21157 Features 11 issues

ASSUR-16462 NPM-20315 Update oracle database server root disk to support customer-like installs. F

ASSUR-12901 NPM-18383 NAC 21 platform customer-like installs. R

ASSUR-14045 NPM-22198 NAC 21 platform customer-like installs. G

ASSUR-10844 NPM-17358 Installation of oracle database node in openstack R

ASSUR-11944 NPM-21197 NAC 21 platform customer-like installs. R

ASSUR-16468 NPM-21868 NAC 21 platform customer-like installs. R

ASSUR-11998 NPM-17883 NAC 21 platform customer-like installs. P

ASSUR-3556 Oracle VM creation fails if volume bigger than 3TB None

ASSUR-11373 HIC: dracpp tool in nolsa F&O: provide proper command to integrate to dynamic R

ASSUR-11374 HIC: dracpp for openstack in oracle integration pipe (oracle product build) Installation of Oracle DB in external env. 2020915_AE_NAC 21 FP2101 None

ASSUR-11375 HIC: dracpp for openstack in oracle integration pipe (oracle product build) Installation of Oracle DB in external env. 2020915_AE_NAC 21 FP2101 None

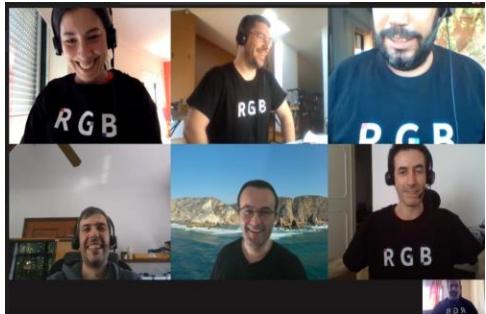
ASSUR-10844 HIC: CSP 21 FP1 onboarding - DoR ready NAC 21 FP2101 on top of CSP 21 FP1 None

ASSUR-11399 Support for solving Kafka Consumer groups issue in Yablob (cf1152) HardCamele - Support and Tech Topics None

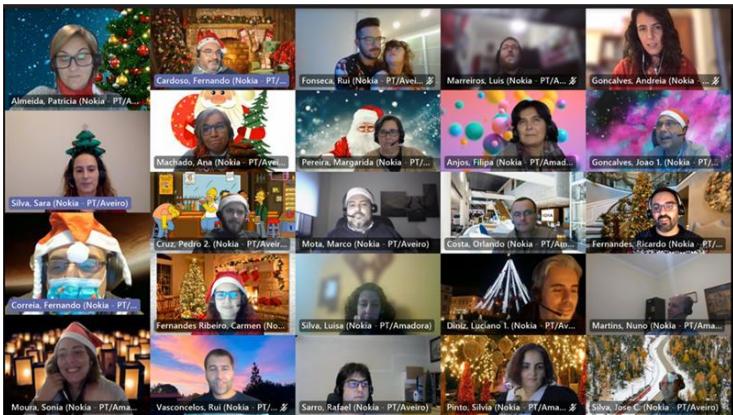
ASSUR-11399 Support for solving Kafka Consumer groups issue in Yablob (cf1152) HardCamele - Support and Tech Topics None

How we work during and after the Pandemic (part 1of2)

Being Nokia is adapted to remote work

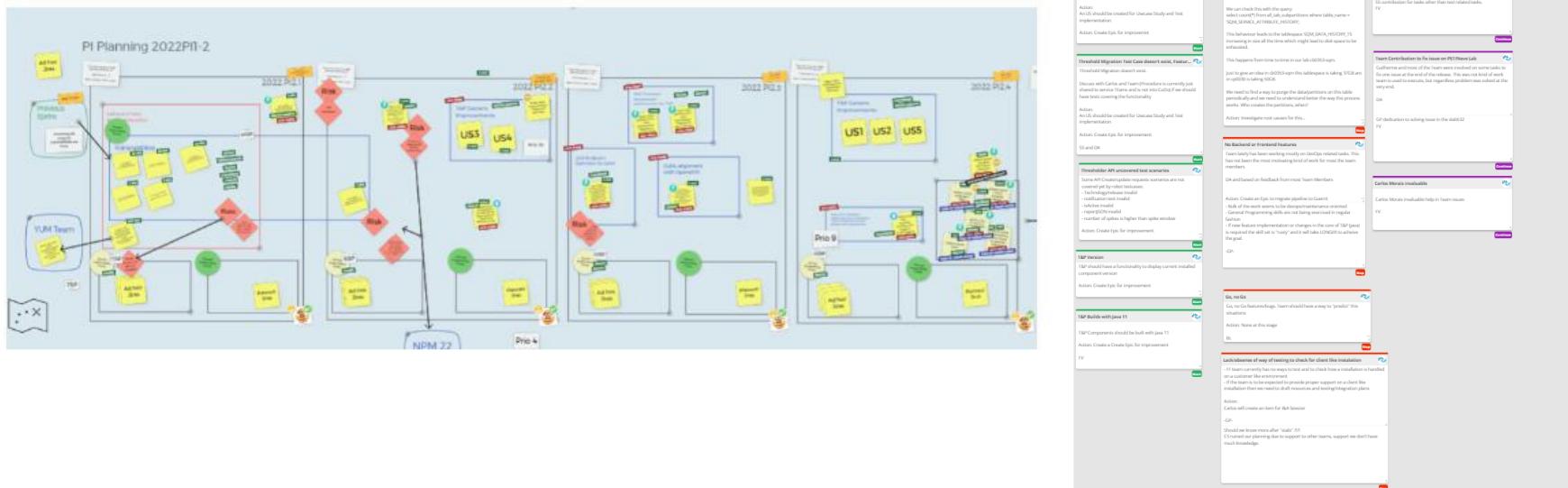


Meetings



How we work during and after the Pandemic (part 2of2)

Remote Team Plannings and Retrospectives



Focus on Quality (1/3)

Definition of Ready

The screenshot shows a Jira board with the following columns: TO DO (0), IN PROGRESS (7), ON HOLD (2), IN REVIEW (0), IN TESTING (0), and DONE (8). The IN PROGRESS column contains three user stories:

- AOA-23855 (SAFE EXECUTION) - 5 sub-tasks: Nifi Authentication - email is not suitable for LDAP user
- AOA-10426 (SAFE COMPLETED) - 37 sub-tasks: Nokia PM Adaptations on top of NAC21x - NA22 FP2202
- ASSUR-24990 (IN DEVELOPMENT) - 2 sub-tasks: Create EMIS integration endpoint bookkeeping test

Below the board, two user stories are shown in detail:

- ASSUR-27190 (USDoD): Status is In Progress.
- ASSUR-27171 (USDoR): Status is In Progress.

DoR (as in CNS User Story DoR page)

The Product Owner is responsible for verifying and approving DoR on User Story level.
The Agile Team (Development Team, Scrum Master, PO) is responsible for delivering the DoR items.
Once the DoR is completed the User Story can be prioritized as part of the Team Backlog.

ID	Definition of Ready Criteria	Status ?	Comment / Description
USDoR-01	Details and Acceptance Criteria are documented	✓	<p>The value is described with good story elements (INVEST mode - Independent, Negotiable, Valuable, Estimable, Small, Testable)</p> <p>The acceptance criteria contains functional and non-functional requirements e.g. performance and test automation levels</p>
USDoR-02	Initial story/PBI estimation is done	✓	All effort is estimated by the Team including effort to provide inputs for the technical writer and all other needed actions
USDoR-03	The User Story is refined by the Team and the Product Owner	✓	An understanding of acceptance criteria is established by discussions between the agile team members

Note on USDoR-01: Current list of **NFRs** that we need to support in NAC can be found [here](#).

Focus on Quality (2/3)

Definition of Done

ID	Definition of Done Criteria	Status ?	Comment / Description
USDoD-01	Acceptance criteria is fulfilled	✓	Requirements are fulfilled, demonstrated and approved by the Product Owner
USDoD-02	Static Code analysis is done and technical debt targets are met	N/A	New technical debt cannot be introduced. Unit test coverage is measured and targets need to be met for new code implementation.
USDoD-03	Code is reviewed, approved, and checked in to the version control system and main software branch	✓	Code review is successfully done and documented. Coding standards have been followed
USDoD-04	Tests in all levels are documented, automated and passed according to the test plan	✓	All requirements according to the acceptance criteria must be tested and tests automated
USDoD-05	There are no open faults	✓	Virtual zero targets must be adhered according to the virtual zero guidance
USDoD-06	The User Story is demonstrated in Iteration Review	✓	Demo at iteration review (10/3/2022)

USD oD-10	Privacy Data Scrambling implementation is completed and Privacy Data Definitions (PDD) document is updated	N/A	If the privacy data definitions (PDD) for one or more components have been changed, component teams complete implementation of PDD changes and trigger needed updates to decoder and scrambler tools. Final updates are done in the PDD and checked into version control by the Product Security Lead. Privacy Champion concurs these activities with the Product Security Lead (if they are different individuals in the organization).
USD oD-11	Privacy Engineering and Assurance Process (PEAP) activities are completed	✓	PEAP activities include Scope, User Stories, Data Flows, PDI and PTA from the template. The Risk Assessment (PIA) is completed at the release level. The activities need to be approved by the Privacy Champion.
USD oD-12	SVM and TALKO tools information is updated for the 3rd party components introduced or updated in the User Story	✓	3rd party components (OEM/OSSW/Firmware) introduced or updated as a part of the user story must be updated in the SVM tool (VAMS) as well as TALKO tool (for licensing checks). This ensures that all 3rd party components are monitored for vulnerabilities as well as licensing breaches. https://talko.int.net.nokia.com/certificates/details.php?certificate_id=10771

