Estratégia CONCURSOS

# Aul

Informática p/ ISS Porto Alegre (Auditor Fiscal) Com Videoaulas - Pós-Edital

Professor: André Castro, Diego Carvalho, Equipe Informática e TI, Fábio Alves, Renato da Costa, Thiago Rodrigues Cavalcanti

# Sumário

# • ATENÇÃO •

**Existem muitos exercícios sobre esse tema em sites de questões, no entanto a imensa maioria foi aplicada em provas para cargos específicos de Tecnologia da Informação (TI), os quais podem demandar um conhecimento muito mais aprofundado da matéria.**

**Dessa forma, recomendo que vocês tenham muita atenção na seleção das questões realizadas para que não extrapolem o nível cobrado em provas da área fiscal.**

**Qualquer dúvida, estou à disposição para maiores esclarecimentos!**

# CRONOGRAMA

Confira também o cronograma atualizado em:

https://www.estrategiaconcursos.com.br/curso/informatica-p-iss-porto-alegre-auditor-fiscal-com-videoaulas-pos-edital/

| AULA | CONTEÚDO | DATA |
|---|---|---|
| **Aula 00** | 6. Fundamentos de engenharia de software. 6.1. Modelos de processo de software cascata, prototipagem. (Prof. Diego Carvalho e Fernando Pedrosa) | 07/10 |
| **Aula 01** | Processo unificado e RUP. (Prof. Diego Carvalho e Fernando Pedrosa) | 10/10 |
| **Aula 02** | 6.2. Framework ágeis SCRUM e XP. (Prof. Diego Carvalho e Fernando Pedrosa) | 12/10 |
| **Aula 03** | 6.3. Técnicas de especificação de requisitos em ciclos de vida tradicional e ágil de desenvolvimento de sistemas. (Prof. Diego Carvalho e Fernando Pedrosa) | 15/10 |
| **Aula 04** | 6.5. Testes de software: fundamentos de testes, tipos de testes. 6.4. Qualidade de software. (Prof. Diego Carvalho e Fernando Pedrosa) | 21/10 |
| **Aula 05** | 6.6. Estimativa de software (Análise de Pontos de Função), utilizando técnicas estimada e detalhada. 6.7. Utilização do CPM, Nesma e SISP. (Prof. Diego Carvalho e Fernando Pedrosa) | 23/10 |
| **Aula 06** | 3. Fundamentos de Bancos de Dados. 3.1. Sistemas de gerenciamento de banco de dados. 3.2. Conceitos básicos. 3.3. Independência de dados, modelos. (Prof. Diego Carvalho e Thiago Cavalcanti) | 19/10 |
| **Aula 07** | 3.4. Abordagem relacional. 3.5. Modelo de dados e restrições de integridade. 3.6. Normalização e dependências funcionais. 3.7. Modelagem entidade-relacionamento. (Prof. Diego Carvalho e Thiago Cavalcanti) | 23/10 |
| **Aula 07.1** | 3.6. Normalização e dependências funcionais. | 31/10 |
| **Aula 08** | 3.10. Ferramentas ETL e OLAP. 3.11. Técnica de modelagem dimensional. 3.12. Mineração de dados e aprendizado de máquina: conceitos. 3.9. Arquitetura e análise de requisitos para sistemas analíticos. - Parte 1. (Prof. Diego Carvalho e Thiago Cavalcanti) | 24/10 |
| **Aula 09** | 3.10. Ferramentas ETL e OLAP. 3.11. Técnica de modelagem dimensional. 3.12. Mineração de dados e aprendizado de máquina: conceitos. 3.9. Arquitetura e análise de requisitos para sistemas analíticos. - Parte 2. (Prof. Diego Carvalho e Thiago Cavalcanti) | 25/10 |
| **Aula 10** | 3.8. Linguagem SQL padrão ANSI. (Prof. Diego Carvalho e Thiago Cavalcanti) | 31/10 |

| | | |
|---|---|---|
| **Aula 11** | 3.13. Utilização do Qlik View. 3.14. Utilização do MS-Access. (Prof. Diego Carvalho e Renato da Costa) | 04/11 |
| **Aula 11.1** | 3.13 Utilização do Qlik View | 05/11 |
| **Aula 12** | 4. Modelagem de processos. 4.1. BPM (Business Process Management). (Prof. Diego Carvalho e Thiago Cavalcanti) | 05/11 |
| **Aula 13** | 4.2. BPMN (Business Process Modeling Notation). 4.3. Software de modelagem Bizagi. (Prof. Diego Carvalho e Thiago Cavalcanti) | 07/11 |
| **Aula 13.1** | 4.3 Software de Modelagem Bizagi | 14/11 |
| **Aula 14** | 5. Gerenciamento de serviços de TI. 5.1. Fundamentos da ITIL (Versão 3). (Prof. Diego Carvalho e Fernando Pedrosa) | 12/11 |
| **Aula 15** | 2. Gerenciamento de projetos. 2.1. Conceitos básicos. 2.2. Conhecimento em gerenciamento de projetos - Guia do PMBOK (6a. Edição). 2.3. o papel do gerente de projetos. 2.4. Gerenciamento da integração do projeto. 2.5. Gerenciamento do escopo do projeto. 2.6. Gerenciamento do cronograma do projeto. 2.7. Gerenciamento dos custos do projeto. 2.8. Gerenciamento da qualidade do projeto. 2.9. Gerenciamento dos recursos do projeto. 2.10. Gerenciamento dos riscos do projeto. 2.11. Gerenciamento das aquisições do projeto. 2.12. Gerenciamento das partes interessadas do projeto - Parte 1. (Prof. Fábio Alves) | 14/11 |
| **Aula 16** | 2. Gerenciamento de projetos. 2.1. Conceitos básicos. 2.2. Conhecimento em gerenciamento de projetos - Guia do PMBOK (6a. Edição). 2.3. o papel do gerente de projetos. 2.4. Gerenciamento da integração do projeto. 2.5. Gerenciamento do escopo do projeto. 2.6. Gerenciamento do cronograma do projeto. 2.7. Gerenciamento dos custos do projeto. 2.8. Gerenciamento da qualidade do projeto. 2.9. Gerenciamento dos recursos do projeto. 2.10. Gerenciamento dos riscos do projeto. 2.11. Gerenciamento das aquisições do projeto. 2.12. Gerenciamento das partes interessadas do projeto - Parte 2. (Prof. Fábio Alves) | 16/11 |
| **Aula 17** | 5.2. Fundamentos de COBIT (versão 5). (Prof. Fábio Alves) | 18/11 |
| **Aula 18** | 6.8. Modelo de referência CMMI. (Prof. Fábio Alves) | 20/11 |
| **Aula 19** | 1 Segurança da informação. 1.1. Confiabilidade, integridade e disponibilidade. 1.2. Mecanismos de segurança: criptografia, assinatura digital, garantia de integridade, controle de acesso e certificação digital. (Prof. Diego Carvalho e Renato da Costa) | 22/11 |
| **Aula 20** | 1.3. Segurança na Internet, golpes na Internet, códigos maliciosos, spam, mecanismos de segurança, privacidade, uso seguro da Internet, segurança de computadores. (Prof. Diego Carvalho e Renato da Costa) | 24/11 |

| | | |
|---|---|---|
| **Aula 21** | 1.4. Gerência de riscos: ameaça, vulnerabilidade e impacto. 1.5. Políticas de segurança: ISO/IEC 27000:2018, ABNT NBR ISO/IEC 27001:2013, ABNT NBR ISO/IEC 27002:2013. (Prof. André Castro) | 26/11 |
| **Aula 22** | ABNT NBR ISO/IEC 27003:2011 (versão corrigida 2015), ABNT NBR ISO/IEC 27004:2017. (Prof. André Castro) | 28/11 |
| **Aula 23** | ABNT NBR ISO/IEC 27017:2016. 1.6. Classificação e controle dos ativos de informação. (Prof. André Castro) | 30/11 |

