

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Desenvolvimento de Sistemas de Software

LEI/MiEI - 3º ano / 1º semestre

2021/2022

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

António Nestor Ribeiro
anr@di.uminho.pt

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



Desenvolvimento de Sistemas de Software

Ciclo de vida do desenvolvimento

Engenharia de Software



- Engenharia

*“The creative application of scientific principles to **design** or **develop** structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to **construct** or **operate** the same with full cognizance of their design; or to **forecast their behavior** under specific operating conditions; all as respects an intended function, economics of operation or safety to life and property” (Engineers' Council for Professional Development)*

- Engenharia de Software

“Aplicação de um método/processo sistemático, disciplinado e quantificável à concepção, desenvolvimento, operação e manutenção de software” (IEEE CS)

Processo de Desenvolvimento de Software

- Define como se estrutura o desenvolvimento de software
- Identifica as fases de desenvolvimento e como se passa de umas para outras
 - Quem faz o quê
 - Quando é feito
 - e Durante quanto tempo





“Code and fix”

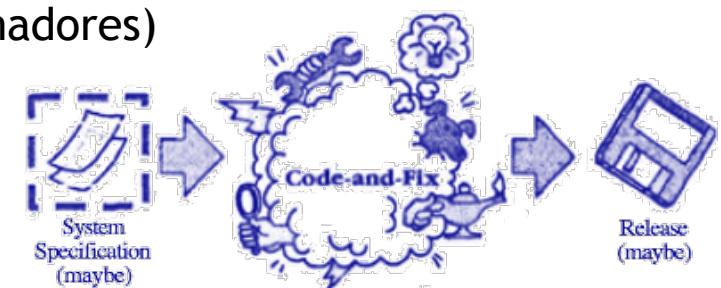


S. McConnell (2002) Rapid Development. Microsoft Press.



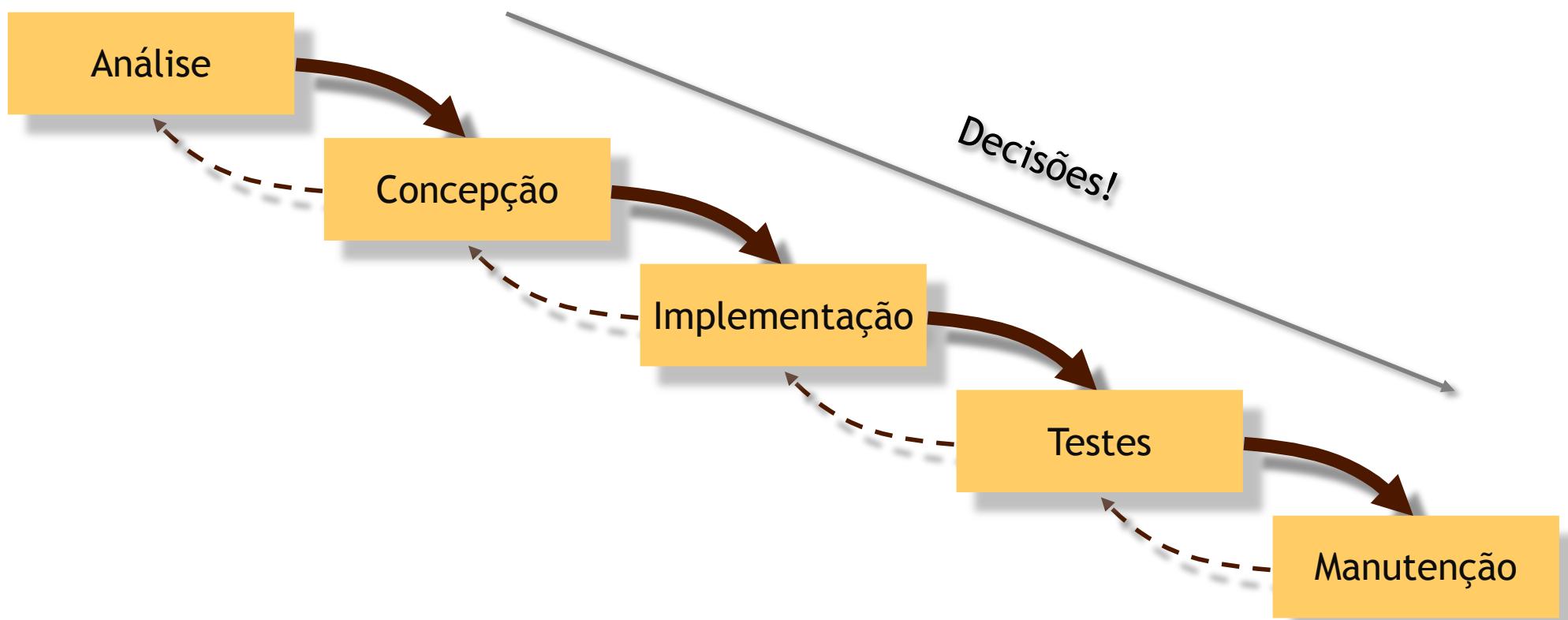
“Code and fix”

- Raramente útil, mas muito utilizado
- Não tanto uma estratégia deliberada, mais o resultado de falta de planeamento e tempo
 - Sem muita análise, começa-se a produzir código de imediato
 - Eventualmente fazem-se testes e os inevitáveis erros têm que ser resolvidos
- “*Vantagens*”
 - Produção de código é imediata
 - Não requer muita competência - qualquer programador pode utilizar
- *Problemas*
 - Não existe forma de avaliar progresso
 - Não existe forma de avaliar qualidade e/ou de prever riscos
 - Não escala bem para para equipas (multiplos programadores)
- Apenas viável para pequenos projectos
 - provas de conceito ou *demos*