Focus on Quality (3/3)

Acceptance Criteria (Conditions of Satisfaction)

 TM_Assure / ASSUR-19495

Storing Period refactor - List adaptations available

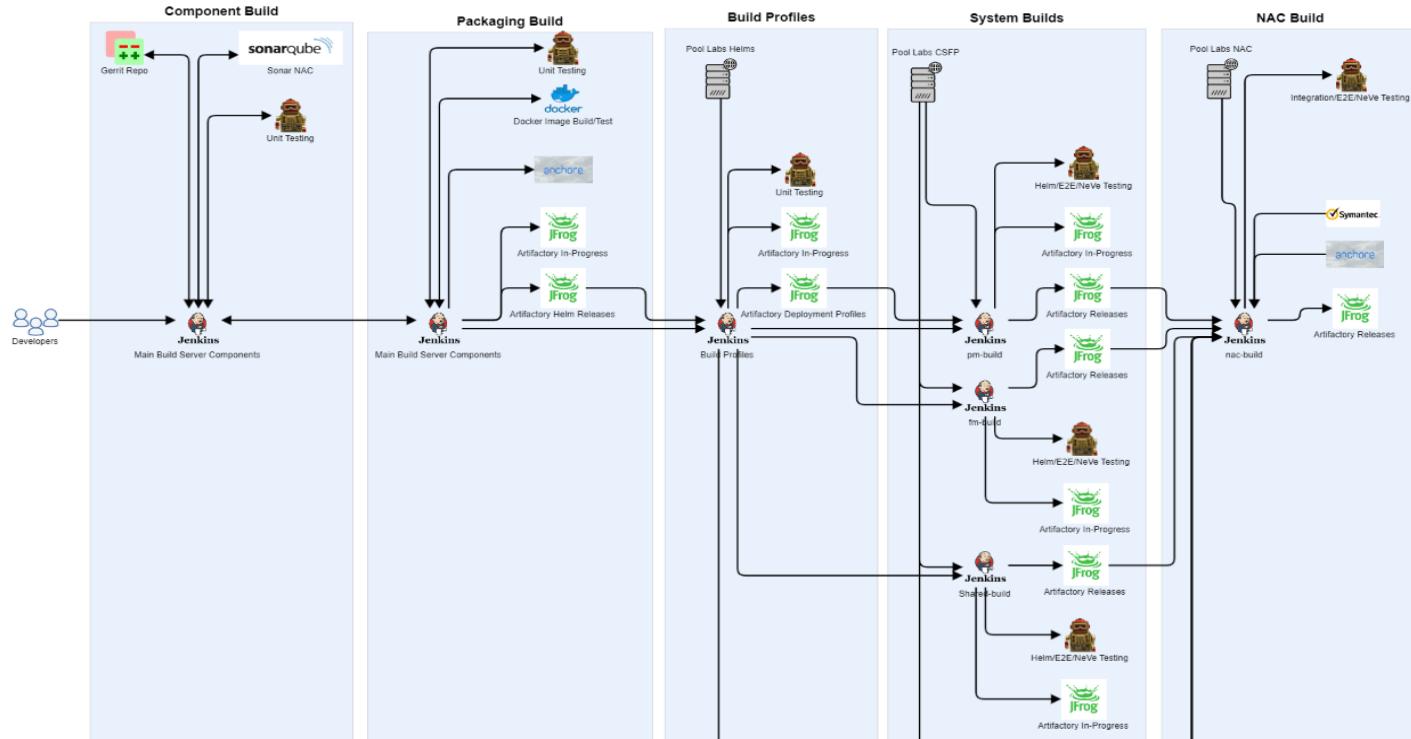
[Edit](#) [Add comment](#) [Assign](#) [More ▾](#) [Cancelled](#)

Details

Type:	<input checked="" type="checkbox"/> User Story	Status:	 DONE (View Workflow)
Priority:	<input type="radio"/> Unknown	Resolution:	Resolved
Affects Version/s:	None	Fix Version/s:	None
Component/s:	None		
Labels:	None		
Issue Category :	Enabler		
Acceptance Criteria:	<ul style="list-style-type: none">▼ Given storing period ui<ul style="list-style-type: none">When the user access storing periodThen the list of adaptations is available. Given storing period ui<ul style="list-style-type: none">When the user inserts text in the adaptation search boxThen the list of adaptations is filtered based on the inserted text		

Continuous Improvement

CI Overview



Continuous Integration

NOKIA A&A Jenkins

search Novais Maranhao, Ana (Nokia - PT/Aveiro) log out

Jenkins OSS NPM product oracle nac21

enable auto refresh

Up Status View Configuration People Build History Project Relationship Check File Fingerprint Job Config History Open Blue Ocean Config Files Wall Display

nac21

Jenkins configured automatically by Jenkins Configuration as Code Plugin

All Dashboard NAC21 NAC21 Monitor

S	W	Name	Last Success	Last Failure	Last Duration	Built On	Fav	# Issues
●	✖	nac-solution-build	1 day 5 hr - #1230	5 hr 32 min - #1231	2 min 10 sec	star	-	
●	✖	nac-solution-build-parallel	16 hr - #13-master-review	30 min - #17	2 min 3 sec	star	-	
●	✖	nac-solution-build_vlog	N/A	N/A	N/A	star	-	
●	✖	cm-product-build	22 hr - #3283-master-t2	1 day 0 hr - #3282-master-t2	1 hr 20 min	star	-	

Build Queue (6)

NM > Neo > environment > vnfinstallation
NM > Neo > environment > e2e
BSS > SPS > production > branches > SPSMAIN > v3394_sps_instantate
NSW > IF > cdk-tenant-operator
NM > Neo > environment > neo-product
part of NM > Neo > environment > e2e neo-psc-geert-verify #142

Build Executor Status

master 1 idle
dcdk-tenant-operator-735-qhkh-Tmkl9

Stage View

Average stage times (Average [all] duration = 9 hours)

Declarative	Checkouts	Init	Untimed	Check	Version	Deploy all	Reserve Lab	Reserve Oracle	Redeploy Oracle	Namespace Creation	Deploy SHARED	Deploy FM	Deploy PH	Deploy TP	Deploy PH-TP	Deploy UIV	Deploy ADAPTATIONS	Release Info	Test-Next	Status	InProgress	Candidate	Declarative Post Actions
9s	20s	30s	2s	55ms	1min 48s	10s	3s	1min 15s	1min 45s	30min 33s	40min 8s	10min 32s	1min 10s	17min 30s	4min 43s	5s	3min 10s	2s	89ms	402ms	32s		
24s	20s	34s	23s	40ms	1min 34s	15s	4s	2min 0s	1min 4s	35min 22s	42min 17s	17min 51s	3min 4s	38min 54s	8min 54s	5s	3min 10s	2s	89ms	402ms	32s		
9s	15s	39s	24s	40ms	1min 10s	15s	4s	1min 8s	10min 4s	17min 25s	24min 53s	10min 3s	35min 29s	4min 31s	12s	24min 39s	12s	2s	30s	30s			
9s	22s	35s	23s	10ms	1min 10s	15s	4s	1min 8s	10min 4s	17min 25s	24min 53s	10min 3s	35min 29s	4min 31s	12s	24min 39s	12s	2s	30s	30s			
8s	23s	38s	23s	24ms	1min 14s	15s	4s	1min 13s	13min 44s	34min 0s	55min 8s	36min 55s	2min 30s	35min 29s	4min 31s	12s	24min 39s	12s	2s	30s	30s		
10s	29s	50s	25s	60ms	1min 10s	15s	4s	1min 39s	14min 12s	1h 52min	82min	83min	88min	83min	85s	85min	85min	85min	85min	85min	85min		
9s	23s	35s	23s	40ms	1min 14s	15s	4s	1min 13s	10min 51s	17min 37s	18min 18s	34min 42s	2min 8s	15min 30s	1min 20s	10s	3min 42s	12s	2s	30s	30s		
8s	24s	39s	23s	40ms	1min 14s	15s	4s	1min 11s	10min 51s	17min 37s	18min 18s	34min 42s	2min 8s	15min 30s	1min 20s	10s	3min 42s	12s	2s	30s	30s		
9s	41s	47s	42s		77ms	1min 3s	18s	3s	24s								1h 10min						
24s	31s	39s																					

Build Queue (6)

NM > Neo > environment > vnfinstallation
NM > Neo > environment > e2e
BSS > SPS > production > branches > SPSMAIN > v3394_sps_instantate
NSW > IF > cdk-tenant-operator
NM > Neo > environment > neo-product
part of NM > Neo > environment > e2e neo-psc-geert-verify #142

Build Executor Status

master 1 idle
dcdk-tenant-operator-735-qhkh-Tmkl9

Stage View

Average stage times (Average [all] duration = 9 hours)

