

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Desenvolvimento de Sistemas de Software

LEI - 3º ano/1º semestre

Apresentação 2025/2026

António Nestor Ribeiro
anr@di.uminho.pt

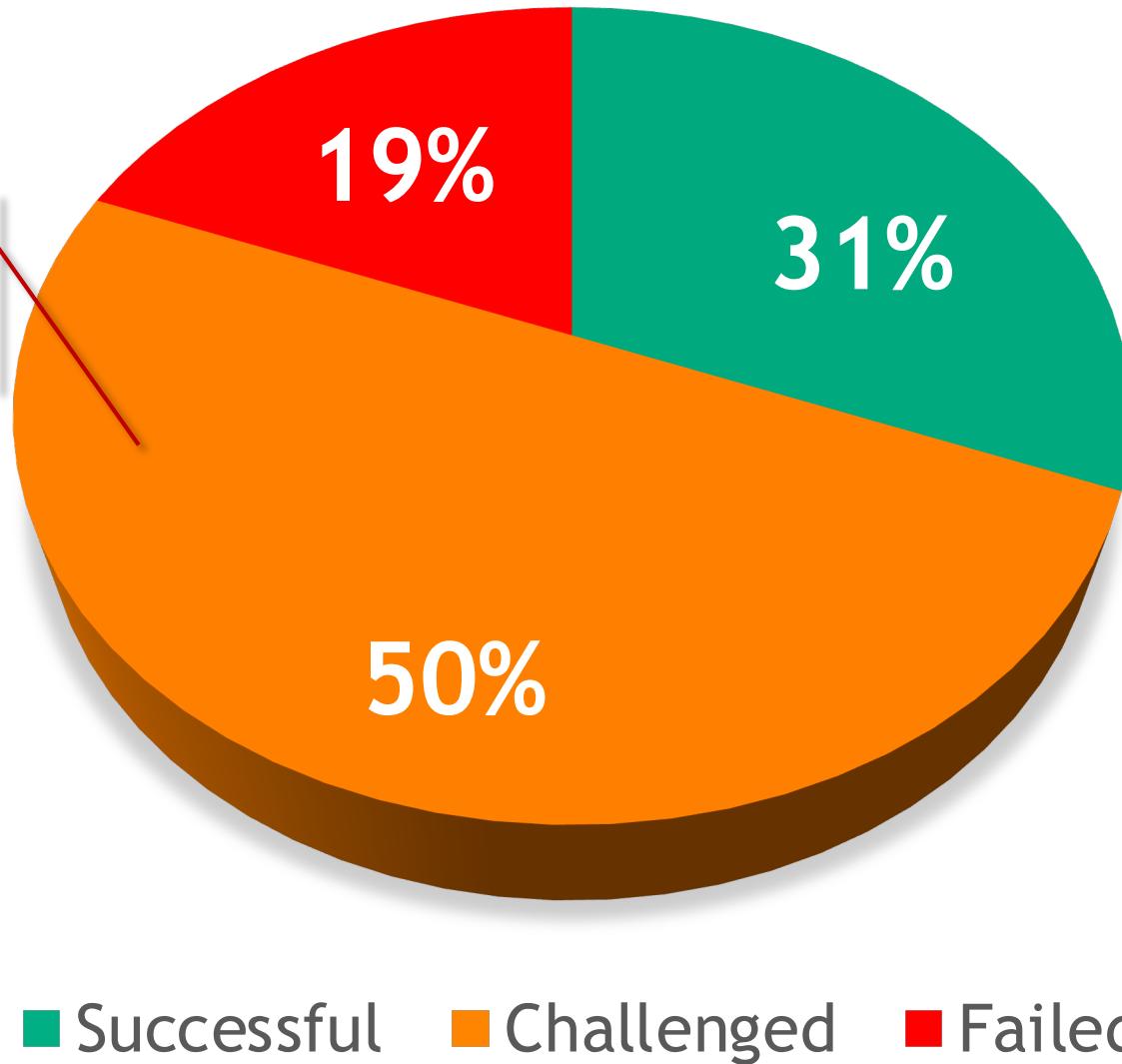
José Creissac Campos
jose.campos@di.uminho.pt

<http://www.di.uminho.pt>

Como vai o desenvolvimento de Software?

Standish Group CHAOS Report 2020*

over cost, over time,
or lacking promised
functionality



Por exemplo...

18:40

Qual é o maior desafio enfrentado pelo grupo VW no que diz respeito ao lançamento de novos modelos de carros?



Copilot 18:40

O maior desafio enfrentado pelo grupo Volkswagen no lançamento de novos modelos de carros tem sido **problemas no desenvolvimento de software**. Esses problemas têm afetado especialmente os projetos de veículos elétricos (EV), incluindo modelos importantes da Audi, Porsche e Bentley [1](#) [2](#).

Há algum modelo específico ou marca do grupo VW que te interessa mais?

Learn more:

- [1. razaoautomovel.com](#) [2. observador.pt](#)



Volkswagen has given a name to its pain, and it is 'software'

The German automaker's ID family of electric cars are plagued with buggy software, causing customer headaches and leading to major setbacks for the company's next-gen Trinity project. How can VW patch the problem?