# 1 – ENGENHARIA DE SOFTWARE

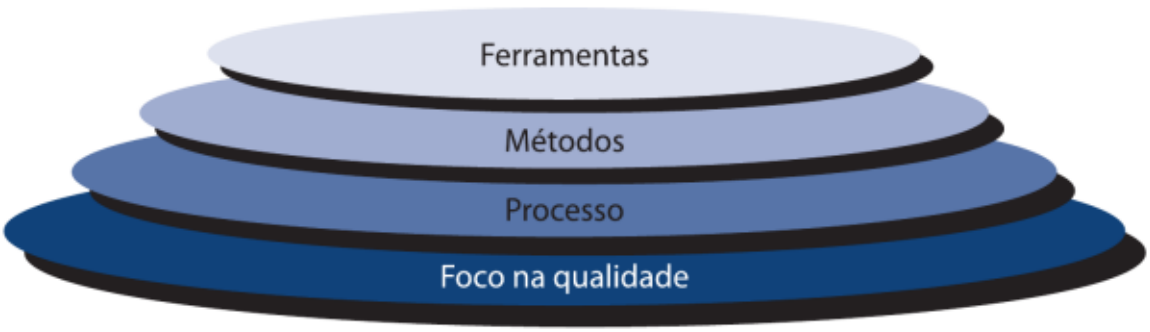
Vamos começar falando um pouquinho sobre Engenharia de Software! *Em primeiro lugar, o que é um software?* Bem, em uma visão restritiva, muitas pessoas costumam associar o termo software aos programas de computador. **No entanto, software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente.**

*E a Engenharia de Software?* **A IEEE define engenharia de software como a aplicação de uma abordagem sistemática, disciplinada e quantificável de desenvolvimento, operação e manutenção de software.** Já Friedrich Bauer conceitua como a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais.

Em suma, trata-se de uma disciplina de engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após sua entrada em produção **A meta principal da Engenharia de Software é desenvolver sistemas de software com boa relação custo-benefício.** Conceito bem simples, vamos prosseguir...

**De acordo com Roger Pressman:** "*A Engenharia de Software ocorre como consequência de um processo chamado Engenharia de Sistemas. Em vez de se concentrar somente no software, a engenharia de sistemas focaliza diversos elementos, analisando, projetando, e os organizando em um sistema que pode ser um produto, um serviço ou uma tecnologia para transformação da informação ou controle*".

Além disso, nosso renomadíssimo autor afirma que a engenharia de sistemas está preocupada com todos os aspectos do desenvolvimento de sistemas computacionais, **incluindo engenharia de hardware, engenharia de software e engenharia de processos.** Percebam, então, que a Engenharia de Sistemas está em um contexto maior – junto com várias outras engenharias. *Entenderam direitinho?*




A Engenharia de Software tem por objetivo a aplicação de teorias, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para um desenvolvimento sistemático de

software. Logo, associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas**, sendo que a pedra fundamental que sustenta a engenharia de software é o foco na qualidade conforme apresenta a imagem anterior.**

Tudo isso envolve planejamento de custos e prazos, montagem da equipe e garantia de qualidade do produto e do processo. Finalmente, a engenharia de software visa a produção da documentação formal do produto, do processo, dos critérios de qualidade e dos manuais de usuários finais. **Todos esses aspectos devem ser levados em consideração**.

**Aliás, nosso outro renomadíssimo autor – Ian Sommerville – afirma que:** *"A engenharia de software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software"*.

A Engenharia de Software surgiu em meados da década de sessenta como uma **tentativa de contornar a crise do software e dar um tratamento de engenharia ao desenvolvimento de software completo.** Naquela época, o processo de desenvolvimento era completamente fora de controle e tinha grandes dificuldades em entregar o que era requisitado pelo cliente.

Já na década de oitenta, **surgiu a Análise Estruturada e algumas Ferramentas CASE** que permitiam automatizar algumas tarefas. Na década de noventa, surgiu a orientação a objetos, linguagens visuais, processo unificado, entre outros conceitos diversos. E na última década, surgiram as metodologias ágeis e outros paradigmas de desenvolvimento muito comuns hoje em dia no contexto de desenvolvimento de software.

A Engenharia de Software possui alguns princípios fundamentais, tais como: **Formalidade**, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; **Abstração**, em que existe uma preocupação com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

Há a **Decomposição**, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica; **Generalização,** maneira usada para resolver um problema, de forma genérica, com o intuito de reaproveitar essa solução em outras situações; **Flexibilização** é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução. *Professor, como a formalidade pode reduzir inconsistências?*

- **Cenário 1:** Estamos na fase de testes de software. O testador afirma que fez todos os testes, você - líder de projeto – acredita nele sem pestanejar, e passa o software ao cliente homologar se está tudo certo. **O cliente tenta utilizar o software e verifica que várias partes estão com erros grosseiros de funcionamento.** *Viram como a ausência de uma formalidade pode gerar inconsistências?*

- **Cenário 2:** Estamos na fase de testes de software. **O testador afirma que fez todos os testes e entrega um documento de testes com tudo que foi verificado no sistema.** Você - líder de projeto - lê o documento de testes e verifica que não foram feitos testes de carga e testes de segurança. Retorna para o testador e pede para ele refazer os testes. Feito isso, ele passa o software ao cliente, que fica feliz e satisfeito porque está tudo funcionando corretamente.

*Vocês percebem que essas formalidades evitam aquele "telefone-sem-fio"?* Quanto mais eu seguir o processo, o passo-a-passo, o que foi definido por várias pessoas a partir de suas experiências com vários projetos, **maior a minha chance de obter êxito na construção do meu software.** *Bacana?* Vamos ver um resuminho...


A Engenharia de Software é uma disciplina dentro do contexto de Engenharia de Sistemas que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema após a sua entrada em produção (apesar de o nome indicar o contrário, em produção significa que o software já está disponível no ambiente real de utilização do cliente), passando por aspectos humanos, hardware, etc.