Declarative	Checkouts	Init	Untimed	Check	Version	Deploy all	Reserve Lab	Reserve Oracle	Redeploy Oracle	Namespace Creation	Deploy SHARED	Deploy FM	Deploy PH	Deploy TP	Deploy PH-TP	Deploy UIV	Deploy ADAPTATIONS	Release Info	Test-Next	Status	InProgress	Candidate	Declarative Post Actions
9s	20s	30s	2s	55ms	1min 48s	10s	3s	1min 15s	1min 45s	30min 33s	40min 8s	10min 32s	1min 10s	17min 30s	4min 43s	5s	3min 10s	2s	89ms	402ms	32s		
24s	20s	34s	23s	40ms	1min 34s	15s	4s	2min 0s	1min 4s	35min 22s	42min 17s	17min 51s	3min 4s	38min 54s	8min 54s	5s	3min 10s	2s	89ms	402ms	32s		
9s	15s	39s	24s	40ms	1min 10s	15s	4s	1min 8s	10min 4s	17min 25s	24min 53s	10min 3s	35min 29s	4min 31s	12s	24min 39s	12s	2s	30s	30s			
9s	22s	35s	23s	10ms	1min 10s	15s	4s	1min 8s	10min 4s	17min 25s	24min 53s	10min 3s	35min 29s	4min 31s	12s	24min 39s	12s	2s	30s	30s			
8s	23s	38s	23s	24ms	1min 14s	15s	4s	1min 13s	13min 44s	34min 0s	55min 8s	36min 55s	2min 30s	35min 29s	4min 31s	12s	24min 39s	12s	2s	30s	30s		
10s	29s	50s	25s	60ms	1min 10s	15s	4s	1min 39s	14min 12s	1h 52min	82min	83min	88min	83min	85s	85min	85min	85min	85min	85min	85min		
9s	23s	35s	23s	40ms	1min 14s	15s	4s	1min 13s	10min 51s	17min 37s	18min 18s	34min 42s	2min 8s	15min 30s	1min 20s	10s	3min 42s	12s	2s	30s	30s		
8s	24s	39s	23s	40ms	1min 14s	15s	4s	1min 11s	10min 51s	17min 37s	18min 18s	34min 42s	2min 8s	15min 30s	1min 20s	10s	3min 42s	12s	2s	30s	30s		
9s	41s	47s	42s		77ms	1min 3s	18s	3s	24s								1h 10min						
24s	31s	39s																					

Best Practices

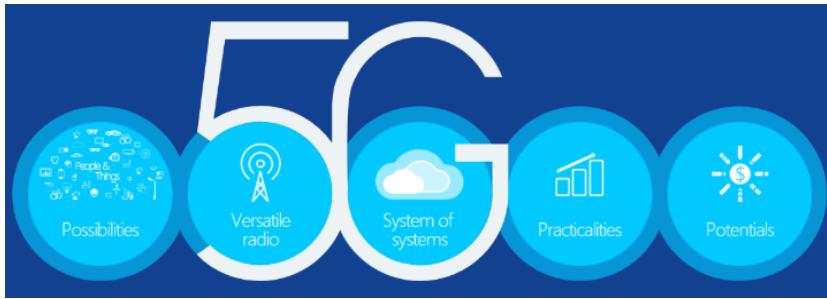
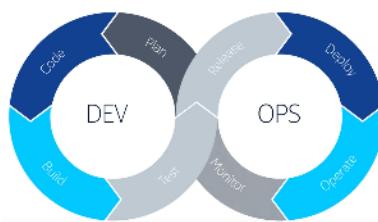
- Pair Programming
- Code Review
- Component/CodeGuardians
- Pre-Studies
- TDD/ATDD
- Continuous Integration
- Testing
- Refactoring
- Static Code Analysis
- Version Control System (Git and Gitlab)



Continuous Innovation



EXPLORATION DAY



CoPs
Communities of
Practice

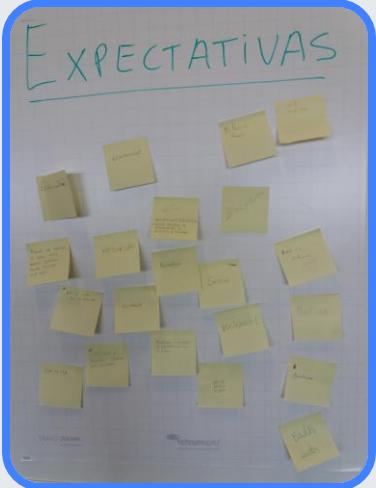
BROWN
BAG Series
Lunchtime discussion



Wellbeing & Social Responsibility



Being Nokia!



References

- Robot (Test) Framework <https://robotframework.org/>
- Zombie Scrum Survival Guide, 2020 (Christiaan Verwijs, Johannes Schartau, Barry Overeem)
- Scrum: The Art of Doing Twice the Work in Half the Time, 2014 (Jeff Sutherland)
- Clean Agile: Back to Basics, 2019 (Robert C. Martin – aka Uncle Bob)
- Uncle Bob: Clean Agile, Back to Basics, 2020, [Youtube talk](#)
- Uncle Bob Clean Code Lessons, 2019, [Youtube talk](#)
- Scrum Master Toolbox [podcast](#)

Contacts for further Questions

- Daniel Almeida daniel.almeida@nokia.com
- Orlando Costa orlando.costa@nokia.com



NOKIA

ANÁLISE DE SISTEMAS

Integração contínua & Entrega contínua (CI/CD)

Ilídio Oliveira

v2022/06/14

Objetivos de aprendizagem

Identificar os passos de um ciclo de CI

Distinguir entre *C. Integration*, *C. Deployment* e *C. Delivery*

Relacionar o CI/CD com a natureza iterative e incremental dos métodos ágeis de desenvolvimento

Explicar as tarefas englobadas numa “build”

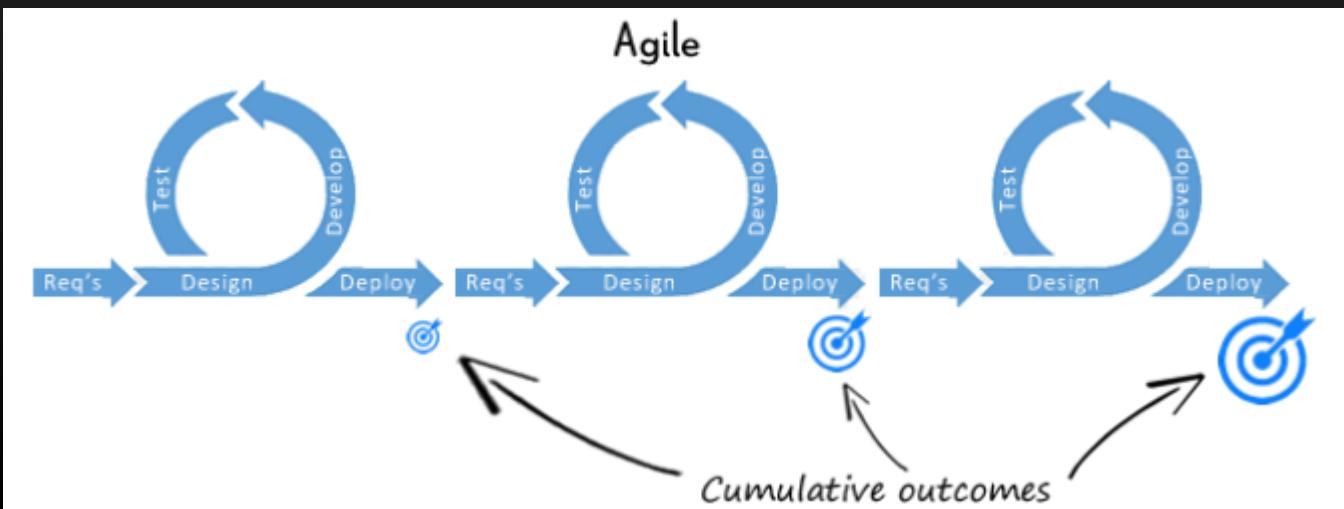
Explicar as principais boas práticas na implementação de CI numa equipa

Relacionar o CI/CD e DevOps.

Desenvolvimento iterativo

Cada iteração envolve escolher um pequeno subconjunto dos requisitos, para projetar/desenhar, implementar e testar.

- Desenvolvimento em ciclos curtos
- Cada ciclo dá um incremento executável (parcial)
- Cada incremento é testado e integrado
- O feedback de cada iteração leva ao refinamento e adaptação da próxima.



Continuously integrating the “units”

The essence of it lies in the simple practice of everyone on the team integrating frequently.

Feel comfortable and set up the tools to integrate at any time.

CI makes the development process smoother and less risky

↓ “it runs on my computer”

Early detection of failures (react quickly)

spot errors earlier
shared code ownership
everybody is co-responsible

big, unpredictable effort to integrate
app state is not executable most of the time

