FASE 12 DO PROJETO
Diagrama de Instalação e Outros

Modelação Estrutural + Notas Finais
(Diagrama de Instalação)

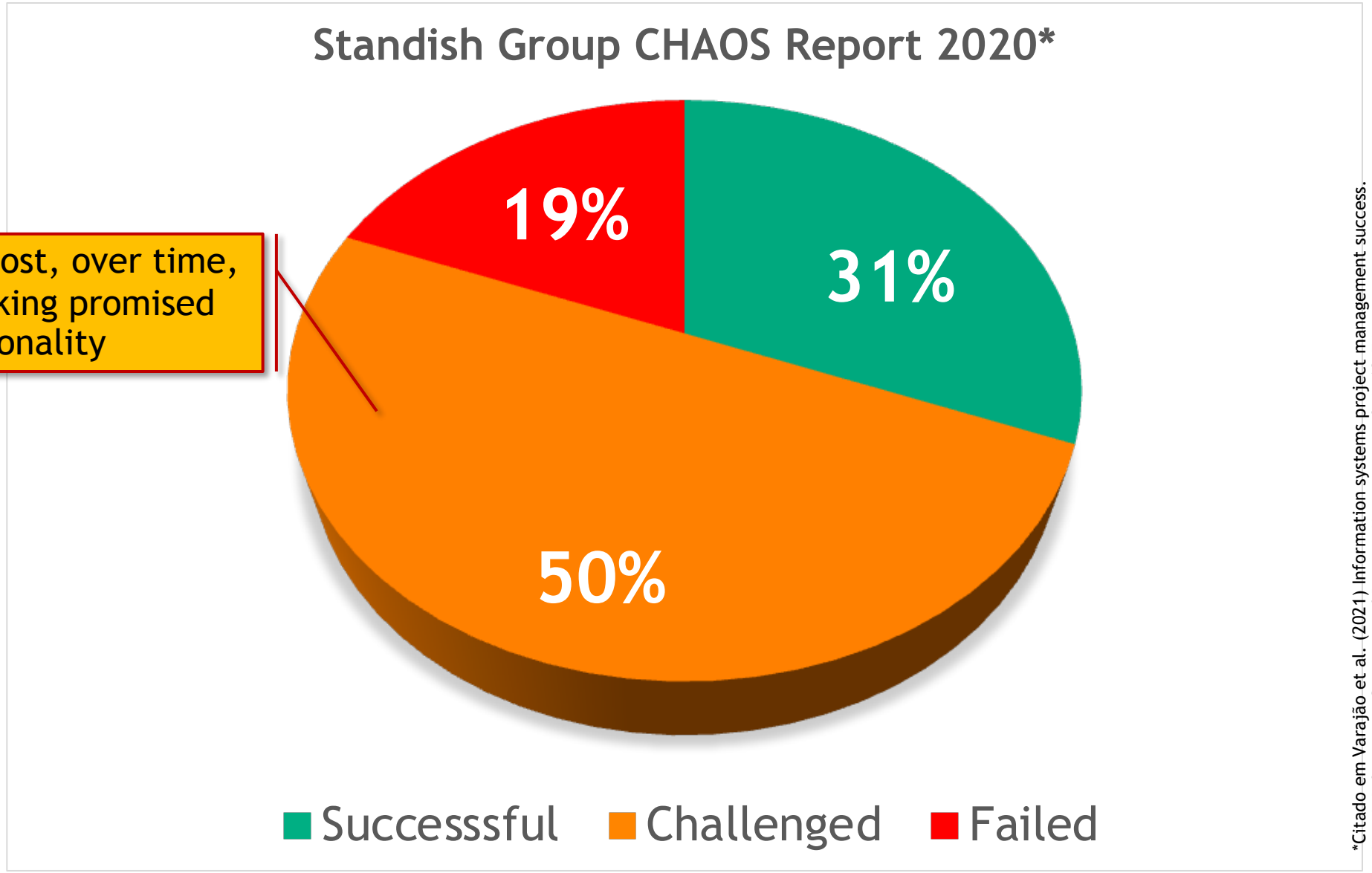
Da primeira aula...

Version 2 announcement:

Among Us 1 (…) started as a tiny local-multiplayer-only game and **has grown and grown and grown**. (…) Because of this, it's **extremely hard to add more things** (…) **because the game is so fragile**. Fixing [it would be] (…) **harder than just making a new game**.

So the first goal of **Among Us 2 is to be made to withstand growth**. We want to add to it at least as long as Among Us, but **with fewer bugs along the way**.

Da primeira aula...



over cost, over time, or lacking promised functionality

*Citado em Varajão et al. (2021) Information systems project management success. Int. J. Information Systems and Project Management, 9(4):62-74.

Standish Group CHAOS Report 2020

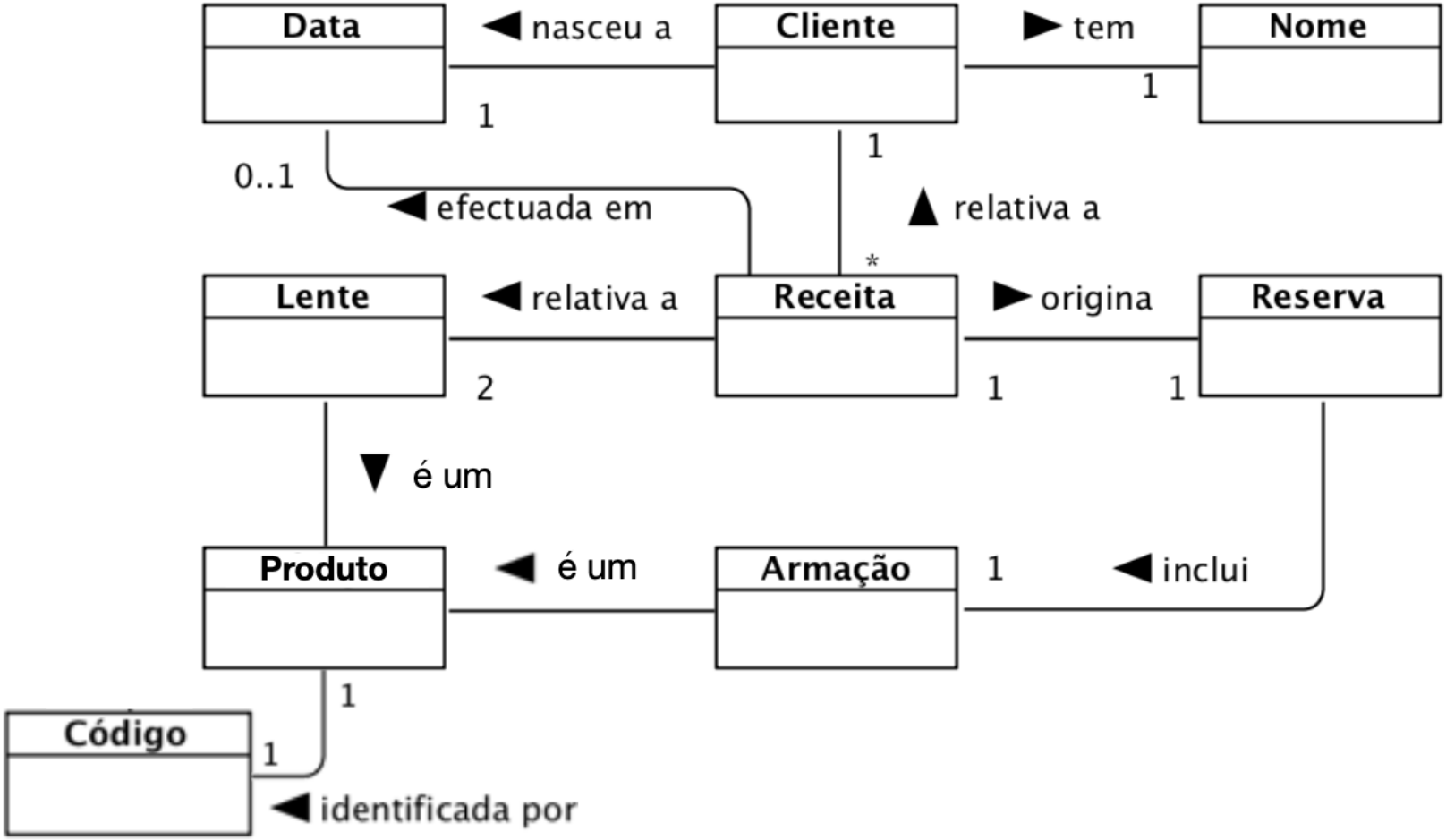
- **Top 5 Factors in Successful IT Projects**
  1. User involvement
  2. Executive management support
  3. Clear Statement of Requirements
  4. Proper planning
  5. Realistic expectations

- **Top 5 Indicators in Challenged IT Projects**
  1. Lack of user input
  2. Incomplete Requirements & Specifications
  3. Changing Requirements & Specifications
  4. Lack of executive support
  5. Technical incompetence

- **Top Factors in Failed IT Projects**
  1. Incomplete Requirements
  2. Lack of user involvement
  3. Lack of resources
  4. Unrealistic expectations
  5. Lack of executive support
  6. Changing Requirements & Specifications
  7. Lack of planning
  8. Didn't need it any longer
  9. Lack of IT management
  10. Technical illiteracy