Abordagens Estruturadas

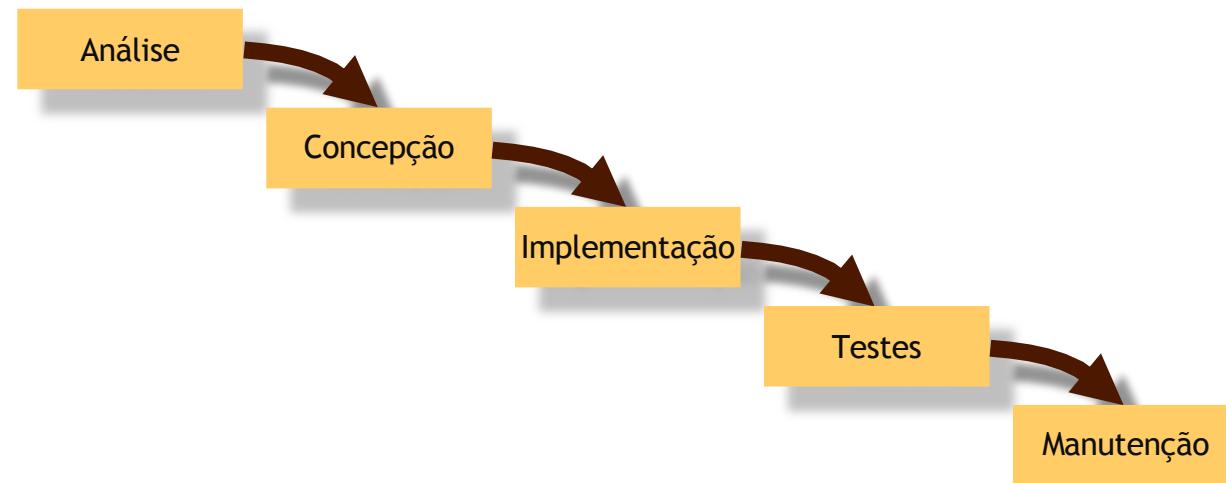
- Modelo em Cascata (Waterfall)



- Define uma série de etapas executadas sequencialmente.

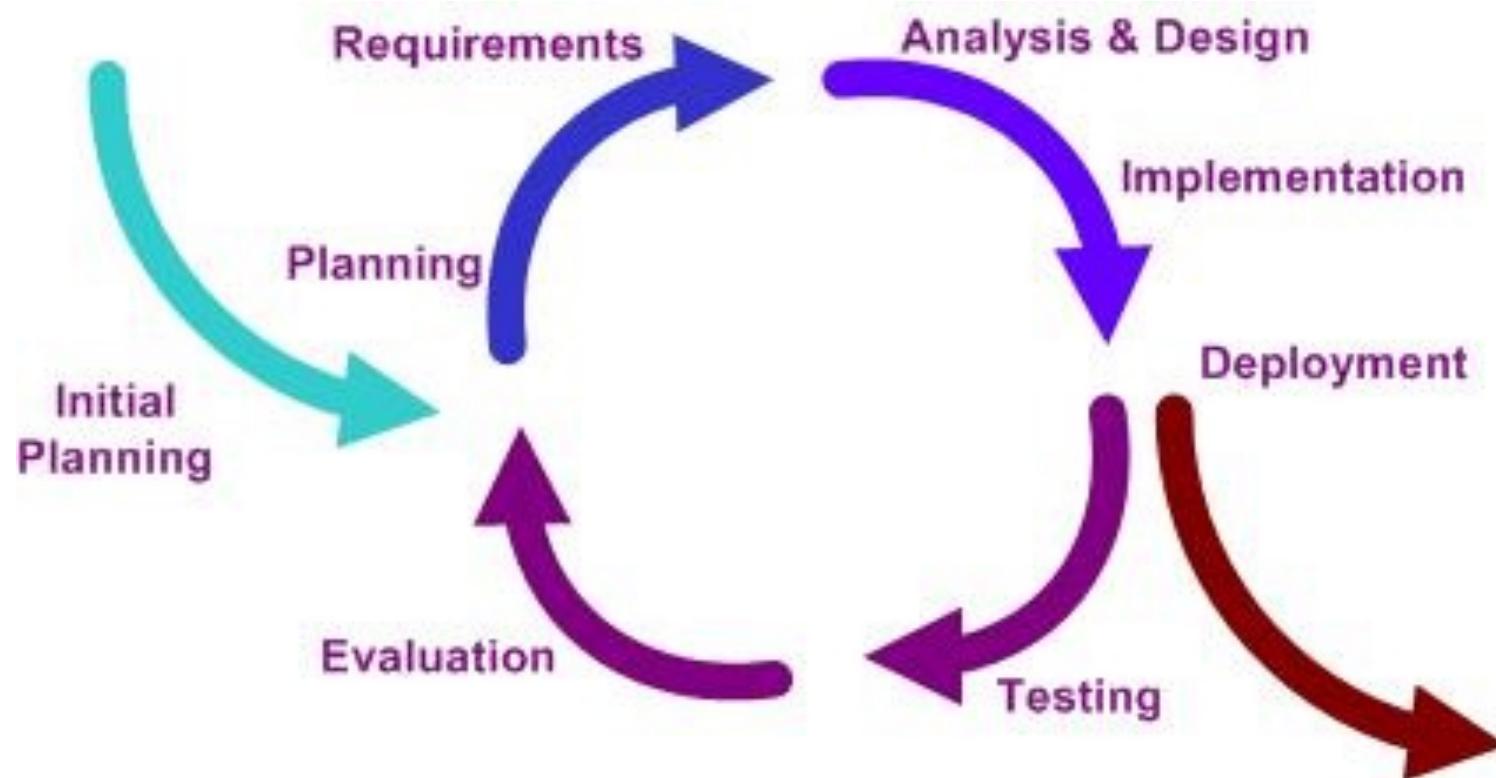
Modelo em Cascata

- Assume que é sempre possível tomar as decisões mais correctas
 - como prever situações de alto risco mais à frente?
 - como favorecer reutilização?
 - como lidar com alterações (p.e. de requisitos) durante o processo?
- **Quanto mais tarde um problema é encontrado, mas caro será corrigi-lo!**