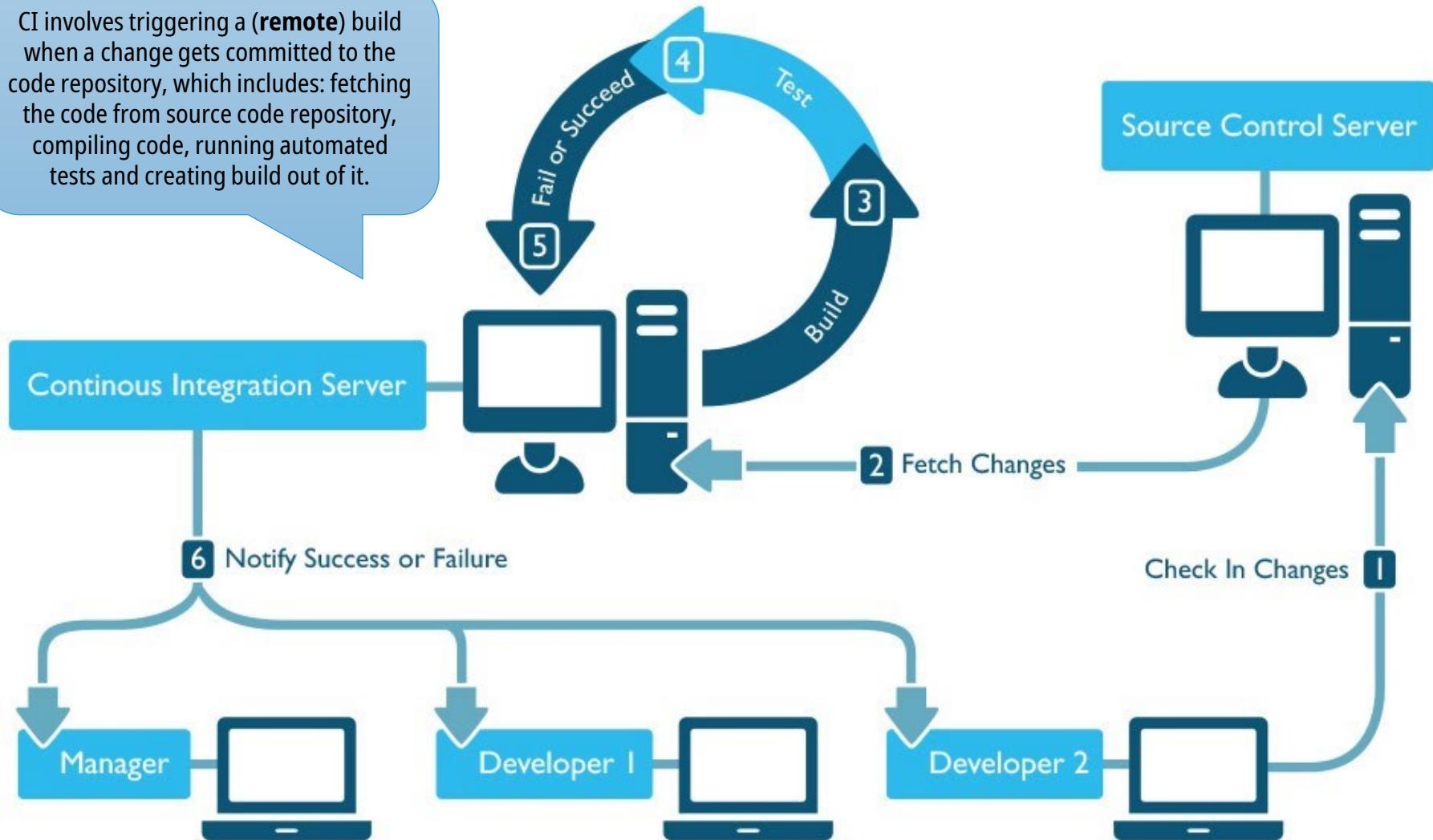
integrate early and often

Integration hell

\(ツ)\/

IT WORKS
on my machine

CI involves triggering a (**remote**) build when a change gets committed to the code repository, which includes: fetching the code from source code repository, compiling code, running automated tests and creating build out of it.



<https://insights.sei.cmu.edu/blog/continuous-integration-in-devops/>

Suggested development workflow

1- Update from shared SCM

2- Code a new feature (tests + code) in a dedicate feature branch

3- Run automated build on local machine

Repeat #2 and #3 till tests pass

4- Commit (integrate with "central")

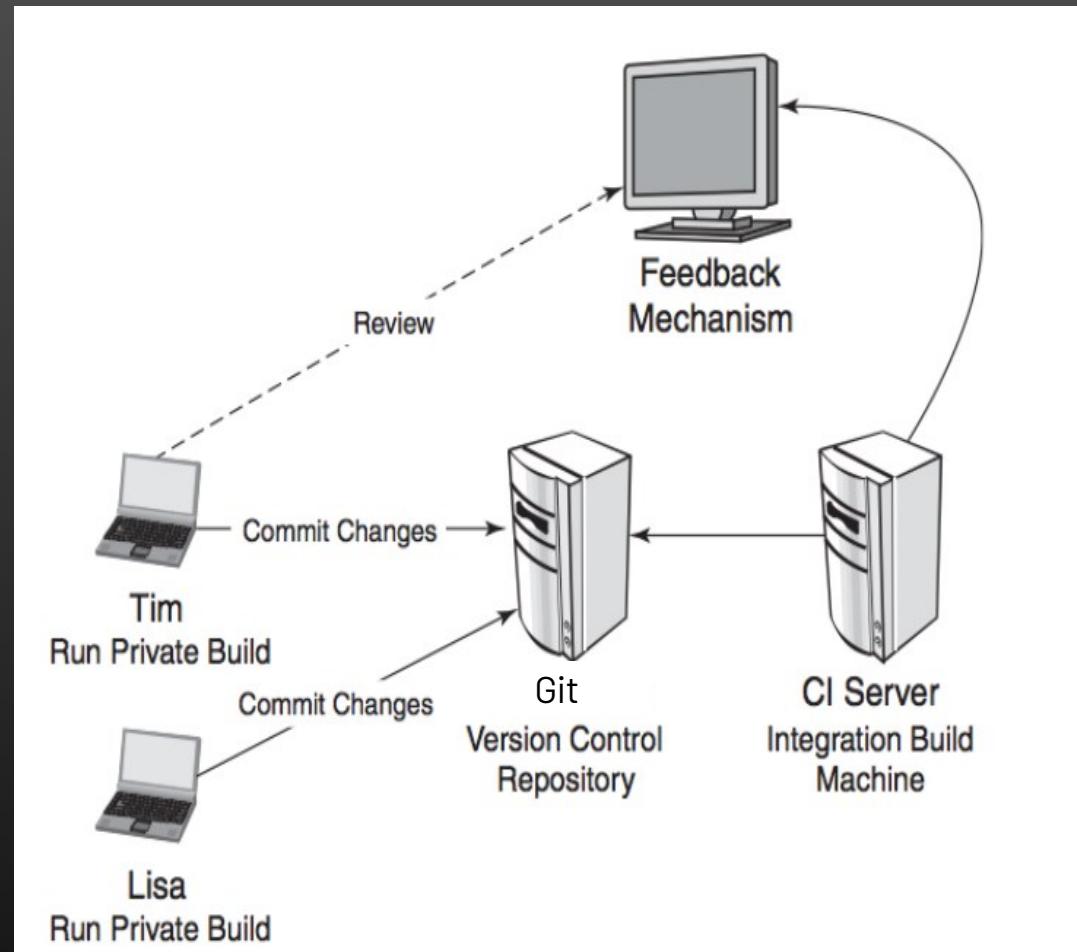
Pull requests advised (with peer code review)

5- Run a build on a clean machine

Update artifacts, update build status (feedback)

Immediately fix bugs and integration issues

Not only tools: CI culture required!



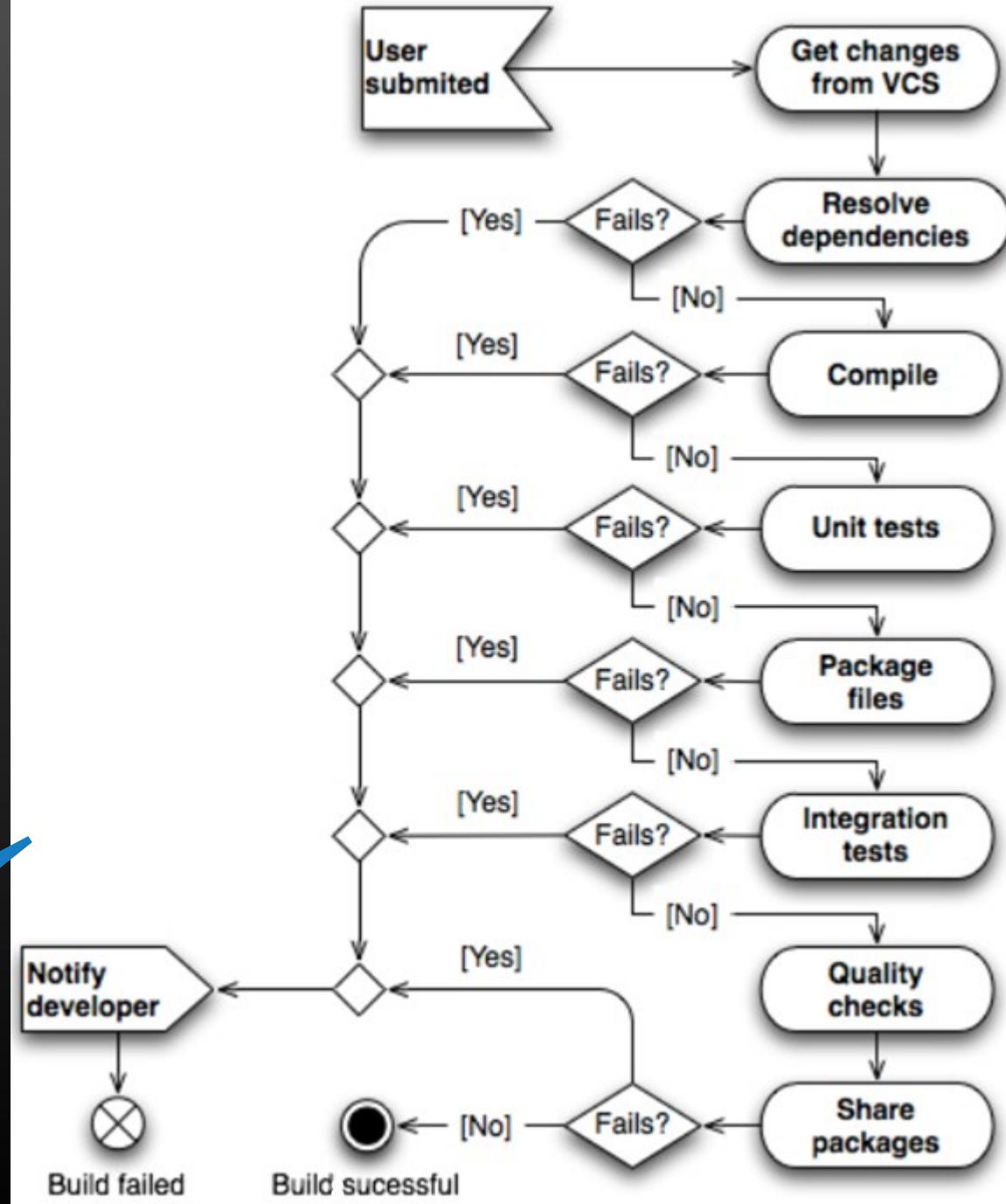
The build process

A build has several stages.