Além disso, ela é composta de quatro princípios fundamentais: decomposição, generalização, flexibilização e abstração. Por fim, é importante notar que ela se baseia em um conjunto de ferramentas, métodos e processos – sendo que a pedra fundamental que sustenta a engenharia de software é o foco na qualidade. Isso é apenas uma contextualização inicial – esse não é um assunto que cai com frequência em prova. *Fechou?* Vamos seguir então...

## 1.1 – Processos de Desenvolvimento de Software

Vamos começar falando sobre ciclo de vida! Esse termo surgiu lá no contexto da biologia. **Ele trata do ciclo de vida de um ser vivo, ou seja, trata do conjunto de transformações pelas quais passam os indivíduos de uma espécie durante toda sua vida**. *Vocês se lembram lá no ensino fundamental quando a gente aprende que um ser vivo nasce, cresce, reproduz, envelhece e morre?* Pois é! Aqui a metáfora ainda vale...

Vejam essa imagem: nós temos um bebê, que depois se torna uma criança, que depois um adolescente, que depois um jovem, que depois um adulto, que depois um velho, depois um ancião e depois morre! **Logo, o ciclo de vida de um ser vivo trata das fases pelas quais um ser vivo passa desde seu nascimento até a sua morte.** E esse conceito pode ser aplicado a diversos outros contextos.

**Exemplos: ciclo de vida de uma organização, ciclo de vida de sistemas, ciclo de vida de produtos, ciclo de vida de projetos, ciclo de vida de planetas, entre vários outros.** *E como esse conceito se dá em outros contextos?* Exatamente da mesma forma! Logo, o ciclo de vida de um produto trata da história completa de um produto através de suas fases (Ex: Introdução, Crescimento, Maturidade e Declínio).

Existe também o ciclo de vida de um projeto, que trata do conjunto de fases que compõem um projeto (Ex: Iniciação, Planejamento, Execução, Controle e Encerramento). *O que todos esses ciclos de vida têm em comum?* **Eles sempre tratam das fases pelas quais algo passa desde o seu início até o seu fim.** Então, é isso que vocês têm que decorar para qualquer contexto de ciclo de vida: fases pelas quais algo passa do seu início ao seu fim.

**Na Engenharia de Software, esse termo é geralmente aplicado a sistemas de software com o significado de mudanças que acontecem na vida de um produto de software**. O ciclo de vida

trata das fases identificadas entre o nascimento e a morte de um software. *Vocês viram?* É exatamente a mesma coisa – são as fases ou atividades pelas quais passa um software durante o decorrer da sua vida.

Uma definição interessante de ciclo de vida de software afirma que se trata das fases de um produto de software que vão desde quando ele é concebido inicialmente até quando ele não está mais disponível para uso. **Já Steve McConnell afirma que um modelo de ciclo de vida de software é uma representação que descreve todas as atividades que tratam da criação de um produto de software.**

Dessa forma, nós podemos concluir que um ciclo de vida de software se **refere às fases pelas quais um sistema de software atravessa desde sua concepção até sua retirada de produção**. Bem, entender esse conceito não é o problema, esse é um conceito muito simples. *Concordam comigo? E qual é o problema, professor?* O problema é que existem dezenas de fases diferentes para os ciclos de vida de acordo com cada autor.

*Como assim, professor?* Infelizmente, não há um consenso entre os autores. **Cada um que lança um livro decide criar o ciclo de vida com as fases que ele acha mais corretas e isso acaba prejudicando nosso estudo.** Na UnB, eu tive um professor de Engenharia de Software chamado Fernando Albuquerque que já tinha escrito vários livros sobre esse assunto e ele tinha o seu próprio ciclo de vida com suas respectivas fases.

**Agora imaginem a quantidade de autores de livros de Engenharia de Software lançados nas últimas três ou quatro décadas.** Além disso, acontece de as vezes o próprio autor mudar seu entendimento sobre as fases. Se vocês olharem a quarta edição do livro do Roger Pressman, ele tem um conjunto de fases; se vocês olharem da sexta edição para frente, ele define um novo conjunto de fases. *O que aconteceu?* Ele mudou de ideia!

Então, eu já vi vários ciclos de vida diferentes em provas! *Qual a solução para esse problema, professor?* Galera, vocês sempre têm que pensar no custo/benefício. *Vale a pena decorar todos?* Na minha opinião, nem um pouco! *Por que?* Porque isso não é algo que cai muito em prova – principalmente nas provas recentes. **Guardem o espaço que vocês têm sobrando na cabeça para memorizar coisas que realmente caem.**

Sendo assim, percebam que há autores que descrevem as fases do ciclo de vida de software de maneira mais ampla e abstrata (com poucas e grandes fases) e autores que o fazem de maneira mais diminuta e detalhada (com muitas e pequenas fases). **Vamos começar a ver alguns ciclos de vida que existem por aí para que vocês visualizem isso com maior clareza!**



Vamos lá! Do lado esquerdo, temos um ciclo de vida de software só com quatro fases bem genéricas: Definição do Software, Desenvolvimento do Software, Operação do Software e Retirada do Software. Eu só vi cair esse ciclo de vida uma vez, mas ele é útil para que vocês visualizem um ciclo de vida com poucas fases. **Observem que ele trata desde o início até a aposentadoria ou retirada do software.**

Do lado direito, eu coloquei um ciclo de vida um pouco mais detalhado. Vejam que ele destrincha mais as fases de desenvolvimento, separando em análise, projeto, implementação, testes, etc. *Por que eu coloquei esse ciclo de vida?* **Porque alguns autores afirmam que, em tese, todo ciclo de vida deveria contemplar todas essas atividades ou fases.** Relaxa, nós vamos ver isso com mais detalhes depois...



Outros exemplos: em verde, nós temos um ciclo de vida de software de acordo com nosso querido autor: Ian Sommerville. Ele contém quatro fases: Especificação, Desenvolvimento, Validação e Evolução. Sendo que, atenção aqui, o desenvolvimento pode ser dividido em Projeto e Implementação. *Tudo certo?* **Esse é um ciclo de vida de software que já cai com uma maior frequência (nada excepcional, só cai mais que os outros).**

Por fim, em laranja, nós temos um ciclo de vida de software de acordo com nosso outro querido autor: Roger Pressman. **Ele contém cinco fases: comunicação, planejamento, modelagem, construção e implantação**. Esse não cai tanto, mas é importante saber também. *Tudo bem até aqui?* Então vejam que não é tão complicado! Esses são os quatro ciclos de vida de software mais comuns em prova... e são raros!

Então, vamos recapitular: inicialmente, nós vimos o conceito de um Ciclo de Vida! Depois nós vimos o que é um Ciclo de Vida de Software. **E, agora, nós vamos ver um terceiro conceito, que é o conceito de Modelo de Ciclo de Vida de Software.** *Por que, professor?* Porque Ciclo de Vida de Software é diferente de Modelo de Ciclo de Vida de Software. *E o que é um modelo de ciclo de vida de software?*

**Bem, um modelo de ciclo de vida de software é um modelo que apresenta não só as fases do ciclo de vida do software, mas também a forma como essas fases se relacionam.** *Diego, não entendi isso muito bem!* Galera, um modelo contém as fases que nós acabamos de ver, mas ele também nos diz como essas fases se relacionam! *Vocês querem um exemplo?* Existe um modelo de ciclo de vida chamado Modelo em Cascata.

