

# Engenharia de Software II



2<sup>a</sup> ano  
1º semestre

## Contexto e conceitos fundamentais

Por:

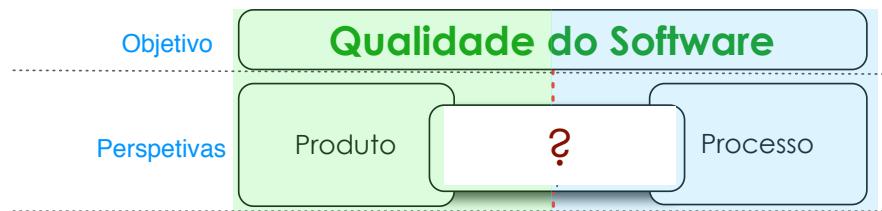
Cristóvão Sousa e Carla Pereira

### Roadmap ESII

Objetivo

**Qualidade do Software**

## Roadmap ESII



Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

3

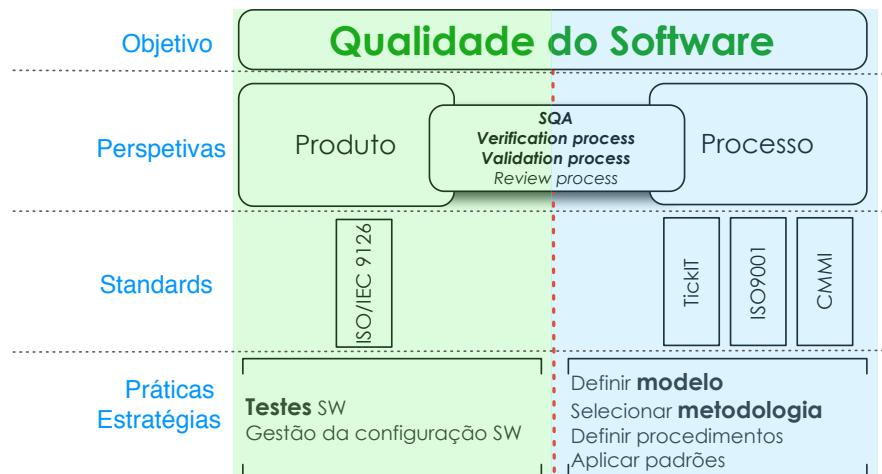
## Roadmap ESII



Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

4

## Roadmap ESII

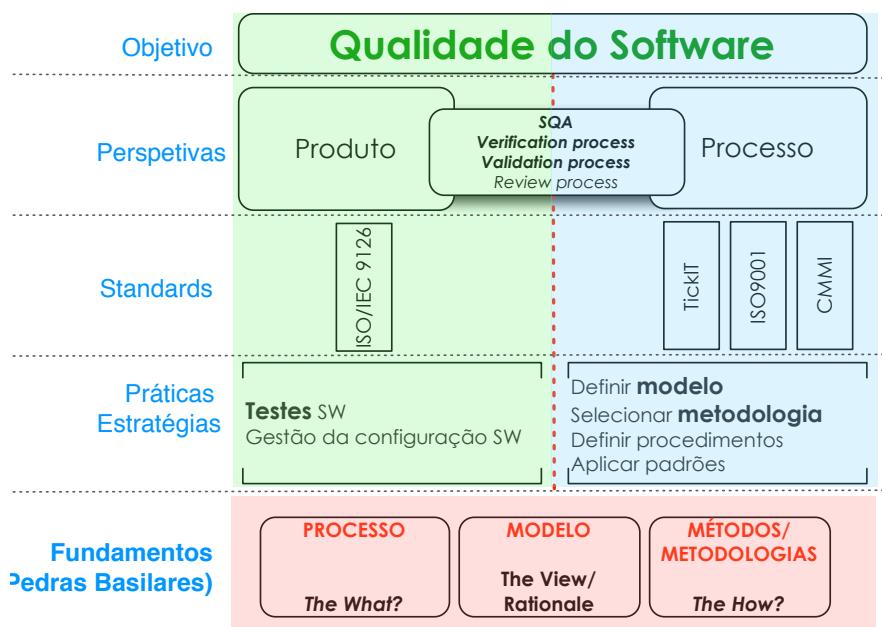


Engenharia de Software II - Engenharia Informática

\* Crístovão Sousa e Carla Pereira\*

5

## Roadmap ESII

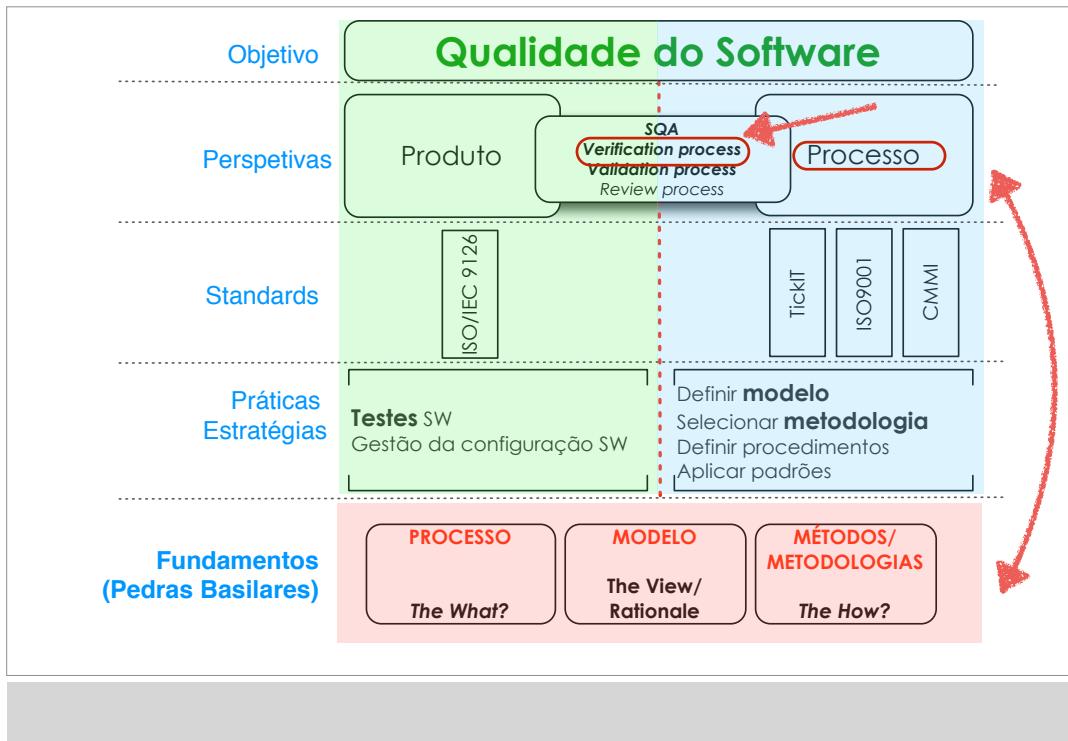


Engenharia de Software II - Engenharia Informática

\* Crístovão Sousa e Carla Pereira\*

6

## Roadmap ESII



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

7

## Desambiguação de conceitos

**Metodologia**  **Processo**

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

8

## Desambiguação de conceitos

### ► Processo

- sequência lógica de tarefas executadas de modo a alcançar um determinado objectivo.
- define o que é necessário fazer (“WHAT”), sem especificar como as tarefas são executadas (“HOW”)
- encerra uma estrutura de com diferentes níveis de agregação de modo a permitir a análise e definição em diferentes níveis de detalhe, dando suporte a diferentes necessidades de decisão.



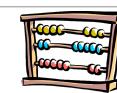
(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

9

## Desambiguação de conceitos

### ► Método



- conjunto de técnicas para execução de uma determinada tarefa.
- especifica como uma tarefa é executada (“HOW”)
- encerra ele próprio um processo (embora a um nível mais baixo)
- o “HOW” num determinado nível de abstracção, torna-se “WHAT” num nível mais baixo.

(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

10

## Desambiguação de conceitos

### ► Ferramenta

- instrumento que quando aplicado a um determinado método, aumenta a eficiência de uma determinada tarefa.
- facilita a execução das tarefas ("HOW")



(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

11

## Desambiguação de conceitos

### ► Metodologia

- colecção/compilação de processos, métodos e ferramentas relacionadas entre si.
- consiste num conjunto de orientações para aplicação de processos, métodos e ferramentas relacionadas entre si a um conjunto típico de problemas.



(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

12

## Desambiguação de conceitos

### ► Ambiente

- composto pelo meio, objectos externos, condições ou factores que possam influenciar as acções sobre um objecto, pessoa ou grupo.
- as condições poderão ser de índole:
  - social
  - cultural
  - física
  - organizacional
  - funcional



(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

13

## Desambiguação de conceitos

### ► Ambiente

- o ambiente de um projecto deverá suportar o uso de ferramentas e métodos no projecto
- um ambiente condiciona o porquê ("WHAT") e o como ("HOW")

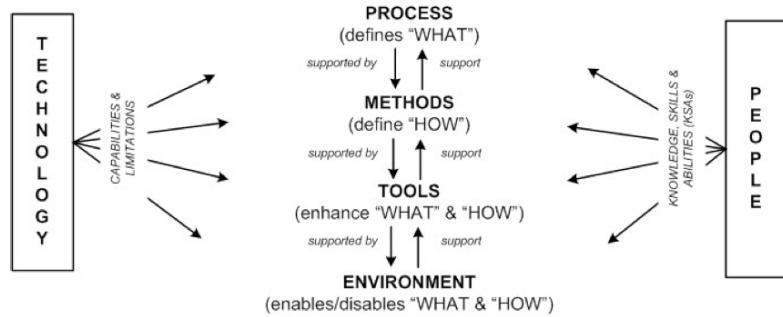


(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

14

## Desambiguação de conceitos



(Martin et al., 1996)

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

15



Escola Superior De Tecnologia e Gestão

## Engenharia de Software II



2<sup>a</sup> ano  
1º semestre

Processo do ciclo de vida do software

Por:

Cristóvão Sousa e Carla Pereira

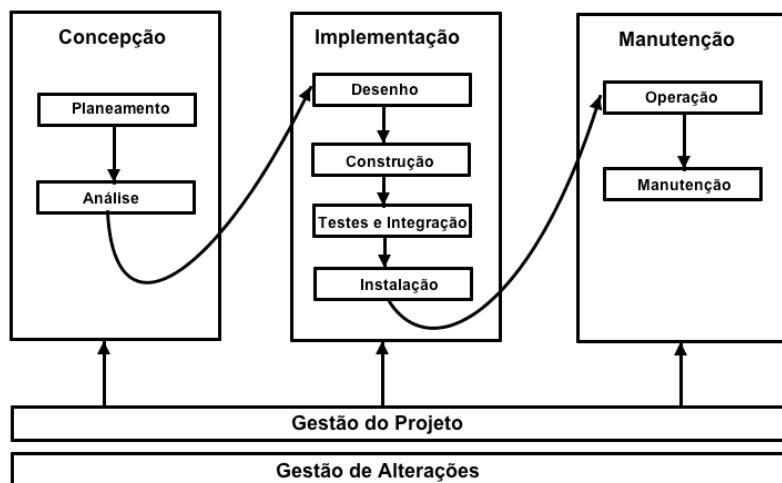
## Definição genérica de processo

**Sequência de actividades, normalmente agrupadas em fases e tarefas, que são executadas de forma sistemática e uniformizada, que são realizadas por intervenientes com responsabilidades bem definidas, e que a partir de um conjunto de inputs produzem um conjunto de outputs**

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

17

## Esquema genérico



In Engenharia de Software I

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

18

## Normas/standards

- ▶ ISO/IEC 12207 - Standard Internacional apresenta um framework (estrutura) comum para os processos de ciclo de vida de software .
- ▶ IEEE/EIA 12207.0 - apresenta algumas clarificações e alterações aceites pela IEEE (Institute of Electric and Electronic Engineers) and EIA (Electronic Industrial Association)
- ▶ IEEE/EIA 12207.2 - fornece um guia para implementar IEEE/EIA 12207.0
- ▶ SWEBOOK - Guide to the Software Engineering Body of Knowledge
  - “The purpose of the Guide to the Software Engineering Body of Knowledge is to **(a)** provide a **consensually validated characterization** of the bounds of the software engineering discipline and to **(b)** provide a **topical access** to the Body of Knowledge supporting that discipline.”