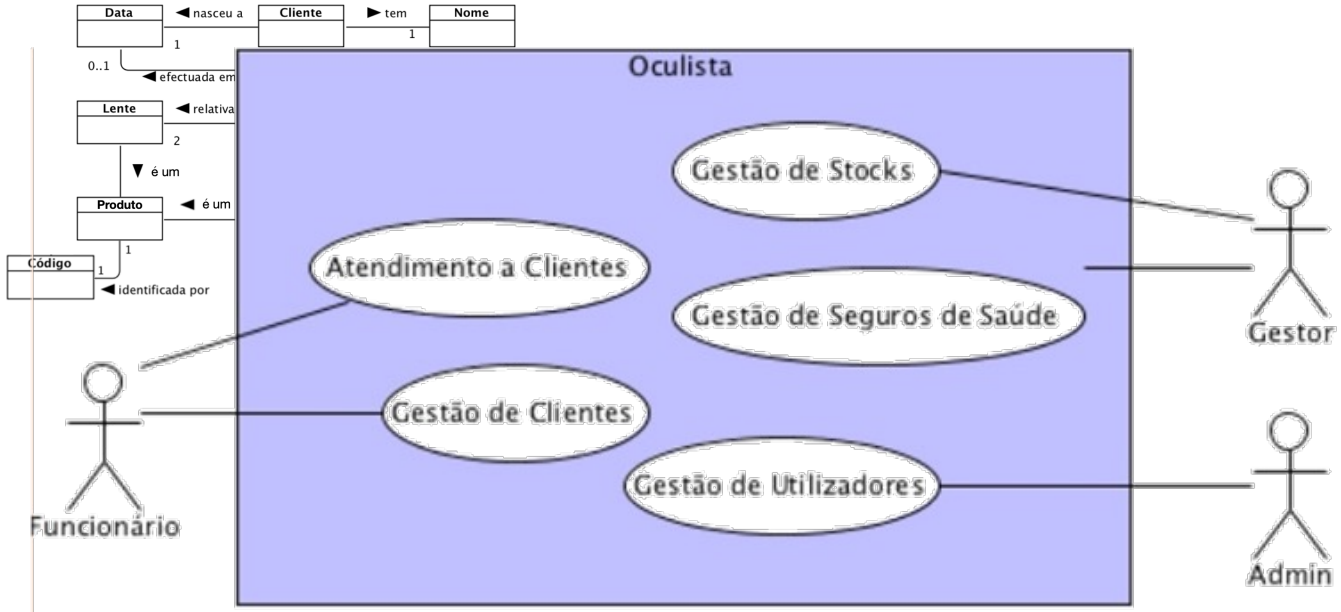
Análise - Perceber o problema

- Modelar o Domínio



Análise - Perceber os requisitos

- Modelo de Use Case



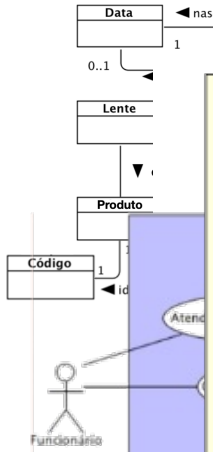
Use Case: Reservar armação e lentes

Descrição: Funcionário regista uma reserva de armação e lentes
Pós-condição: Reserva fica registada
Fluxo normal:
1. Funcionário indica nome e/ou data de nascimento do cliente
2. Sistema procura clientes
3. Sistema apresenta lista de clientes
4. Funcionário selecciona cliente
5. Sistema procura cliente
6. Sistema apresenta detalhes do cliente
7. Funcionário confirma cliente
8. Sistema procura produtos e apresenta lista
9. Funcionário indica Código de armação e lentes
10. Sistema procura detalhes dos produtos
11. Sistema apresenta detalhes dos produtos
12. Funcionário confirma produtos
13. Sistema regista reserva dos produtos
14. <<include>> imprimir talão

Fluxo de excepção: [cliente não quer produto] (passo 12)
11.1. Funcionário regista produtos
11.2. Sistema termina processo

Concepção - Conceber a solução

- Identificar a API da LN



Use Case: Reservar armação e lentes

Descrição: Funcionário regista uma reserva de armação e lentes

Pós-condição: Reserva fica registada

Fluxo normal:

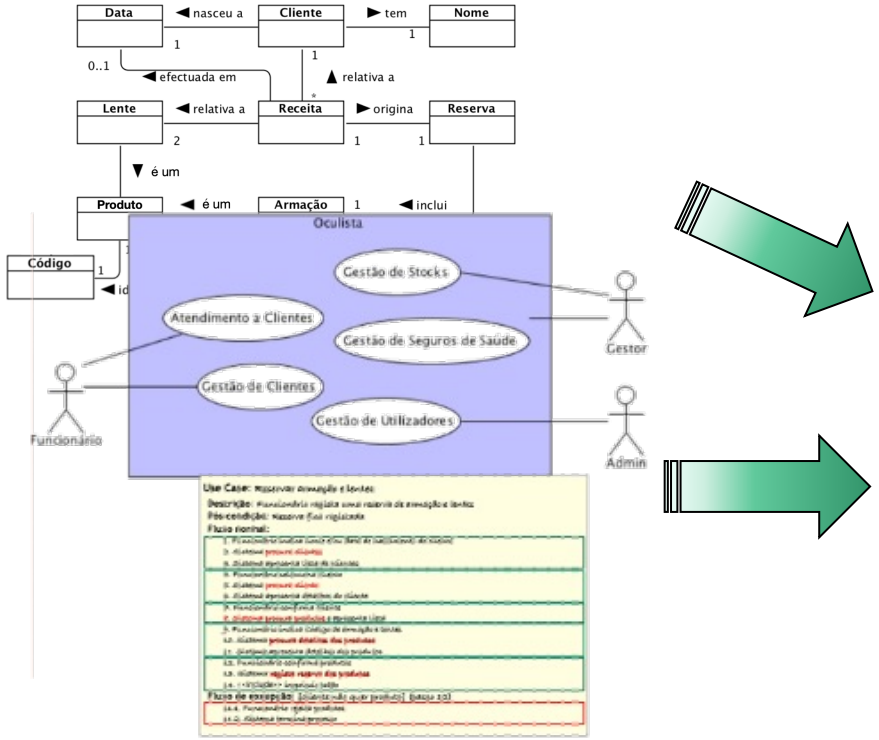
1. Funcionário indica nome e/ou data de nascimento do cliente
2. Sistema procura clientes
3. Sistema apresenta lista de clientes
4. Funcionário selecciona cliente
5. Sistema procura cliente
6. Sistema apresenta detalhes do cliente
7. Funcionário confirma cliente
8. Sistema procura produtos e apresenta lista
9. Funcionário indica Código de armação e lentes
10. Sistema procura detalhes dos produtos
11. Sistema apresenta detalhes dos produtos
12. Funcionário confirma produtos
13. Sistema regista reserva dos produtos
14. <<include>> imprimir talão

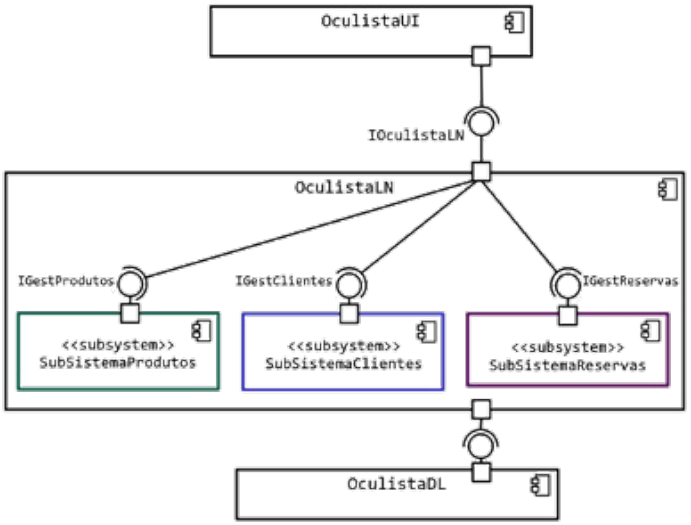
Fluxo de excepção: [cliente não quer produto] (passo 12)