Esse modelo foi pioneiro – ele foi o primeiro modelo a aparecer na Engenharia de Software e nós vamos vê-lo em mais detalhes em outro momento. **O importante sobre o Modelo em Cascata é a forma como as fases se relacionam.** Nesse modelo, nós temos uma regra de ouro: uma fase somente se inicia após o término completo da fase anterior (exceto a primeira, evidentemente) – não é possível fazer fases paralelamente. *Viram como as fases se relaciona?*

Dessa forma, um Modelo de Ciclo de Vida de Software contém suas respectivas fases, **mas também contém como essas fases se relacionam.** Vejam os conceitos abaixo:

*E o que seria um processo de software?* **Então, um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software.** Como o Modelo de Ciclo de Vida nos diz como as fases do ciclo de vida se relacionam, nós podemos – em termos de prova – considerar Modelo de Ciclo de Vida como sinônimo de Modelo de Processo! (Aliás, pessoal... infelizmente muitas questões ignoram essas diferenças!).


Um processo de software não pode ser definido de forma universal. **Para ser eficaz e conduzir à construção de produtos de boa qualidade, um processo deve ser adequado às especificidades do projeto em questão.** Deste modo, processos devem ser definidos caso a caso, considerando-se diversos aspectos específicos do projeto em questão, tais como:

**#1.** Características da aplicação (domínio do problema, tamanho do software, tipo e complexidade, entre outros);

**#2.** Tecnologia a ser adotada na sua construção (paradigma de desenvolvimento, linguagem de programação, mecanismo de persistência, entre outros);

**#3.** Organização onde o produto será desenvolvido e a equipe de desenvolvimento alocada (recursos humanos).

**Há vários aspectos a serem considerados na definição de um processo de software.** No centro da arquitetura de um processo de desenvolvimento estão as atividades-chave desse processo: análise e especificação de requisitos, projeto, implementação e testes, que são a base sobre a qual o processo de desenvolvimento deve ser construído. *Entendido?*

No entanto, **a definição de um processo envolve a escolha de um modelo de ciclo de vida (ou modelo de processo)**, o detalhamento (decomposição) de suas macro-atividades, a escolha de métodos, técnicas e roteiros (procedimentos) para a sua realização e a definição de recursos e artefatos necessários e produzidos – é bastante coisa!

Podemos dizer que se trata da representação abstrata de um esqueleto de processo, incluindo tipicamente algumas atividades principais, a ordem de precedência entre elas e, opcionalmente,

artefatos requeridos e produzidos. **Em geral, um modelo de processo descreve uma filosofia de organização de atividades, estruturando-as em fases e definindo como essas fases estão relacionadas.**

A escolha de um modelo de ciclo de vida (para concursos, sinônimo de modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. **Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas.** *Tudo certo?*

Conforme eu disse anteriormente, alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: **Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção.** Abaixo eu trago uma descrição genérica sobre cada uma dessas fases.

| FASES | DESCRIÇÃO |
|---|---|
| PLANEJAMENTO | O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, com os requisitos esboçados, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto. |
| ANÁLISE E ESPECIFICAÇÃO DE REQUISITOS | Nesta fase, o processo de levantamento de requisitos é intensificado. O escopo deve ser refinado e os requisitos mais bem definidos. Para entender a natureza do software a ser construído, o engenheiro de software tem de compreender o domínio do problema, bem como a funcionalidade e o comportamento esperados. Uma vez capturados os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo o que o software tem de fazer (e não como fazê-lo). |
| PROJETO | Esta fase é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes. O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados. |
| IMPLEMENTAÇÃO | O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada. |
| TESTES | Inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada e os resultados documentados. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado. |

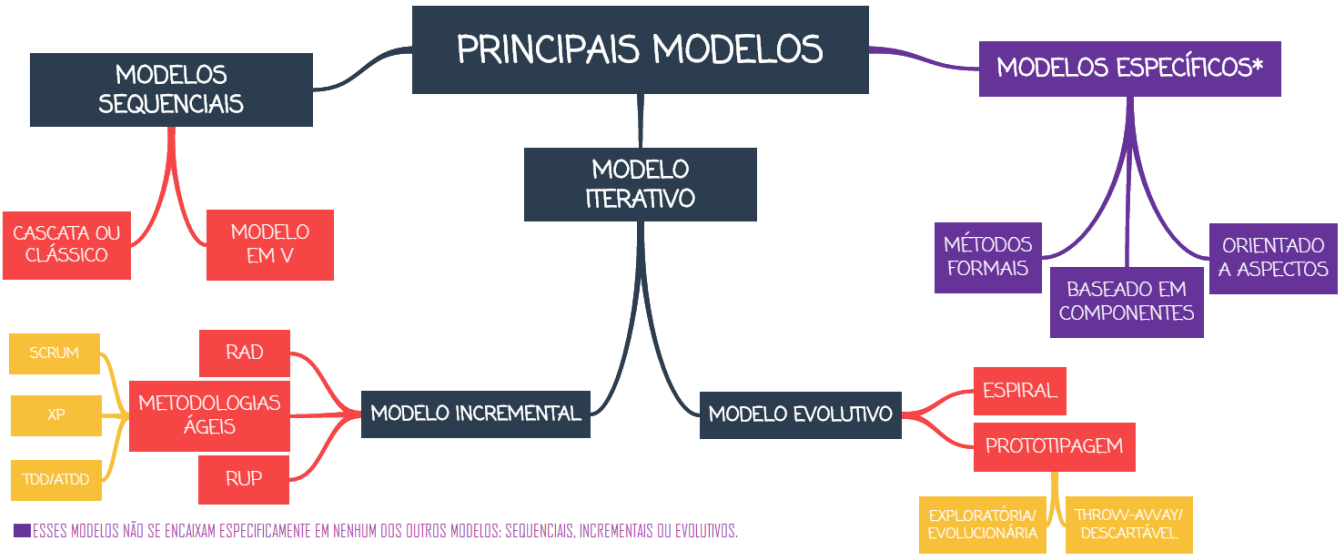
| | |
|---|---|
| ENTREGA E IMPLANTAÇÃO | Uma vez testado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação. Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada. |
| OPERAÇÃO | Nesta fase, o software é utilizado pelos usuários no ambiente de produção, isto é, no ambiente real de uso do usuário. |
| MANUTENÇÃO | Indubitavelmente, o software sofrerá mudanças após ter sido entregue para o usuário. Alterações ocorrerão porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, essa fase pode requerer a definição de um novo processo, onde cada uma das fases precedentes é reaplicada no contexto de um software existente ao invés de um novo. |
| PLANEJAMENTO | O objetivo do planejamento de projeto é fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custos e prazos. Uma vez estabelecido o escopo de software, com os requisitos esboçados, uma proposta de desenvolvimento deve ser elaborada, isto é, um plano de projeto deve ser elaborado configurando o processo a ser utilizado no desenvolvimento de software. À medida que o projeto progride, o planejamento deve ser detalhado e atualizado regularmente. Pelo menos ao final de cada uma das fases do desenvolvimento (análise e especificação de requisitos, projeto, implementação e testes), o planejamento como um todo deve ser revisto e o planejamento da etapa seguinte deve ser detalhado. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto. |