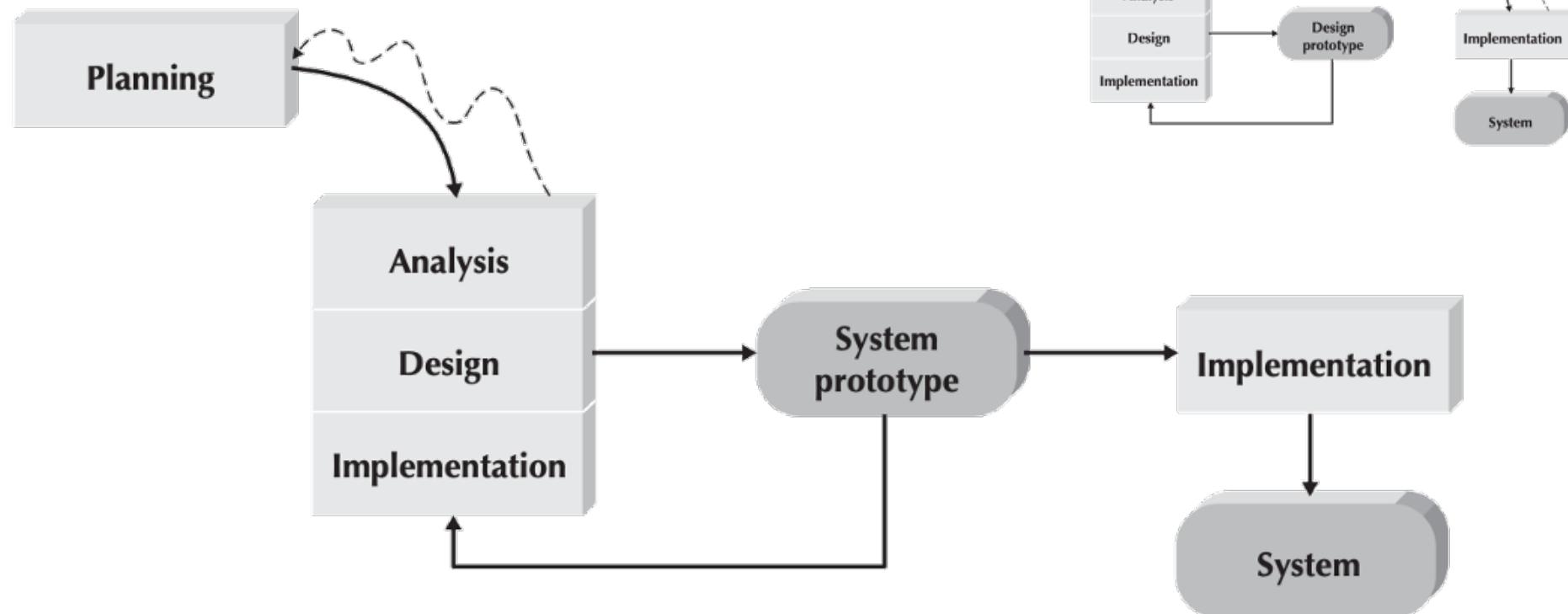


Desenvolvimento Iterativo e Incremental



Desenvolvimento Iterativo e Incremental

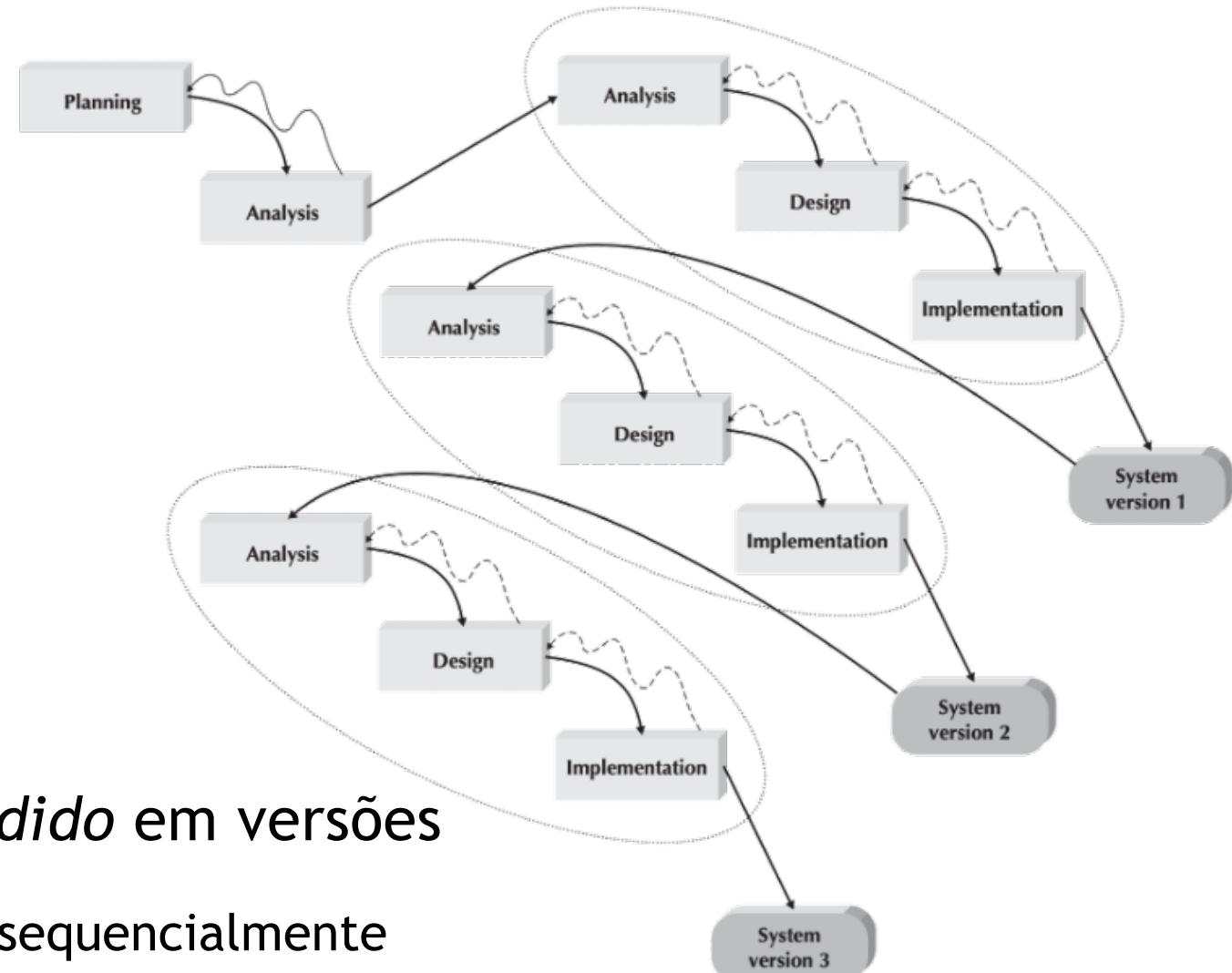
- Rapid Application Development (RAD)
 - Baseado na prototipagem do sistema