During the year that he owned a 2021 Volkswagen ID.4, Branden Flasch came to believe that the electric crossover SUV did many things very well. But when it came to the car's software-based functions — the infotainment system, the various controls, smartphone integration and so on — “it was death by a

EU launches investigation into Chinese EVs to protect European automakers

SCOTT SIMON, HOST:

There's a battle between Europe and China that's brewing over electric vehicles. The European Union has accused China of dumping cheap, government-subsidized electric cars on its market, contending that it's unfair for European automakers. The EU has launched an investigation. NPR's Rob Schmitz joins us now from Berlin. Rob, thanks so much for being with us.

Software problems at Cariad delay Volkswagen's model planning



Because Volkswagen's software subsidiary Cariad is far behind schedule in its development, the model planning of the group's premium brands is being thrown into disarray, according to a media report. At Bentley, even the timetable for the transformation to a pure electric vehicle brand by 2030 is said to be wobbling.

ed legacy carmakers would blow Tesla out of the water with the bogged down with software problems, showing that a modernts. Volkswagen is probably the best example, seeing how Cariad, aged to derail the new car programs for all Volkswagen brands.



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

“In preparing for battle I have always found that plans are useless, but planning is indispensable.”

(Eisenhower)

Dívida técnica (*technical debt*)



- Resulta de priorizar entregas rápidas em vez de qualidade da solução
- Software desenvolvido de forma parcial (com problemas por resolver)
- Erros e limitações afectam evolução do produto

Dívida técnica (*technical debt*)



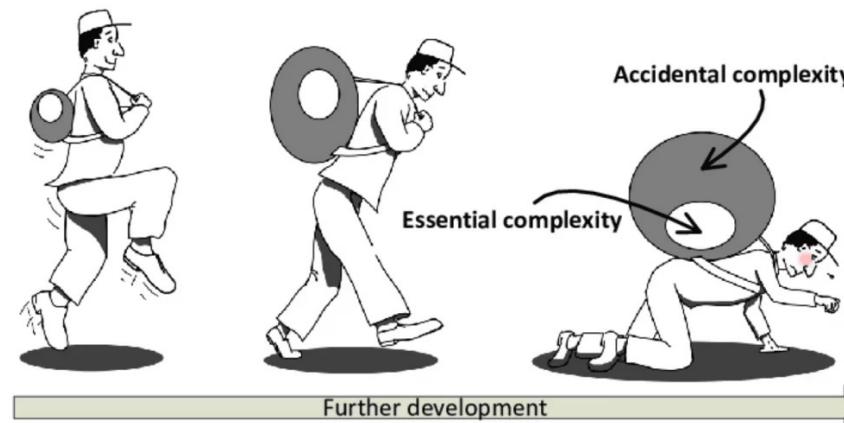
Version 2 announcement (August 2020):

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**.

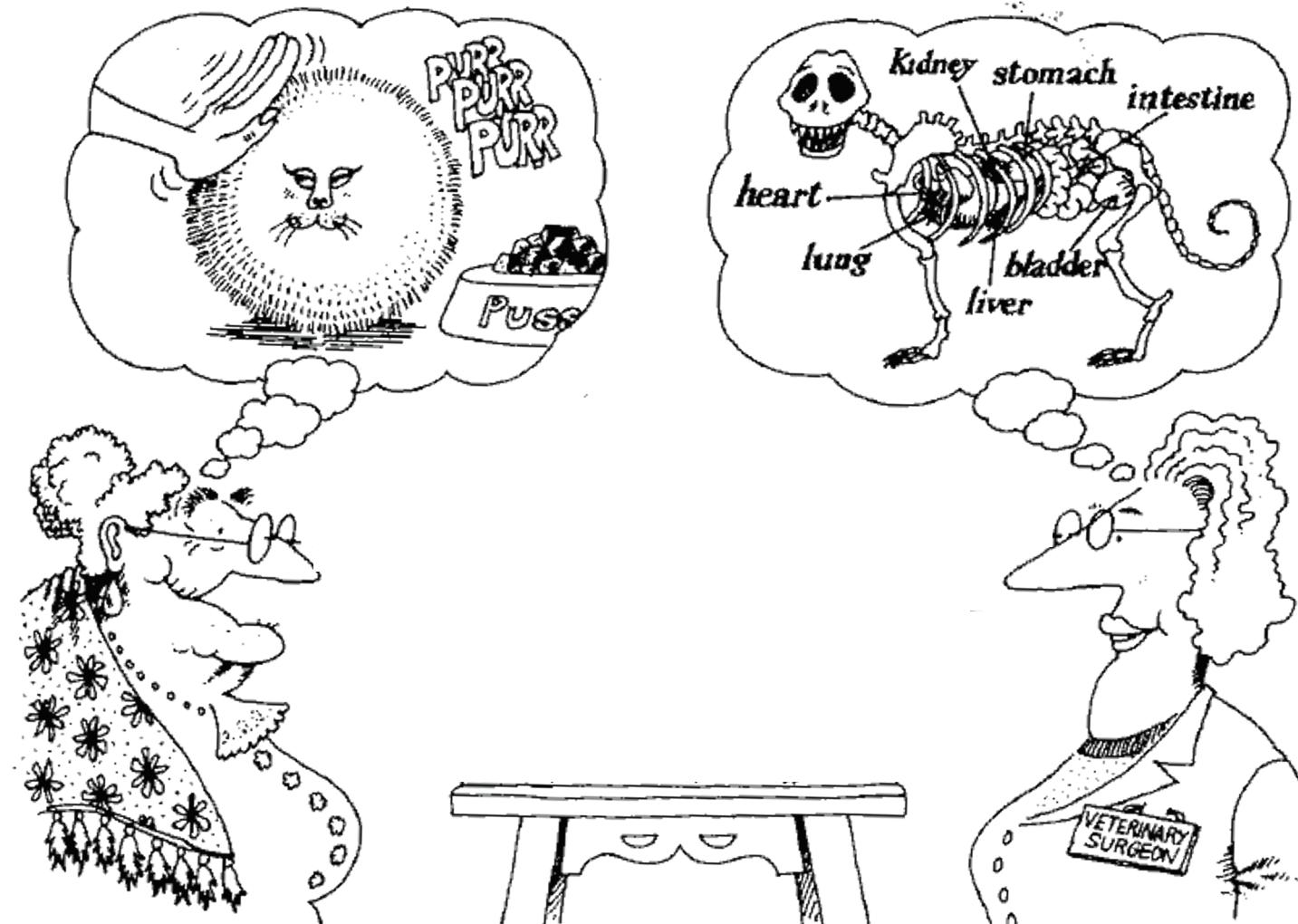
So the first goal of **Among Us 2 is to be made to withstand growth**. We want to add to it at least as long as Among Us, but **with fewer bugs along the way**.