**Existem outras fases em outros modelos, tais como:** análise, responsável por modelar o problema (diferente do projeto, responsável por modelar a solução do problema); homologação, responsável pela aceitação pela parte interessada do produto; gerência de configuração, responsável pela estruturação sistemática dos produtos, artefatos, documentos, modelos, entre outros. *Bacana?*

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: **Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.**

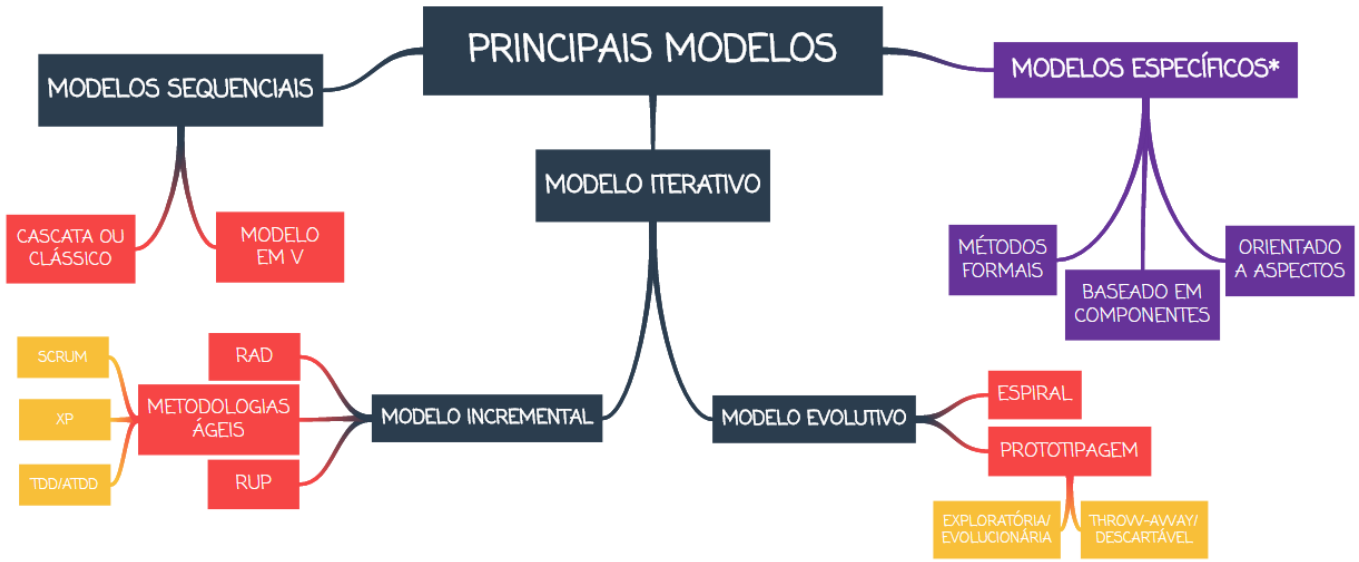
**Também de acordo nosso querido autor, um modelo de processo de software é uma descrição simplificada desse processo de software que apresenta uma visão dele.** Os modelos de processo incluem as atividades, que fazem parte do processo de software, e eventualmente os produtos de software e os papéis das pessoas envolvidas na engenharia de software. E não para por aí...

Ele ainda afirma que a maioria dos processos de software é baseada em três modelos gerais: modelo em cascata; desenvolvimento iterativo e engenharia de software baseada em componentes. Isso entra em contradição com o que dizem outros autores, isto é, **os principais modelos podem ser agrupados em três categorias: modelos sequenciais, modelos incrementais e modelos evolutivo.**



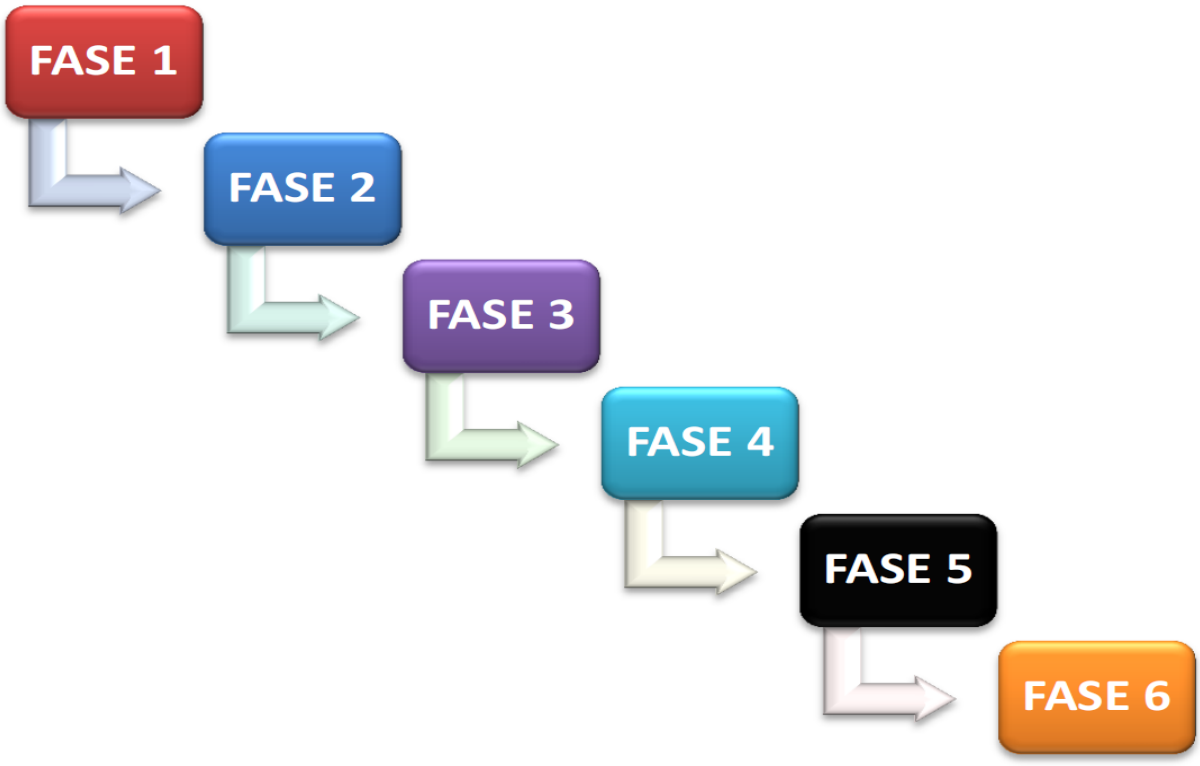

Por fim, existe mais um conceito importante nessa aula! É o conceito de Metodologia de Desenvolvimento de Software (também chamada de Processo de Desenvolvimento de Software). *O que é isso, professor?* **É basicamente uma caracterização prescritiva ou descritiva de como um produto de software deve ser desenvolvido**, isto é, ela define o quê, como e quando fazer algo para desenvolver um software. Calma, tudo ainda fará sentido...

