Aula 1

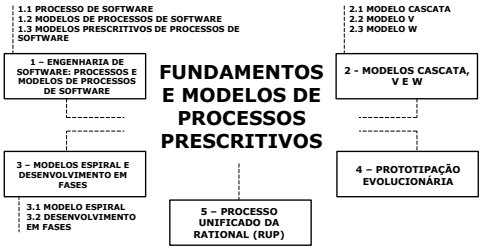
Engenharia de Software

Prof. Alex Mateus Porn

1

Conversa Inicial

2



3

Engenharia de *Software*: processos e modelos de processos de *software*

4

Engenharia de *software*

- **"... existem muitos tipos diferentes de sistemas de *software*, variando de sistemas embarcados simples até sistemas de informação complexos" (Sommerville, 2018, p. 3)**
- **Serviços públicos nacionais**
- **Indústrias completamente informatizadas**
- **Setores de entretenimento**
- **Telefonia**
- **Entre outros**

5

- **Os sistemas de *software* podem rapidamente se tornar bastante complexos, difíceis de entender e caros de modificar**
- **Não há notações, métodos ou técnicas universais para o desenvolvimento de uma variedade completamente heterogênea de sistemas de *software***

6

- **"A engenharia de *software* se destina a apoiar o desenvolvimento de *software* profissional em vez da programação individual. Ela inclui técnicas que apoiam a especificação, o projeto e a evolução do *software*" (Sommerville, 2018, p. 5)**

7

Processos de *software*

- **"[...] referem-se a um conjunto de atividades relacionadas que levam à produção de um sistema de *software*. [...] não há um método universal de engenharia de *software* que seja aplicável a todos os tipos diferentes de sistemas de *software* existentes. Desse modo, não existem processos de *software* universalmente aceitos" (Sommerville, 2018, p. 29)**

8

- **Sommerville (2018) destaca que os processos devem incluir quatro atividades fundamentais:**

1. **Especificação: a funcionalidade do *software* e as restrições sobre sua operação devem ser definidas**
2. **Desenvolvimento: o *software* deve ser produzido para atender a especificação**
3. **Validação: o *software* deve ser validado para garantir que atenda o que o cliente deseja**
4. **Evolução: o *software* deve evoluir para atender as mudanças nas necessidades dos clientes**

9

Modelos de processos de *software*

- **Processos prescritivos: modelos de processos tradicionais, que surgiram com o propósito de organizar e direcionar as atividades inerentes ao desenvolvimento de *software***
- **Processos ágeis: possuem iterações curtas, nas quais o resultado é medido através do produto pronto, ao contrário dos modelos prescritivos, em que o desenvolvimento é divido em etapas bem definidas**

10

- **"Alguns modelos de processos vêm caindo em desuso, enquanto outros evoluem a partir desses. O engenheiro de *software* deve escolher o modelo que for mais adequado para sua equipe e ao projeto que ele vai desenvolver" (Wazlawick, 2013, p. 22)**

11

Modelos cascata, V e W

12

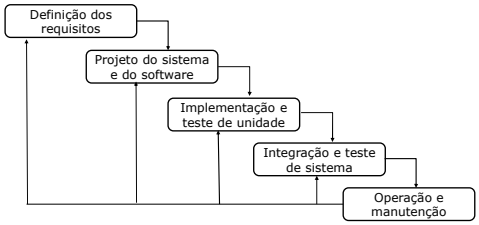
Modelo cascata

- **Desenvolvido na década de 1970**
- **Derivado dos modelos utilizados na engenharia de grandes sistemas militares**
- **Apresenta o processo de desenvolvimento de *software* como uma série de fases**

UniqueVector/SHUTTERSTOCK

13

Modelo cascata



Fonte: Adaptação de sommerville (2018. p. 33)

14

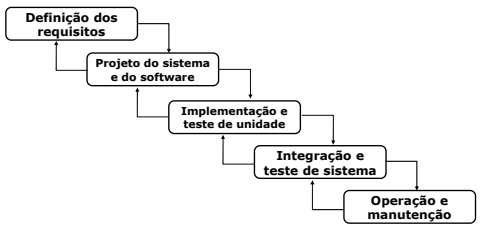
- **Vantagens**
  - **Fases bem definidas ajudam a detectar erros mais cedo**
  - **Procura promover a estabilidade dos requisitos**
  - **Funciona bem quando os requisitos são conhecidos e estáveis**
  - **É adequado para equipes tecnicamente fracas ou inexperientes**

15

- **Desvantagens**
  - **Não produz resultados tangíveis até a fase de codificação**
  - **É difícil estabelecer requisitos completos antes de começar a codificar**
  - **Desenvolvedores sempre reclamam que os usuários não sabem expressar aquilo de que precisam**
  - **Não há flexibilidade com requisitos**