11.1. Funcionário rejeita produtos
11.2. Sistema termina processo

Concepção - Conceber a solução

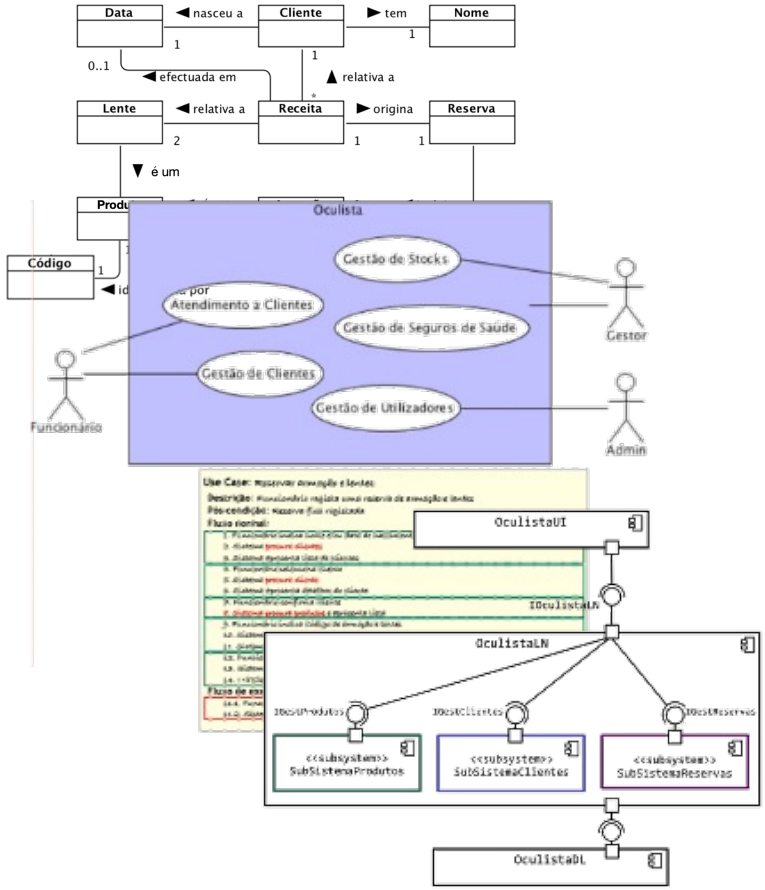
- Identificar subsistemas e suas APIs

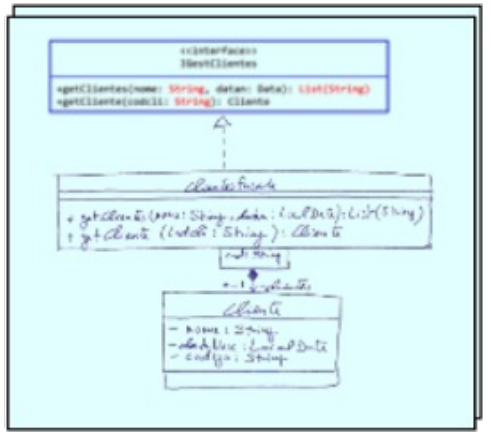


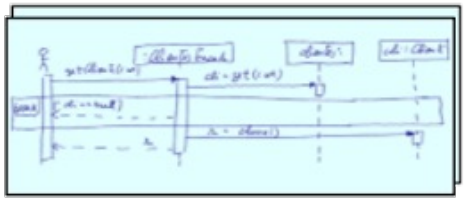


Concepção - Conceber a solução

- Conceber os subsistemas

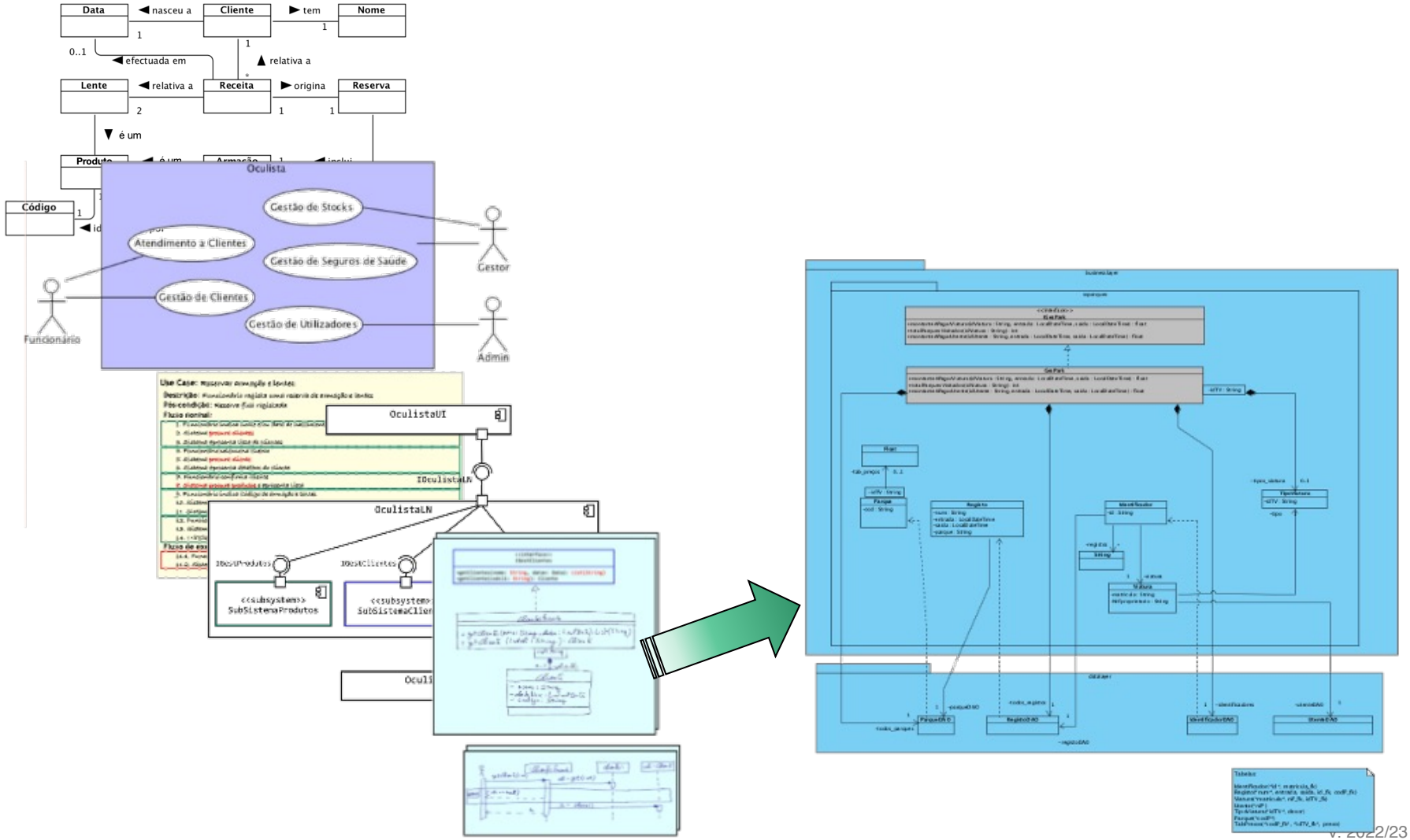






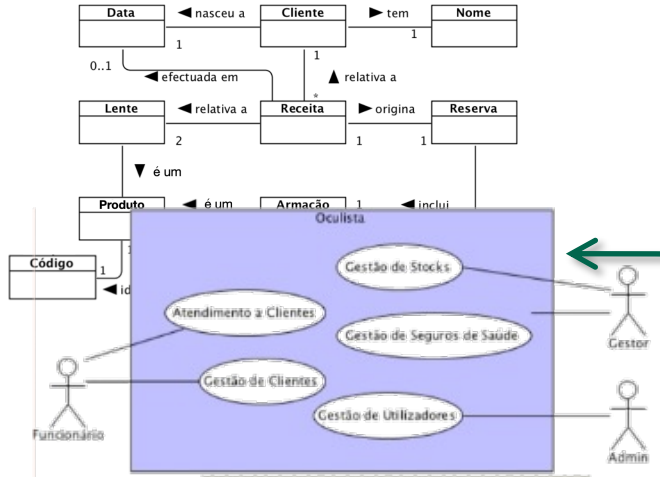
Concepção - Conceber a solução

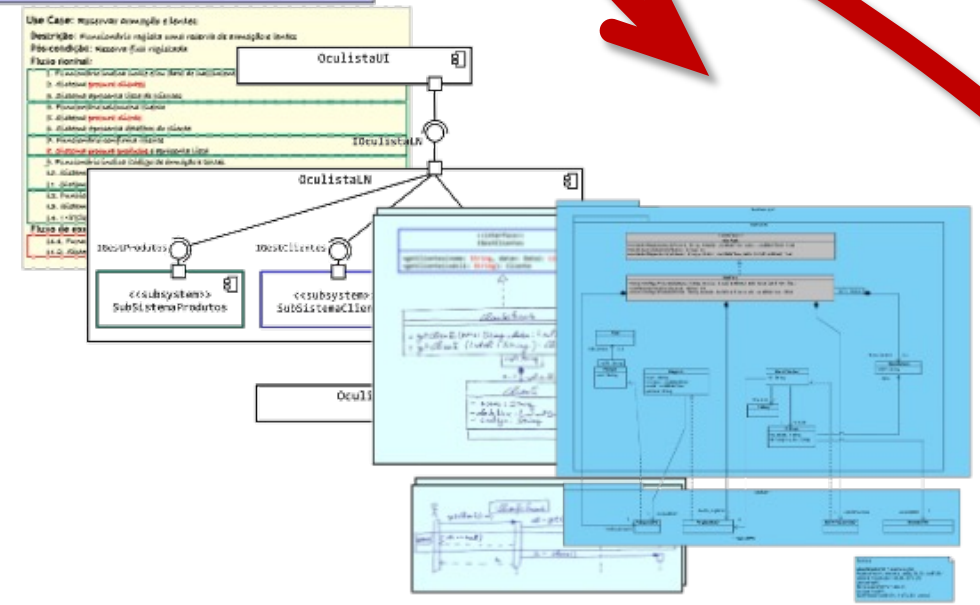
- Ajustar à solução tecnológica concreta



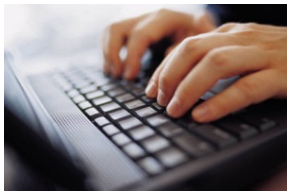
Análise ➜ Implementação

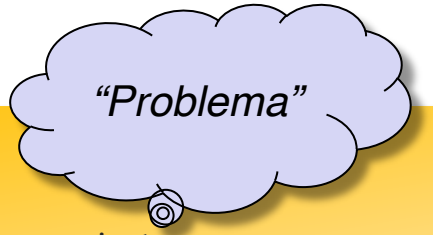
- Implementar, testar e instalar





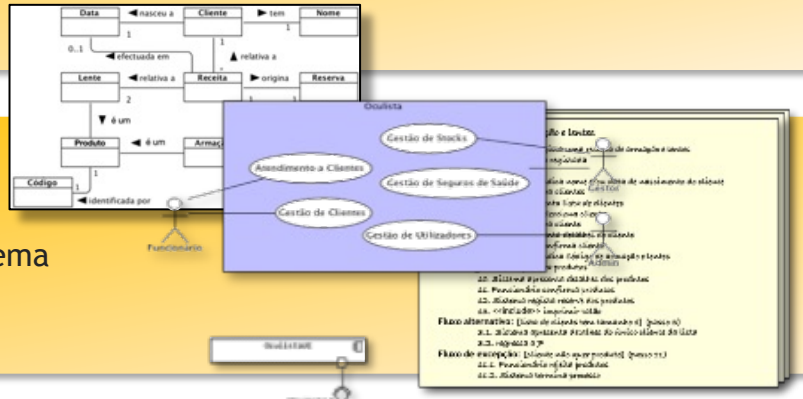
Atenção!
Não é um processo em cascata!





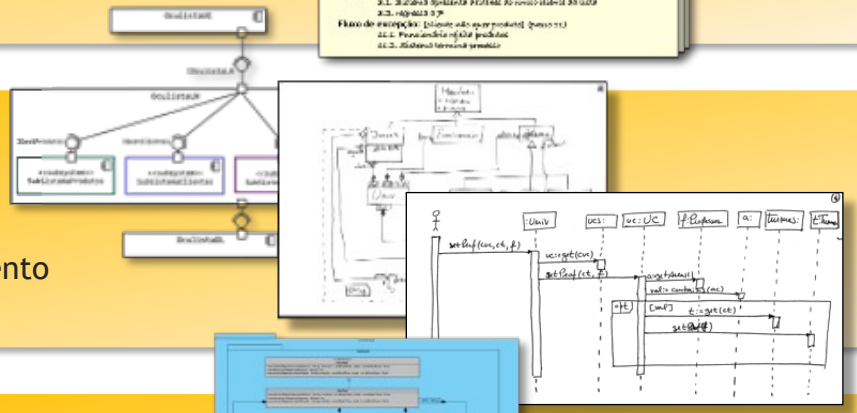
Planeamento

- Decisão de avançar com o projecto
- Gestão do projecto



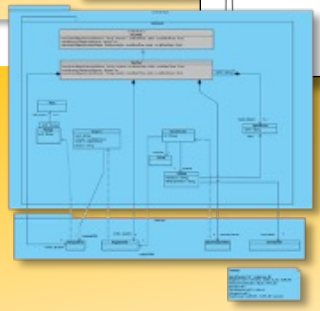
Análise

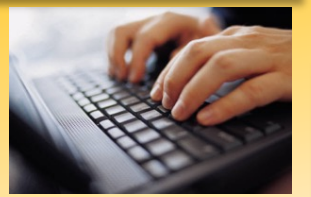
- Análise do domínio do problema
- Análise de requisitos