## 1.2 – Modelo em Cascata




Citado inicialmente em 1970 por W. Royce, é também denominado Modelo em Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. No Modelo em Cascata, **uma fase só se inicia após o término e aprovação da fase anterior**, isto é, há uma sequência de desenvolvimento do projeto.

*Como assim, Diego?* Por exemplo: a Fase 4 só pode ser iniciada após o término e aprovação da Fase 3; a Fase 5 só pode ser iniciada após o término e aprovação da Fase 4; e assim por diante!



*Mas que fases são essas, Diego?* Bem, agora complica um pouco porque cada autor resolve criar suas fases! Vejam só na tabelinha a seguir:

| POR SOMMERVILLE | POR ROYCE | POR PRESSMAN (4ª ED) | POR PRESSMAN (6ª ED) |
|---|---|---|---|
| Análise e Definição de Requisitos | Requisitos de Sistema | Modelagem e Engenharia do Sistema/Informação | Comunicação |
| Projeto de Sistema e Software | Requisitos de Software | Análise de Requisitos de Software | Planejamento |
| Implementação e Teste de Unidade | Análise | Projeto | Modelagem |
| Integração e Teste de Sistema | Projeto | Geração de Código | Construção |
| Operação e Manutenção | Codificação | Teste e Manutenção | Implantação |
| | Teste | | |
| | Operação | | |

**Percebam que há grandes diferenças entre os autores**! Inclusive, há divergências até entre autor e ele mesmo, dependendo da versão do livro (Exemplo: Pressman mudou as fases na última edição de seu livro). *Professor, você já viu isso cair em prova?* Sim, já vi! *E o que aconteceu?* Bem, polêmica, recursos, etc – não há o que fazer! Enfim... a minha classificação preferida é a do Royce por conter as fases que eu acho que realmente ocorrem no desenvolvimento de um software.

**Galera, na prática, esses estágios do modelo em cascata não são rigorosamente sequenciais, isto é, eventualmente eles se sobrepõem e trocam informações entre si.** Na teoria, são fases rigorosamente sequenciais, sendo que o resultado de cada fase consiste em um ou mais documentos aprovados ou não, dependendo dos problemas. Por exemplo: durante o projeto, podem ser identificados problemas com os requisitos da fase anterior.

Em suma: o projeto segue uma série de passos ordenados em que, ao final de cada fase, a equipe do projeto finaliza uma revisão. Além disso, **o desenvolvimento não continua até que o cliente esteja satisfeito com os resultados alcançados**. *Vocês conseguem perceber como essas restrições engessam o desenvolvimento?* Se não, vocês vão entender em breve! De modo geral, grande parte dos modelos possuem as seguintes fases:

a) **Planejamento:** faz-se o esboço do escopo e dos requisitos do software, além de levantar estimativas razoáveis sobre recursos, custos e prazos.

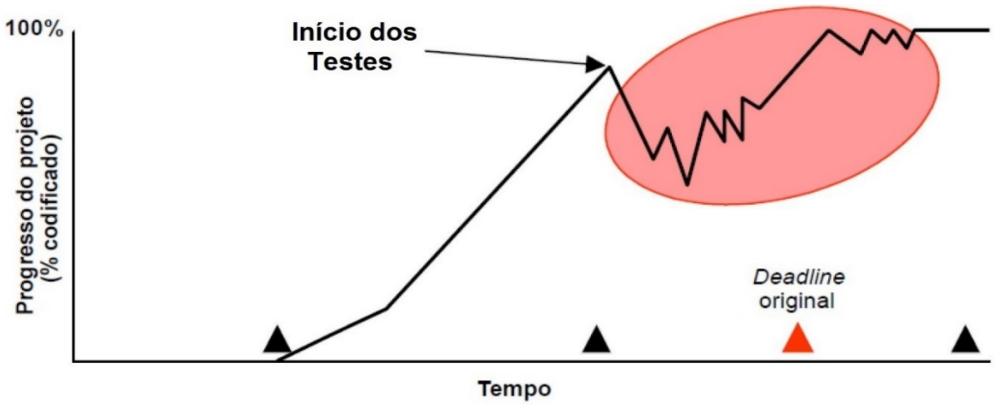
b) **Análise e Especificação de Requisitos:** durante essa fase, refinam-se os requisitos e o escopo, e desenha-se o problema a ser resolvido pelo software.

c) **Projeto:** durante essa fase, incorporam-se requisitos tecnológicos aos requisitos essenciais do sistema e projeta-se a arquitetura do sistema.

**d) Implementação:** durante essa fase, codifica-se o software como um conjunto de programas executáveis pela máquina.

**e) Teste:** o programa é testado como um sistema completo para garantir que os requisitos de software foram atendidos.

**f) Implantação, Operação e Manutenção:** o sistema de software é liberado para o cliente, treinam-se usuários, gerenciam-se os serviços e realizam-se manutenções.




O MODELO EM CASCATA ATRASA A REDUÇÃO DE RISCOS...

O Modelo em Cascata tem um grande problema: ele atrasa a redução de riscos! *Como assim, Diego?* Bem, essa é uma desvantagem recorrente em provas! Como uma fase só se inicia após o término e aprovação da fase anterior, **em geral só é possível verificar se ocorreram erros nas fases finais, que é quando o sistema é efetivamente testado** – isso gera grandes riscos! Em outros modelos, os riscos são reduzidos desde as primeiras fases do processo de desenvolvimento.

Percebam que os riscos deveriam ser descobertos logo no início do processo de desenvolvimento, no entanto eles são descobertos somente após o início dos testes e implantação. Vocês podem notar no gráfico acima que, a partir da região vermelha, **o progresso do projeto sobe e desce diversas vezes**, porque provavelmente o sistema está sendo corrigido devido a requisitos modificados. *Descobriu um erro?* Desce! *Corrigiu o erro?* Sobe!