- Análise, concepção e implementação realizadas em, paralelo e repetidamente
 - Protótipo pode ou não evoluir para o sistema final

Desenvolvimento Iterativo e Incremental

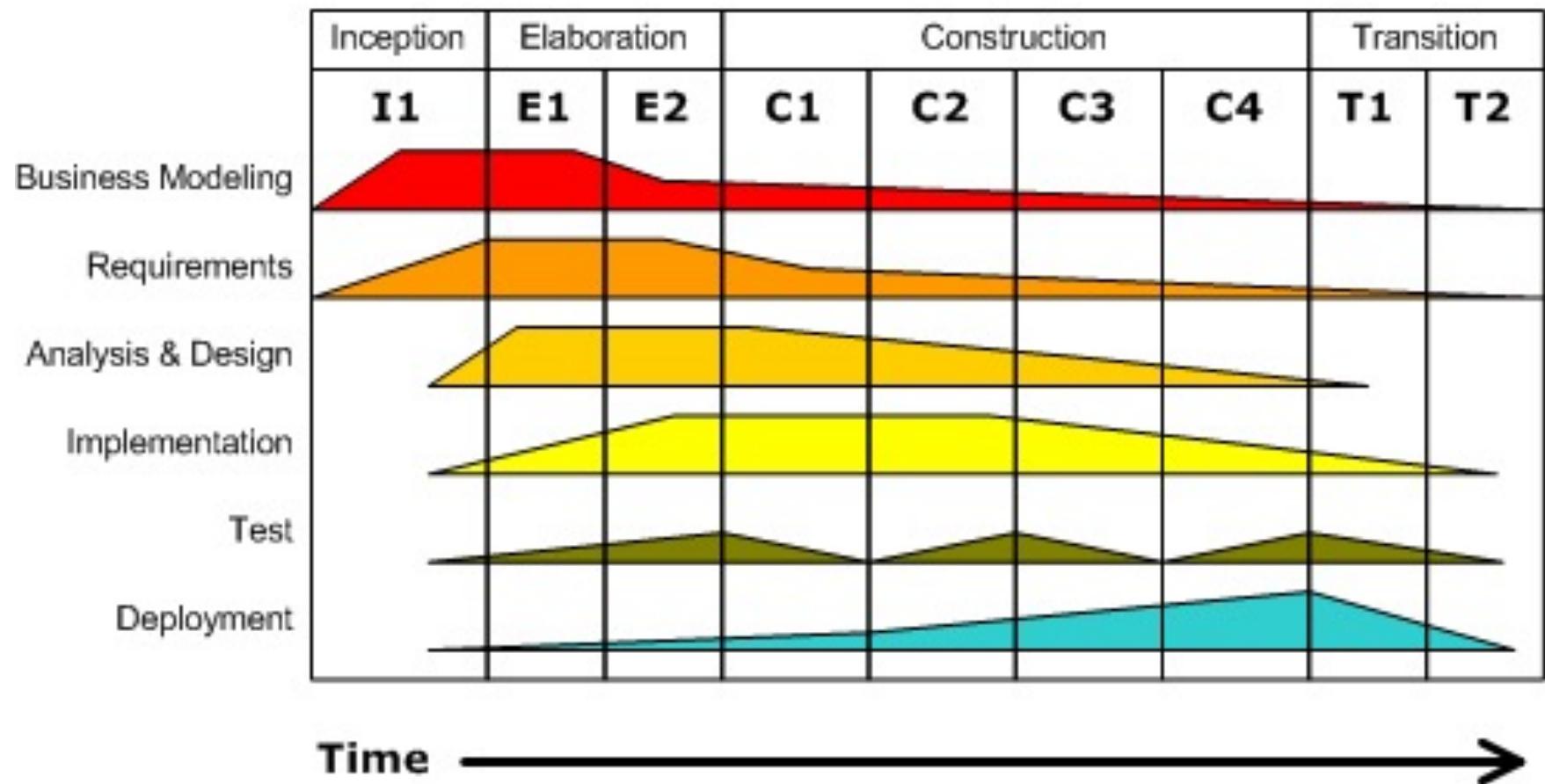
- Desenvolvimento faseado (*Phased development*)



- Sistema é *dividido* em versões
 - Desenvolvidas sequencialmente

Desenvolvimento Iterativo e Incremental

- (Rational) Unified Process (RUP)