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

19

## Normas/standards

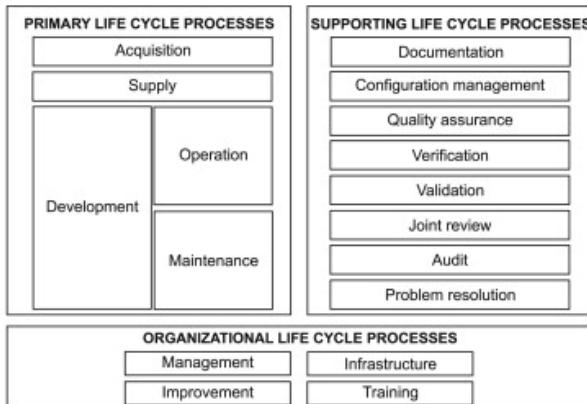
- ▶ ISO/IEC 12207:2008 ou IEEE/EIA 12207.0\*
  - Divide as actividades realizadas durante o ciclo de vida do software em 5 processos primários, 8 processos de suporte e 4 processos organizacionais
  - Processos Primários (primary life cycle processes)
  - Processos de suporte (supporting life cycle processes)
  - Processos Organizacionais (organizational life cycle processes)

\* Doravante, irei referir-me apenas a ISO&IEEE 12207 de 2008

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

20

## SDLC - ISO&amp;IEEE 12207 de 2008



§

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

21

## SDLC - Processos fundamentais

- Necessários para que um software seja executado. Eles iniciam o ciclo de vida e comandam outros processos.
  - (1) Aquisição; (2) Fornecimento; (3) **Desenvolvimento\***; (4) Operação; (5) Manutenção.
- Os envolvidos são:
  - acquirer, supplier, planner, developer, operator and maintainer de um sistema de SW
- As actividades e tarefas iniciais são da responsabilidade da organização que inicia o processo.

\*Atividades nas quais nos iremos focar

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

22

## Processos de suporte

- ▶ **Apoiam os outros processos** contribuem para o sucesso e qualidade do projecto de SW
- ▶ **Existem 8 subprocessos** e qualquer um pode ser usado nos processos primários
- ▶ Podem ser “activados” quer pela organização que os usa, por outra organização parceira ou pelo cliente.
- ▶ São eles:
  - (1) Documentação; (2) Aderência de configuração; (3) **Garantia da qualidade**; (4) **Verificação**; (5) **Validação**; (6) Revisão conjunta; (7) Auditoria; (8) Resolução de problema; (9) Usabilidade; (10) Contrato

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

23

## Processos organizacionais

- ▶ **Auxiliam a organização e gestão geral** dos processos e podem ser empregados fora do domínio de projetos e contratos específicos, servindo para toda a organização.
  - (1) Gestão; (2) Infra-estrutura; (3) Melhoria; (4) Recursos humanos; (5) Gestão de ativos; (6) Gestão de programa de reutilização; (7) Engenharia de domínio.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

24

## Desenvolvimento: Implementação

- ▶ Actividades de Implementação do Processo Fundamental de Desenvolvimento
  - A implementação consiste na **definição ou seleção de um modelo de ciclo de vida** de software apropriado ao âmbito, magnitude e complexidade do projeto e na execução de documentação dos resultados, de acordo com o processo de documentação (...)

FOCO da componente de modelos , métodos e metodologias de ESII

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

25

## Desenvolvimento:Codificação de testes de software

- ▶ Actividades de Codificação e Testes de Software do Processo Fundamental de Desenvolvimento
  - Para, executar a codificação e os testes é necessário desenvolver e documentar cada unidade de software.
  - Os testes devem garantir que os requisitos documentados são contemplados.
  - **Os resultados dos testes devem ser documentados. Durante esta fase, a atualização e documentação do utilizador pode ser feita, se necessário.**
  - Após a codificação e testes é importante fazer a avaliação do código do software e dos resultados dos testes.
  - Os resultados das avaliações devem ser documentados.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

26

## Desenvolvimento: Integração de software

- ▶ Actividades de integração de Software do Processo Fundamental de Desenvolvimento
  - **Para homologar o sistema é necessário desenvolver um plano de integração** para integrar as unidades e componentes de software.
  - **O plano deve incluir requisitos de teste**, procedimentos, dados, responsabilidades e cronograma. Deve-se testar essas agregações à medida que forem sendo integradas, de acordo com o plano de integração.
  - **Após a codificação e testes é importante fazer a avaliação do plano de integração**, projeto, código, testes, resultados dos testes e a documentação do utilizador.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

27

## Desenvolvimento: testes e qualificação de software

- ▶ Actividades de Testes de Qualificação de Software do Processo Fundamental de Desenvolvimento
- ▶ Teste de Qualificação do Software
  - Deve-se desenvolver e documentar os requisitos de qualificação de software e **elaborar casos de teste (entradas, saídas e critérios de teste) e procedimentos de teste** para conduzir o Teste de Qualificação do Software de acordo com os requisitos de qualificação para o item de software.
  - Após a codificação e testes é importante fazer a avaliação do projeto, código, testes, resultados dos testes e a documentação do utilizador.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

28

## Desenvolvimento: testes e qualificação de software

- ▶ Actividades de Testes e Qualificação do Sistema do Processo Fundamental de Desenvolvimento
- ▶ Teste de Qualificação do Sistema
  - Para garantir a qualidade do produto entregue é importante conduzir o teste de qualificação do sistema e fazer a avaliação do sistema.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

29

## Desenvolvimento: apoio à aceitação de software

- ▶ Actividades de Apoio á aceitação do Software Fundamental de Desenvolvimento
  - No apoio à aceitação do software é preciso garantir o apoio à **revisão de aceitação do cliente** e testes do produto de software.
  - A revisão de aceitação e testes deve considerar os resultados de:
    - Revisões Conjuntas, Auditorias, Teste de Qualificação do Software e Teste de Qualificação do Sistema (se executado).
  - Conclusão e entrega do produto de software deve ser feita, conforme especificado no contrato e o desenvolvedor deve dar formação e suporte ao cliente, conforme especificado no contrato.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

30

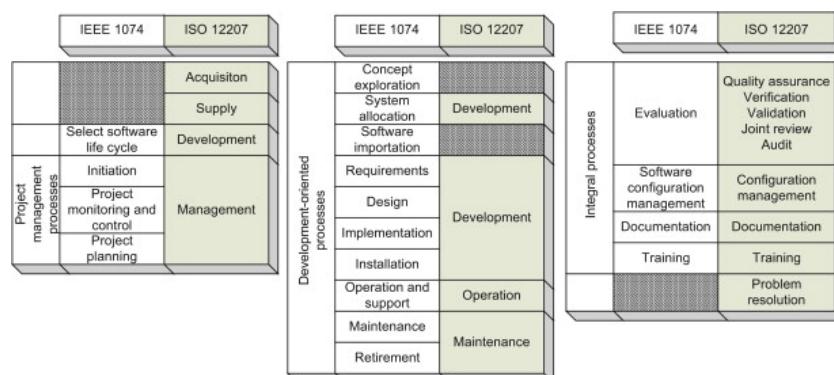
## IEEE/EIA 12207:2008 vs. IEEE Std. 1074

- ▶ IEEE/EIA 12207.0 estabelece um framework comum para o ciclo de vida do software, em termos dos processos que poderão ser usados ou aplicados para:
  - Adquirir/obter (acquire) , fornecer (supply), operar (operate) e manter (maintain) software, e
  - Gerir (manage), controlar (control) e melhorar (improve) os processos;
- ▶ **Fornece uma espécie de “guia base” para o comércio de software.**
- ▶ **Mas**, IEEE/EIA 12207.0 não especifica o detalhe da implementação desses processos.
- ▶ IEEE Std 1074 vem complementar a aplicação da IEEE/EIA 12207.0 e é direcionada para o que se pode chamar de “**process architect**”, ou seja, o indivíduo ou grupo responsável por estabelecer/definir o “**software project life cycle process (SPLCP)**” a ser seguido em determinado projeto.
- ▶ O “**process architect**” faz uso da may use IEEE Std 1074 para especificar processos para projectos cumprindo os requisitos da IEEE/EIA 12207.0.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

31

## mapear ISO12207 vs. IEEE Std. 1074

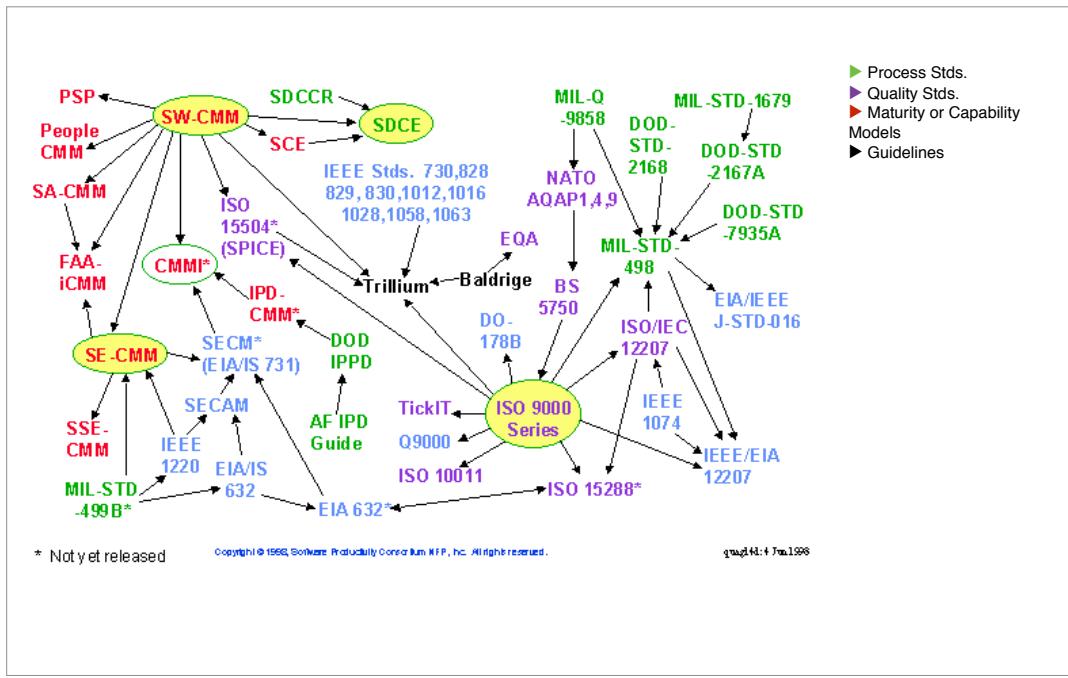


Mapping ISO 12207 to IEEE Std 1074.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

32

## o pântano dos standards



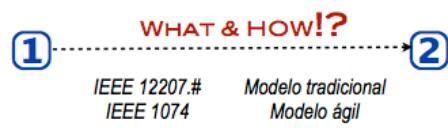
Fonte: S.A. Sheard, "Evolution of the Framework's Quagmire", ;presented at IEEE Computer, 2001, pp.96-98.

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

33

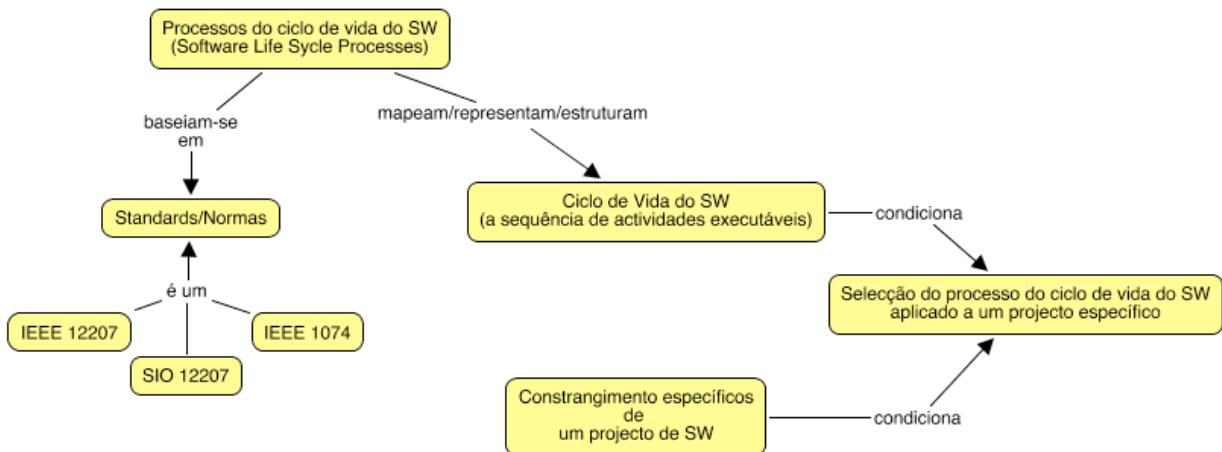
## como evitar ficar atolado no pântano?

