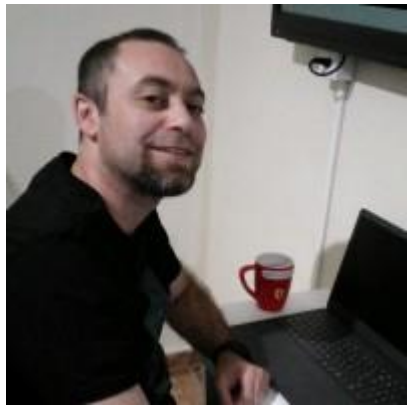
Engenharia de Software

Anhanguera

Apresentação!



Ivan Grand Champs

E-mail:
ivan.pereira@kroton.com.br

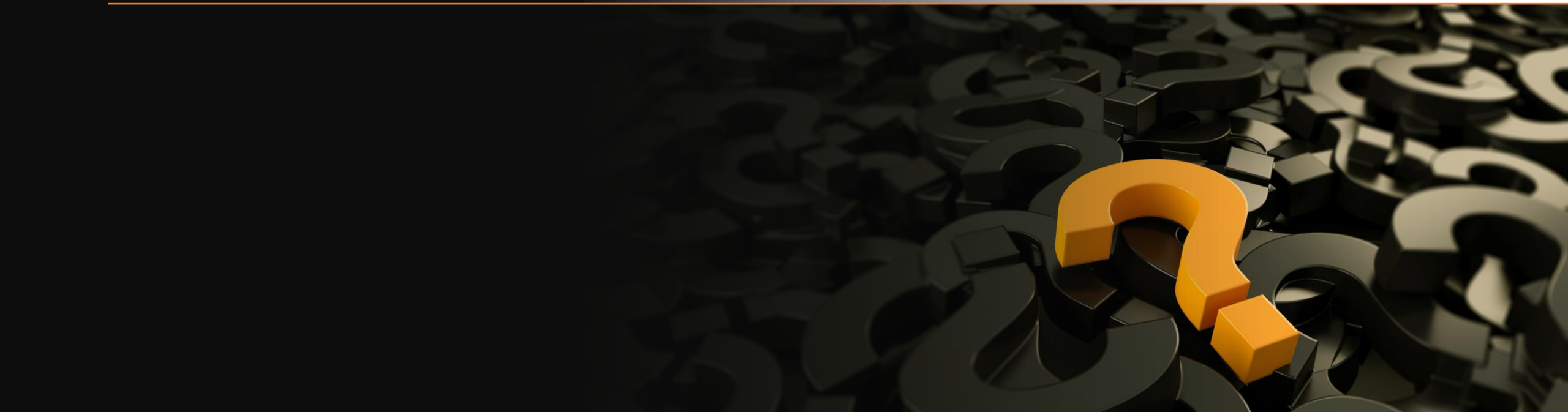
Experiência

- [Professor em Ciências da Computação e Engenharias/ Líder Técnico de Segurança da Informação
- Bacharel em Computação Científica / Tecnólogo em Redes / Especialização em Gestão de Processos/MBA Liderança Digital


AVALIE
SUA PROFISSÃO
QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:
NOTAS
9 OU 10
SIGNIFICA QUE VOCÊ INDICA
NOTAS
7 OU 8
SIGNIFICA QUE VOCÊ NÃO INDICA
Anhanguera

Anhanguera

O que é Engenharia de Software??



Ementa

Unidade 1, são abordados os fundamentos da Engenharia de Software, incluindo a visão da metodologia tradicional e das metodologias ágeis de desenvolvimento.

Unidade 2, o foco se volta a aspectos de qualidade envolvidos em uma iniciativa de desenvolvimento de um software, com a clara separação entre qualidade do produto e qualidade do processo.

Unidade 3, são investigados os testesde software, com ênfase para as definições relacionadas ao tema, para os tipos de testes e para os testes automatizados.

Unidade 4, tratamento de auditoria de sistemas e manutenção e evolução de software.

O QUE É A ENGENHARIA DE SOFTWARE?

A Engenharia de Software abrange um processo, um conjunto de métodos (práticas) e um leque de ferramentas que possibilitam aos profissionais desenvolverem software de altíssima qualidade (PRESSMAN; MAXIM, 2016).