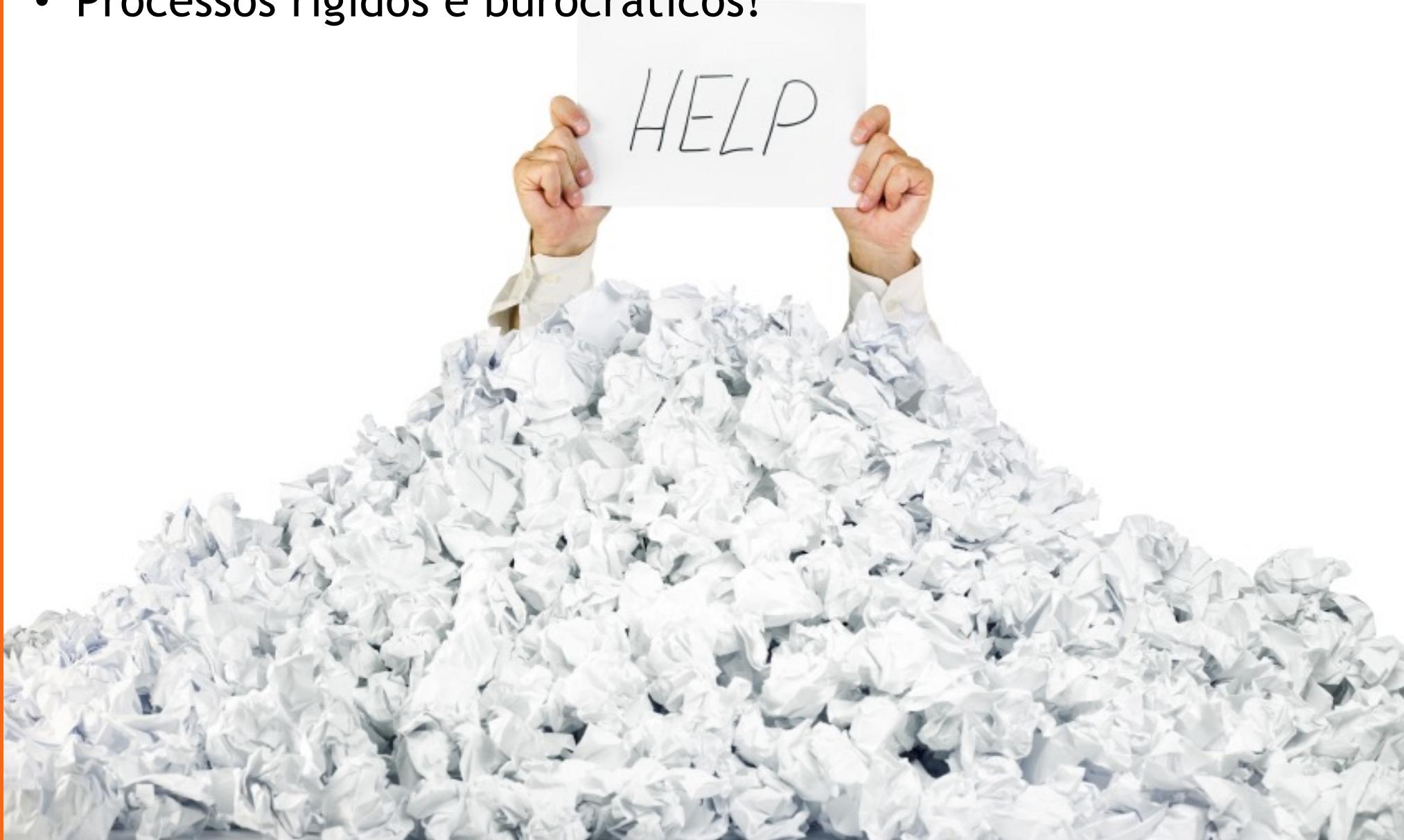


Abordagens interativas e incrementais - prós e contras

- Vantagens
 - Entregas intermédias permitem demonstração/exploração do sistema desde cedo
 - Identificação de novos requisitos / identificação de problemas
 - Demonstração de progresso (potencia satisfação dos clientes)
 - Maior capacidade de lidar com incerteza, maior controlo de risco
 - Promovem a inclusão de objectivos de qualidade no processo de desenvolvimento
 - Promovem a utilização de ferramentas de automatização do desenvolvimento
- Problemas
 - Utilizadores começam por trabalhar com sistemas que estão incompletos
 - Gestão de expectativas
 - Dependência da qualidade da avaliação de risco/identificação de funcionalidades mais relevantes
 - Decisões tomadas no início podem revelar-se inadequadas mais tarde
 - Produto não deve ser monolítico
 - Entregas causam trabalho adicional

Problemas...

- Processos rígidos e burocráticos!





Agile Manifesto (Beck et al. 2001)

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.