A successful build implies success in code correctness and quality checks.

Automatic build tools run quality checks (e.g.: unit testing, code inspections)

Not just compiling...



Quality checks → análise estática do código

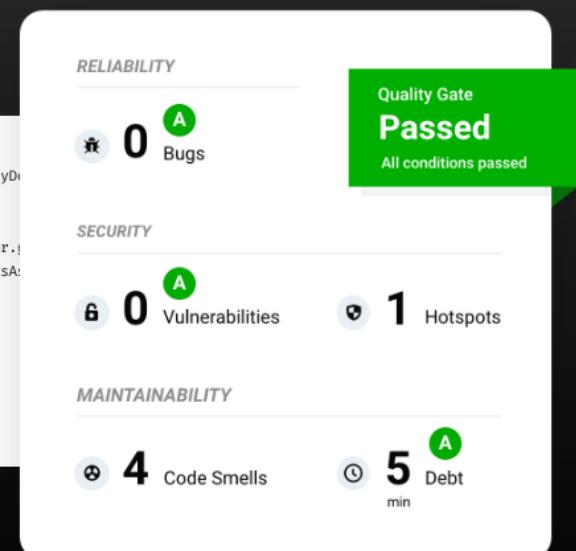
Analizar a qualidade do Código sem o executar

- vulnerabilidades conhecidas
- potenciais erros (e.g.: *null pointers*,...)
- construções desaconselhadas

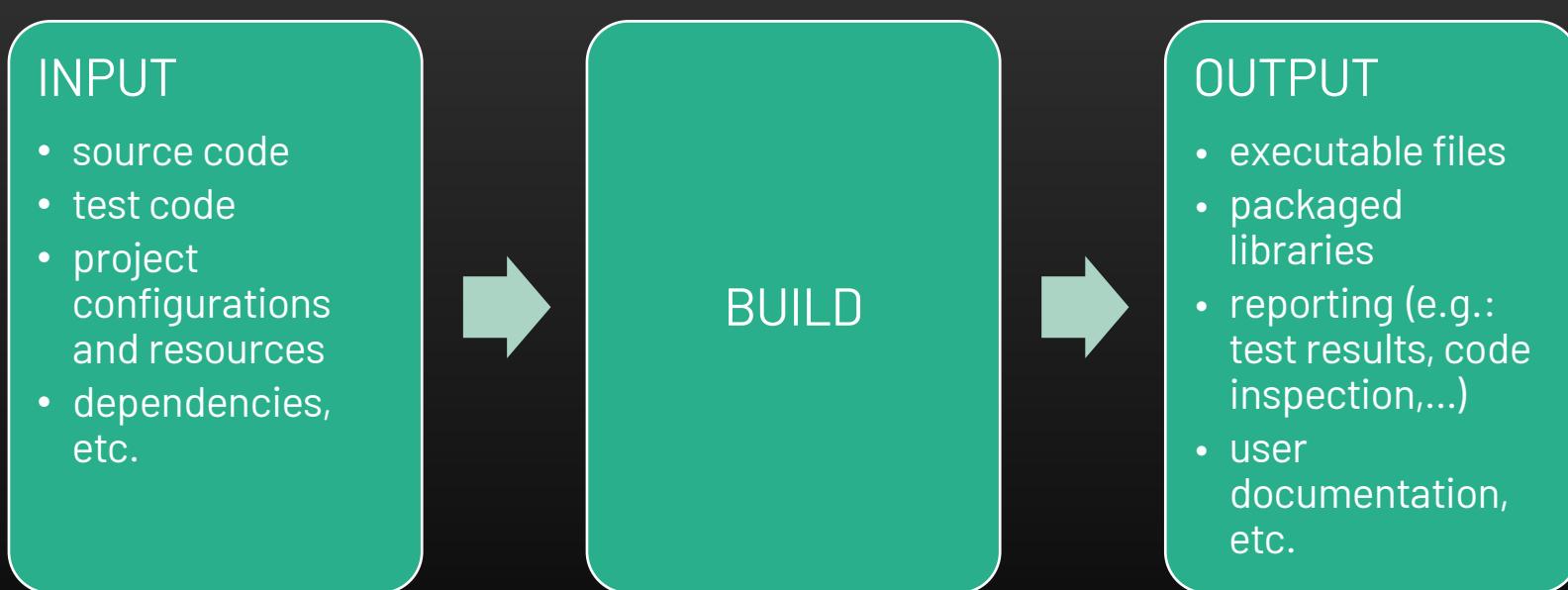
<https://www.sonarqube.org/>

A screenshot of the SonarQube interface. On the left, there is a code editor window displaying Java code. A red box highlights a line of code: "Class providedClass = ReflectionUtils.getTypeClass(providedType);". Below this line, another red box highlights "providedClass.isAssignableFrom(List.class)". A tooltip appears below the code editor stating: "A 'NullPointerException' could be thrown; 'providedClass' is nullable here." At the bottom of the code editor, there are three status indicators: a bug icon labeled "Bug", a major icon labeled "Major", and a link labeled "cert, cwe". The code editor shows lines 246 through 253.

```
246 if (Provider.class == roleTypeClass) {  
247     Type providedType = ReflectionUtils.getLastTypeGenericArgument(dependencyDi-  
248     2 Class providedClass = 1 ReflectionUtils.getTypeClass(providedType);  
249  
250     if (this.componentManager.hasComponent(providedType, dependencyDescriptor.  
251         || 3 providedClass.isAssignableFrom(List.class) || providedClass.isA:
```



A *build* é mais que compilar...



Práticas da integração contínua

- Os *developers* gravam para o repositório partilhado com regularidade (*commit*)
- As alterações no SCM são monitorizadas e acionam automaticamente o ciclo de CI
- Feedback imediato sobre falhas nas *builds* (as *builds* falhadas têm alta prioridade)
- Opcional: instalação de artefactos num repositório de referência
- Opcional: acionamento de testes de integração/aceitação

Quanto mais frequente for o processo de integração, menos penoso será

“Continuous” → mindset e ferramentas preparados para a integração a qualquer altura

Fowler's 10 CI practices

- Maintain a Single Source Repository.
- Automate the Build
- Make Your Build Self-Testing
- Everyone Commits To the Mainline Every Day
- Every Commit Should Build the Mainline on an Integration Machine
- Keep the Build Fast
- Test in a Clone of the Production Environment
- Make it Easy for Anyone to Get the Latest Executable
- Everyone can see what's happening
- Automate Deployment

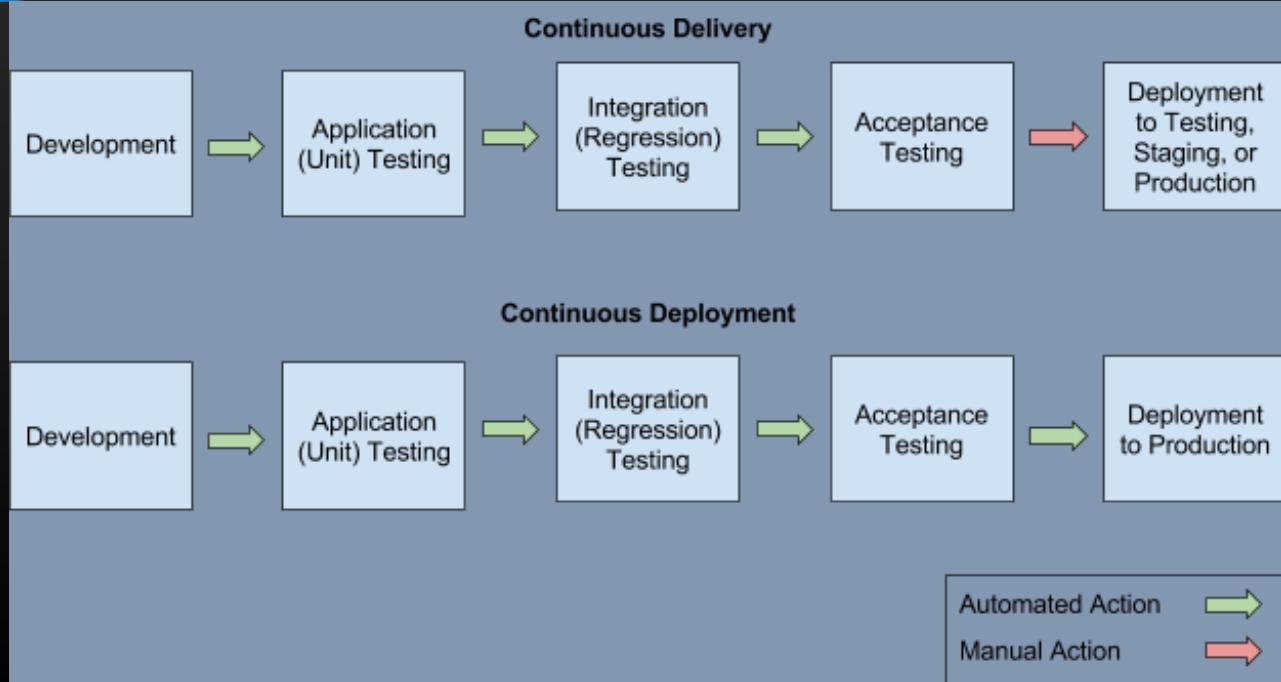
<http://martinfowler.com/articles/continuousIntegration.html>

Termos relacionados

continuous
delivery / deployment

continuous
integration

É possível fazer instalações frequentes, mas pode-se optar por não o fazer (geralmente relacionadas com a estratégia empresarial)



Continuous...