Complexidade

- O grau de dificuldade ou complicações de um sistema ou de um problema.
 - Afeta a qualidade, o desempenho e a manutenibilidade dos sistemas.
- Segundo [Brooks](#), existem dois tipos de complexidade na eng. de software:
 - **Complexidade essencial** é inerente ao domínio do problema
 - Determinada pelos requisitos e funcionalidades do sistema e não pode ser evitada ou simplificada.
 - Pode ser gerida usando técnicas adequadas (abstração, decomposição, encapsulamento, herança, polimorfismo, etc.)
 - **Complexidade acidental** surge da implementação ou do design do sistema
 - É evitável, mas pode ser introduzida usando ferramentas, métodos ou práticas inadequados.
 - Pode ser reduzida ou eliminada usando melhores **ferramentas, métodos ou práticas**.

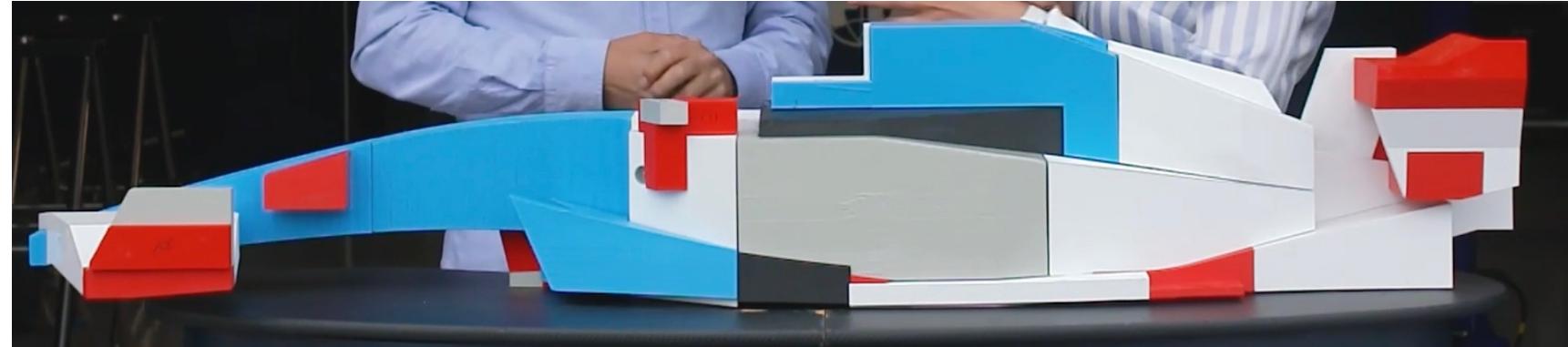


Abstração!