16

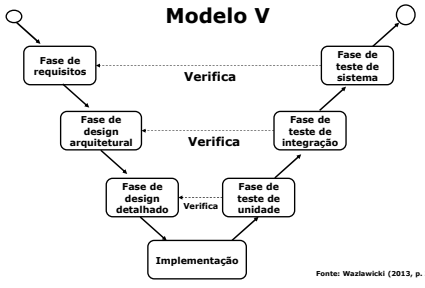
Modelo cascata dupla



Fonte: Adaptação de Royte (1970)

17

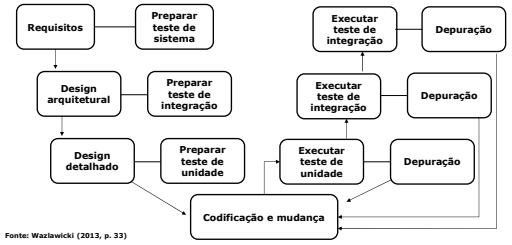
Modelo V



Fonte: Wazlawicki (2013, p. 32)

18

Modelo W



Fonte: Wazlawicki (2013, p. 33)

19

- **Fase de requisitos: apenas requisitos que possam ser testados são aceitáveis ao final dessa fase**
- **Fase de *design* arquitetural: arquiteturas simples devem ser fáceis de testar. Caso contrário, talvez a arquitetura seja demasiadamente complexa e necessite ser refatorada**
- ***Design* detalhado: a mesma questão se coloca em relação às unidades. Unidades coesas são mais fáceis de testar**

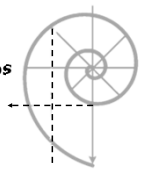
20

Modelos espiral e desenvolvimento em fases

21

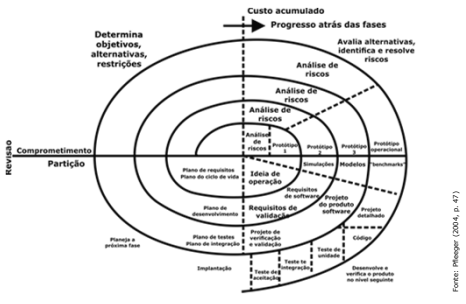
Modelo espiral

- **Criado em 1986**
- **Apresenta a ideia em ciclos iterativos**
- **O projeto é dividido em subprojetos**
- **"Cada subprojeto aborda um ou mais elementos de alto risco, até que todos os riscos identificados tenham sido tratados" (Wazlawick, 2013, p. 35)**



Himash47/SHUTTERSTOCK

22



Fonte: Pfleeger (2004, p. 47)

23

Iterações do modelo espiral

- **Primeira iteração: concepção das operações**
- **Segunda iteração: requisitos de *software***
- **Terceira iteração: desenvolvimento do sistema**
- **Quarta iteração: testes**

24

Vantagens e desvantagens do modelo espiral

- As primeiras iterações são as mais baratas do ponto de vista de investimento de tempo e de recursos
- Também são as que resolvem os maiores problemas do projeto
- Esse modelo não provê a equipe com indicações claras sobre a quantidade de trabalho esperada a cada ciclo
- Pode tornar o tempo de desenvolvimento nas primeiras fases bastante imprevisível (Wazlawick, 2013)

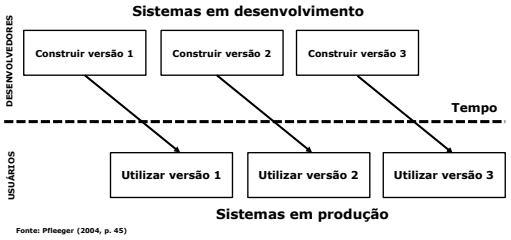
25

Modelo de desenvolvimento em fases

- "... o sistema é projetado de modo que possa ser entregue em partes, possibilitando aos usuários dispor de alguns recursos enquanto o restante do sistema está sendo desenvolvido" (Pfleeger, 2004, p. 45)
- Sistema em produção: é o sistema que está sendo atualmente utilizado pelo cliente
- Sistema em desenvolvimento: é a versão seguinte que está sendo desenvolvida para substituir o sistema em produção

26

Modelo desenvolvimento em fases



Fonte: Pfleeger (2004, p. 45)

27

Abordagens do modelo de desenvolvimento em fases

- Desenvolvimento incremental: o sistema é dividido por funcionalidades e cada versão entregue corresponde a uma delas ou a um conjunto dessas funcionalidades
- Desenvolvimento iterativo: a primeira versão entregue corresponde ao sistema completo, porém, com funcionalidades limitadas. Cada versão posterior corresponde a mudanças realizadas nas funcionalidades limitadas

28

Exemplificando

- O projeto a ser desenvolvido refere-se a um *software* de processamento de textos. Esse *software* deve oferecer os seguintes recursos:

1. Criar textos
2. Organizar textos (recortar e colar)
3. Formatar textos

29

- Desenvolvimento no modelo incremental
  - Versão 1: função de criação de textos
  - Versão 2: criação e organização de textos
  - Versão 3: criação, organização e formatação de textos