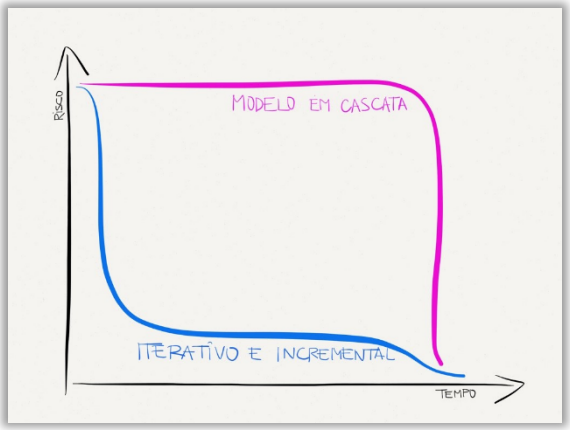
Vejam, também, **que o projeto não terminou em seu deadline (prazo) original**. Como a redução de riscos atrasou, todo andamento do projeto também atrasou. Dessa forma, não se cumpriu nem o prazo do projeto e, provavelmente, nem o orçamento e talvez nem seu escopo – tendo em vista que, quanto mais ao fim do projeto um erro é identificado, mais caras se tornam as modificações.

*Entenderam essa parte direitinho?* **Um erro na fase de requisitos, por exemplo, que não foi corrigido e foi descoberto no final do processo de desenvolvimento, terá um custo de correção altíssimo**, visto que provavelmente terá que se refazer tudo novamente. *Vocês conhecem a Torre de Pisa?* Ela fica na Itália e é famosa por ser torta. Respondam: *seria mais fácil corrigir a torre logo no início da construção do primeiro andar ou no último?*

Ora, se alguém tivesse notado que a torre estava torta logo no início da construção do primeiro andar, seria possível corrigi-la sem ter tanto trabalho! Agora se somente ao final da construção alguém notasse que a torre estava bastante torta, seria muito mais trabalhoso e caro para corrigir. *Concordam comigo?* A mesma coisa acontece com um software! Se o cliente me pede uma coisa, mas eu entendo outra e somente mostro para ele quando estiver tudo pronto, é evidente que eu estou atrasando a redução de riscos. *Por que?* Porque se eu tivesse mostrado para ele cada passo do desenvolvimento desde o início, ele poderia identificar o erro de forma que eu pudesse corrigi-lo tempestivamente. **Em outras palavras, eu teria adiantado a redução de riscos** – eu estaria reduzindo os riscos de fazer algo errado de maneira adianta! *Sacaram?*

Portanto não confundam essas duas coisas: **se o erro ocorreu no início e foi identificado no início, terá baixo custo de correção; se o erro ocorreu no início e foi identificado no final, terá alto custo de correção.** Dessa forma, o custo de correção de um erro está mais focado no momento em que um erro é identificado do que no momento em que ele de fato ocorreu. *Vocês entenderam legal? Então, vamos seguir...*

Outra maneira de visualizar o atraso é por meio de um gráfico Risco x Tempo, comparando o modelo em cascata com o Modelo Iterativo e Incremental (que veremos a seguir). Observem que o Modelo Iterativo e Incremental já começa a reduzir os riscos desde o início do processo de desenvolvimento, **em contraste com o Modelo em Cascata que acumula os riscos até a fase de testes, integração e implantação do sistema**. Vejam o gráfico ao lado e tentem entender essa interpretação que acabamos de ver!



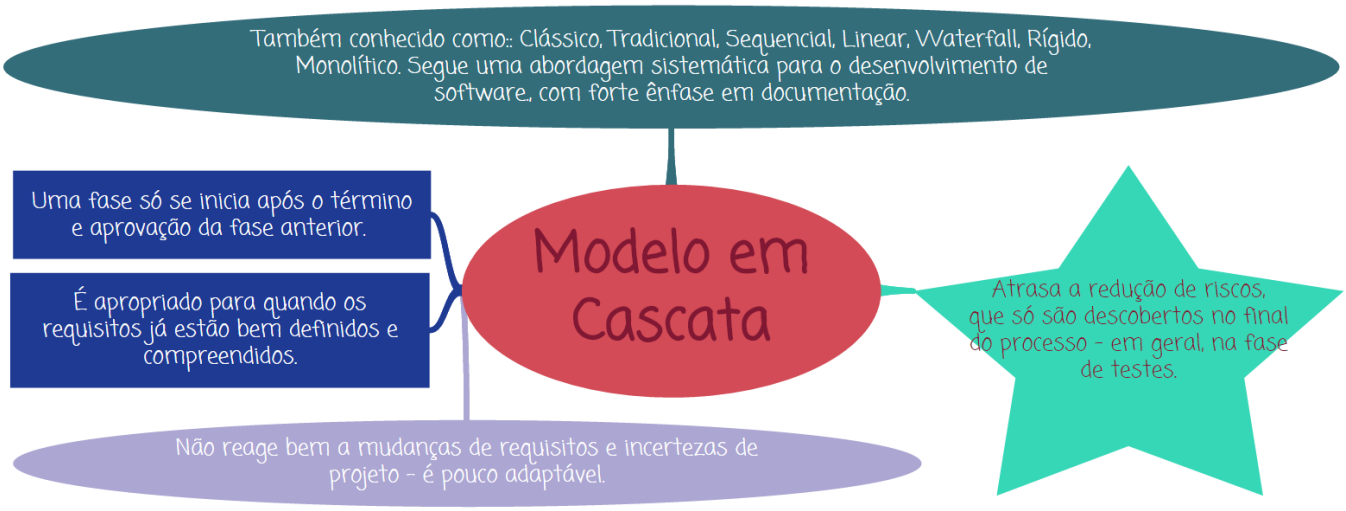

Galera, a grande vantagem do Modelo em Cascata é que o desenvolvimento é dividido em fases distintas, padronizadas, entre outras. *Vantagem, professor?* Em relação ao que existia antes, sim! Antigamente, os softwares eram construídos quase que de maneira artesanal. Ademais, **é possível colocar pessoas com perfis específicos para cada fase**, isto é, quem tem facilidade de se comunicar vai ser analista de requisitos, programadores vão fazer a codificação, etc.

A grande desvantagem é que - em projetos complexos – demora-se muito para chegar até a fase de testes, sendo que o cliente não vê nada rodando até a implantação. Então, pode acontecer de, nas fases finais, os requisitos da organização não serem mais os mesmos daqueles do início **e o software não ter mais utilidade para organização**. Imaginem que vocês contratam um pedreiro para construir a casa de vocês.

Só que ele não vai te mostrar nada de como a casa está ficando, ele só vai te mostrar quando já estiver tudo pronto daqui um ano. *É interessante?* Não! Primeiro, porque você vai morrer de ansiedade. Segundo, porque isso atrasará a redução de riscos, aumentando a probabilidade de erros graves. Terceiro, porque demora tanto que durante esse ano você pode ter mudado de ideia – queria uma casa de um andar e mudou de ideia para uma casa de dois andares, por exemplo.