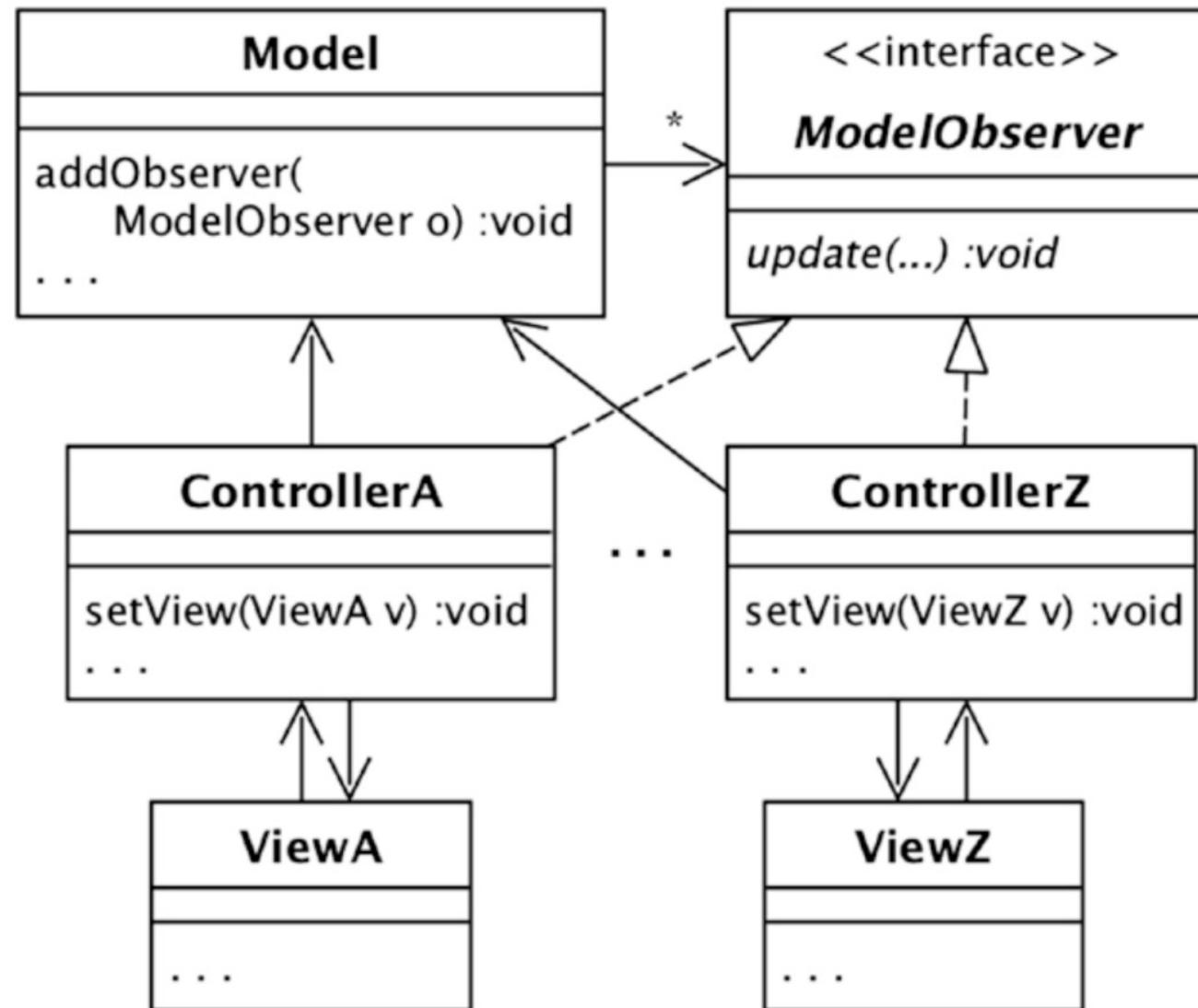


Capacidade de isolar mentalmente um aspecto ou uma propriedade de um todo,
para considerá-lo separadamente

Abstração - Modelos

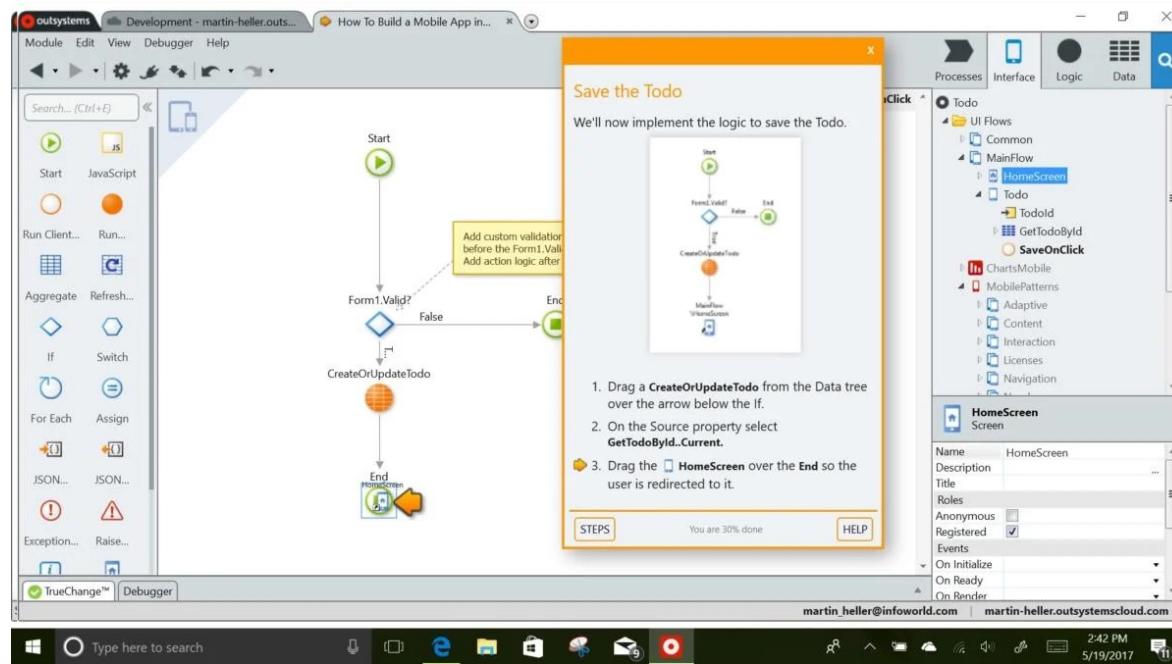


Abstração - Modelos: Diagramas de Classe



Abstração - Low-code development

- Desenvolvimento com recurso a linguagens visuais, minimizando a necessidade de escrever código.
- Foco no **Desenvolvimento baseado em modelos**





Write me a basic java implementation of the observer-observable pattern



The Observer-Observable pattern, also known as the Publish-Subscribe pattern, is a behavioral design pattern in which an object, known as the subject (or observable), maintains a list of its dependents, called observers, and notifies them of state changes, typically by calling one of their methods.