DEFINIÇÃO DE ENGENHARIA DE SOFTWARE

A literatura nos oferece diversas definições para a Engenharia de Software, desdeas sucintas até as mais elaboradas. Porém, a grande maioria utiliza-se de termoscomo procedimentos, métodos e ferramentas para indicar uma natural convergência: a qualidade do produto a ser desenvolvido. Se o resultado da aplicação de procedimentos, métodos e ferramentas não for uma entrega de qualidade, de pouco terá valido todo o esforço em adotá-los. Como primeiro passo,convém conhecermos a definição para Engenharia de Software dada por um importante autor dessa área.

Metodologia tradicional de desenvolvimento, criada nos primeiros anos de existência da Engenharia de Software e que ainda baseia muitos procedimentos de software. Como não poderia deixar de ser, a metodologia tradicional será abordada nesta primeira seção, juntamente com os princípios, desde sempre, norteadores daEngenharia de Software.

A produção de programas de computador possui situações bastanteparticulares, que requerem soluções especializadas e, portanto, diferentesdaquelas adotadas em um processo fabril.
Avançando para o próximo termo, temos uma definição satisfatória para software como:

1. Instruções que, quando executadas, produzem a função desejada.

2. Estruturas de dados que possibilitam os programas manipularem ainformação;

3. E documentação relativa ao sistema.

PRINCÍPIOS DA ENGENHARIA DE SOFTWARE

Formada a base conceitual da disciplina, avançamos agora para elementos que costumamos chamar de princípios da Engenharia de Software justamente pordarem uma feição própria a ela e por suportarem suas práticas. São esses princípios que ajudam a estruturar os processos e a padronizar as soluções propostas por esse ramo da Computação. Eles foram propostos pelo SWEBOK(Software Engineering Body of Knowledge ou Conjunto de Conhecimentos deEngenharia de Software), que se trata de uma importante publicação do IEEE na área.

O primeiro princípio proposto pelo IEEE (2004) foi o da organização hierárquica, o qual estabelece que os componentes de uma proposta de solução ou aorganização de elementos de um problema devem ser apresentados em formato hierárquico, em uma estrutura do tipo árvore, com quantidade crescente dedetalhamento nível após nível.

Seguir uma Metodologia!

O SOFTWARE

Software consiste em: (1) instruções (programas de computador) que, quando executados, fornecem características, funções e desempenho desejados; (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; (3) informação descritiva, tanto naforma impressa quanto na virtual, descrevendo a operação e o uso do programa (PRESSMAN, MAXIM, 2016).

MITOS SOBRE DESENVOLVIMENTO DE SOFTWARE

Conforme nos ensinam Pressman e Maxim (2016), daquela época ficaram algunsmitos sobre o desenvolvimento de um software. Embora os profissionais atuaisestejam mais capacitados a reconhecê-los e a evitarem seus efeitos, ainda énecessário dar atenção a eles. Trataremos de dois deles aqui.

MITOS DE GERENCIAMENTO

Sob pressão, gestores podem apoiar-se em crenças relacionadas ao gerenciamento de seus projetos de software. Um mito bastante comum é que umlivro repleto de práticas e padrões vai ser instrumento suficiente para o sucesso da empreitada, bastando segui-lo à risca. O questionamento necessário à validade detal livro é sua completude e adequação às modernas práticas da Engenharia de Software. Outro mito comum é que, mediante atraso em uma entrega, a inclusão de mais desenvolvedores no projeto ajudaria a evitar atraso maior. Por fim, é perigosa a crença de que, ao terceirizar o desenvolvimento de um produto, aempresa que o terceirizou pode deixar a cargo do terceiro todo odesenvolvimento, sem necessidade de gerenciá-lo (PRESSMAM, MAXIM, 2016).