Concepção

- Concepção da Arquitectura
- Concepção do Comportamento





Implementação

- Construção
- Teste
- Instalação
- Manutenção

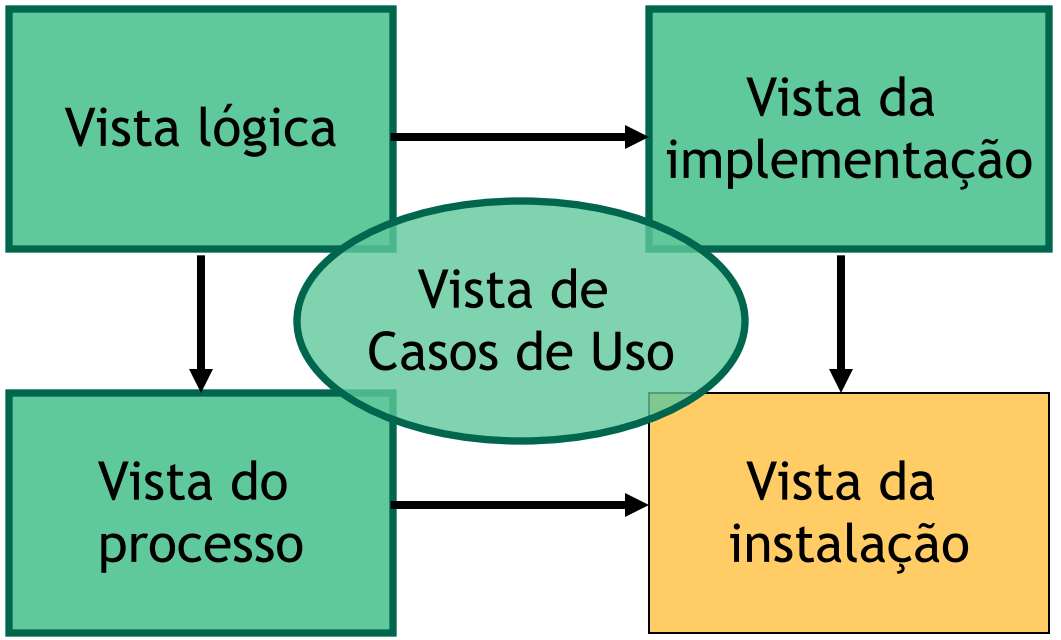
Processo iterativo e incremental - o trabalho

- Fase 1 – Perceber os requisitos
  - 5 cenários para perceber o problema
- Fase 2 – **Primeira** iteração de conceber o sistema
  - Focar concepção no Cenário 5
- Fase 3 – **Primeira** iteração de implementar o sistema
  - Focar implementação no Cenário 5 - mas fazer iterativamente
  - Ordem sugerida
    - Simular Corridas
    - Configurar Corridas
    - Configurar Campeonato
  - Implementar… mas também testar e instalar!

Sobre testar…

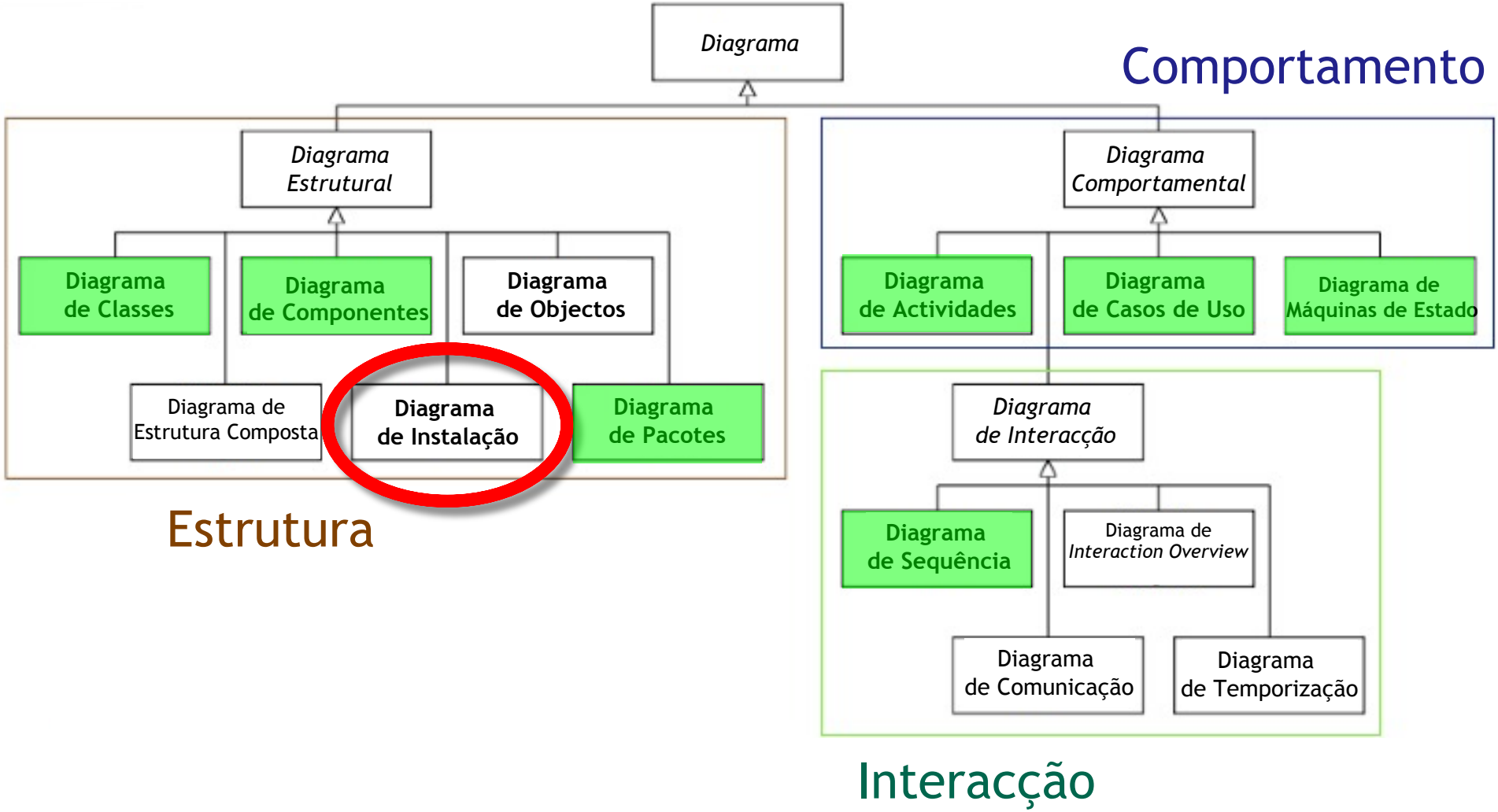
- Os Use Case permitem a melhorar a qualidade do processo de testes → sistema mais robusto e confiável
- Fornecem uma **descrição clara e concisa** dos passos necessários à utilização **de uma funcionalidade**
  - Ajudam a compreender o que é necessário fazer para testar o sistema
  - Ajudam a garantir que todas as etapas necessárias são incluídas no plano de teste
- Ajudam a identificar potenciais problemas ou áreas do sistema mais propensas a falhas
  - A análise da descrição do Use Case permite identificar potenciais pontos de falha ou áreas onde o sistema pode não se comportar como esperado

Sobre a instalação do sistema…

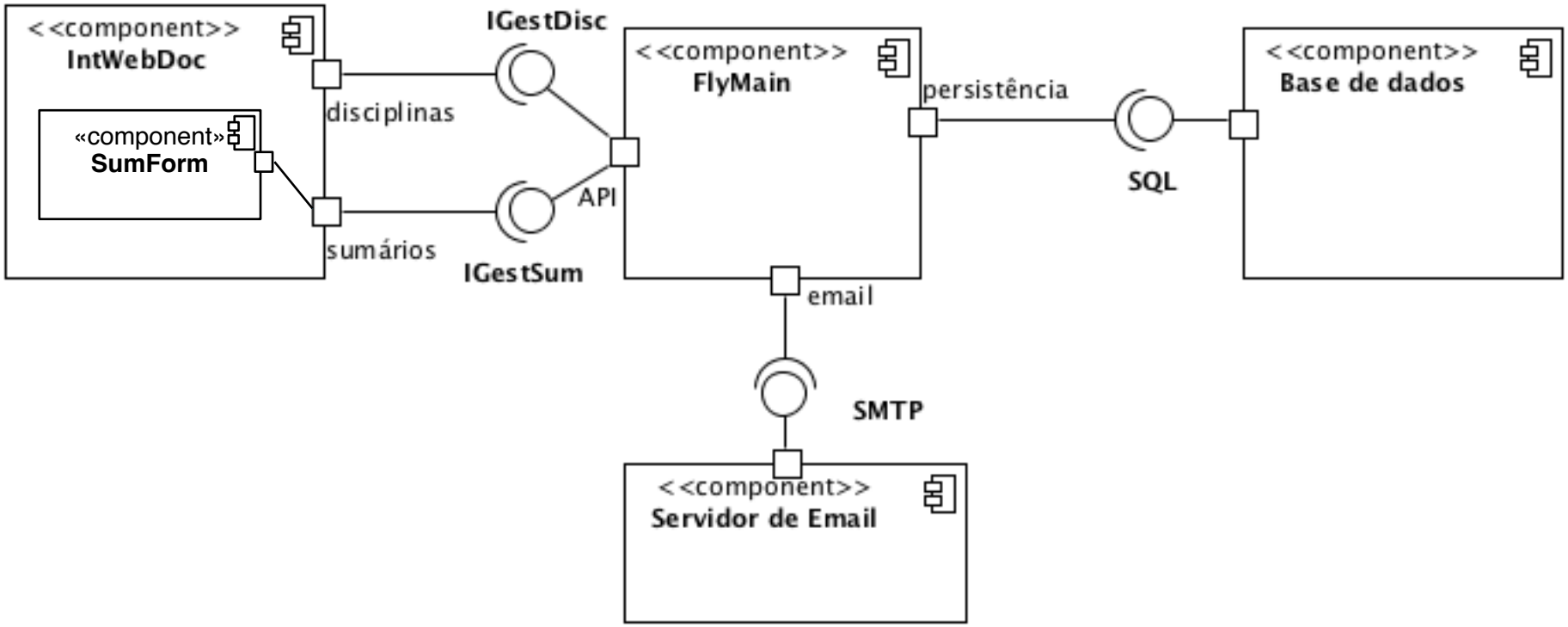


(Kruchten, 1995)