Here's a basic Java implementation of the Observer-Observable pattern:

java

Copy code

```
import java.util.ArrayList;
import java.util.List;

// Observable (Subject)
class WeatherStation {
    private List<Observer> observers = new ArrayList<>();
    private int temperature;

    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    public void removeObserver(Observer observer) {
        observers.remove(observer);
    }

    public void setTemperature(int temperature) {
        this.temperature = temperature;
        notifyObservers();
    }

    private void notifyObservers() {
        for (Observer observer : observers) {
            observer.update(temperature);
        }
    }
}

// Observer
interface Observer {
    void update(int temperature);
}

// Concrete Observer
class WeatherDisplay implements Observer {
    private int temperature;

    @Override
    public void update(int temperature) {
        this.temperature = temperature;
        display();
    }

    private void display() {
        System.out.println("Weather Display: Current Temperature is " + temperature + " degrees Celsius");
    }
}

// Concrete Observer
class MobileApp implements Observer {
    private int temperature;

    @Override
    public void update(int temperature) {
        this.temperature = temperature;
        display();
    }

    private void display() {
        System.out.println("Mobile App: Temperature Alert - " + temperature + " degrees Celsius");
    }
}

public class ObserverPatternExample {
    public static void main(String[] args) {
        WeatherStation weatherStation = new WeatherStation();
        WeatherDisplay weatherDisplay = new WeatherDisplay();
        MobileApp mobileApp = new MobileApp();

        weatherStation.addObserver(weatherDisplay);
        weatherStation.addObserver(mobileApp);

        // Simulate a temperature change
        weatherStation.setTemperature(25);

        // Remove the weather display observer
        weatherStation.removeObserver(weatherDisplay);

        // Simulate another temperature change
        weatherStation.setTemperature(30);
    }
}
```

Regenerate

Send a message





Automação, segundo o Bard...

- A automação é o uso de máquinas para realizar tarefas que antes eram feitas por humanos.
- À medida que se difunde, a automação substitui alguns empregos e altera as competências necessárias para outros.
- Os trabalhadores mais afetados são aqueles com empregos repetitivos e rotineiros.
- Esses trabalhadores precisarão adquirir novas competências para encontrar emprego na economia automatizada.
- As competências que serão mais requisitadas no futuro incluem **pensamento crítico, resolução de problemas e criatividade**.
- Os trabalhadores que conseguirem adquirir essas habilidades terão mais possibilidades de ter sucesso na economia automatizada.

Aulas Teóricas - Programa

- O ciclo de vida do Desenvolvimento de Software
 - **Desenvolvimento de Software baseado em modelos:** análise e concepção orientada aos objectos, com UML
 - Modelação de Domínio
 - Análise e modelação de requisitos
 - Concepção da solução
 - Refinamento e implementação da solução - mapeamento de objectos no modelo relacional
 - Modelação de comportamento
 - Pré-requisito (oficioso): POO!
- 

Práticas Laboratoriais - Programa

- Estudos de caso:
 - pequenos exemplos para aprendizagem dos conceitos
 - realização do projecto
- Apresentação da Ferramenta de Modelação Visual Paradigm:
 - modelação em UML (Unified Modelling Language)
 - geração de código



Objectivos de aprendizagem

No final da UC os alunos deverão ser capazes de:

1. Caracterizar as fases típicas de um processo de desenvolvimento de software
2. Interpretar os diferentes tipos de diagramas da UML
3. Avaliar qual o diagrama mais adequado em função das diferentes necessidades de modelação
4. Avaliar criticamente modelos (de requisitos / estruturais / comportamentais)
5. Conceber sistemas de software utilizando modelos
6. Implementar sistemas de software a partir de modelos

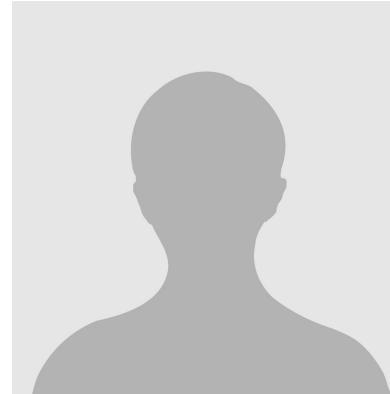
Desenvolvimento de Sistemas Software 2023/24

- **Equipa Docente**



António N. Ribeiro (PL)

anr@di.uminho.pt



Tiago Oliveira (PL)

tiago@di.uminho.pt



Afonso Sousa (PL)