- ▶ (1) Perceber a situação actual;
- ▶ (2) Clarificar a situação desejada;
- ▶ (3) Definir o caminho adequado entre 1 e 2;



:: Um processo define o “**what**”  
:: O modelo define o “**how**”

## Resumo



Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

35

**ESTGF** | POLitécnico  
do Porto  
Felgueiras

Escola Superior De Tecnologia e Gestão



2<sup>a</sup> ano  
1º semestre

Modelos do ciclo de vida do desenvolvimento de software

Por:

Cristóvão Sousa e Carla Pereira

## Sumário

- ▶ Modelos do ciclo de vida do desenvolvimento do software
- ▶ Como seleccionar um modelo?
- ▶ Pequeno exercício

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

37

## implementação

- ▶ Recuperando um dos “slides” anteriores:
  - A implementação consiste na definição ou **selecção de um modelo de ciclo de vida de software** apropriado ao âmbito, magnitude e complexidade do projeto e na execução de documentação dos resultados, de acordo com o processo de documentação (...)

**QUAIS OS MODELOS?  
COMO ESCOLHER?**

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

38

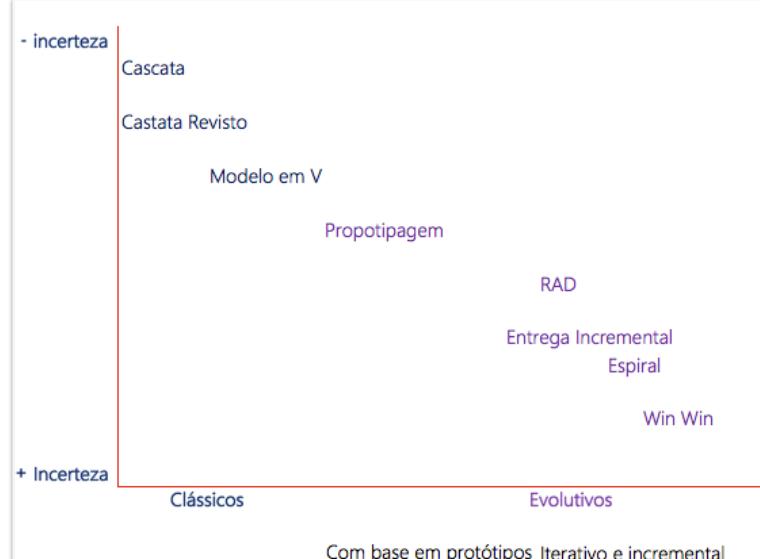
## modelos

- ▶ Modelo em Cascata
- ▶ Modelo em Cascata revisto
- ▶ Modelo V
- ▶ Modelos evolutivos
  - ▶ Modelo de Prototipagem
  - ▶ Modelo RAD
  - Iterativo e Incremental
  - Espiral
  - Espiral WinWin
- ▶ Baseado em componentes
- ▶ Técnicas de 4<sup>a</sup> Geração

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

39

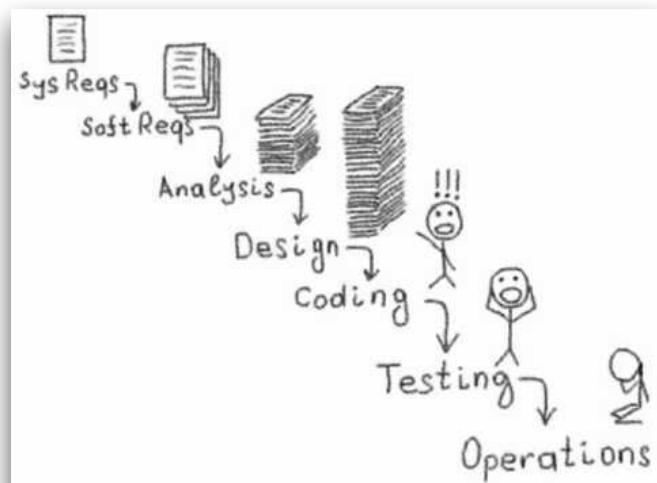
## Visão Geral



Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

40

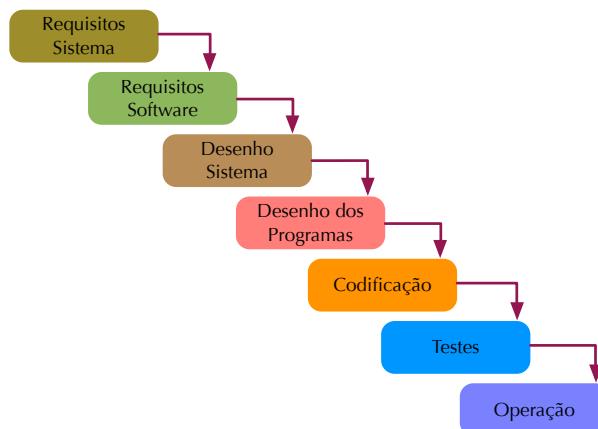
## Modelo em Cascata (WaterFall)



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

41

## Modelo em Cascata p/ Cascata Revisto

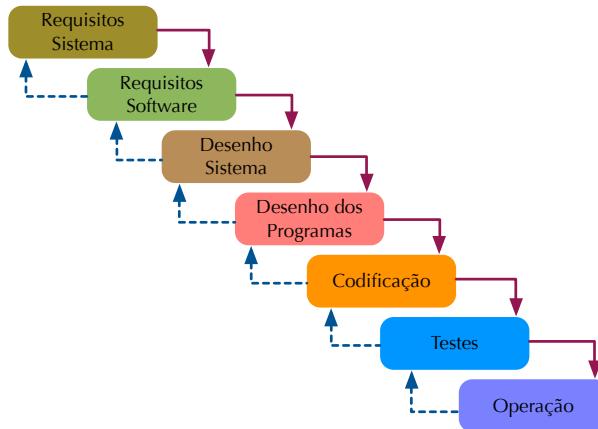


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

42

## Modelo em Cascata p/ Cascata Revisto

- Possibilidade de mais do que uma **iteração** entre duas fases

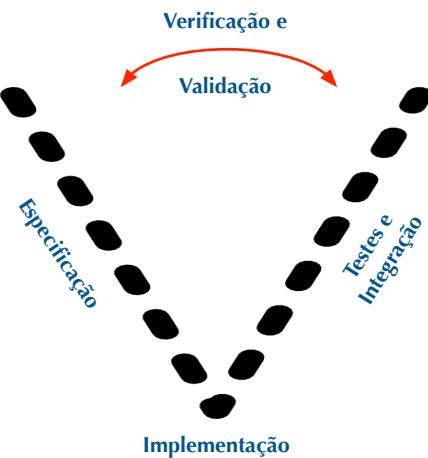


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

43

## Modelo em Cascata Revisto p/ Modelo em V

- Abordagem

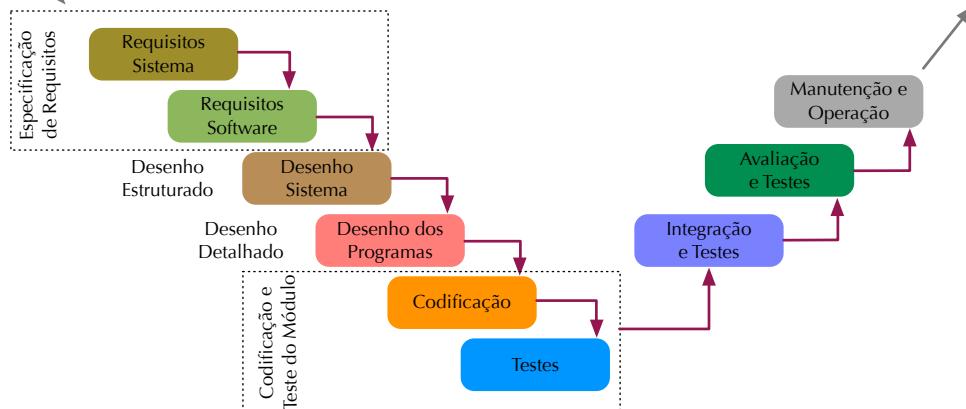


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

44

## Modelo em Cascata Revisto p/ Modelo em V

- ▶ **Alteração do nome das fases:** agrupam-se umas, extende-se outras ... mas o processo mantém a sua estrutura de base.

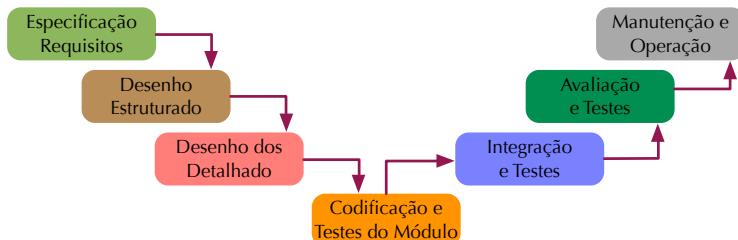


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

45

## Modelo em Cascata Revisto p/ Modelo em V

- ▶ Fases do **V-Model**

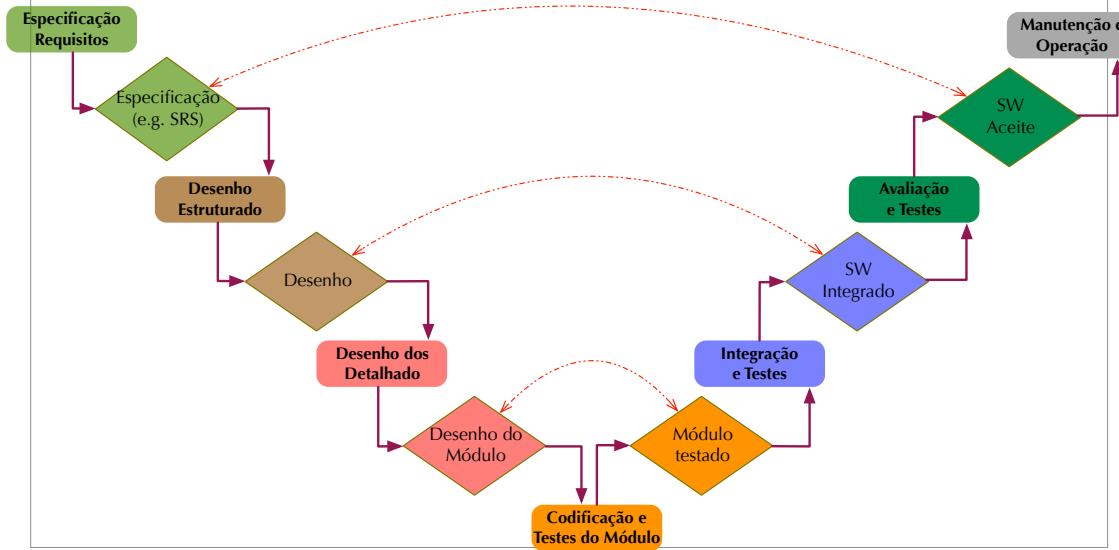


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

46

## Modelo em Cascata Revisto p/ Modelo em V

- ▶ **Acrescentam-se produtos entre fases de modo a verificar o progresso**
- ▶ **Estabelece-se relações entre produtos para validar resultados**

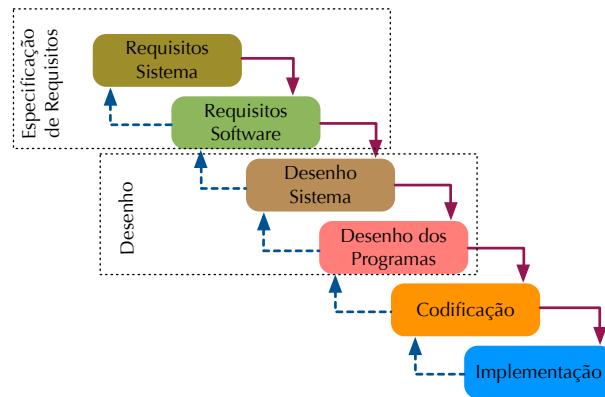


Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

- ▶ e quando o **cliente não sabe em concreto o que quer**, tornando-se difícil compreender a complexidade do projeto e consequentemente, orçamentar, calendarizar, definir arquiteturas de selecionar tecnologias?