Diagramas da UML 2.x



Um sistema divide-se em Componentes

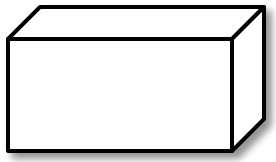


Diagramas de Instalação *(Deployment)*

- Qual a disposição física dos componentes que constituem o sistema?
  - Qual é a configuração do sistema em tempo de execução?
- Diagramas de Instalação especificam a arquitectura física do sistema
  - Topologia (ambiente) de hardware sobre a qual são executados os componentes de software
- Permitem:
  - Especificar a distribuição de componentes
  - Identificar estrangulamentos de desempenho

Diagramas de Instalação

- Elementos de um diagrama de deployment
  - Nós
  - Ligações

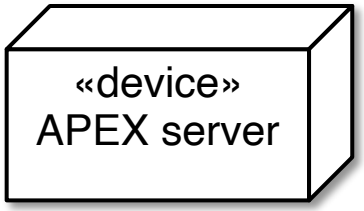


- **Nós (*nodes*):**
  - Computadores ou outros dispositivos (**hardware**)
  - Sistema operativo, *web servers*, *application servers*, etc. (**ambientes de execução**)
  - Os componentes localizados (*deployed*) em cada nó são representados explicitamente
  - É possível agrupar nós em pacotes (*packages*)

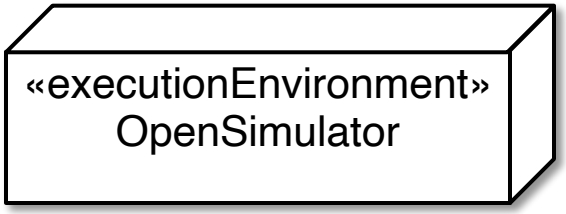
- **Ligações (*connections*):**
  - Representam **comunicação** entre os nós.
  - Podem ser decoradas com multiplicidades.
  - Podem ter estereótipos que indicam o tipo de ligação.
    - Exemplo: «TCP/IP» ou «RMI»

Diagramas de Instalação

- Por vezes utiliza-se o estereótipo **«device»** para identificar os nós de **hardware**



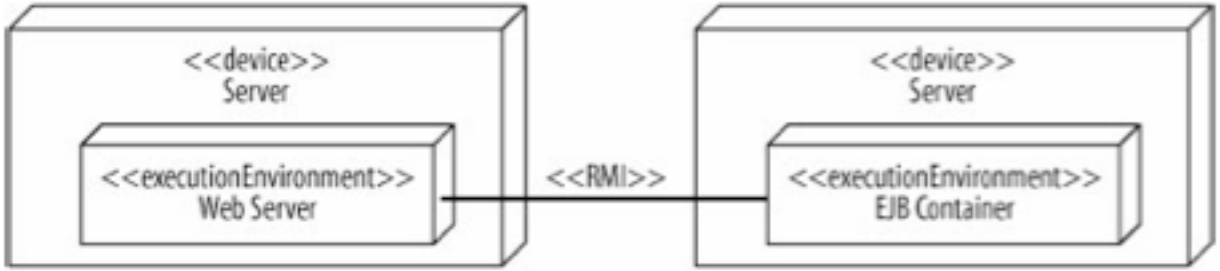
- Para identificar os **ambientes de execução** pode utilizar-se o estereótipo **«executionEnvironment»**



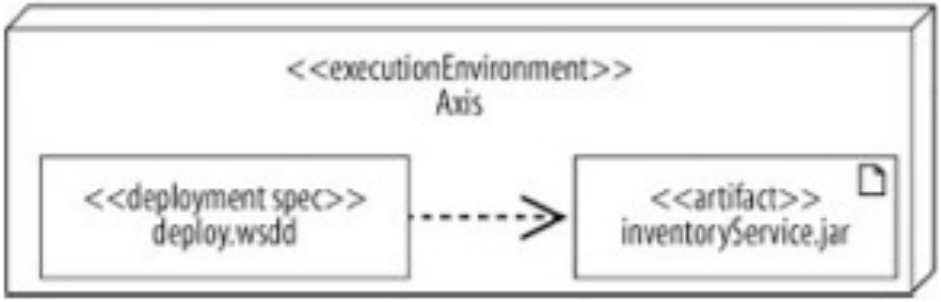
- Comunicação entre dois nós

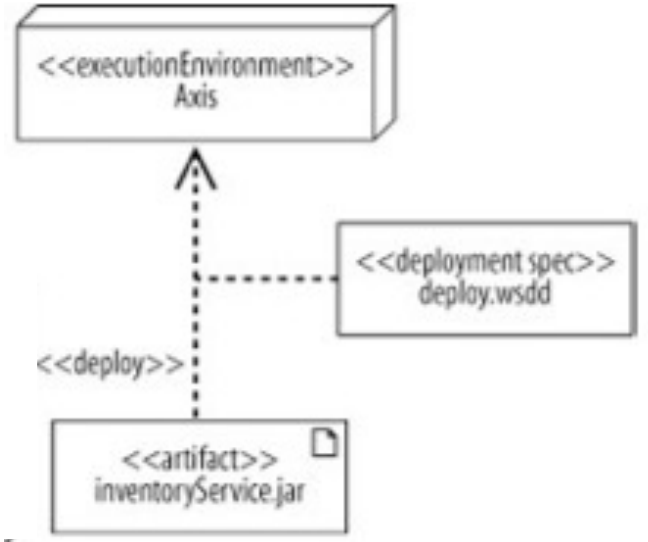
Diagramas de Instalação

- A descrição pode ser refinada para detalhar os ambientes de execução em cada nó