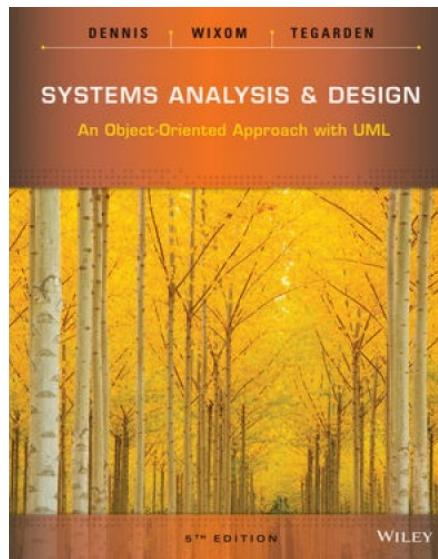
d14998@di.uminho.pt

- **Canais de comunicação**

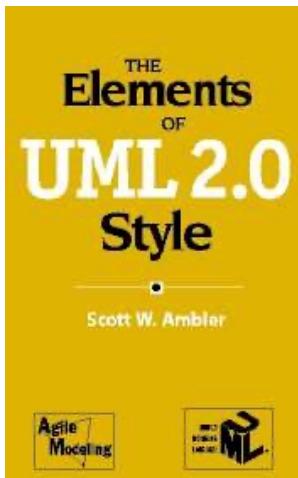
- Aulas teóricas (**canal principal**)
- Aulas PL (turno)
- Blackboard (elearning)
- Horário de atendimento (o meu!): quintas 11:00-13:00
 - Horário flexível (marcar com antecedência)



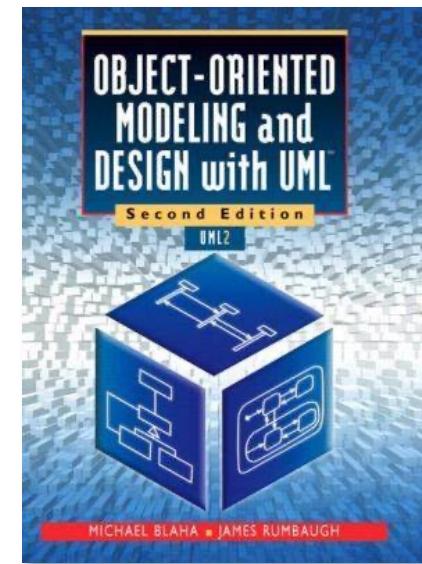
Bibliografia



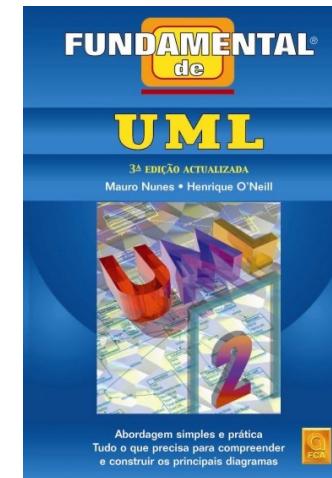
Dennis, Wixom & Tegarden. *System Analysis & Design: An Object-Oriented Approach with UML* (5th ed). Wiley, 2015.



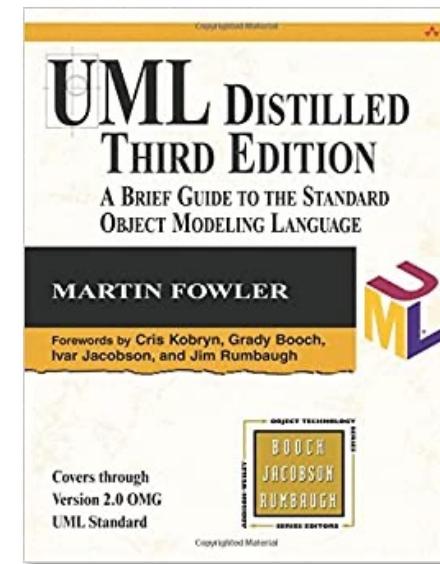
Wembler, *The Elements of UML 2.0 Style*. Cambridge University Press, 2005.



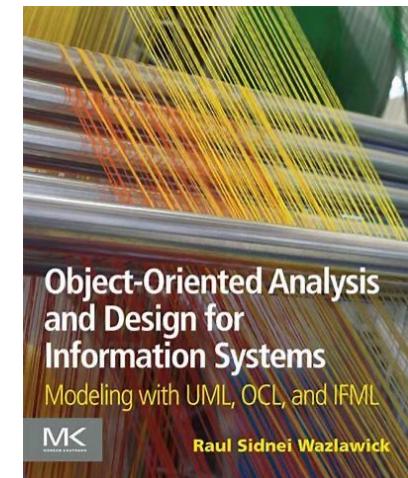
Blaha & Rumbaugh. *Object-Oriented Modeling and Design with UML* (2nd ed). Prentice Hall, 2005.



Nunes & O'Neill. *Fundamental de UML* (3^a ed). FCA, 2007.