## Evolução para modelos de prototipagem

- **Motivação:** melhor **compreender o detalhe dos requisitos** e validar abordagens para implementação dos mesmos

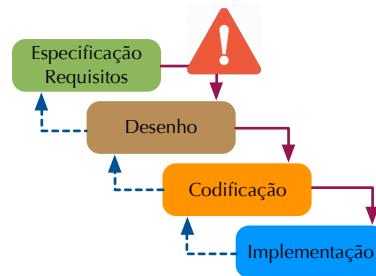


Como lidar com a incerteza de um projeto? Apostar em abordagem iterativas mais inovadoras.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

49

## Evolução para modelos de prototipagem

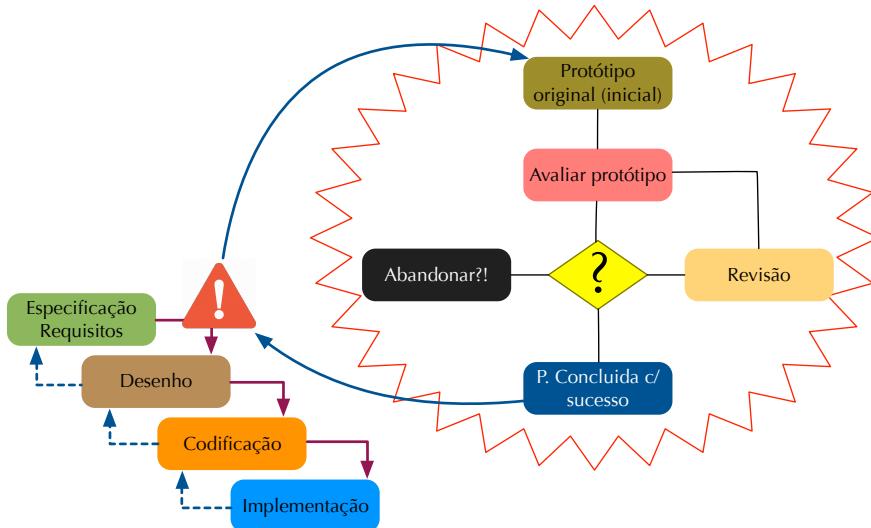


- Se é altamente provável haver incertezas num projeto e o cliente, tipicamente, não sabe o que quer, então
- a **especificação dos requisitos** será uma dura tarefa e o desenho da solução poderá não corresponder ao pretendido pelo cliente

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

50

# Evolução para modelos de prototipagem

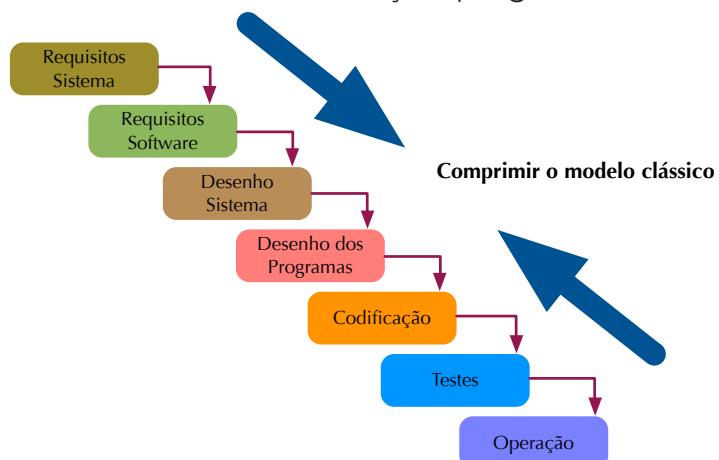


Engenharia de Software II - Engenharia Informática  
\* Cristóvão Sousa e Carla Pereira\*

51

RAD

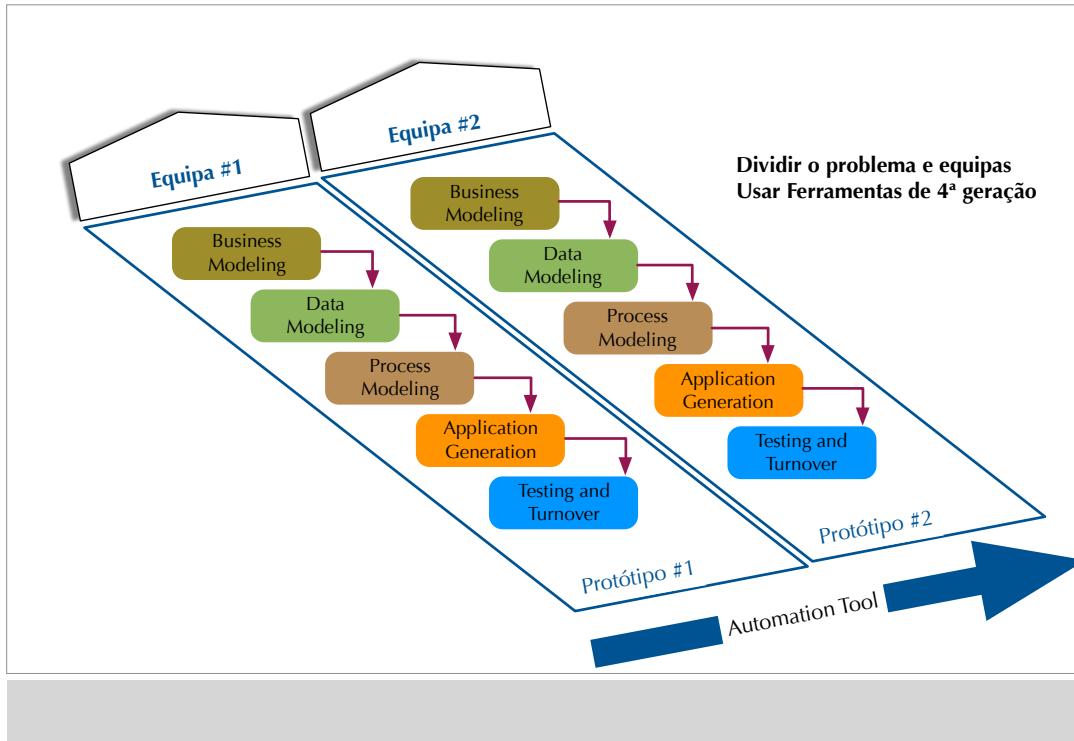
- ▶ **Motivação:** E se para além da necessidade de lidar com a incerteza associada a um projeto, o **tempo é um fator crítico?**
    - como **rapidamente** compreender o detalhe dos requisitos e automatizar a sua codificação progressiva?



Engenharia de Software II - Engenharia Informática  
\* Cristóvão Sousa e Carla Pereira\*

52

## Waterfall (iterativo) to RAD

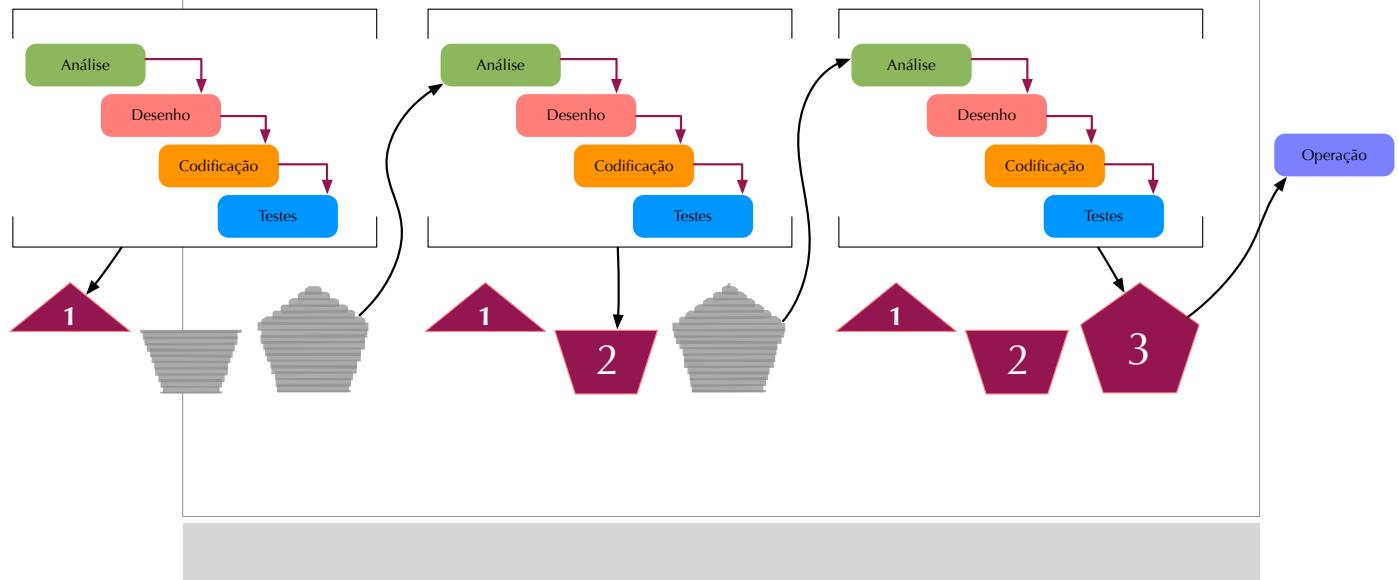


Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

- e quando o a partir de uns requisitos **surgem outros que quer o cliente, quer a equipa de desenvolvimento não tinham identificado?** Como lidar com o **alargamento do âmbito do sistema?**

## Modelos Iterativo e Incremental

► **Motivação:** Abordagem **incremental**?

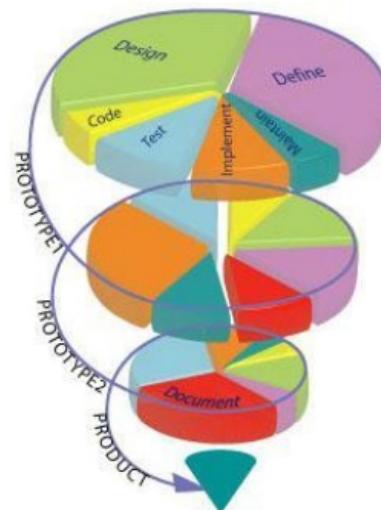


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

55

## Modelo Expiral

► **Motivação:** apostar na gestão de risco relativamente às interações e incrementos, uma vez que o projeto poderá “fugir” do controlo **incremental**?

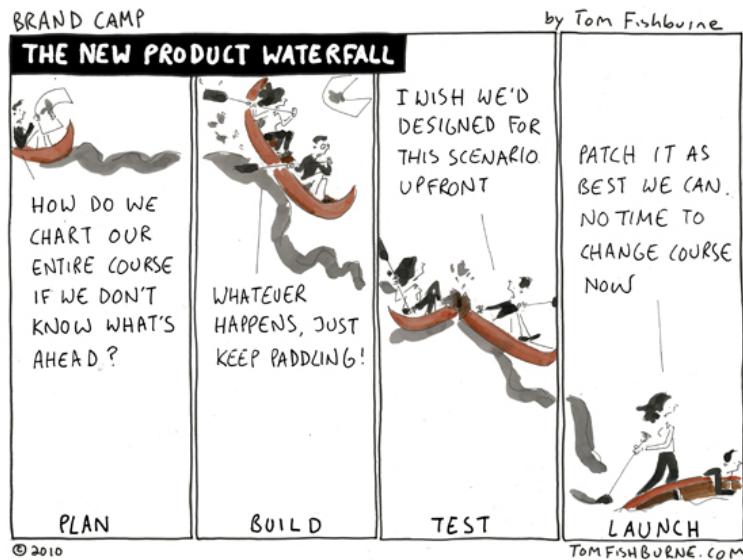


Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

56

# Os modelos em detalhe

## Classificação dos modelos

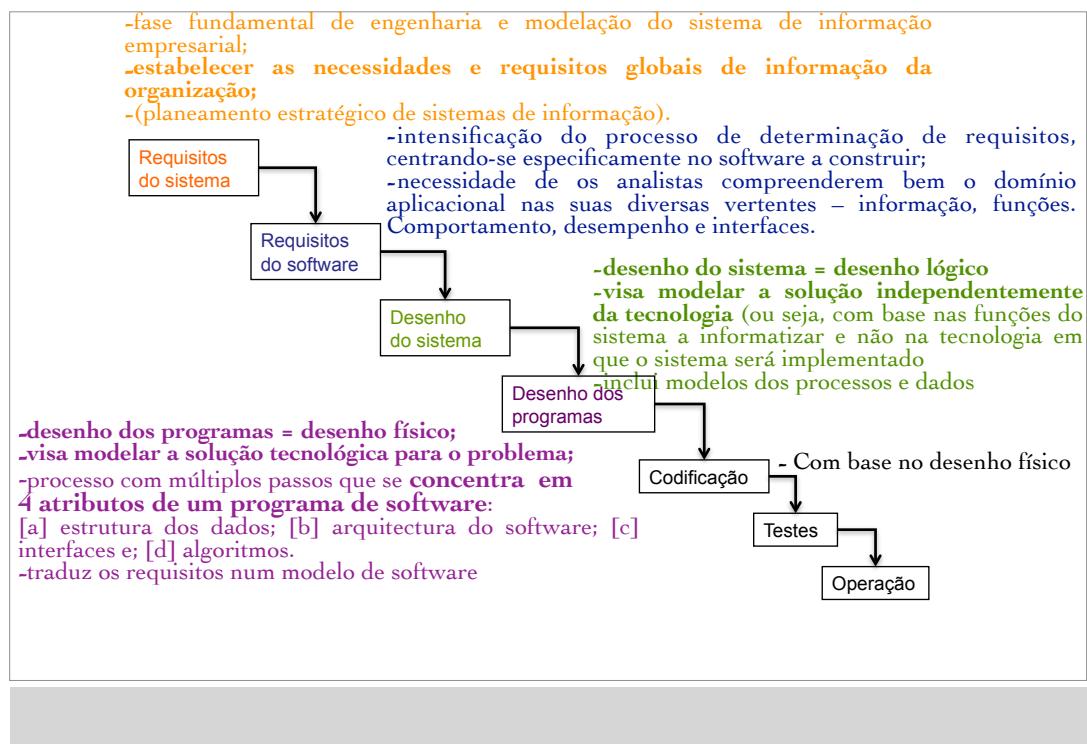


# Waterfall

## modelo em cascata

- ▶ Apareceu no início dos anos 70 e foi o primeiro paradigma que veio tentar disciplinar e sistematizar o DSI.
- ▶ **Processos tradicionais de desenvolvimento de software** são caracterizados por adoptarem este modelo.
- ▶ **As actividades a executar são agrupadas em tarefas, executadas sequencialmente**, de forma a que uma tarefa só tem início quando a tarefa anterior tiver terminado.
- ▶ Tem a vantagem que **só se avança para a tarefa seguinte quando o cliente valida e aceita os produtos finais** (documentos, modelos, programas) da tarefa actual.
- ▶ Baseia-se no pressuposto que o cliente participa activamente no projecto e que sabe muito bem o que quer.