MITOS DOS CLIENTES

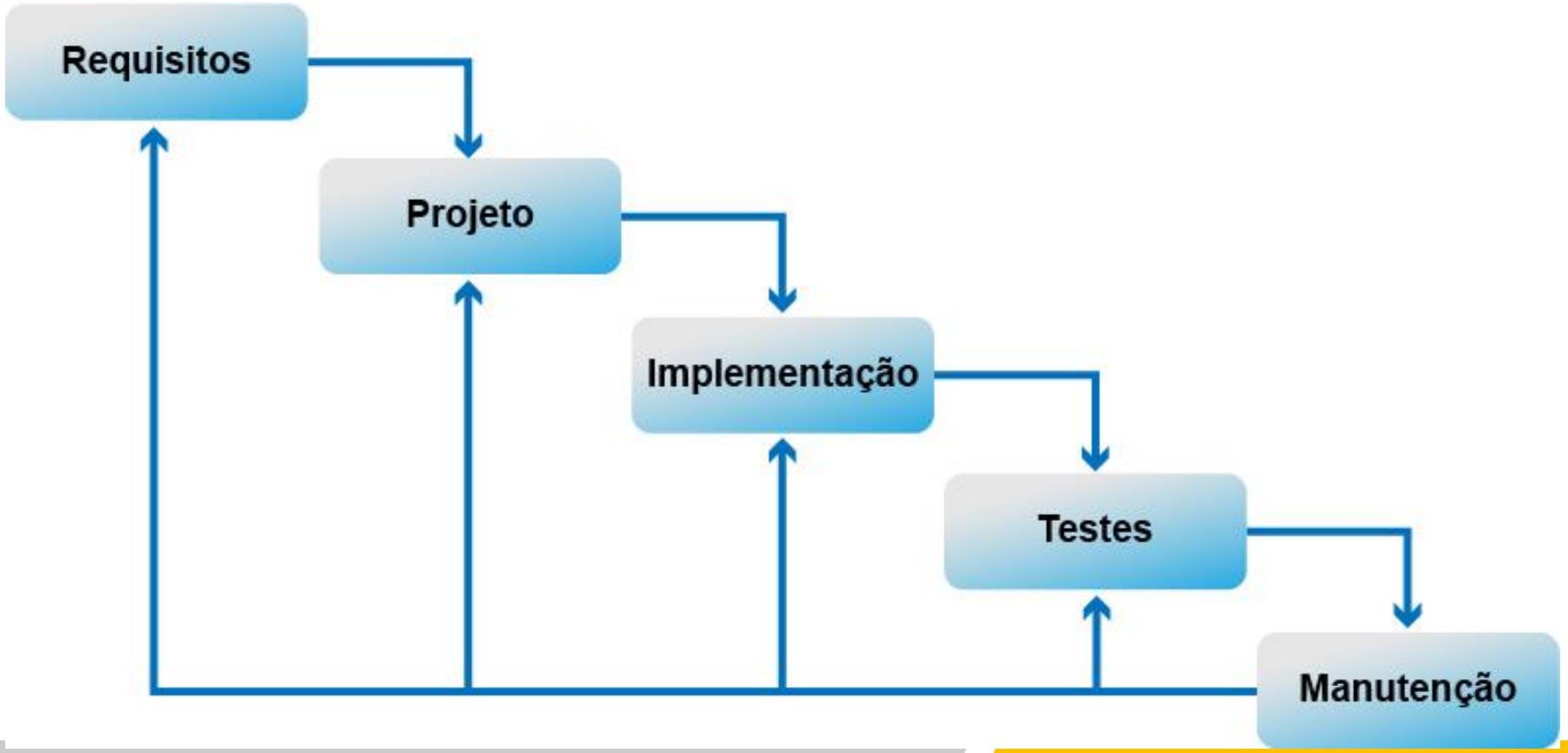
Aqui, os autores destacam crenças relacionadas ao papel e ao comportamento dos clientes em um projeto. Segundo um dos mitos, basta ao cliente fornecer uma definição geral dos objetivos do software para que seja possível iniciar sua elaboração. Na verdade, a realidade é que a definição mais clara e mais completa possível dos requisitos é o caminho mais seguro para o bom início de um projeto(PRESSMAM, MAXIM, 2016).

(Balanço de Pneu)

MODELO CASCATA

Para o bem do futuro da Computação como uma atividade humana viável, umametodologia estável e bem estruturada de desenvolvimento de software deveria ser criada. A resposta da comunidade ligada ao software, então, veio através do modelo em cascata, também conhecido como modelo tradicional, o qual ainda é utilizado para desenvolvimento de produtos de software. Ele descreve, por meio de etapas bem definidas, o ciclo que o software cumprirá durante o período compreendido entre sua concepção e sua descontinuidade.

O ciclo de vida natural de um software, de acordo com Resende (2005), abrange as seguintes fases: concepção, construção, implantação, implementações, maturidade, declínio, manutenção e descontinuidade. Essas fases são mais comumente descritas como fase de requisitos, projeto, implementação, teste emanutenção. A Figura 1.1 mostra o esquema geral das fases do modelo cascata.


Requisitos
Projeto
Implementação
Testes
Manutenção

Observe que uma fase do processo depende do artefato gerado pela fase anterior. Um artefato nesse contexto não é exatamente um produto de software acabado.As setas de retroalimentação, dispostas no sentido contrário à cascata, indicam a possibilidade de retornos às fases anteriores, considerando a ocorrência de falhas. A volta a uma fase anterior serve, em tese, para sanar problemas que, se levados adiante, acarretariam mais prejuízo ao desenvolvimento.