Continuous Delivery

sw development practice in which you build software in such a way that it **can be released** to production at any time.

You're doing continuous delivery when:

Focus on quality of working software

Your software is deployable throughout its lifecycle

Your **team prioritizes keeping the software deployable over working on new features**

Anybody can get fast, automated feedback on the production readiness

Continuous Deployment/release

every change goes through the pipeline and **automatically gets put into production**.

Focus on speed and agility to deploy to production

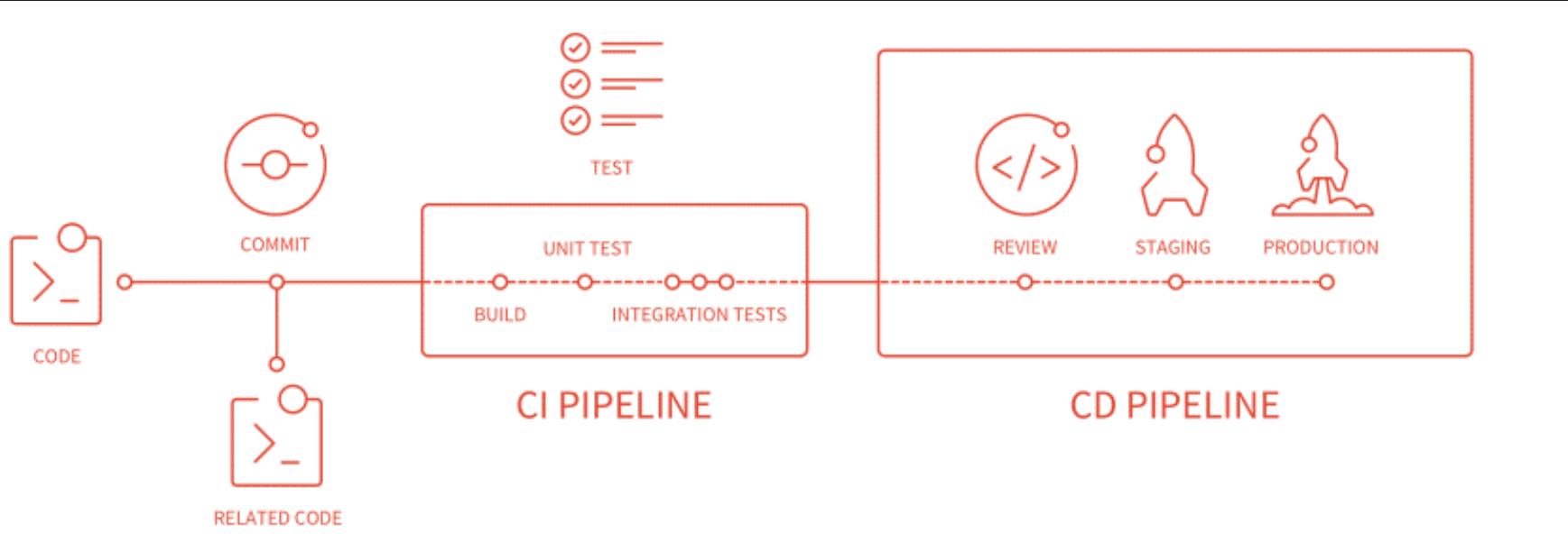
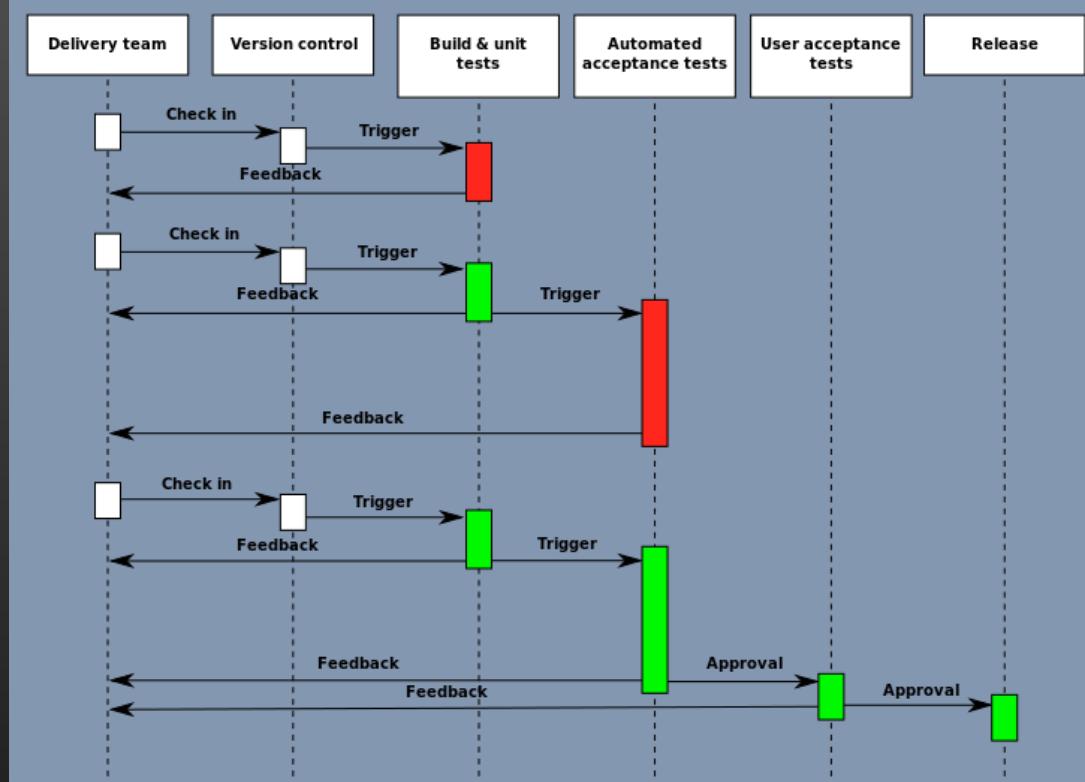
Continuous Integration

Automatically integrating, building, and testing code within the development environment.

Pre-delivery steps.

Entrega continua

<https://about.gitlab.com>



smartcode

localhost:8080/jenkins/view/Search%20Services/

Search Services

Librarian

Claimed by Alice Jenkins: There's a bunch of flaky tests that keep breaking. Fixing it now, 1.5.0+build.18.sha.7d22a8

19s | 25s

Performance Benchmark

#1 1m 13s

Promote to UAT

#12 Identified Problems: Invalid credentials 22s | 2m 7s

Promote to PROD

#2 1m 2s

Search API Contract Tests (PROD)

#1 17s

Search API Contract Tests (UAT)

#2 37s

Settings

Font size 0.5 1.0 2.0

Columns 1 2 6

Colour blind mode?

Add/Remove Jobs

Done

Brought to you by Jan Molak



Continuous feedback

Errors are easier to detect in an earlier stage, near the point where they have been introduced:

The detection mechanism of such bugs becomes simpler because the natural step in diagnosing the problem is to check what was the latest submitted change.

problems followed by atomic commits are easiest to correct than to fix several problems at once, after bulk commits

There must be an effective mechanism that automatically informs programmers, testers, database administrators and managers about the status of the build

Feedback → generate reaction in a more accurate and prompter way



Continuous testing

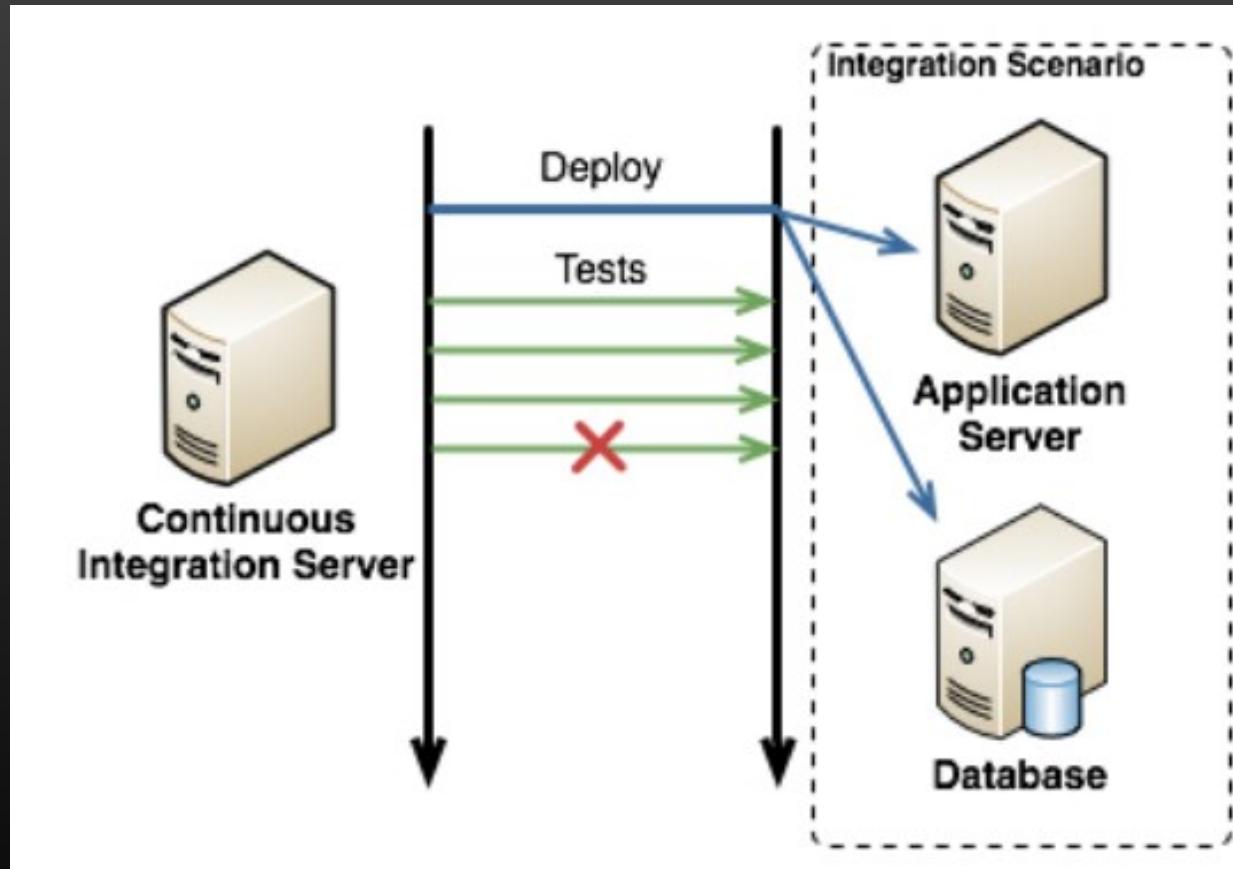
Quality checks at all system levels
and involve all individuals, not just the
elements of the QA team