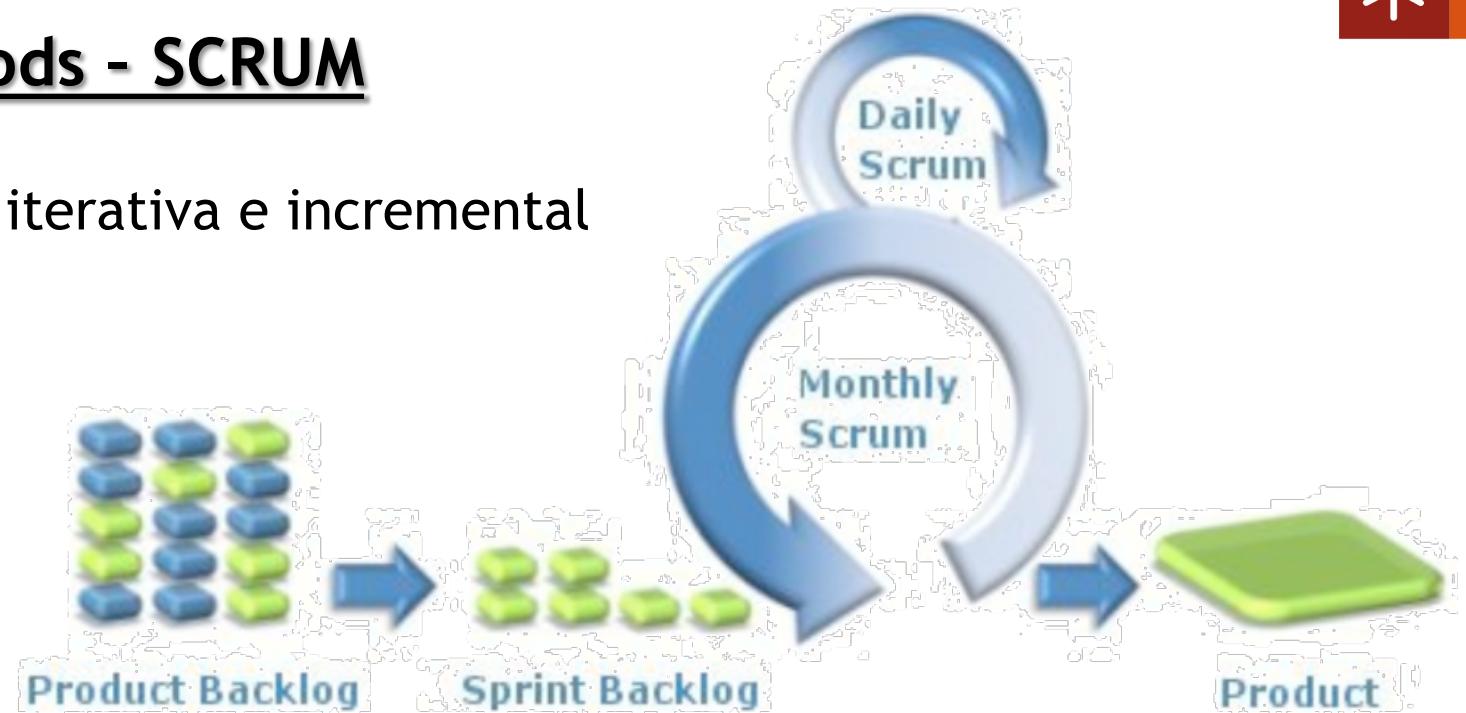


Os Doze Princípios do Manifesto Ágil

1. A nossa maior prioridade é, desde as primeiras etapas do projecto, satisfazer o cliente através da entrega rápida e contínua de software com valor.
2. Aceitar alterações de requisitos, mesmo numa fase tardia do ciclo de desenvolvimento. Os processos ágeis potenciam a mudança em benefício da vantagem competitiva do cliente.
3. Fornecer frequentemente software funcional. Os períodos de entrega devem ser de poucas semanas a poucos meses, dando preferência a períodos mais curtos.
4. O **cliente e a equipa de desenvolvimento devem trabalhar juntos**, diariamente, durante o decorrer do projecto.
5. Desenvolver projectos com base em **indivíduos motivados**, dando-lhes o ambiente e o apoio de que necessitam, confiando que irão cumprir os objectivos.
6. O método mais eficiente e eficaz de passar informação para e dentro de uma equipa de desenvolvimento é através de **conversa pessoal e directa**.
7. A principal **medida de progresso** é a entrega de software funcional.
8. Os processos ágeis promovem o **desenvolvimento sustentável**. Os promotores, a equipa e os utilizadores deverão ser capazes de manter, indefinidamente, um ritmo constante.
9. A atenção permanente à **excelência técnica** e um bom desenho da solução aumentam a agilidade.
10. A **simplicidade** - a arte de maximizar a quantidade de trabalho que não é feito - é essencial.
11. As melhores arquitecturas, requisitos e desenhos surgem de **equipas auto-organizadas**.
12. A **equipa reflecte regularmente** sobre o modo de se tornar mais eficaz, fazendo os ajustes e adaptações necessárias.

Agile Methods - SCRUM

- Framework iterativa e incremental



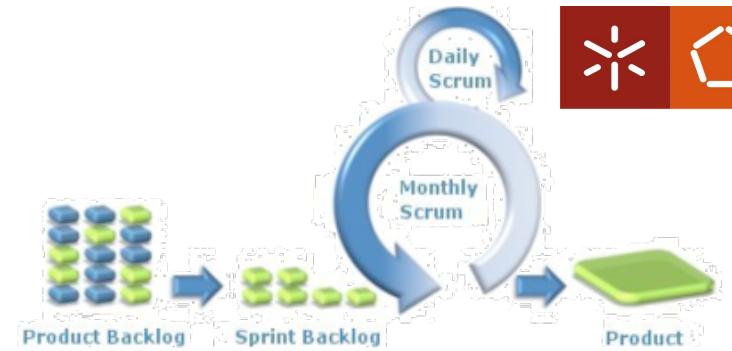
- Papeis/roles
 - *Product owner* - interesses dos *Stakeholders*; voz do cliente
 - Cria o *Product Backlog* a partir de um processo de análise de requisitos
 - *Team* (Equipa - 3 a 5 pessoas) - implementa incrementos em cada Sprint (iteração)
 - Responsável pela produção de software de qualidade a partir do Backlog
 - *Scrum Master* - buffer entre a Team e factores externos
 - Responsável pelo processo. Não faz parte da Team

Scrum

- *Sprint*
 - Unidade basica de desenvolvimento
 - Duração fixa -1 semana a 1 mês (2 semanas usual)
 - Começa com Sprint planning - definir objectivos; *Sprint backlog*
 - Termina com
 - *Sprint review* - revisão do trabalho feito/por fazer; demo
 - *Sprint retrospective* - melhoria do processo
- *Daily Scrum*
 - Reunião diária - 15 minutos
 - Cada elemento da Equipa responde às questões:
 - O que completei ontem?
 - O que planeio completar hoje?
 - Vejo algum impedimento?



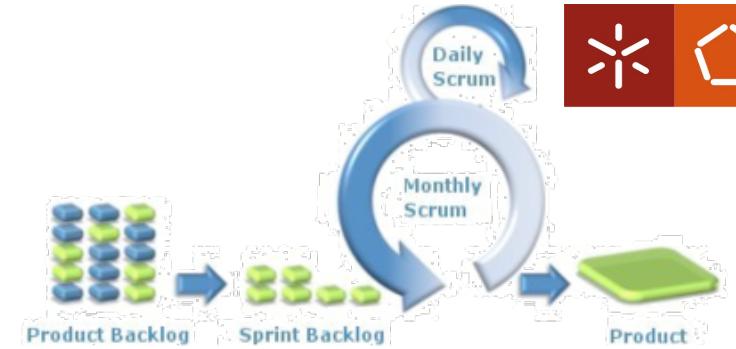
Scrum - prós e contras



- Vantagens
 - Focado na produção de *deliverables* e na utilização eficiente de recursos
 - Gestão de complexidade via divisão em *Sprints*
 - Focado nas necessidades dos *Stakeholders* (mas...)
 - Permite reagir a mudanças nos requisitos e a *feedback* dos *Stakeholders*
 - Esforço de cada elemento da equipa é visível nas Scrum meetings
- Limitações
 - *Scope creep* - Difícil definir o *scope* do projecto (não há data de fim)
 - Preço e duração do projecto?
 - Muito dependente da qualidade e empenho da equipa
 - Difícil de aplicar em projectos/equipas de grande dimensão
 - Focado nas necessidades dos clientes e não na qualidade técnica da solução
 - *Technical debt!*
 - Muito *refactoring* - utilização eficiente de recursos?