## modelo em cascata



Engenharia de Software II - Engenharia Informática

\* Crístovão Sousa e Carla Pereira\*

61

## modelo em cascata: limitações

- ▶ Promove a **compartimentação dos esforços** <=> **falta de comunicação** e partilha de visões
- ▶ **Rigidez**
  - minimiza o impacto da compreensão adquirida no decurso de um projecto, uma vez que se um processo não pode voltar atrás de modo a alterar os modelos e as conclusões das tarefas anteriores, é normal que as novas ideias sobre o sistema não sejam aproveitadas
- ▶ Raros são os projectos que seguem um fluxo sequencial
- ▶ **Raramente o cliente consegue transmitir os requisitos de forma explícita**
- ▶ Requer elevadas doses de paciência por parte do cliente. Uma versão “usável” só está pronta num estado avançado do projecto.

Engenharia de Software II - Engenharia Informática

\* Crístovão Sousa e Carla Pereira\*

62

## modelo em cascata: conclusão

- ▶ Resistência a mudanças de requisitos ...Adequado quando os requisitos estão muito bem definidos
- ▶ Método pouco flexível...permite pouca interacção entre as diferentes fases
- ▶ Para projectos grandes permite prever com maior precisão as datas de conclusão envolvidas
- ▶ Mas, infelizmente, não evita que o projecto não obtenha os resultados originalmente esperados, que o cliente fique desapontado, e que se tenha desperdiçado tempo e recursos indevidamente.
- ▶ O utilizador tem de esperar, por vezes bastante tempo, até lhe ser entregue uma versão operacional do sistema de software
- ▶ o actual ritmo de mudança dos negócios não é compatível com esse compasso de espera (quando o sistema é entregue , há o sério risco de os requisitos iniciais se terem alterado substancialmente...)

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

63

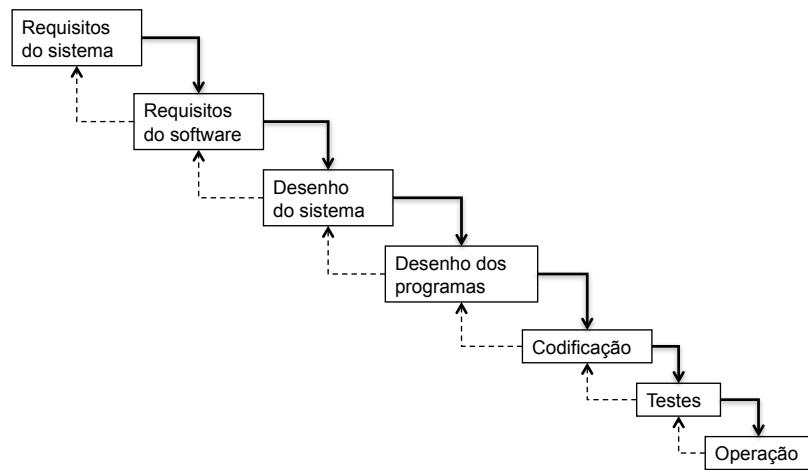
## modelo em cascata revisto

- ▶ Foi definido, **baseado no modelo clássico em cascata, para eliminar os problemas apresentados atrás**
- ▶ Prevê a possibilidade de **a partir de qualquer tarefa do ciclo se poder regressar a uma tarefa anterior de forma a contemplar alterações funcionais e/ou técnicas que tenham surgido**, em virtude de um maior conhecimento que se tenha obtido  
*é necessário que exista* ↗
- ▶ Na ausência de **um processo de gestão do projecto e de controlo** das alterações bem definido, podemos passar o tempo num ciclo sem fim, sem nunca atingir o objectivo final. (**Risco!**)

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

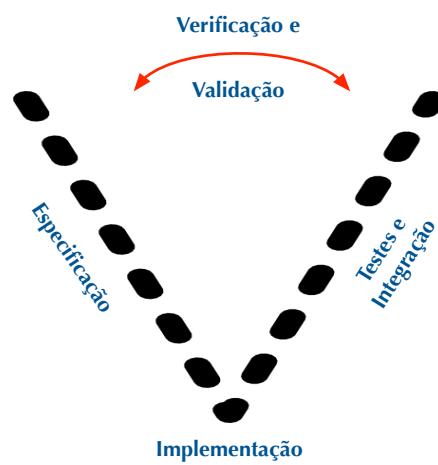
64

## modelo em cascata revisto



Engenharia de Software II - Engenharia Informática  
\* Crístóvão Sousa e Carla Pereira\*

65



V-Model

## Modelo em V

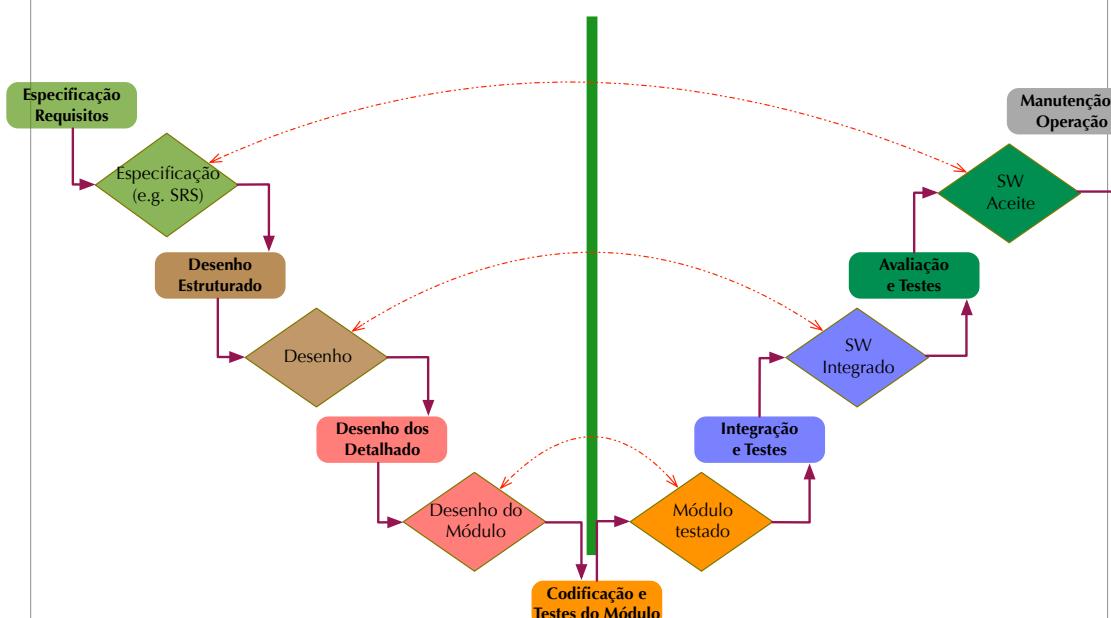
- ▶ V de Verificação
- ▶ V de Validação
- ▶ Evolução do modelo em cascata e em muito semelhante a esse modelo:
  - sequencial; avança para a fase seguinte com a primeira está concluído, mas
- ▶ A fase de testes é planeada em **paralelo** com a fase de desenvolvimento correspondente

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

67

## modelo em V

### Planeamento



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

68

## modelo em V

- ▶ O processo de desenvolvimento é basicamente dividido em duas partes, as duas pernas do V, a parte da especificação e da verificação e validação
- ▶ No diagrama (fig. anterior) as caixas rectangulares representam as fases e os ovais representam os produtos, que não são mais do que o resultado de uma revisão satisfatória da fase anterior, apresentadas sob a forma de documentos, e que vão servir de base de trabalho para a fase posterior.
- ▶ Este paradigma sugere que nenhuma fase pode ser considerada completa.

Esta aproximação não é realista, principalmente em grandes projectos, e tendo em conta o aspecto dinâmico dos requisitos no tempo actual.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

69

## Modelo em V

- ▶ Vantagens principais
  - **Simples é fácil de usar:** dá uma estrutura global ao projecto, sob a forma de uma sequência bem definida de fases.
  - Cada fase tem “deliverables” específicos
  - Maior probabilidade de sucesso do que o modelo clássico
  - Funciona bem para projectos pequenos em que os requisitos são facilmente entendidos
  - **reconhece que não existem soluções perfeitas e que não é comercialmente sensato procurar uma.**
  - **As atividades de planeamento e desenho têm lugar antes da codificação**

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

70

## modelo em V

### ► Desvantagens

- **Muito rígido** - tal como acontecia com o modelo em cascata
- **Pouca flexibilidade** - os ajustamentos são difíceis e caros
- O software é desenvolvido apenas na fase de construção e por isso não existem protótipos (funcionais ou não funcionais)
- o facto de um compromisso poder, no limite, transformar-se num erro e um projecto acabar por descobrir, talvez demasiado tarde no processo de desenvolvimento, que seguiu um caminho errado. (Falta de gestão dos riscos...)

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

71

### ► Bases dos modelos anteriores:

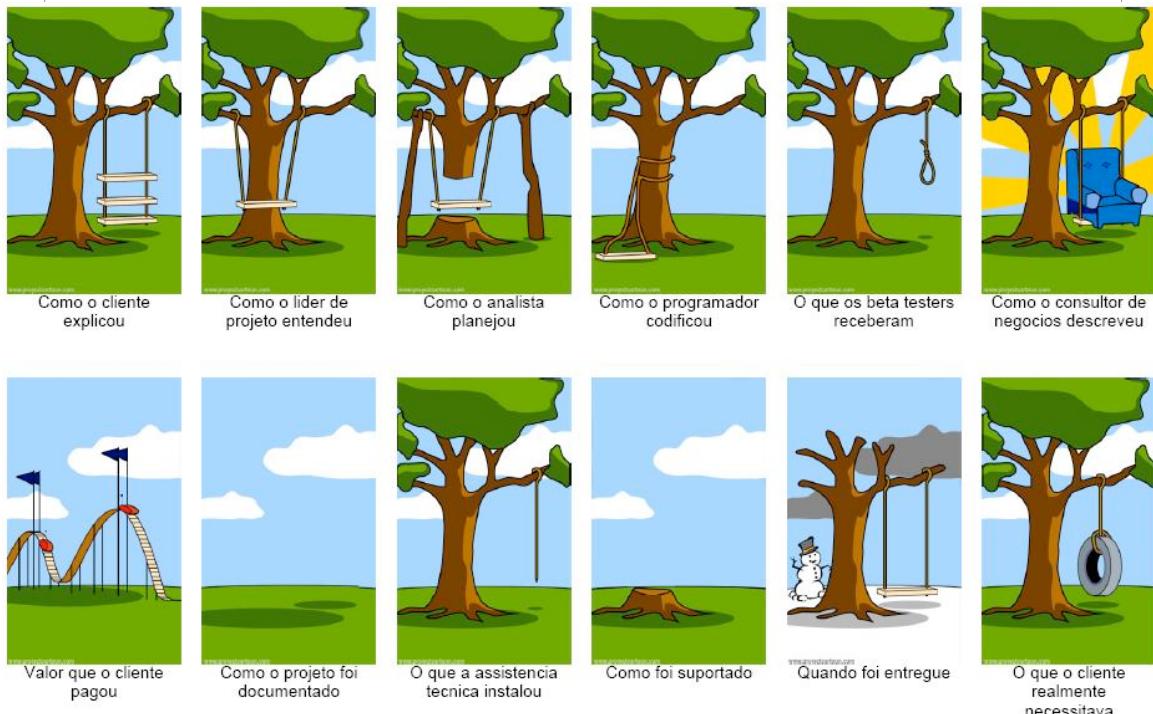
- não há grandes incertezas no projecto!!??