Requisitos: a fase de requisitos de software preocupa-se com a descoberta, a análise, a especificação e a validação das propriedades que devem ser apresentadas para resolver tarefas relacionadas ao software a ser desenvolvido. Fazem parte dos requisitos de um software suas funções, suas características, suas restrições e todas as demais condições para que ele existae cumpra seu objetivo. O projeto de um software fica comprometido quando olevantamento dos requisitos é executado de forma não apropriada. Eles expressam as necessidades e as restrições colocadas num produto de software, as quais contribuem para a solução de algum problema do mundo real (SCHACH, 2009).

Projeto: uma vez tratados os requisitos, o modelo nos leva ao desenho doproduto. Se os requisitos nos mostram o que o sistema deverá fazer, o projetodeverá refletir como ele o fará. Por meio do projeto, os requisitos são refinados de modo a se tornarem aptos a serem transformados em programa. Otrabalho principal de um projetista é o de decompor o produto em módulos, que podem ser conceituados como blocos de código, que se comunicam com outros blocos por meio de interfaces (SCHACH, 2009).

Implementação: nessa fase, o projeto é transformado em linguagem deprogramação para que, de fato, o produto passe a ser executável. Como estratégia de implementação, a equipe poderá dividir o trabalho de forma quecada programador (ou um pequeno grupo deles) fique responsável por um módulo do sistema. Idealmente, a documentação gerada pela fase de projeto deve servir como principal embasamento para a codificação, o que não afasta anecessidade de novas consultas ao cliente e à equipe de projetistas (SCHACH,2009).

Testes: testar significa executar um programa com o objetivo de revelar apresença de defeitos. Caso esse objetivo não possa ser cumprido, considera-seo aumento da confiança sobre o programa. As técnicas funcional e estrutural são duas das mais utilizadas a fim de se testar um software. A primeira baseia-se na especificação do software para derivar os requisitos de teste e a segunda no conhecimento da estrutura interna (implementação) do programa (SCHACH,2009).

Manutenção: os esforços de desenvolvimento de um software resultam naentrega de um produto que satisfaça os requisitos do usuário. Espera-se,contudo, que o software sofra alterações e evolua. Uma vez em operação,defeitos são descobertos, ambientes operacionais mudam e novos requisitos dos usuários vêm à tona. A manutenção de software é definida como modificações em um produto de software após a entrega ao cliente a fim decorrigir falhas, melhorar o desempenho ou adaptar o produto ao um ambiente diferente daquele em que o sistema foi construído (SCHACH, 2009).

A engenharia de requisitos começa com a concepção (uma tarefa que define a abrangência e a natureza do problema a ser resolvido). Ela prossegue para o levantamento (uma tarefa de investigação que ajuda os envolvidos a definir o que é necessário) e, então, para a elaboração (na qual os requisitos básicos são refinados e modificados) (PRESSMAN; MAXIM, 2016).

Considerando o conteúdo de requisitos de software no contexto do modelo em cascata, analise as afirmações a seguir.

I. Requisitos são elementos desejáveis, porém opcionais em um processo de software conduzido com base no modelo em cascata.

II. As características gerais, as funções a serem executadas e as restrições de um software são partes dos seus requisitos.

III. A prática tem ensinado que o levantamento dos requisitos pode ser postergado para a fase de implementação do produto.

É verdadeiro o que se afirma em:

a. II e III apenas.
b. II apenas.
c. I e II apenas.
d. III apenas.
e. I, II e III.

Quando executado mediante aplicação de uma metodologia definida e conhecida, o processo de desenvolvimento de um software deverá alcançar um resultado muito mais satisfatório do que quando executado em um ambiente de ausência de formalidade.
Considerando esse fato, assinale a alternativa que contém a real vantagem em seadotar uma metodologia definida no processo de desenvolvimento de umsoftware.

a. Facilita a descoberta de maus profissionais da organização, já que o processo padronizado ajuda adestacar pessoas sem aptidão.

b. Promove a produção meramente baseada em padrões, o que desestimula iniciativas potencialmentenocivas ao projeto.

c. Possibilita que os membros do projeto foquem mais nas tarefas de implementação do produto do queem atividades de menor importância.

d. Permite sejam identificados os membros da equipe que, de fato, conhecem os processos da Engenhariade Software.

e. Promove a produção de artefatos mais uniformizados, já que a previsibilidade do processo ajuda aequipe a trabalhar de forma mais padronizada.