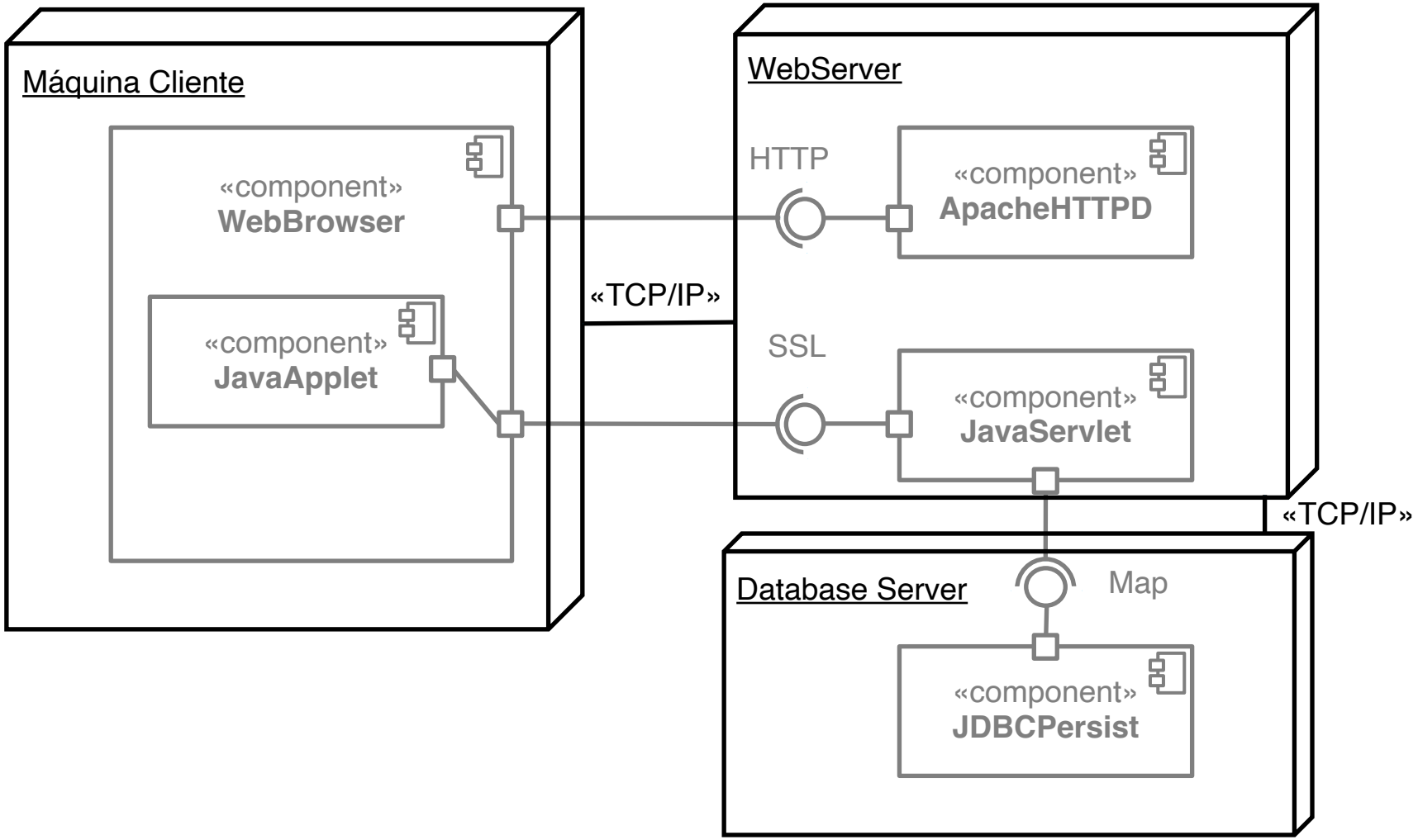


- Especificação de dependências em tempo de execução



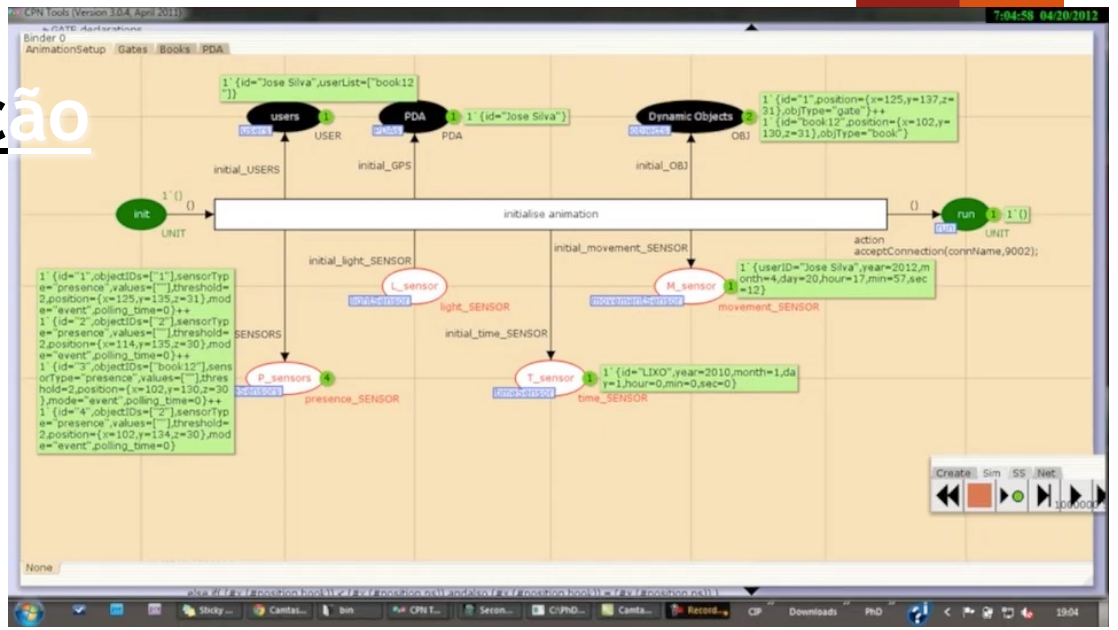


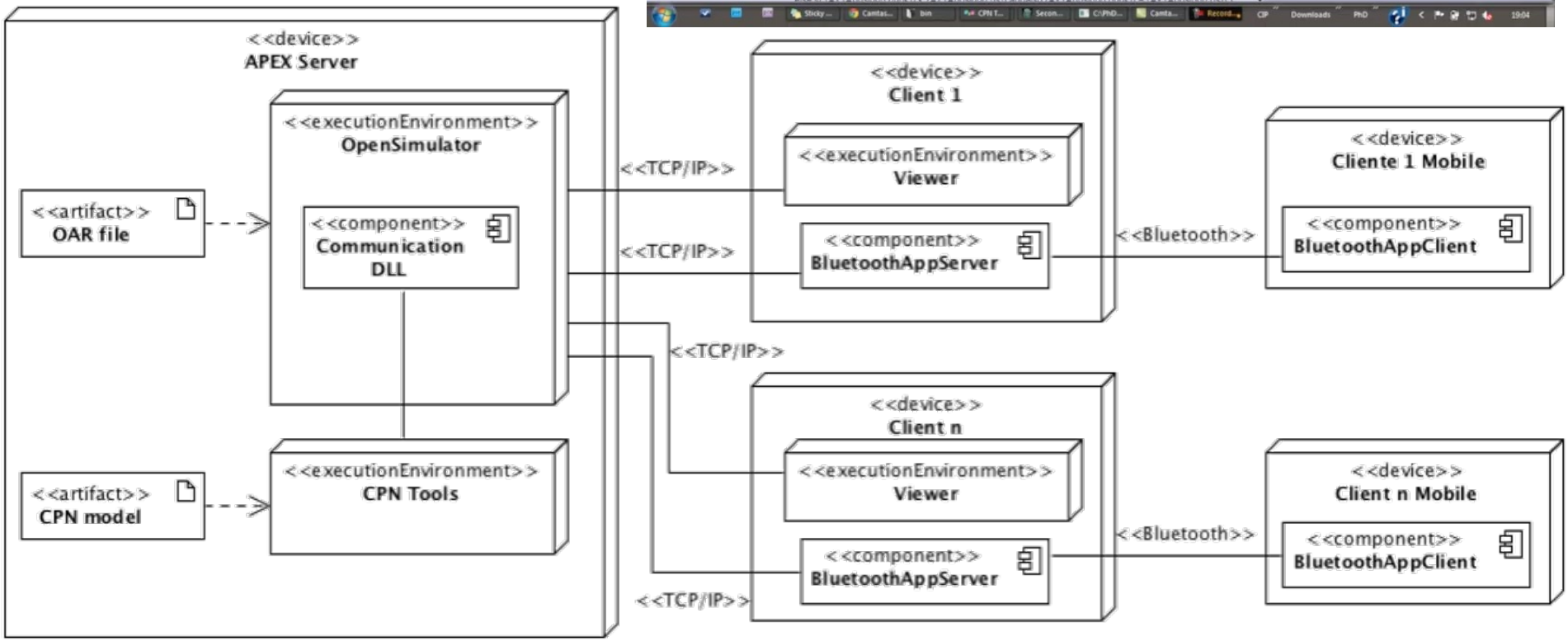
Instalação e componentes



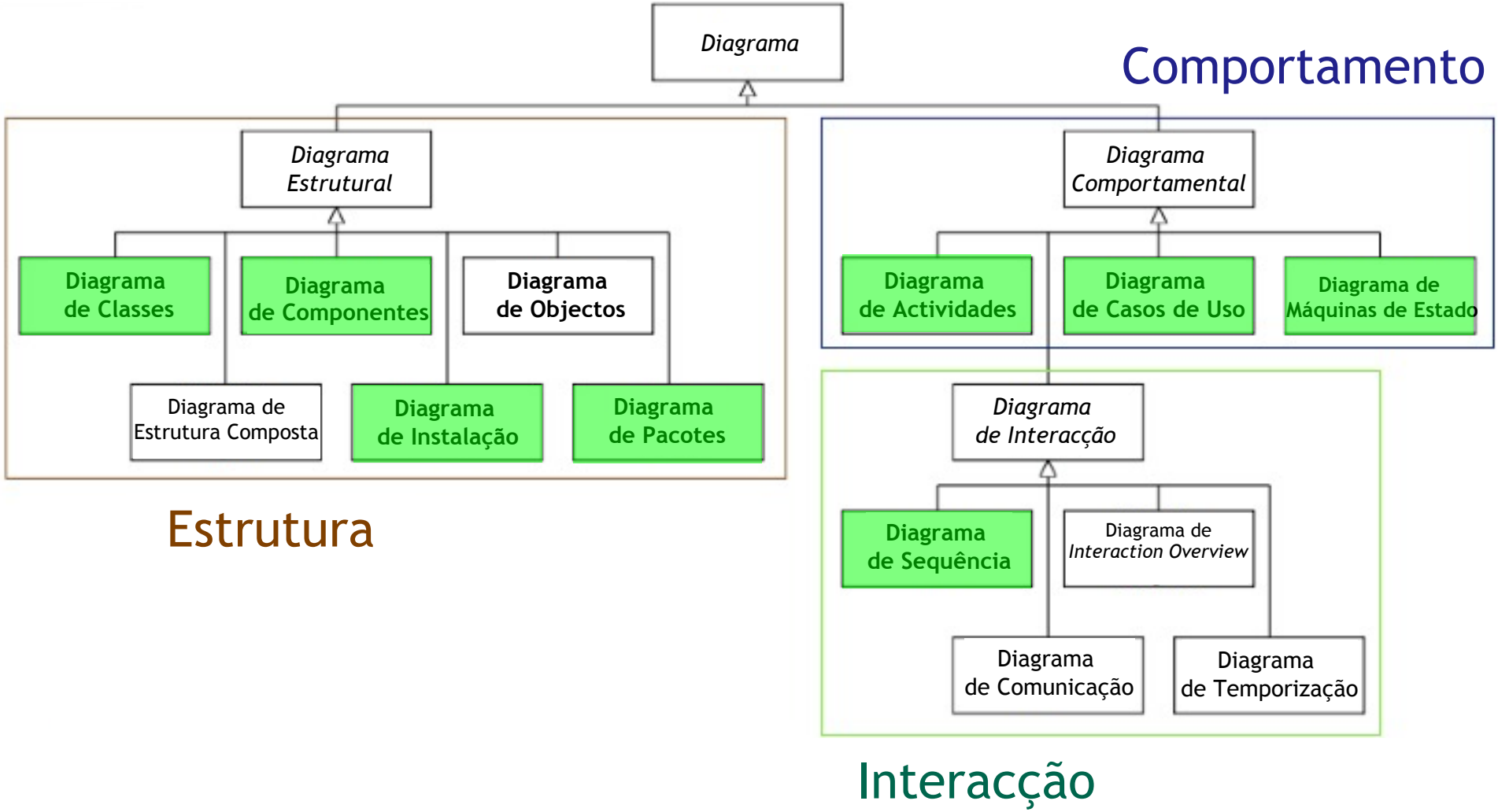
Diagramas de Instalação

- Um exemplo



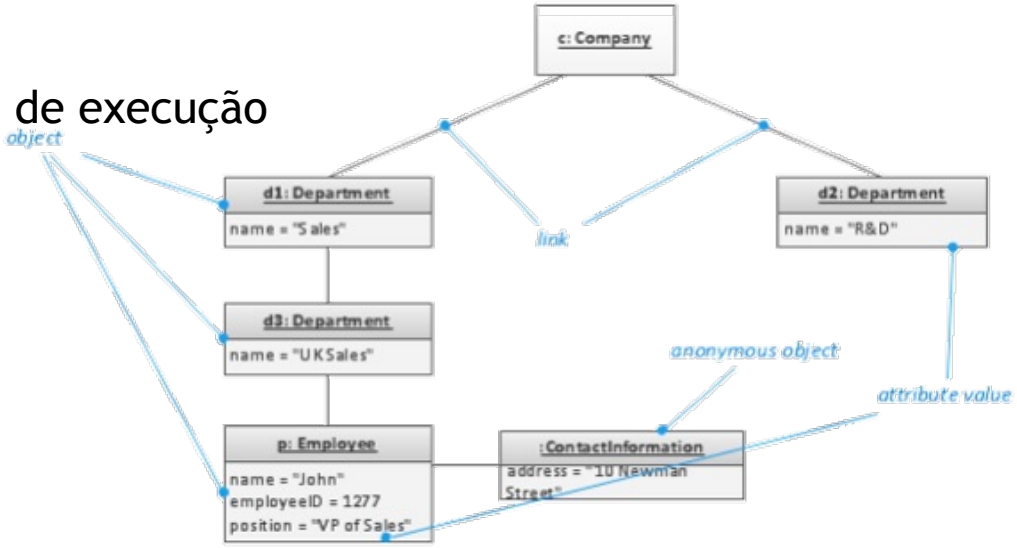


Diagramas da UML 2.x

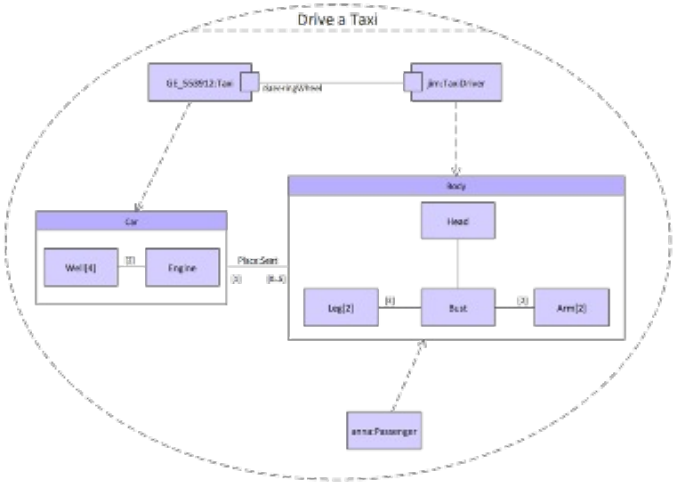


Diagramas de que não falamos