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

72

## o “verdadeiro” processo :)

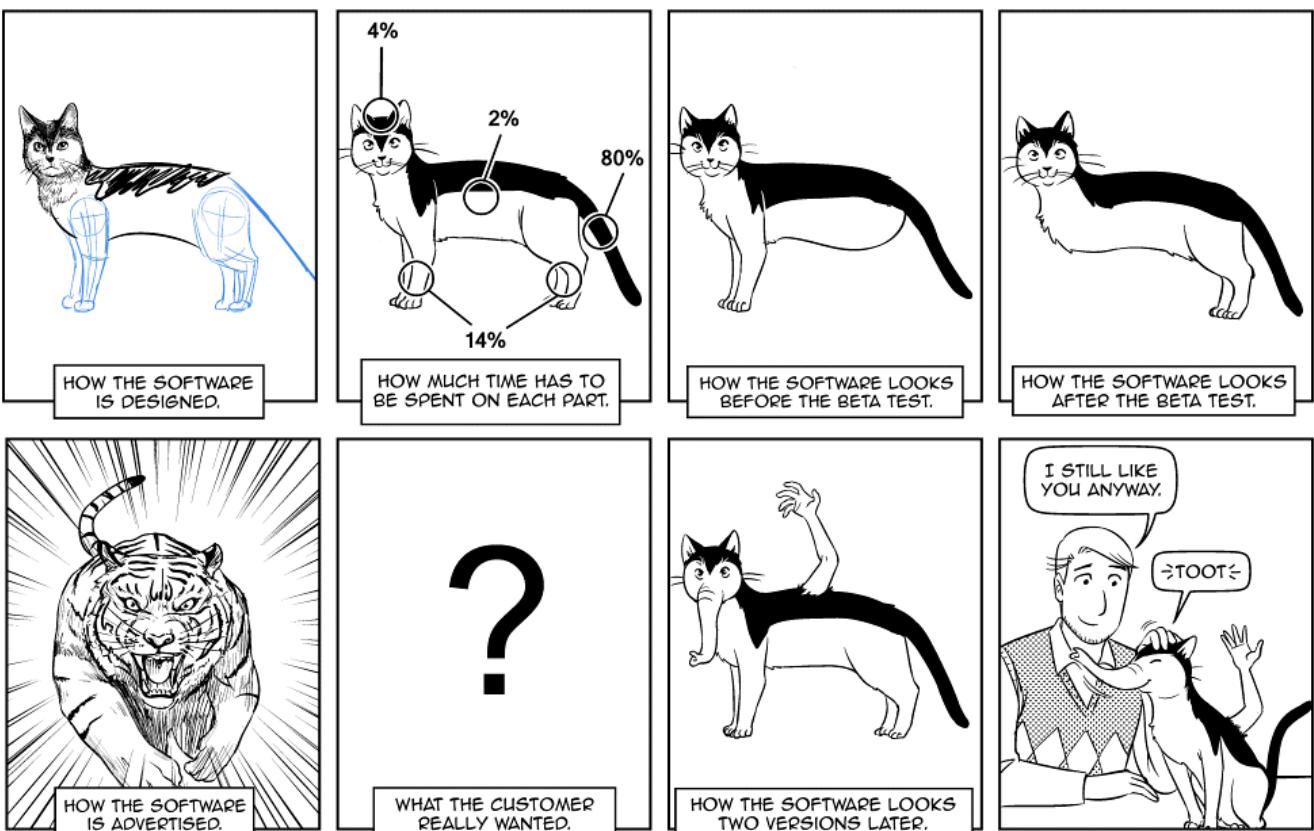


Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

73

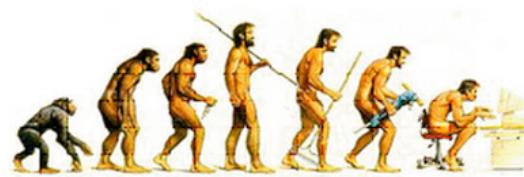
ESI

## Richard's guide to software development



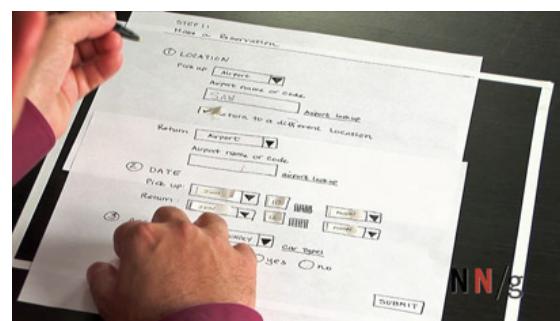
## modelos evolutivos

- Possibilitam a construção de versões cada vez mais completas do software



Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

75



# Prototipagem

## modelo de prototipagem

### ► Tipicamente:

- o cliente define um conjunto de objectivos globais para o sistema, **mas não identifica detalhadamente os inputs, o processamento ou os outputs;**

### ► Por vezes:

- é o engenheiro de software que pode não estar seguro acerca da eficiência (ou mesmo da aplicabilidade) de **um dado algoritmo**, da adequação de um SGBD ou da forma que deverá assumir a interface com o utilizador
- Nestas e noutras situações em que existem incertezas, **a prototipagem constitui uma das medidas(\*) de redução do risco** que se podem utilizar.

(\*) Pode não ser considerada um modelo completa mas sim uma abordagem para lidar com partes (maiores) de outros modelos, como: Espiral, Incremental, ou RAD.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

77

## modelo de prototipagem

- Processo Iterativo
- Tenta reduzir riscos inerentes ao projecto através da divisão do projecto em segmentos mais pequenos possibilitando uma menor resistência à mudança durante o processo de desenvolvimento.
- O utilizador é envolvido no processo o que aumenta a probabilidade de o utilizador aceitar a implementação final
- “**Small-scale mockups**” do sistema são desenvolvidos de acordo com um processo de modificação/ajuste iterativo até que o protótipo evolua de forma a atender as necessidades do utilizador.
- A maior parte dos protótipos é desenvolvida na expectativa de que sejam posteriormente “eliminados”, contudo é possível, em alguns casos, evoluir para protótipos completamente funcionais.
- É necessária uma compreensão do problema de modo a evitar uma solução para o problema errado.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

78

## modelo de prototipagem

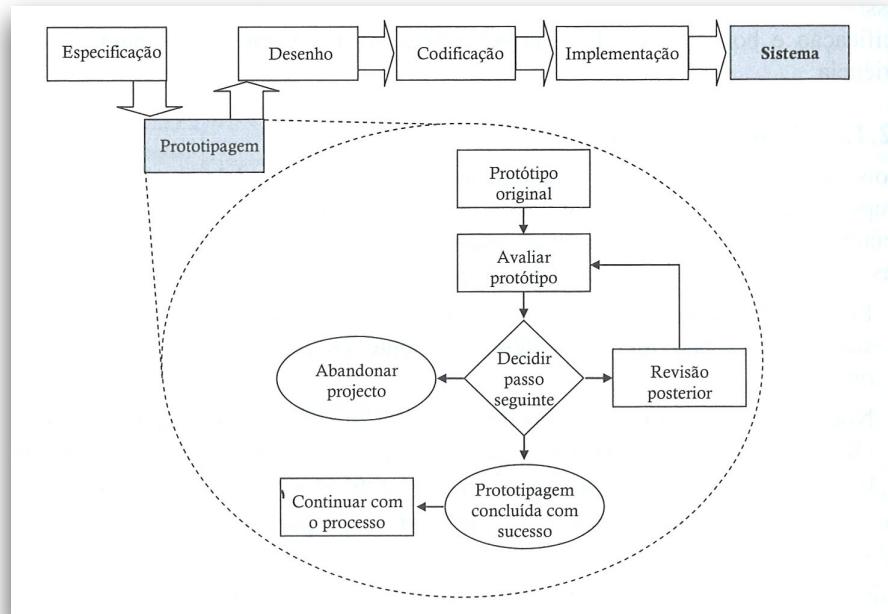
### ► Como?

- inicia-se com uma “expressão” dos requisitos
- o **engenheiro de sistemas** (analista) e o **cliente/utilizador definem**, em conjunto:
  - os objectivos globais para o sistema de software;
  - identificam os requisitos que forem conhecidos no momento
  - definem áreas em que é necessária uma definição posterior
- desenha-se o protótipo do sistema
- é avaliado pelo cliente/utilizador e depois utilizado para refinar os requisitos do software

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

79

## modelo de prototipagem



Experimentação iterativa com o propósito de obter informação para o processo de desenvolvimento

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

80

## modelo de prototipagem (**desvantagens**)

- ▶ Requer processo de **controlo e aprovação - que não é rigoroso!**
- ▶ **Pode ocorrer uma análise incompleta e inadequada do problema** e apenas serem consideradas as necessidades mais básicas e superficiais
- ▶ **O requisitos podem mudar significativamente e frequentemente**
- ▶ **A identificação de elementos não funcionais é difícil de documentar**
- ▶ Os “designers” podem desenvolver um protótipo rapidamente sem a análise prévia das necessidades dos utilizadores. **O resultado pode ser um desenho inflexível com um enfoque muito estreito em relação á realidade.**
- ▶ Há a necessidade de “Designers” qualificados

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

81

## modelo de prototipagem (**desvantagens**)

- ▶ Caso não haja uma correcta **gestão das expectativas**, quer do cliente/utilizador, quer do engenheiro de software:
  - o cliente/utilizador vê aquilo que aparenta ser uma versão **funcional do sistema** que ele pretende, sem se aperceber que se trata apenas de algo construído sem grandes preocupações
  - a fim de ter um protótipo a funcionar num intervalo de tempo reduzido, o engenheiro de software faz muitas vezes compromissos de implementação (p.e., a utilização de uma linguagem de programação, ou um algoritmo, pouco eficientes)
- ▶ As várias iterações do processo devem ser consideradas na **gestão do projecto**, “pesadas” em relação aos benefícios. Em projectos muito pequenos pode não se justificar esta abordagem. Apenas as partes maiores e de risco elevado de projectos complexos poderão justificar esta abordagem.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

82

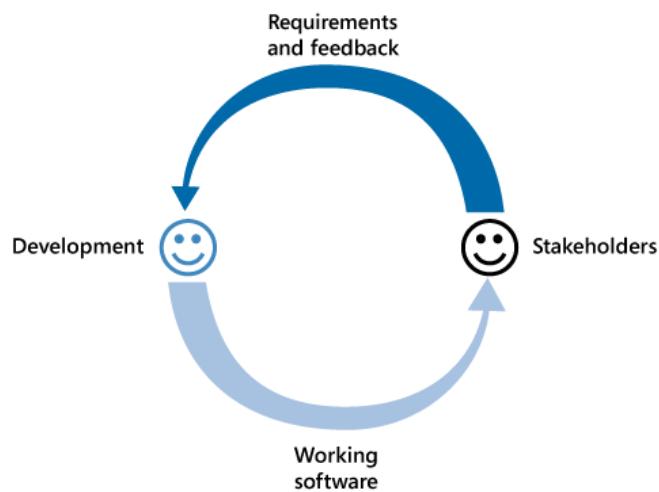
## modelo de prototipagem (**vantagens**)

- ▶ Lida com a incapacidade que a maioria dos utilizadores tem em especificar as suas necessidades
- ▶ Usada para modelar de forma realista aspectos específicos do sistema em cada fase do ciclo de vida tradicional do desenvolvimento de SW
- ▶ Melhora a participação do utilizador do desenvolvimento do sistema e a comunicação com os stakeholders
- ▶ **Útil** na: i) resolução de objectivos pouco claros; ii) desenvolvimento e validação dos requisitos do utilizador; iii) experimentação e comparação de várias desenhos de soluções; iv) investigação na interacção homem-máquina;
- ▶ Existe potencial para exploração do conhecimento ganho numa iteração enquanto se desenvolvem as outras.
- ▶ Encoraja a inovação e flexibilidade
- ▶ Possibilita uma rápida implementação do sistema que, embora incompleta , pode ser funcional

---

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

83



# RAD

## modelo RAD

### ► **Processo iterativo de desenvolvimento de SW**

- Versão “high-speed” do modelo em cascata, usando uma abordagem ao desenvolvimento baseada em componentes.
- Ideia básica:
  - desenvolver e entregar um sistema com elevados índices de qualidade ao mais baixo custo possível
- Divide o problema em problemas mais simples o que lhe confere uma menor resistência à mudança
- Pode fazer uso de protótipos fazendo bastante uso de ferramentas GUI e CASE (Computer Aided Software Engineering) e Database Management Systems (DBMS) e ainda linguagens de 4<sup>a</sup> geração, geradores de código e técnicas orientadas por objectos.