Most of the tests can be automated
and should be run in the CI pipeline to
be carried out repeatedly:

unit testing, integration testing,
regression testing, system testing, load
and performance testing, etc.

Build tools can take a crucial role on
automating tests

Integration tests



Jenkins

Easy of use and extremely extensible

Plugins-oriented

Hosted

vs cloud-centric

Distributed builds

Master/slaves architecture



Jenkins

Jenkins vocabulary

Job: a runnable task

Node: a master or slave machine

Build Executor: a stream of builds to run

Plugin: module that extends the core functionality.

Pipeline: definition of the steps to be executed

Jenkins supports building Java projects since its inception, and for a reason! It's both the language Jenkins is written in, plus the language in use by many if not all the projects Kohsuke Kawaguchi wanted to watch out when he created the tool many years ago.

If you want to build a Java project, there are a bunch of different options. The most typical ones nowadays are generally Apache Maven, or Gradle.

Apache Maven

In any FreeStyle job, as **currently** Maven is supported in standard, you can use the dedicated step. One advantage is, as for all Jenkins tools, that you can select a specific Maven version and have Jenkins automatically install it on the build node it's going to run on.

Gradle

As the associated plugin is not installed by default, first install the [Gradle plugin](#). Once done, you should be able to add a Gradle step.

Java plugins for Jenkins

[JUnit plugin](#)
publishes JUnit XML formatted test reports for trending and analysis

[Gradle plugin](#)
support invoking Gradle as a build step and listing executing tasks per build

[Findbugs plugin](#)
generate trending and analysis for FindBugs reports

[PMD plugin](#)
generate trending and analysis for PMD reports

[Cobertura plugin](#)
publish and trend code coverage reports from Cobertura

[SonarQube plugin](#)
integrate reporting from the SonarQube code quality/inspection platform

[Repository Connector plugin](#)
adds features for resolving artifacts from a Maven repository such as Nexus or Artifactory.

→ <https://jenkins.io/solutions/java/>

Pipeline as Code with Jenkins

The default interaction model with Jenkins, historically, has been very web UI driven, requiring users to manually create jobs, then manually fill in the details through a web browser. This requires additional effort to create and manage jobs to test and build multiple projects, it also keeps the configuration of a job to build/test/deploy separate from the actual code being built/tested /deployed. This prevents users from applying their existing CI/CD best practices to the job configurations themselves.

Pipeline

With the introduction of the [Pipeline plugin](#), you can define your build/compile/test/deploy pipeline in a [Jenkinsfile](#) and store the pipeline configuration in version control, alongside another piece of code checked into source control.

A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released.

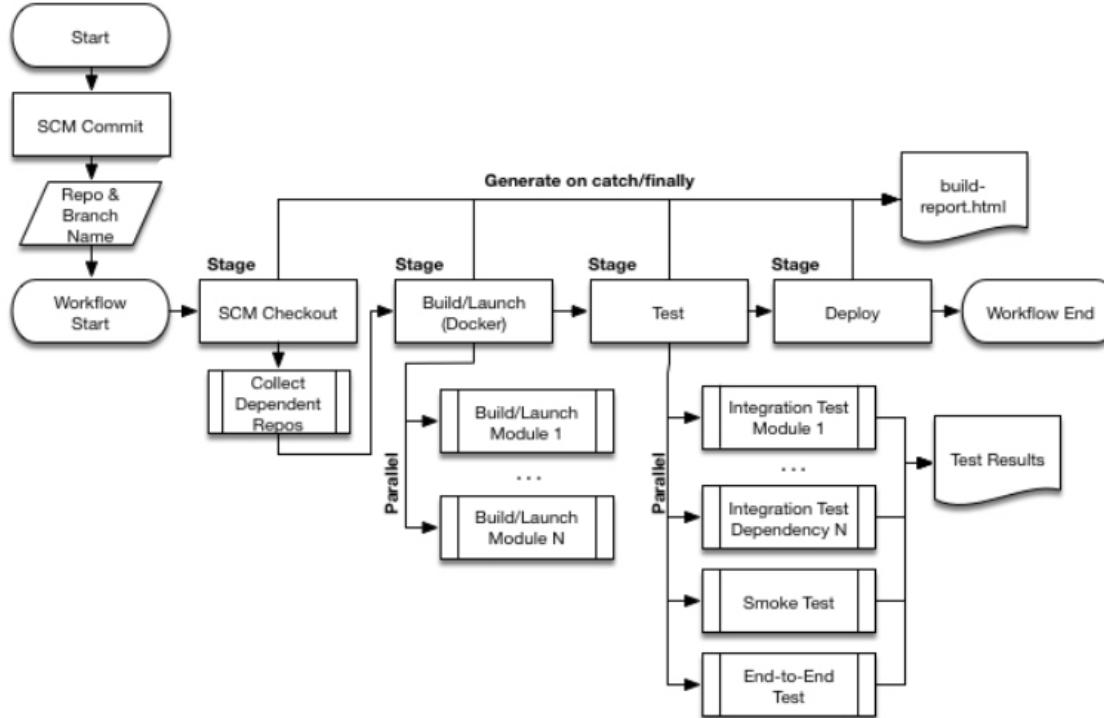
Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the [Pipeline domain-specific language \(DSL\) syntax](#).

Jenkins ❤ Continuous Delivery Articles

[Multibranch Workflows in Jenkins](#)
[jenkins-ci.org](#)

Continuous Delivery

Jenkins pipelines



```
pipeline {
    agent {
        docker {
            image 'maven:3-alpine'
            args '-v /root/.m2:/root/.m2'
        }
    }
    options {
        skipStagesAfterUnstable()
    }
    stages {
        stage('Build') {
            steps {
                sh 'mvn -B -DskipTests clean package'
            }
        }
        stage('Test') {
            steps {
                sh 'mvn test'
            }
            post {
                always {
                    junit 'target/surefire-reports/*.xml'
                }
            }
        }
        stage('Deliver') { ①
            steps {
                sh './jenkins/scripts/deliver.sh' ②
            }
        }
    }
}
```

```
pipeline {
    agent any
    stages{
        stage('Build'){
            steps {
                sh 'mvn clean package'
            }
            post {
                success {
                    echo 'Now Archiving...'
                    archiveArtifacts artifacts: '**/target/*.war'
                }
            }
        }
        stage ('Deploy to Staging'){
            steps {
                build job: 'Deploy-to-staging'
            }
        }
        stage ('Deploy to Production'){
            steps{
                timeout(time:5, unit:'DAYS'){
                    input message:'Approve PRODUCTION Deployment?'
                }
                build job: 'Deploy-to-Prod'
            }
            post {
                success {
                    echo 'Code deployed to Production.'
                }
                failure {
                    echo ' Deployment failed.'
                }
            }
        }
    }
}
```