- Diagramas de objectos
  - Mostra instâncias em tempo de execução

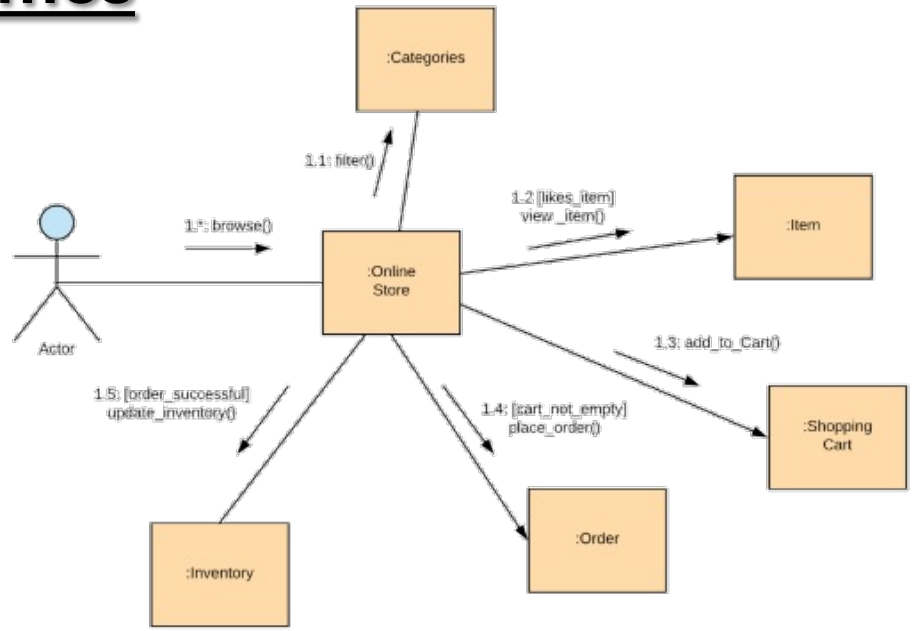


- Diagramas de estrutura composta
  - Mostra a estrutura interna de uma classe em termos de suas partes e das relações entre elas

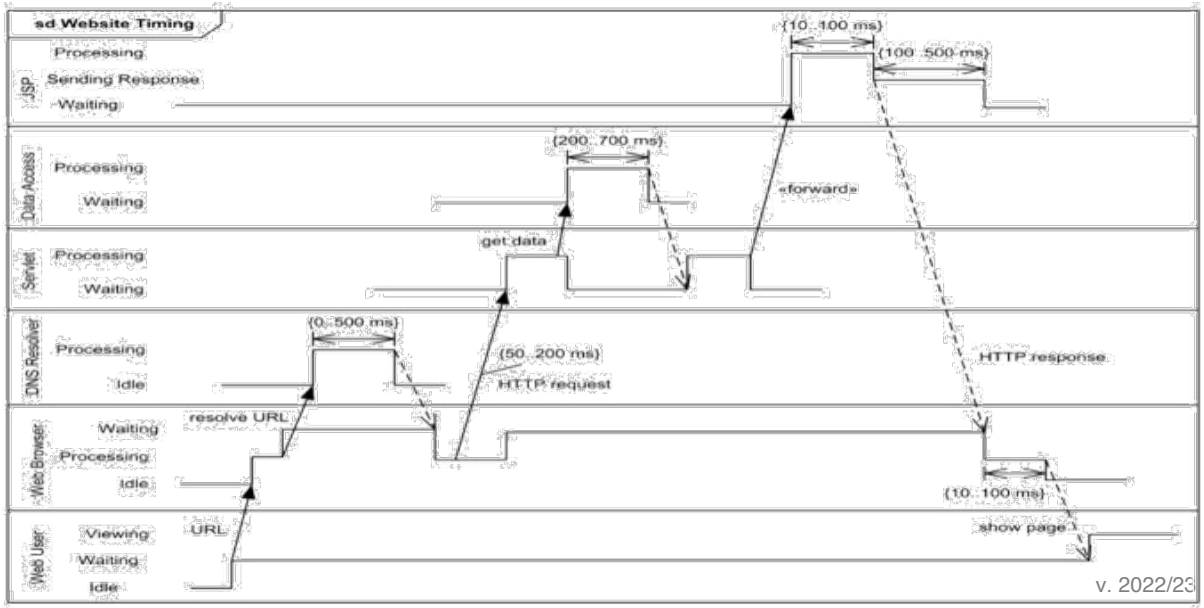


Diagramas de que não falamos

- Diagramas de Comunicação
  - Uma alternativa aos diagramas de sequência, focada na relação estrutural entre os objectos