WatcharaRitian/SHUTTERSTOCK

30

- Desenvolvimento no modelo iterativo
  - Versão 1: formas primitivas das três funções
    - Podemos criar textos, mas não podemos recortar ou colar
    - Podemos formatar o texto, mas não podemos mudar a cor e tamanho da fonte
  - Versão 2: as funções para recortar e colar são implementadas
  - Versão 3: as funções para mudar a cor e o tamanho da fonte são implementadas

31

Prototipação evolucionária

32

- "... a prototipação evolucionária não precisa ser somente uma atividade para incrementar outros modelos de processos, de modo que ela própria pode ser a base de um modelo de processo efetivo" (Pfleeger, 2004, p. 42)

Anton Shaparenko/SHUTTERSTOCK

33

- Permite que todo o sistema, ou parte dele, seja construído rapidamente
- Tem o mesmo objetivo de um protótipo de engenharia
- Auxilia na definição dos requisitos
- Facilita para garantir que desenvolvedor, usuário e cliente cheguem a um consenso sobre o que é necessário e o que é proposto

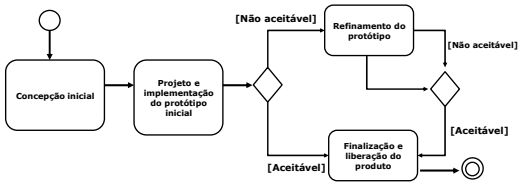
34

Métodos de prototipação evolucionária

- ***Throw away*** ou descartável: refere-se a protótipos usados unicamente para estudar aspectos do sistema, entender melhor seus requisitos e reduzir riscos
- ***Cornerstone*** ou fundamental: refere-se a protótipos para serem parte do sistema final, de modo que o protótipo vai evoluindo até se tornar um sistema que possa ser entregue

35

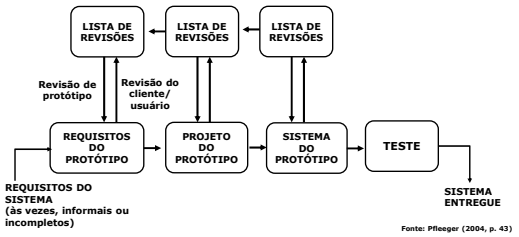
Modelo de prototipação evolucionária



Fonte: Wazlawick (2013, p. 37)

36

Modelo de prototipação evolucionária



Fonte: Pfleeger (2004, p. 43)

37

Exemplificando

- O desenvolvimento do sistema pode começar com
  - Requisitos fornecidos pelos clientes e usuários
    - Examinam-se: telas, tabelas, relatórios e outras saídas do sistema
  - Assim que os clientes e usuários decidem o que querem, os requisitos são revisados
  - Finalmente o sistema é codificado, com uma possível iteração entre requisitos e projeto

38

Processo unificado da rational (RUP)

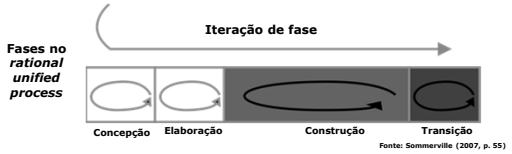
39

RUP

- Concepção: elabora-se um plano de negócios para o sistema, com o objetivo de identificar as entidades externas e os requisitos, a fim de avaliar a contribuição do sistema para o negócio
- Elaboração: são desenvolvidos os requisitos e a arquitetura do sistema
- Construção: implementação e testes
- Transição: implantação em ambiente real

40

- Reúne os elementos de todos os modelos de processos estudados
- Apoia a prototipação e a entrega incremental do *software* (Sommerville, 2018)



Fonte: Sommerville (2007, p. 55)

41

- Disciplinas de projeto
  - Modelagem de negócios: compreender a empresa e seus processos
  - Requisitos: elicitação dos requisitos e *design* da interface
  - Análise e design: detalhamento dos requisitos para a modelagem
  - Implementação: desenvolvimento e testes de unidade
  - Teste: testes de integração e interação
  - Implantação: versões para serem entregues

42

- **Disciplinas de apoio**
  - **Ambiente: ambiente de desenvolvimento. Configuração do processo para desenvolver o projeto. Ferramentas de apoio**
  - **Gerenciamento de mudança e configuração: mantém a integridade dos artefatos produzidos**
  - **Gerência de projeto: planeja o projeto inteiro e cada iteração individualmente. Gerencia os riscos e monitora o progresso**

43

Referências

44

- **PFLEEGER, S. L. Engenharia de *software*: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.**
- **ROYCE, W. W. Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, 1970.**
- **SOMMERVILLE, I. Engenharia de *software*. 10. ed. São Paulo: Pearson Education do Brasil, 2018.**
- **SOMMERVILLE, I. Engenharia de *software*. 8. ed. São Paulo: Pearson Prentice Hall, 2007.**
- **WAZLAWICK, R. S. Engenharia de *software*: conceitos e práticas. São Paulo: Elsevier, 2013.**

45