GitHub Actions

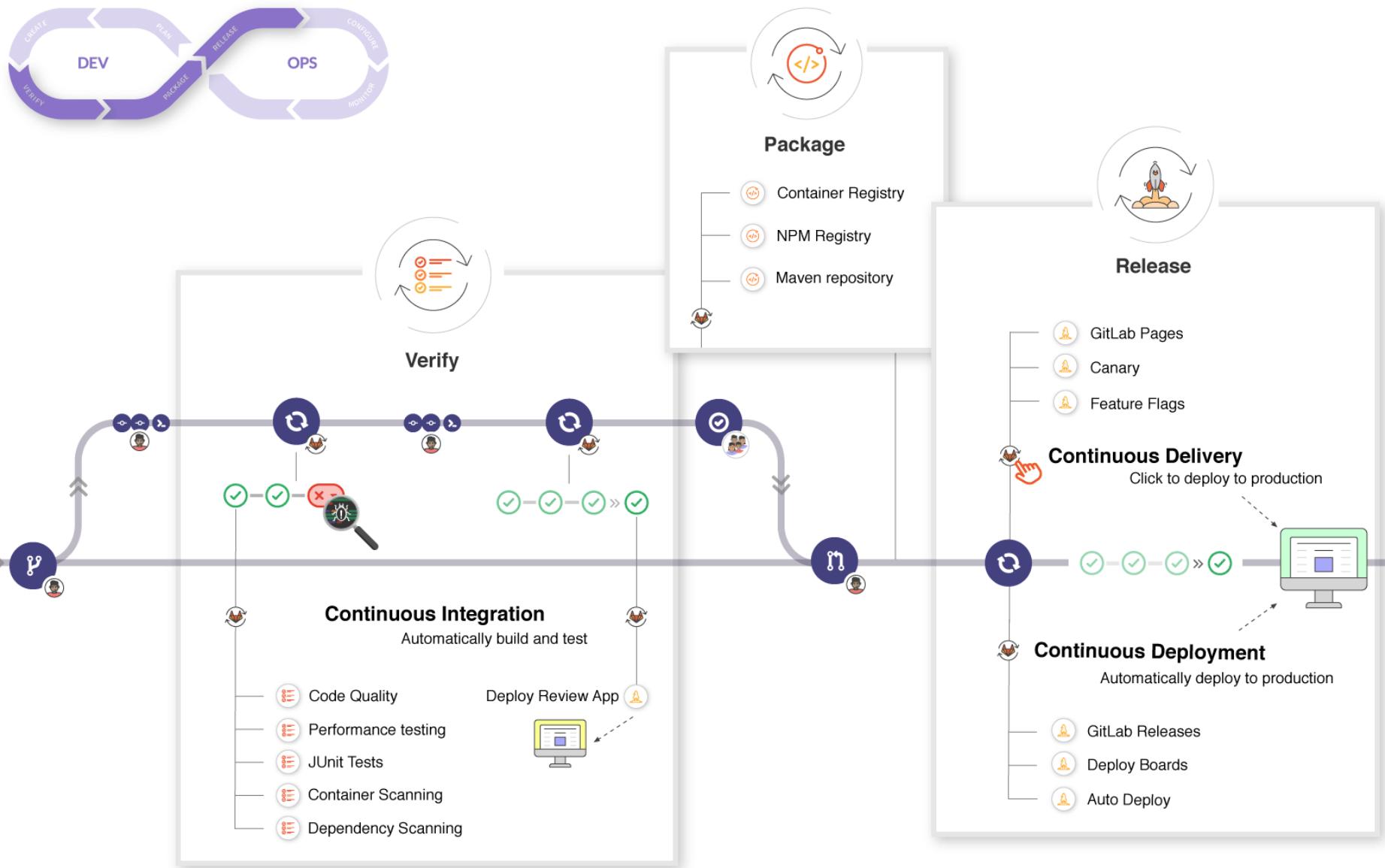
```
name: Node CI

on: [push]

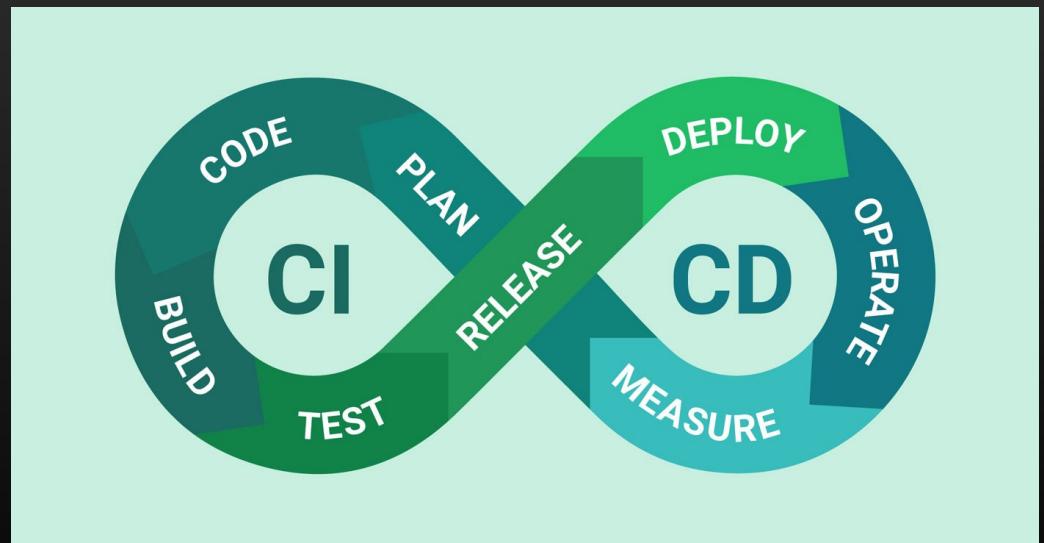
jobs:
  build_and_test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3
      - name: npm install, build, and test
        run: |
          npm install
          npm run build --if-present
          npm test
      - name: Archive production artifacts
        uses: actions/upload-artifact@v3
        with:
          name: dist-without-markdown
          path: |
            dist
            !dist/**/*.{md,markdown}
      - name: Archive code coverage results
        uses: actions/upload-artifact@v3
        with:
          name: code-coverage-report
          path: output/test/code-coverage.html
```

GitLab CI/CD

<https://docs.gitlab.com/ee/ci/introduction/>
→ Includes video demos

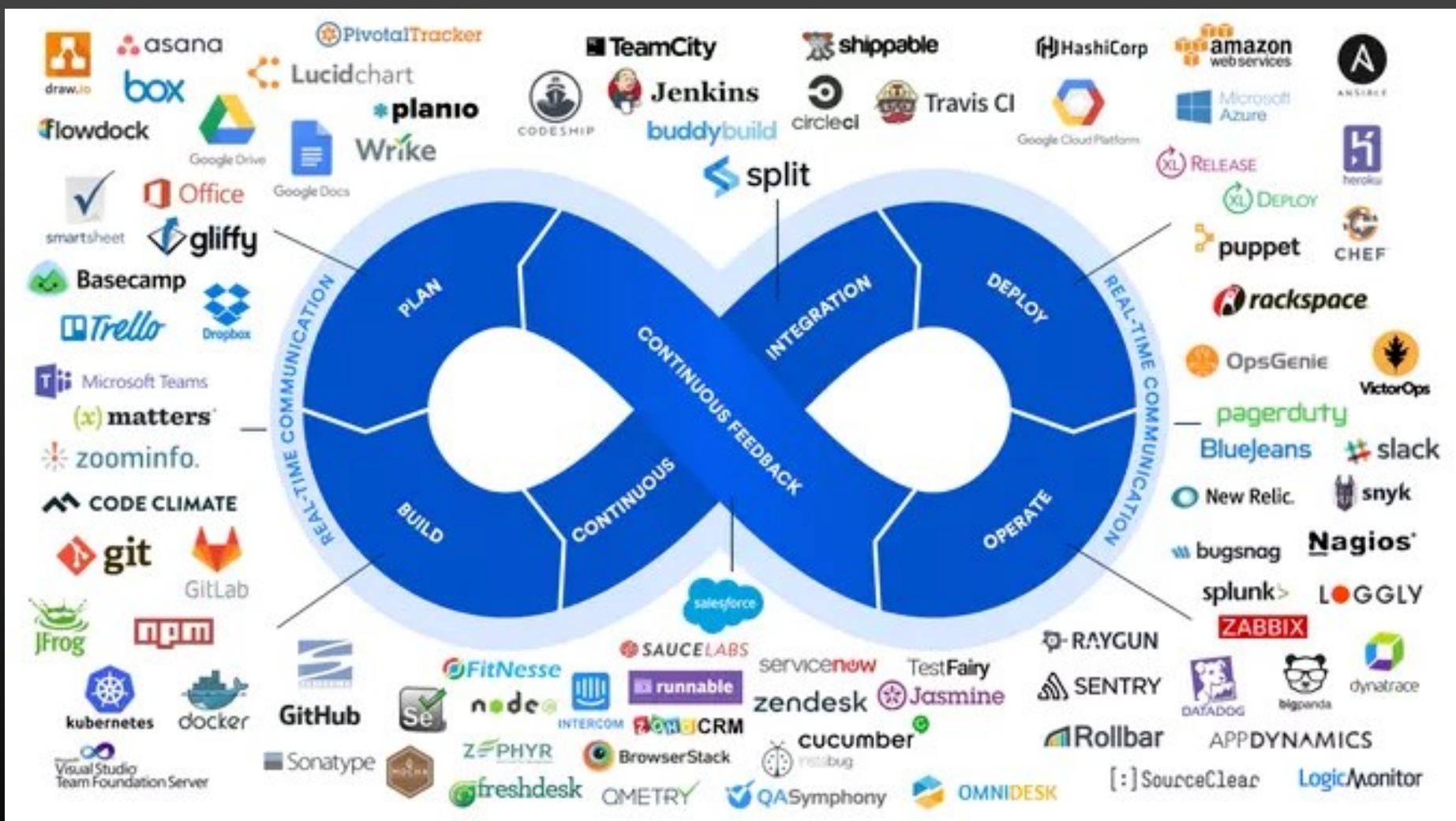


DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams. It emphasizes team empowerment, communication and collaboration, and technology automation.



<https://www.atlassian.com/devops/what-is-devops>

Automação e tecnologias...



More to explore

Books on Continuous integration:

Duvall's Continuous Integration: <http://www.amazon.com/Continuous-Integration-Improving-Software-Reducing/dp/0321336380>

Humble's "Continuous Delivery": <http://www.amazon.com/Continuous-Delivery-Deployment-Automation-Addison-Wesley/dp/0321601912>

Hudson/Jenkins

Extensive information:

<http://www.youtube.com/watch?v=6k0S402PnTc#!>

Maven:

Free ebook: <http://www.sonatype.com/books/mvnref-book/reference/public-book.html>