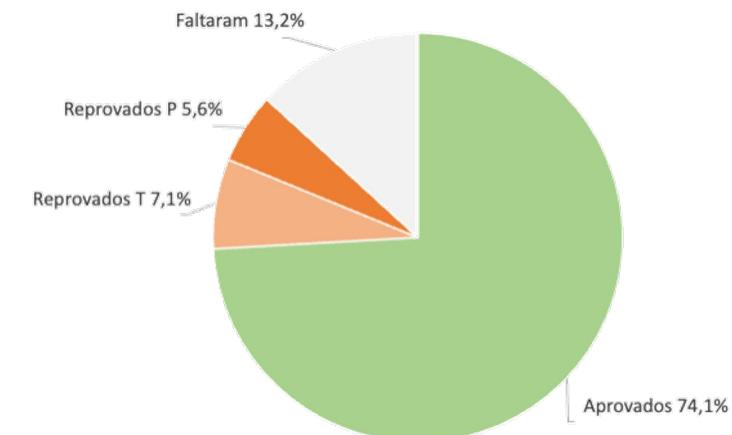
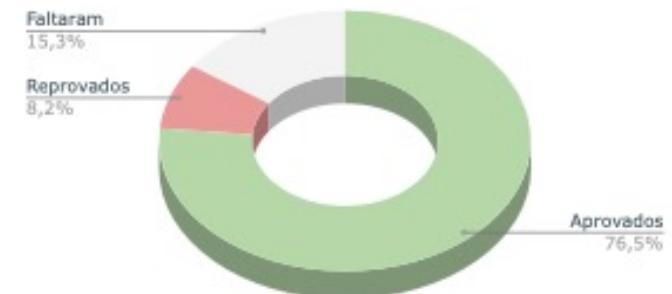
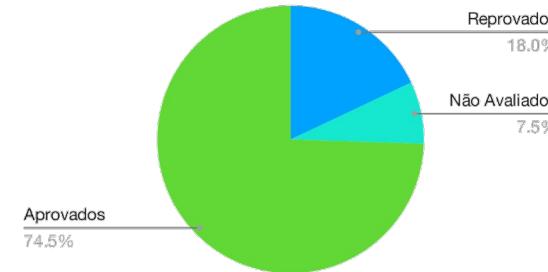
Fowler. *UML Distilled* (3rd ed). Addison-Wesley, 2004.



Wazlawick. *Análise e Design Orientados a Objetos para Sistemas de Informação* (3^a ed). Grupo GEN, 2014.

Avaliação

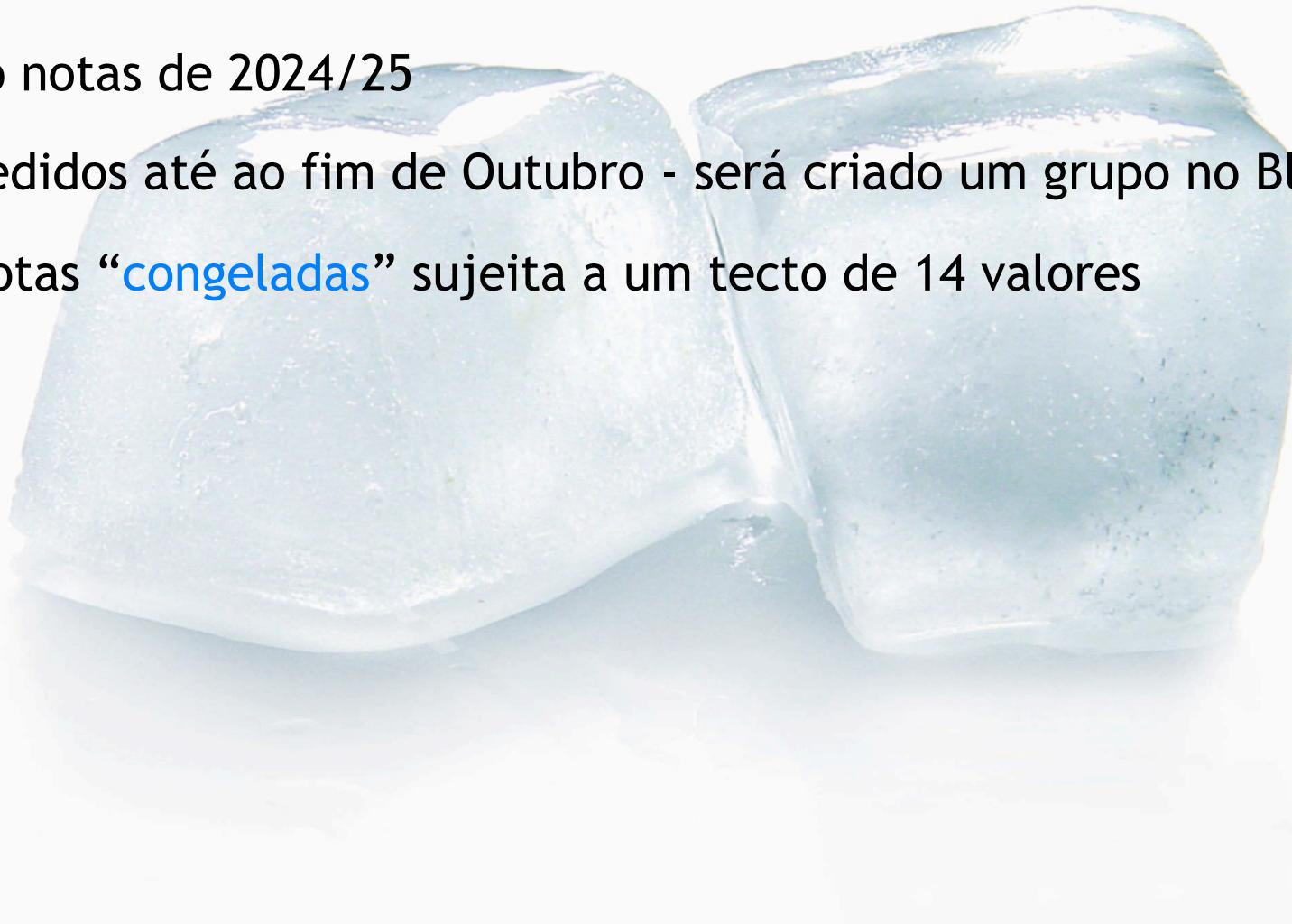
- Trabalho Prático (≥ 10.0)
 - projecto de análise e desenvolvimento de software, em grupo
- Teste/Exame (≥ 8.0)
 - prova individual escrita
- Classificação Final (≥ 10.0)
 - 50% Exame + 50% Trabalho



IMPORTANTE:
Mínimos são condição necessária,
mas não suficiente, para garantir
aprovação à UC.