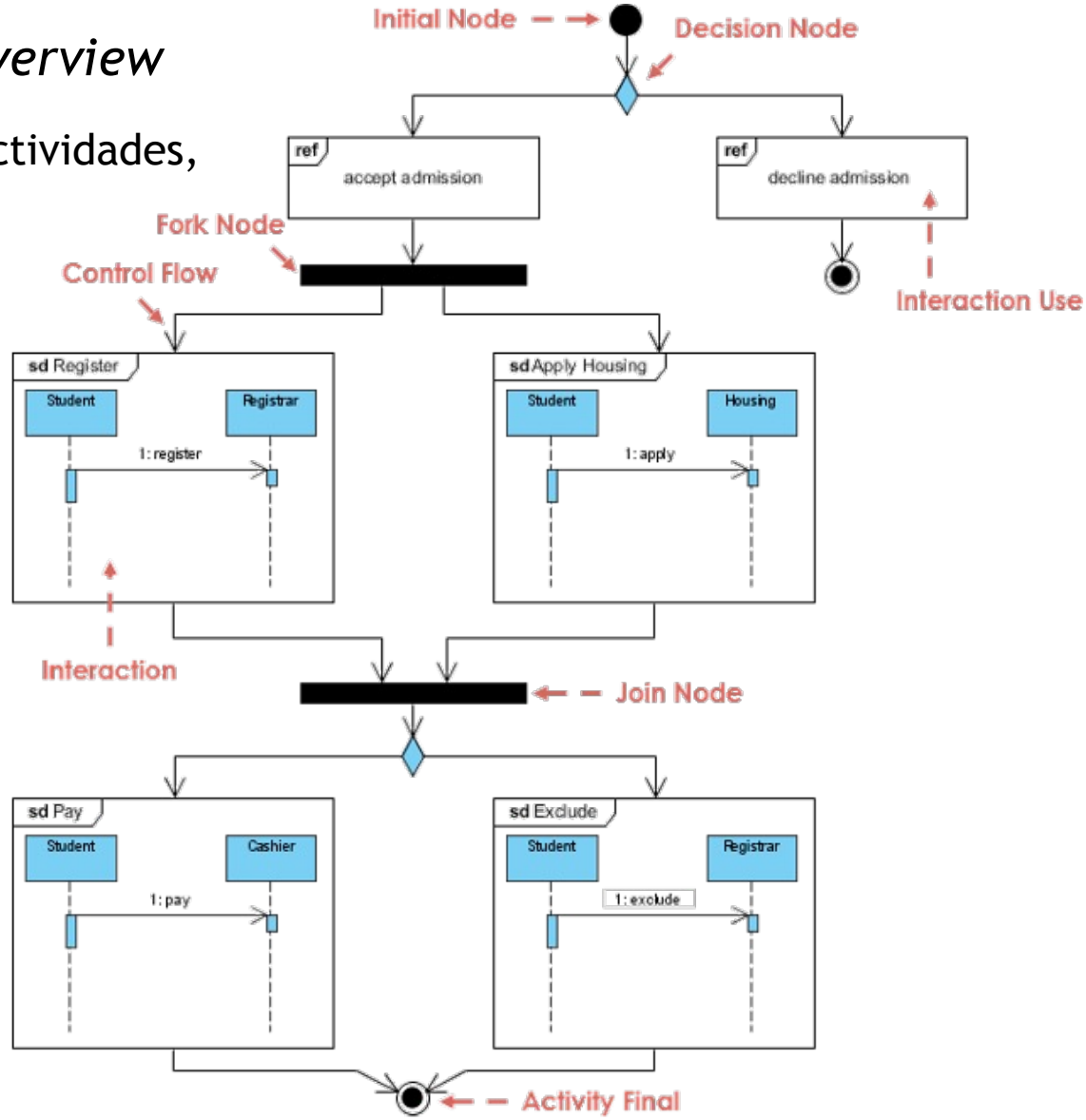


- Diagramas de Temporização
  - Focados na representação de tempo

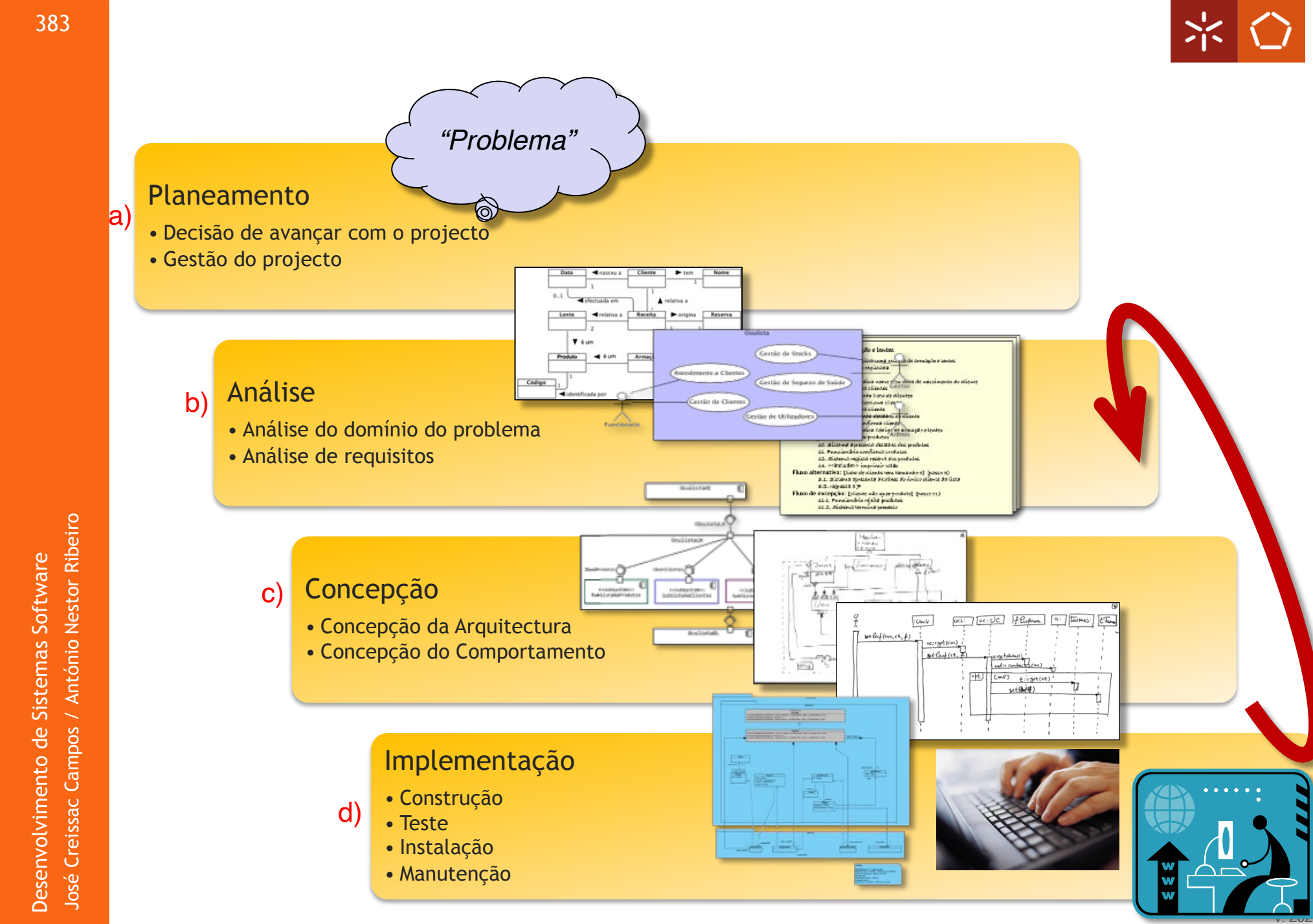


Diagramas de que não falamos

- Diagramas de *Interaction Overview*
  - Semelhante ao diagrama de actividades, com diagramas de sequência como actividades



Notas finais


"Problema"
a)
Planeamento
• Decisão de avançar com o projecto
• Gestão do projecto
b)
Análise
• Análise do domínio do problema
• Análise de requisitos
c)
Concepção
• Concepção da Arquitectura
• Concepção do Comportamento
d)
Implementação
• Construção
• Teste
• Instalação
• Manutenção

Futuro da (na!) programaçã

Forbes / Entrepreneurs / #CuttingEdge

JUL 11, 2016 @ 04:41 AM 22,468

Robots Replacing Developers? This Startup Uses Automation To Build Smart Software

Julian Mitchell, CONTRIBUTOR
I cover entrepreneurs and startups disrupting industries.
FULL BIO

Quora Home Answer Notifications Search Quora

Could artificially intelligent computers replace programmers in the future?

Answer Request Follow 124 Comment 1 Downvote

Promoted by VisionMobile

How well do you know your tools and platforms?
Take the new developer economics survey, along with another 40,000 developers and find out!

Start now at vmob.me

73 Answers

Joscha Bach
Answered Jun 9, 2016

I am surprised that most answers seem to be in the negative, or think that programming is one of the last fields where human labor becomes obsolete. General artificial intelligence will of course replace programmers at some point: the question is not if, but when, and at what level of sophistication of AI.

COMMUNICATIONS OF THE ACM

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE

Home / News / AI Will Replace Coders by 2040, Warn Academics / Full Text

ACM TECHNEWS

AI Will Replace Coders by 2040, Warn Academics

By V3.co.uk
December 8, 2017
Comments

VIEW AS: SHARE:

Display a menu

Coders and programmers could find themselves becoming marginalized by emerging technologies such as artificial intelligence, with humans being replaced in these jobs by 2040, according to a study from academic researchers published by Oak Ridge National Laboratory.

The report predicted by 2040, machine learning and natural language processing technologies will have become so advanced they will be able to write better software code faster than the best human coders.

In addition, "the major technologies that will drive the creation and adoption of machine-generated code already exist, either at

SIGN IN for Full Access
User Name
Password
» Forgot Password?
» Create an ACM Web Account
SIGN IN

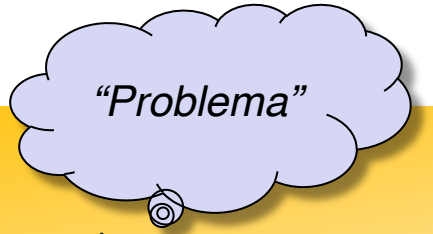
MORE NEWS & OPINIONS
As Scrutiny Of Social Networks Grows, Influence Attacks Continue In Real Time
NPR
Connectivity Hacking Back Makes a Comeback—But It's Still

OVERVIEW LEARN HIGHEST RATED FEATURES

Best No-Code Development Platforms Software

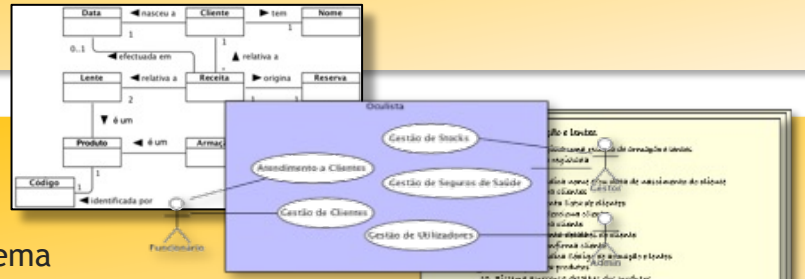
No-Code Development Platform provide drag-and-drop tools that that allow businesses to develop software quickly without coding. The platforms provide WYSIWYG editors and drag-and-drop components to quickly assemble and design applications. Both developers and nondevelopers can use these tools to practice rapid application development with customized workflows and

SHOW MORE



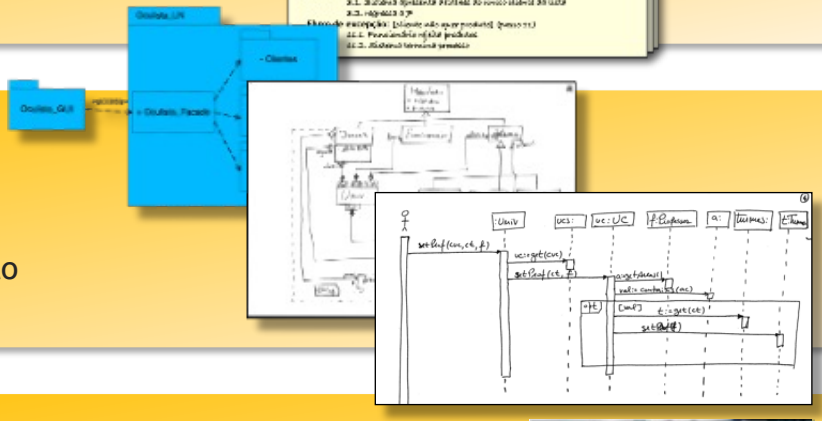
Planeamento

- Decisão de avançar com o projecto
- Gestão do projecto



Análise

- Análise do domínio do problema
- Análise de requisitos



Concepção

- Concepção da Arquitectura
- Concepção do Comportamento

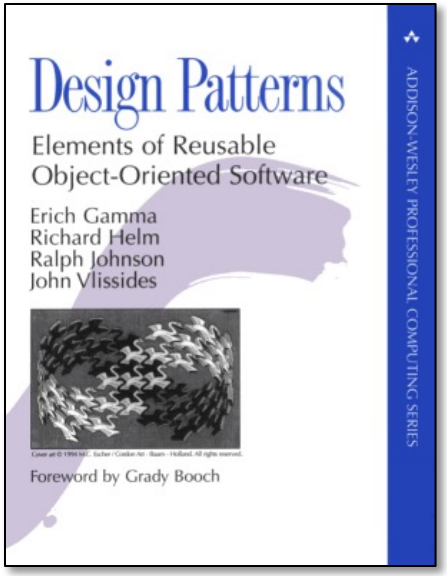
Implementação

- Construção
- Teste
- Instalação
- Manutenção

Relembrar…

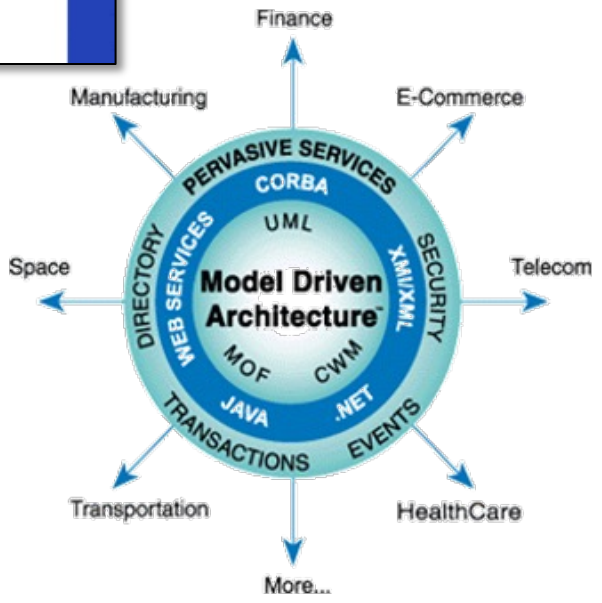
- ECTS
  - 5 ECTS = 140h
    - 15 semanas -> 140/15 = 9h20m/semana (**para DSS!**)
  - Aulas = 4h/semana (na verdade são só 13 semanas!)
    - Trabalho autónomo = 5h20m/semana = ~1h/dia útil
  - 1h por dia = 75h de trabalho autónomo em 15 semanas (**por aluno!**)
- Teste / Exame
  - Podem levar duas folhas (4 páginas) com a informação que considerarem relevante

Mais...



- Software design patterns

- Model Driven Engineering
  - Model Driven Arquitecture (OMG)



- <pub>Engenharia de Aplicações</pub>
  - Desenvolvimento de Aplicações Multi-camada (Web)