---

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

85

## modelo RAD

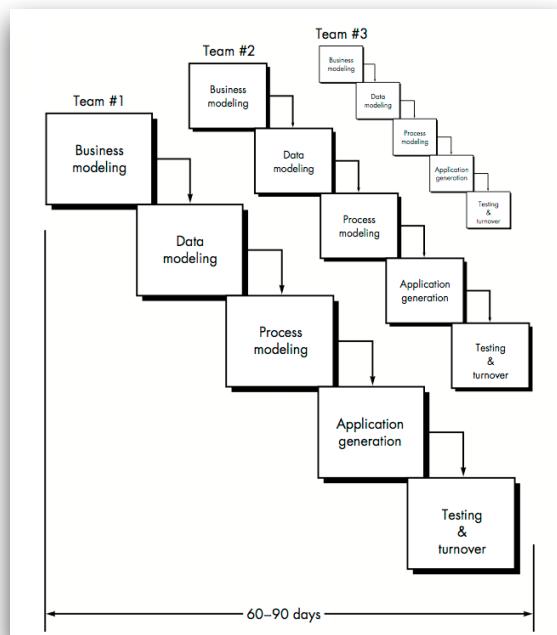
- Criar um sistema funcional num curto espaço de tempo (60 a 90 dias)
- Aplicável se:
  - os requisitos são bem entendidos
  - o âmbito do projecto é restrito

---

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

86

## modelo RAD

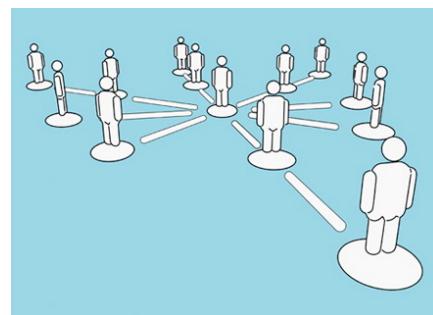


Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

87

## modelo RAD: processo

- ▶ Business modeling
  - Que **informação** suporta o **processo de negócio**?
  - Que **informação** é gerada e **quem a gera**?
  - Para onde flui a **informação** e **quem a processa**?



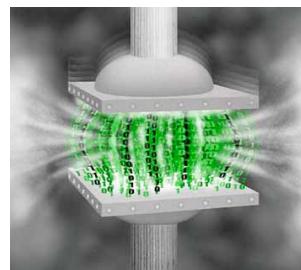
Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

88

## modelo RAD: processo

### ► Data modeling

- **fluxo de informação** => “data objects” de suporte ao negócio



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

89

## modelo RAD: processo

### ► Process modeling

- **descrição do processo** <=> adding; modifying; deleting; or retrieving a data object



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

90

## modelo RAD: processo

### ► Application generation

- RAD implica:

- (a) reusar componentes - quando possível
- (b) criar componentes reutilizáveis - quando necessário
- (a) e (b) implica o uso de técnicas denominadas de 4ª geração (ferramentas automáticas)



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

91

## modelo RAD: processo

### ► Testing and turnover

- RAD implica reutilização

- reusar componentes significa componentes já testados
- componentes já testados significa mais tempo
- tempo significa €€€€€

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

92

## modelo RAD (**desvantagens**)

- ▶ Grandes projectos => recursos humanos suficientes => €€
- ▶ RAD => colaboração (programadores, clientes)
  - a falta desta disponibilização para colaborar pode condicionar o modelo
- ▶ RAD adequa-se a todo o tipo de projectos/aplicações?
  - se não for possível modularizar o sistema, então não
  - se a performance é uma questão muito pertinente, então cuidado
  - quando os ricos técnicos são elevados, então não

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

93

## Características dos modelos iterativos

- ▶ A noção de processo iterativo corresponde à **ideia de “melhorar (ou refinar) pouco-a-pouco” o sistema**
- ▶ O âmbito do sistema não é alterado, mas o seu **detalhe vai aumentando em iterações sucessivas**
- ▶ Num processo iterativo:
  - Os riscos e dúvidas com maior importância são identificados no início do processo (nas primeiras iterações) quando é possível tomar medidas para os corrigir
  - Esta abordagem **encoraja a participação activa** dos utilizadores de modo a identificar os verdadeiros requisitos do sistema
  - **A execução de testes contínuos e iterativos permite uma avaliação objectiva do estado do projecto**
  - **As inconsistências entre a análise, o desenho e a implementação são identificadas atempadamente**

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

94

## Características dos modelos iterativos

► ...

- O esforço dos diversos elementos da equipa é distribuído ao longo do tempo
- A equipa pode aprender com experiências anteriores e melhorar continuamente o processo
- Pode ser demonstrado inequivocamente a todos os envolvidos e interessados no projecto o respectivo avanço

---

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

95



# Entrega incremental

## Modelos baseados em entrega incremental

- ▶ Combina elementos do modelo sequencial com a filosofia iterativa do modelo de prototipagem
- ▶ A noção de processo incremental corresponde à ideia de “aumentar (ou alargar) pouco-a-pouco” o âmbito do sistema.
- ▶ O princípio do processo iterativo e incremental é que a equipa envolvida possa refinar e alargar pouco-a-pouco a qualidade, detalhe e âmbito do sistema envolvido
- ▶ Ideia básica:
  - São executadas um conjunto de “mini-cascatas” em que todas as fases completas do modelo de desenvolvimento em cascata completam uma pequena parte dos sistema
  - Em alternativa ao ponto anterior: O conceito inicial do software (análise de requisitos, desenho da arquitectura e o “core” do sistema) são definidos segundo o modelo em cascata, em seguida através do modelo de prototipagem iterativo que culmina com a instalação do protótipo final.

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

97

## Entrega incremental (desvantagens)

- ▶ Quando neste modelo de opta pela utilização de “**mini-cascatas**” é frequente uma falta de consideração do problema do negócio e os requisitos técnicos do sistema
- ▶ Alguns módulos são completos muito antes de outros o que requer interfaces bem definidos
- ▶ Os problemas tendem a ser “empurrados” para uma altura posterior como forma de evidenciar o sucesso antecipado

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

98

## Entrega incremental (vantagens)

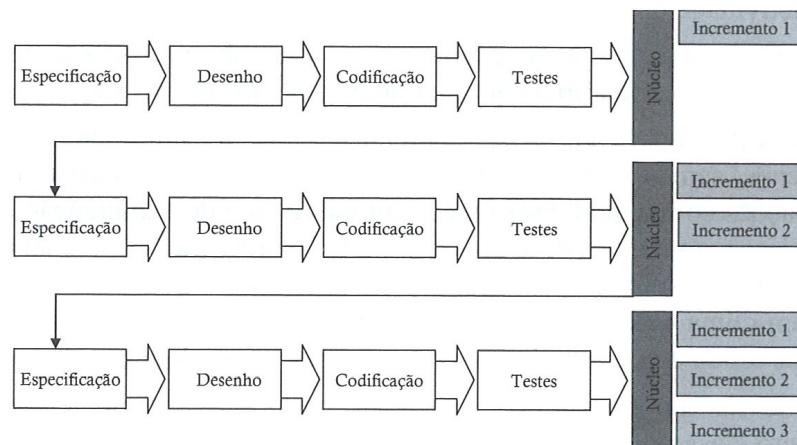
► (por oposição ao modelo em cascata):

- possibilidade de **avaliar mais cedo os riscos** e pontos críticos ou mais significativos do projecto, e identificar medidas para os eliminar ou pelo menos controlar
- **definição de uma arquitectura que melhor possa orientar todo o desenvolvimento**
- disponibilização natural de um conjunto de regras para melhor controlar os inevitáveis pedidos de alterações futuras
- **permite** que os vários intervenientes possam trabalhar mais efectivamente pela **interacção e partilha** de comunicação daí resultante
- Este método reduz o risco de falha global de todo o projecto (desenvolvimento do sistema)

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

99

## Entrega incremental



Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

100

## Modelos baseados em entrega incremental

- ▶ Este modelo de processo é útil quando:
  - Só se consegue visualizar, no início, uma parte da funcionalidade;
  - Não se consegue entregar toda a funcionalidade no mesmo dia;
  - O negócio não consegue suportar toda a mudança, de uma só vez.
- ▶ Este modelo de processo é menos útil quando:
  - Projectos pequenos e de curta duração
  - Os riscos de integração e da arquitectura são baixos
  - Aplicações extremamente interactivas e a informação para o projecto já existe, e quando o projecto comprehende uma forte componente de análise e reporting.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

101



**Modelo espiral**

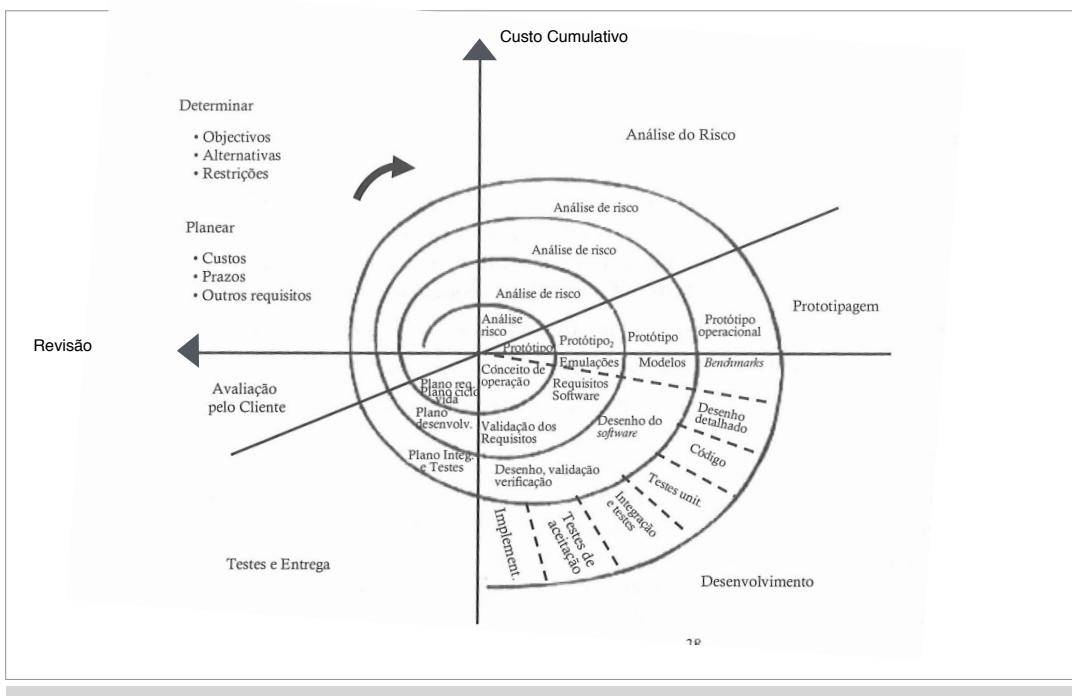
## Modelo em espiral

- ▶ É uma pequena variante do modelo iterativo e incremental
- ▶ Foi desenvolvido para incluir os melhores aspectos do ciclo de vida convencional e da prototipagem (até aqui igual ao incremental), acrescentando uma nova fase, a análise de risco (inexistente em qualquer um dos modelos anteriormente referidos)
- ▶ Desenvolvimento é efectuado mediante a repetição de um processo básico com seis fases:
  - **Determinação**, em conjunto com o cliente/utilizador, dos **objectivos, alternativas e restrições** do desenvolvimento, e **planeamento dos custos, prazos** e outros requisitos;
  - **Avaliação das alternativas e análise dos riscos**;
  - Realização de **protótipos**;
  - **Desenvolvimento do produto de software** para o cliente;
  - **Teste** e disponibilização do software;
  - **Submissão do software à avaliação pelo cliente**.