Congelamentos de nota

- É possível reutilizar a nota do trabalho do ano passado
 - Só notas de 2024/25
 - Pedidos até ao fim de Outubro - será criado um grupo no Blackboard
 - Notas “**congeladas**” sujeita a um tecto de 14 valores





Teste / Exame

- Individual
- Com consulta de apontamentos
 - Podem levar **duas folhas manuscritas** com a informação que considerarem relevante
- No exame aplicam-se as mesma regras de cálculo da nota que para o teste
- Datas
 - Teste: 15 de Dezembro
 - Exame de recurso: 20 de Janeiro



Trabalho Prático

- Grupos de 3-5 elementos
 - Discussão de soluções alternativas
- A realizar **durante** o semestre - 4 momentos relevantes:
 - 1) Apresentação do enunciado - **próxima semana!**
 - 2) Entrega intermédia 1 - até **19 de Outubro** às 24h00 (5 semanas)
 - Análise de requisitos
 - 3) Entrega final - a **09 de janeiro** (7 semanas)
 - Modelação realizada + Sistema desenvolvido
 - 4) Apresentação e discussão: **semana de 12 de Janeiro** de 2025
 - Todos os elementos do grupo presentes; avaliação será individualizada

Trabalho Prático - Avaliação por Pares

- Avaliação dos estudantes pelos seus colegas
- Objectivos
 - **Responsabilizar** o grupo pelo seu desempenho colectivo/de cada elemento
 - Avaliação como parte do processo de **aprendizagem (feedback)**
 - **Classificar** individualmente cada aluno de acordo com a sua contribuição e empenho, tal como precepcionada pelo grupo (incluindo o próprio aluno)
 - Maior **comprometimento e autonomia**



Peer Assessment allows students to make judgements about others work while reflecting on their own.

Avaliação por pares

- Realizada três vezes ao longo do semestre
 - 21-25 de outubro após análise de requisitos (peso: 20%)
 - 25-29 de dezembro antes da entrega final (peso: 40%)
 - 13- 15 de janeiro após entrega final (peso: 40%)
- Preenchimento online de uma grelha de avaliação multi-critério
 - Para cada critério há $p*n$ pontos a distribuir (sendo p o peso do critério e n o número de elementos do grupo - $\sum p/n=100$)
- CAIG - Coeficiente de Avaliação Intra-Grupo
 - O resultado da avaliação por pares
 - CAIG final - média ponderada dos CAIG de cada etapa

Avaliação por pares

- A avaliação individual será: avaliação_do_proj * CAIG
 - CAIG = 1 - a nota individual será a da avaliação do trabalho
 - CAIG < 1 - a nota individual será inferior à avaliação do trabalho
 - CAIG > 1 - a nota individual será superior à avaliação do trabalho
- CAIG < 0.80 - não é atribuída nota prática
- CAIG > 1.10 - subida de nota limitada a 10%
- Atenção
 - Fundamental realizar a avaliação com seriedade
 - Não se estão a avaliar amizades mas o contributo de cada um para o grupo
 - Fundamental acordo quanto a critérios de avaliação e sua ponderação

Avaliação por pares - Exemplo

Avaliação feita pelo Aluno 1				Avaliação feita pelo Aluno 2				
Critério	Aluno 1	Aluno 2	Aluno 3	Critério	Aluno 1	Aluno 2	Aluno 3	Total
Ideias	24	20		Ideias	20	20	20	60
Realização	26	18		Realização	22	20	18	60
Empenho	24	24		Empenho	24	20	16	60
Assiduidade	23	23		Assiduidade	22	20	18	60
Relações	22	22		Relações	22	22	16	60
CAIG (parcial)	1.19	1.07		CAIG (parcial)	1.10	1.02	0.88	
CAIG (parcial)	1.19	1.07		0.74				
Avaliação feita pelo Aluno 3				Final	Aluno 1	Aluno 2	Aluno 3	
Critério	Aluno 1	Aluno 2	Aluno 3	CAIG	1.097	1.030	0.873	
Ideias	20	20	20	Trabalho: 11	12.07	11.33	9.60	
Realização	20	20	20	Trabalho: 14	15.36	14.42	12.22	
Empenho	20	20	20	Trabalho: 18	19.75	18.54	15.71	
Assiduidade	20	20	20	50				
Relações	20	20	20	60				
CAIG (parcial)	1.00	1.00	1.00					

Avaliação por pares

- Critérios propostos

Critério	Peso	Descrição
C1 Ideias para o trabalho	20%	Contribuição com ideias e sugestões para a condução dos trabalhos e resolução de problemas
C2 Realização do trabalho	30%	Qualidade da execução das tarefas necessárias à realização do trabalho
C3 Empenho no trabalho	25%	Esforço realizado para que o trabalho seja entregue no prazo
C4 Assiduidade	15%	Contribuição para a marcação de reuniões do grupo e comparência nas mesmas
C5 Relações interpessoais	10%	Contribuição para a motivação, manutenção de bom ambiente e para a resolução de eventuais conflitos

- Estes critérios estarão à discussão até 29 de setembro
 - enviem comentários e sugestões!!



*"I can explain it to you, but
I can't understand it for you."*