Scrum - prós e contras

- Vantagens
 - Focado na produção de *deliverables* e na utilização eficiente de recursos
 - Gestão de complexidade via divisão em *Sprints*
 - Focado nas necessidades dos *Stakeholders* (mas...)
 - Permite reagir
 - Esforço de cada
- Limitações
 - *Scope creep* - D
 - Preço e duraç
 - Muito depende
 - Difícil de aplica
 - Focado nas nec
 - *Technical deb*
 - Muito *refactori*



Discussão...



- Qual a melhor abordagem?
- Depende do tipo de sistema

Capacidade de desenvolver sistemas...	Waterfall	Phased	Prototyping	Throwaway Prototyping	SCRUM	Low code
com requisitos incertos	👎	👍	👌	👌	👌	👌
com tecnologia desconhecida	👎	👍	👎	👌	👍	👎
complexos e de larga escala	👍	👍	👎	👌	👎	👍
confiáveis	👍	👍	👎	👌	👌	👌
com prazos curtos	👎	👌	👌	👍	👌	👌
com visibilidade do andamento	👎	👌	👌	👍	👌	👌

👎 - Mau | 👍 - Bom | 🤝 - Excelente

Adaptado de (Dennis, Wixom & Tegarden, 2015)

Fases do ciclo de vida do desenvolvimento de sistemas

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

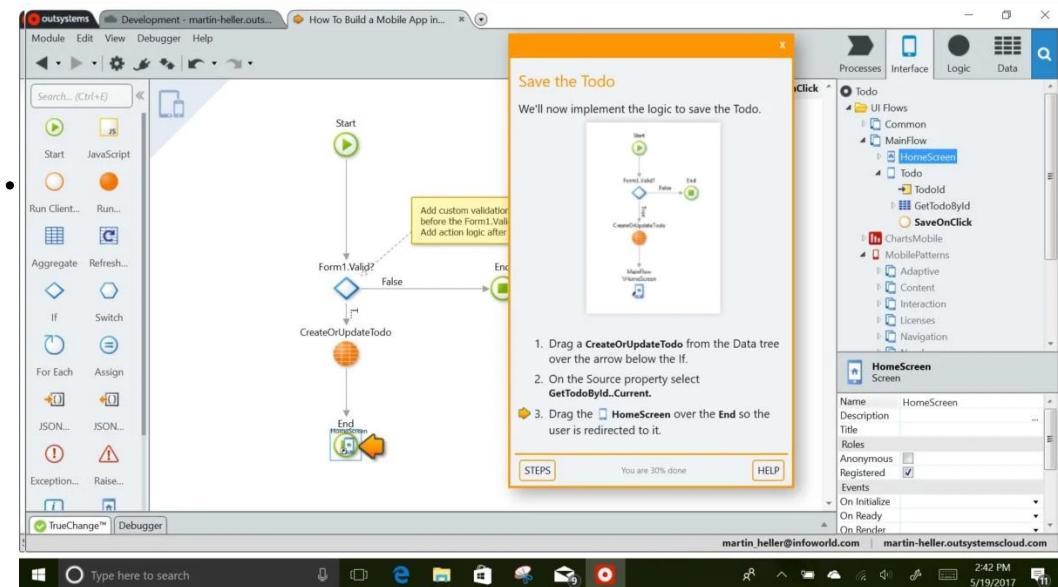


Low-code development

- Desenvolvimento com recurso a linguagens visuais, minimizando a necessidade de escrever código.

- The 9 Essential Principles

1. Model-driven development
2. Collaboration
3. Agility
4. The cloud
5. Openness
6. Multi-User Development
7. Experimentation & Innovation
8. Governance & Control
9. Community



 outsystems

 mendix

 salesforce

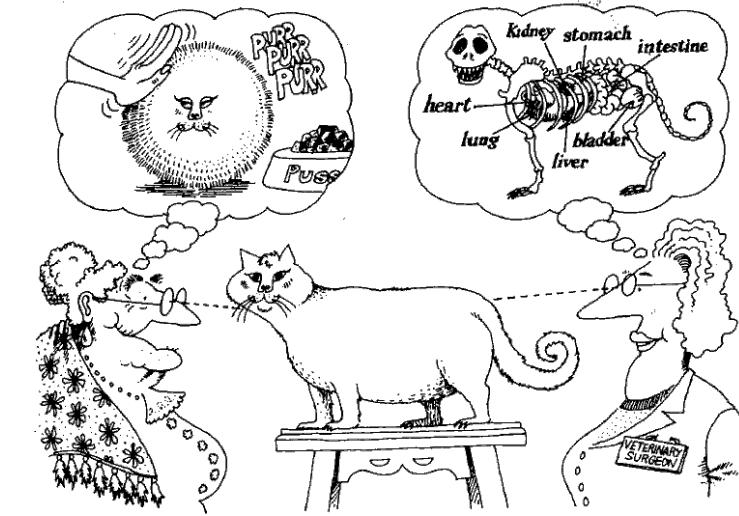
 PowerApps

 service**now**



Modelos são...

- Simplificações da realidade – representações abstractas de um sistema, efectuadas de um ponto de vista determinado
 - Abstracção:
 - O processo de remover informação de uma descrição para ficarem apenas os aspectos relevantes
 - Mecanismo poderoso para lidar com a complexidade
- Úteis para descrever e analisar os problemas e as soluções que queremos desenvolver



(Booch)



Porquê modelar?