Engenharia de Software II - Engenharia Informática  
\* Cristóvão Sousa e Carla Pereira\*

103

## Modelo em espiral



Engenharia de Software II - Engenharia Informática  
\* Cristóvão Sousa e Carla Pereira\*

104

## Modelo em espiral (vantagens)

- ▶ **Aumenta a capacidade e evitar riscos**
  - ▶ Pode incorporar os modelo em cascata, de prototipagem e as metodologias incrementais e fornecendo simultaneamente "pistas" para a selecção da melhor metodologia de acordo com o risco do projecto
    - Exemplo:
      - **Proj-a: elevado risco de "falhar" o orçamento e baixo risco dos requisitos não serem os correctos -> Abordagem: modelo em cascata**
      - **Proj-b: baixo risco de "falhar" o orçamento e elevado risco dos requisitos não serem os correctos -> Abordagem: modelo de prototipagem**
  - ▶ **A dimensão do raio, em cada instante, mede os custos incorridos até ao momento;**
  - ▶ Em cada ciclo são removidos os riscos considerados suficientes para poder prosseguir para o ciclo seguinte;
  - ▶ O processo é orientado para os riscos, em vez de ser orientado para o produto.
  - ▶ Abordagem realista para projectos de desenvolvimento de larga escala

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

105

## Modelo em espiral (desvantagens)

- ▶ **Não é simples de aplicar a projectos subcontratados** que não possuem a flexibilidade e a liberdade que o modelo assume;
- ▶ **Os pressupostos subjacentes ao modelo, isto é, a sua confiança nos métodos de avaliação dos riscos, constituem um problema, na medida em que se pressupõe que os responsáveis pelo desenvolvimento "possuem sempre a capacidade de identificar e gerir riscos";**
- ▶ **O modelo é de aplicação complexa** (devido à grande possibilidade de configuração do modelo). Serão necessários alguns anos de prática até se poder aplicar o modelo com eficácia.
- ▶ Não é possível estipular a composição exacta das metodologias a aplicar em cada iteração do desenvolvimento.
- ▶ **Não existem deadlines rígidas**
- ▶ **Grande probabilidade de este modelo cair simplesmente no modelo em cascata**
  - Aliás, na sua base é um modelo em cascata (iterativo) c/ um nível de abstracção diferente

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

106

## Modelos espiral win-win

- ▶ O processo de levantamento de requisitos é um processo de negociação
- ▶ Um processo de negociação pretende alcançar um equilíbrio em que ambas as partes possam ter algo a ganhar
- ▶ O cliente ganha (win) 1 sistema funcional que satisfaz a maior parte dos requisitos
- ▶ O programador ganha (win) ao trabalhar mediante orçamentos e calendários realistas.

---

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

107

## Modelos espiral win-win

- ▶ **Actividades de negociação**
  - identificação dos “stakeholders” chave
  - determinar as “win conditions” dos “stakeholders”
  - negociação das “win conditions” dos “stakeholders” e
  - conciliá-las com um pacote comum de “win-win conditions” - isto inclui a equipa do projecto.
- ▶ Em cada etapa do processo, o **resultado desta negociação será um critério para prosseguir** com a definição do sistema e do software em particular.

---

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

108

## Modelos espiral win-win

- ▶ introduz **3 milestones** denominados de *anchor points*:
  - life cycle objective (LCO)
    - p.e.: **definição dos requisitos genéricos** do sistema
  - life cycle architecture (LCA)
    - define os **objectivos a alcançar de acordo com a arquitectura** do sistema
  - initial operation capability (IOC)
    - define os **objectivos com a preparação do SW para a devida instalação/distribuição**

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

109



# Selecionar o modelo

## selecionar o modelos adequado?!

Orçamento	Calendário	Requisitos	Possibilidade de refutar os requisitos	Risco associado ao projeto	Entrega antecipada?	Modelo adequado
Baixo	Apertado	Estáveis	Req. Vagos	Alto	Sim	Ágil
Baixo	Apertado	Em evolução	Req. Vagos	Alto	Sim	
Baixo	Flexível	Estáveis	Req. Claros	Alto	Sim	Incremental
Alto/Folgado	Apertado	Estáveis	Req. Claros	Alto	Sim	
Baixo	Flexível	Em evolução	Req. Vagos	Alto	Sim	Evolutivo
Alto/Folgado	Apertado	Em evolução	Req. Vagos	Alto		RAD
Alto/Folgado	Flexível	Estáveis	Req. Claros	Baixo		Cascata
Alto/Folgado	Flexível	Em evolução	Req. Vagos	Alto		Espiral

Fonte: <http://softwaremethodologies.blogspot.pt/2009/04/how-to-select-software-development.html>

Engenharia de Software II - Engenharia Informática

\* Crístovão Sousa e Carla Pereira\*

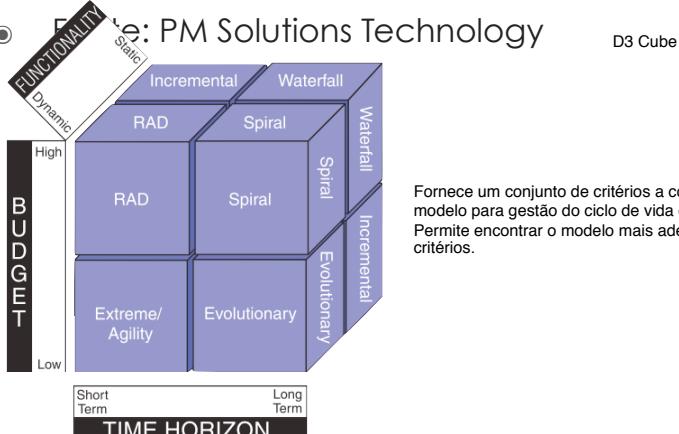
111

## selecionar o modelos adequado?! (outra abordagem)

### ► The D3 Cube

- Selecting a Software Development Life Cycle (SDLC) Methodology - A Practical Decision Framework to Maximize Business Value

- e: PM Solutions Technology



Fornecer um conjunto de critérios a considerar quando da escolha de um modelo para gestão do ciclo de vida de software numa visão tridimensional. Permite encontrar o modelo mais adequado tendo em conta determinados critérios.

Engenharia de Software II - Engenharia Informática  
\* Crístovão Sousa e Carla Pereira\*

112

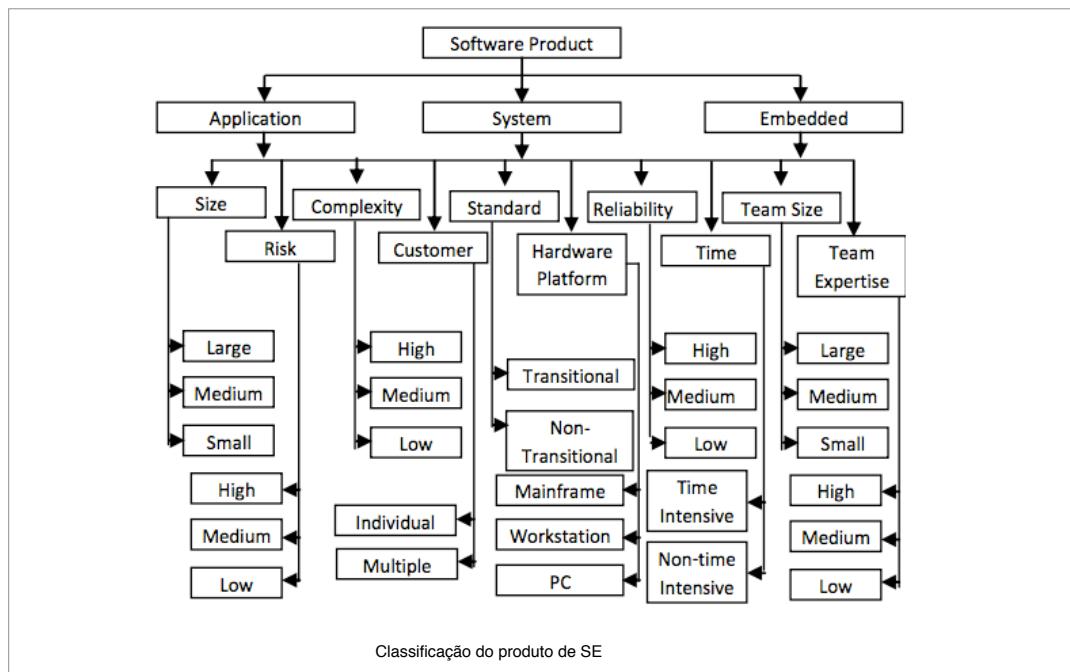
## matriz das inferências possíveis do D3Cube

Methodology & Criteria	Advantages	Disadvantages
<b>Waterfall</b>		
Budget: <i>high</i> Time: <i>long term</i> Functionality: <i>static</i>	Clearly defined stages. Assures delivery of initial requirements. Well documented process and results.	Lack of measurable progress within stages. Cannot accommodate changing requirements. Resistant to time and/or budget compression.
<b>Incremental</b>		
Budget: <i>high</i> Time: <i>short term</i> Functionality: <i>static</i> or  Budget: <i>low</i> Time: <i>long term</i> Functionality: <i>static</i>	Early and periodic results. Measurable progress. Supports parallel development efforts.	Demands increased management attention. Can increase resource requirements. No support for changing requirements.
<b>Evolutionary</b>		
Budget: <i>low</i> Time: <i>long term</i> Functionality: <i>dynamic</i>	Supports changing requirements. Minimises time to initial operating capability (IOC). Achieves economies of scale for enhancements.	Increases management complexity. IOC only partially satisfies requirements and does not have complete functionality. Risk of not knowing when to end the project.
<b>Spiral</b>		
Budget: <i>high</i> Time: <i>long term</i> Functionality: <i>dynamic</i>	Supports changing requirements. Allows for extensive use of prototypes. More accurately captures requirements. Increased management complexity.	Defers production capability to end of the SDLC. Risk of not knowing when to end the project.
<b>RAD (rapid application development)</b>		
Budget: <i>high</i> Time: <i>short term</i> Functionality: <i>dynamic</i>	Minimizes time to delivery. Accommodates changing requirements. Measurable progress.	Increases management complexity. Drives costs forward in the SDLC. Can increase resource requirements.
<b>Extreme/agile development</b>		
Budget: <i>low</i> Time: <i>short term</i> Functionality: <i>dynamic</i> or  Budget: <i>low</i> Time: <i>short term</i> Functionality: <i>static</i>	Rapid demonstrable functionality. Minimal resource requirements. Supports fixed or changing requirements.	Not conducive to handling complex dependencies. Creates quality assurance (QA) risks. Increased risk of sustainability, maintainability, and extensibility.

Table 1: Summary of D3 cube inferences and project life cycles

fonte:(Dillman, 2003:Bhunu et al., 2007)

## A Rule-based Recommendation System for Selection of Software Development Life Cycle Models



Kumar, Kuldeep, and Sandeep Kumar. 2013. "A Rule-based Recommendation System for Selection of Software Development Life Cycle Models." ACM SIGSOFT Software Engineering Notes 38 (4) (July 12): 1. doi:10.1145/2492248.2492269.

## A Rule-based Recommendation System for Selection of Software Development Life Cycle Models

**Rule 1: IF** Type==Application and Size==Small and Risk==Low and Complexity==Less and Customer==Individual and Standard==Non-Transitional and Hardware Platform==PC and Reliability==Low and Time==Non-Intensive and Team-Size== Medium and Team-Expertise == Medium **THEN** Iterative Waterfall Model.

**Rule 2: IF** Type==Application and Size==Large and Risk==low and Complexity==Less and Customer==Individual and Standard== Non-Transitional and Hardware Platform==PC and Reliability==Low and Time==Non-Intensive and Team-Size== Medium and Team-expertise == Medium **THEN** Incremental Model.

**Rule 3: IF** Type==System and Size==Large and Risk==low and Complexity==Less and Customer==Individual and Standard== Non-Transitional and Hardware Platform==PC and Reliability==High and Time==Intensive and Team-Size == Large and Team-Expertise == High **THEN** RAD.

Kumar, Kuldeep, and Sandeep Kumar. 2013. "A Rule-based Recommendation System for Selection of Software Development Life Cycle Models." ACM SIGSOFT Software Engineering Notes 38 (4) (July 12): 1. doi:10.1145/2492248.2492269.

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

115

## A Rule-based Recommendation System for Selection of Software Development Life Cycle Models

**Rule 4: IF** Type==Embedded and Size==Large and Risk==High and Complexity==High and Customer==Individual and Standard== Non-Transitional and Hardware Platform==Mainframe and Reliability==High and Time==Non-Intensive and Team-Size == Small and Team-Expertise == Low **THEN** Spiral Model.

**Rule 5: IF** Type==Application and Size==Medium and Risk==Medium and Complexity==Less and Customer==Multiple and Standard== Non-Transitional and Hardware Platform==PC and Reliability==Medium and Time==Intensive and Team-Size == Small and Team-Expertise == High **THEN** Incremental Model.

**Rule 6: IF** Type==System and Size==Small and Risk==Low and Complexity==High and Customer==Individual and Standard== Non-Transitional and Hardware Platform==PC and Reliability==Low and Time== Intensive and Team-Size == Medium and Team-expertise == Medium **THEN** XP model.

Kumar, Kuldeep, and Sandeep Kumar. 2013. "A Rule-based Recommendation System for Selection of Software Development Life Cycle Models." ACM SIGSOFT Software Engineering Notes 38 (4) (July 12): 1. doi:10.1145/2492248.2492269.

Engenharia de Software II - Engenharia Informática  
\* Crisóstomo Sousa e Carla Pereira\*

116