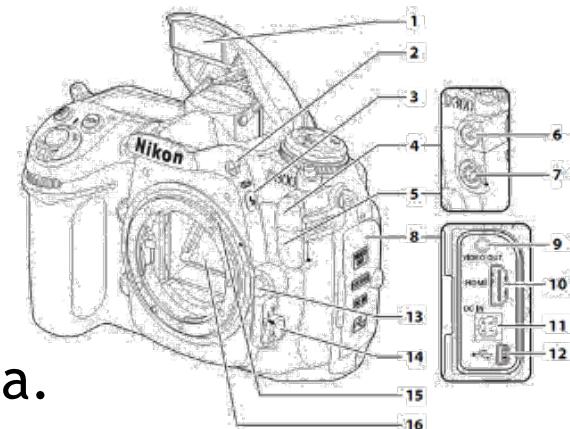
Vantagens da utilização de modelos

- Auxiliam a **compreender** a realidade
- Ajudam a **comunicar** ideias de forma simplificada.
- Ajudam a **documentar** as decisões tomadas durante o desenvolvimento.
- Modelos com uma semântica precisa (rigorosa) suportam **análise e geração de código**.

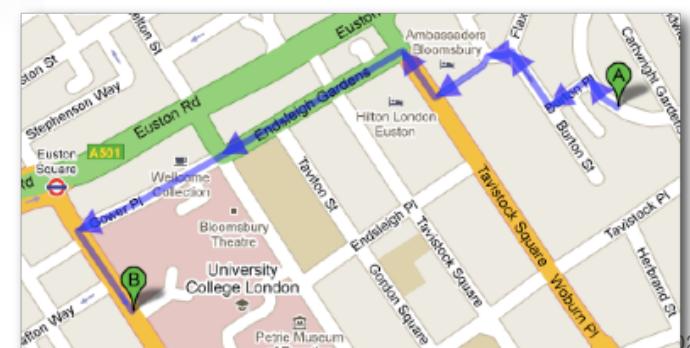
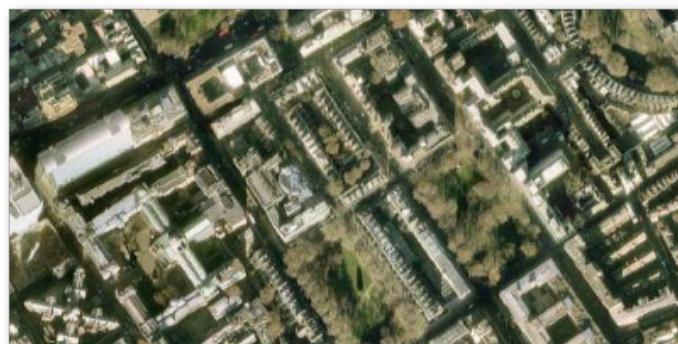
Porquê modelar?

Vantagens da utilização de modelos

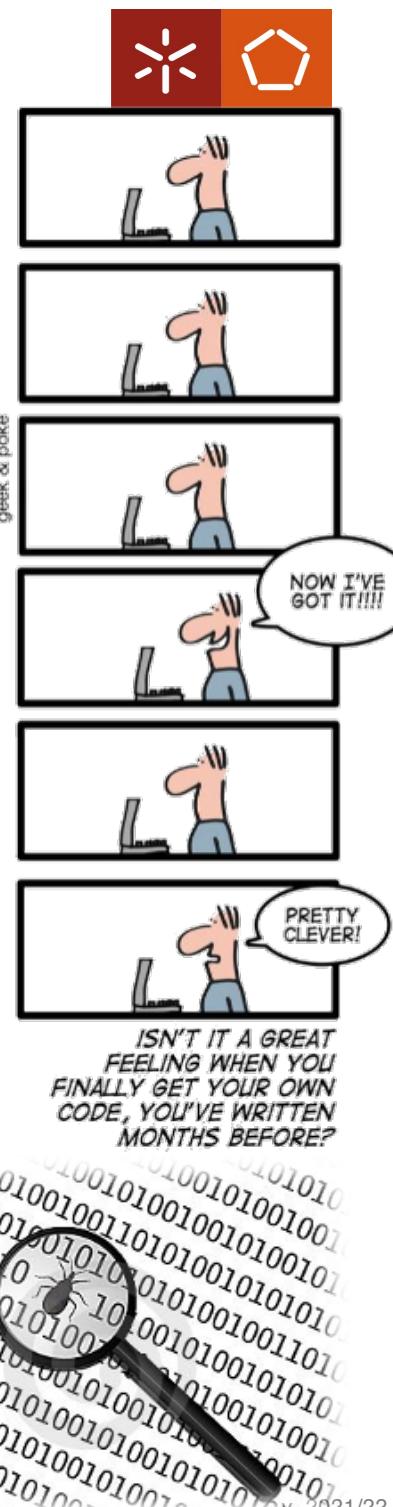
- Auxiliam a **compreender** a realidade.
 - Sendo **abstracções** da realidade, os modelos permitem descrever o que é considerado essencial num dado contexto, escondendo detalhes desnecessários/irrelevantes nesse contexto.



- Ajudam a **comunicar** ideias de forma simplificada.
 - Sendo simplificações da realidade, permitem comunicar apenas os aspectos pretendidos.



Porquê modelar?



Vantagens da utilização de modelos

- Ajudam a documentar as decisões tomadas durante o desenvolvimento.
 - Os modelos desenvolvidos constituem uma base documental para a compreensão do processo de desenvolvimento
 - “*Thinking made public*”.
 - Modelos com uma semântica precisa (rigorosa) permitem análise rigorosa
 - Validação - “estamos a construir o sistema certo?”
 - Verificação - “estamos a construir o sistema bem?”



Porquê modelar?

Problemas com a utilização de modelos

- Mais uma “linguagem” a aprender.
 - Isso acarreta custos, quer monetários (para as organizações), quer cognitivos (para os indivíduos).
- Modelos apresentam uma **visão idealizada da realidade**.
 - Existe o risco de durante o processo de modelação nos esquecermos que os modelos **são representações da realidade e não a realidade**.
 - É necessário encontrar as **abstracções adequadas** para modelar todos os **aspectos relevantes**.
- A fase de modelação **atrasa a produção de código**?
 - É possível passar, de forma (semi-)automática, dos modelos para o código.
 - Código produzido é de melhor qualidade.
 - Regra dos 5/6 – 1/6 (análise e concepção vs. codificação).



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

Eisenhower