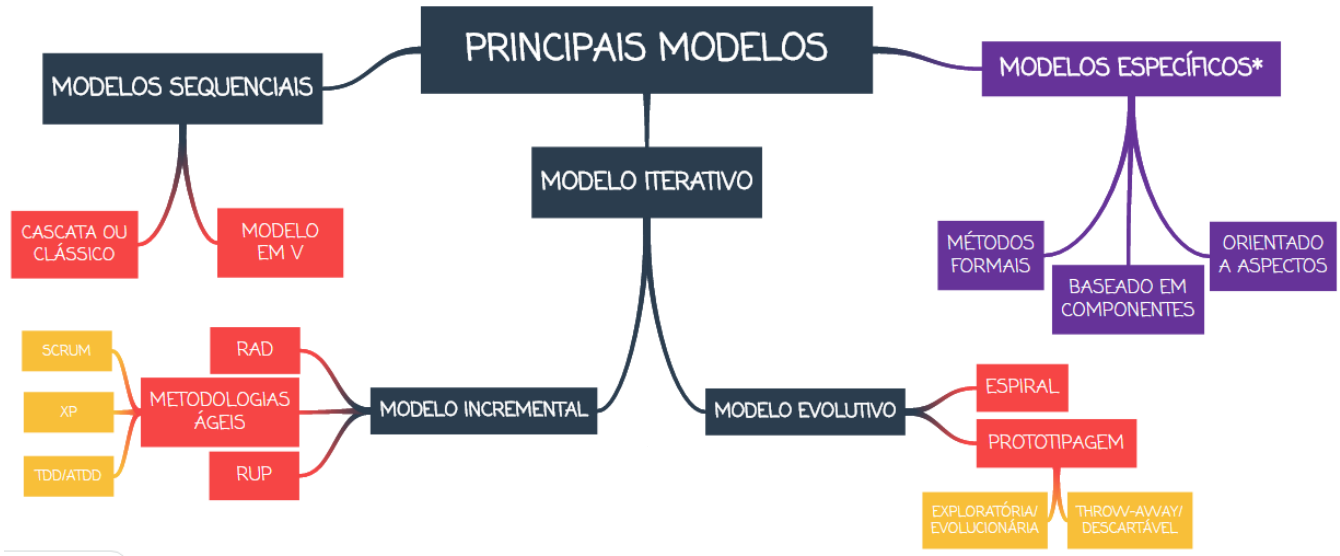
*Então o Modelo em Cascata não deve ser usado em nenhuma hipótese?* Calma lá, ele pode ser utilizado, sim – apesar de atualmente ser bem raro! No entanto, sua utilização deve ocorrer **preferencialmente quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudanças radicais durante o desenvolvimento do sistema**. *Vocês entenderam?* Então vamos ver agora uma lista com as maiores vantagens e desvantagens.

| VANTAGENS | DESVANTAGENS |
|---|---|
| É simples de entender e fácil de aplicar, facilitando o planejamento. | Divisão inflexível do projeto em estágios distintos. |
| Fixa pontos específicos para a entrega de artefatos. | Dificuldade em incorporar mudanças de requisitos. |
| Funciona bem para equipes tecnicamente fracas. | Clientes só visualizam resultados próximos ao final do projeto. |
| É fácil de gerenciar, devido a sua rigidez. | Atrasa a redução de riscos. |
| Realiza documentação extensa por cada fase ou estágio. | Apenas a fase final produz um artefato de software entregável. |
| Possibilita boa aderência a outros modelos de processo. | Cliente deve saber todos os requisitos no início do projeto. |
| Funciona bem com projetos pequenos e com requisitos bem conhecidos. | Modelo inicial (Royce) não permitia feedback entre as fases do projeto. |
| | Pressupõe que os requisitos ficarão estáveis ao longo do tempo. |
| | Não funciona bem com projetos complexos e OO, apesar de compatível. |



Para finalizar, podemos afirmar que **o Modelo em Cascata é considerado um método tradicional e fortemente prescritivo de desenvolvimento de software**, isto é, ele busca sempre dizer o que fazer, em geral, por meio de planos definidos no início do projeto. *Que planos, professor?* Escopo, custo, cronograma... tudo isso bem detalhado em uma extensa documentação. *Mudanças são bem-vindas?* Claro que não! Mudanças são bem vindas no modelo do nosso próximo assunto...

## 1.3 – Modelo Iterativos e Incrementais




Como foi dito anteriormente, o Modelo em Cascata acumulava riscos e vários projetos começaram a fracassar um atrás do outro ao utilizá-lo no mundo inteiro. *Diego, o que você quer dizer com fracassar?* O projeto não era entregue, ou era entregue de forma completamente errada, ou era entregue faltando partes, ou era entregue no dobro do prazo combinado, ou era entregue com o dobro do custo acordado, enfim... diversos motivos!

**Foi quando surgiu o Modelo Iterativo e Incremental como uma tentativa de resolver esse problema de acúmulo de riscos**. Vejamos as diferenças fundamentais:

- **No Modelo em Cascata**, caso haja cem requisitos, analisam-se os cem requisitos, projetam-se os cem requisitos, codificam-se os cem requisitos, testam-se os cem requisitos, e assim por diante sequencialmente;

- **No Modelo Incremental**, caso haja cem requisitos, dividem-se os cem requisitos em vinte miniprojetos de cinco requisitos cada e utiliza-se o modelo em cascata para cada miniprojeto;

- **No Modelo Iterativo**, caso haja cem requisitos, analisam-se, projetam-se, codificam-se, testam-se os cem requisitos, porém os requisitos são entregues incompletos e eu repito esse ciclo de refinamento até chegar ao produto final.

Sim, existe a abordagem incremental e a abordagem iterativa. **No entanto, na prática elas virão sempre de forma combinada no modelo iterativo e incremental para desenvolver os miniprojetos e entregar partes diferentes do projeto**. A imagem a seguir apresenta miniprojetos sendo feitos iterativamente em um modelo iterativo e incremental. Observem que cada iteração (passagem pelas fases de desenvolvimento) refinam cada vez mais a funcionalidade.


Dessa forma, os resultados são mais rápidos, há maior interação com o usuário e há um feedback mais intenso entre usuário e desenvolvedor – sendo possível reagir mais facilmente a mudanças. **Essa abordagem permite gerenciamento e mitigação de riscos**. *Professor, mas eu fiquei com uma dúvida: qual é a diferença entre a abordagem iterativa e abordagem incremental? Ou eles são exatamente a mesma coisa?*

Bem, galera... **eu nunca vi nenhuma prova cobrar essa diferença entre modelo iterativo e modelo incremental!** Como eu disse, quando se fala em modelo iterativo, já se presume que é incremental; e quando se fala em modelo incremental, já se presume que é iterativo. Eles frequentemente andam lado a lado, mas há pequenas diferenças.

**IMPORTANTE**

CUIDADO COM A GRAFIA DAS PALAVRAS ITERATIVO E INTERATIVO. EU JÁ AS VI UTILIZADAS DE FORMA INVERTIDA DEZENAS DE VEZES EM PROVAS E ATÉ EM EDITAIS. POR VEZES, BANCAS NÃO ACATAM OS RECURSOS CONTRA ISSO! DE TODO MODO, SAIBAM QUE:

ITERATIVO: REITERADO OU REPETITIVO / INTERATIVO: PARTICIPAÇÃO OU AÇÃO MÚTUA.

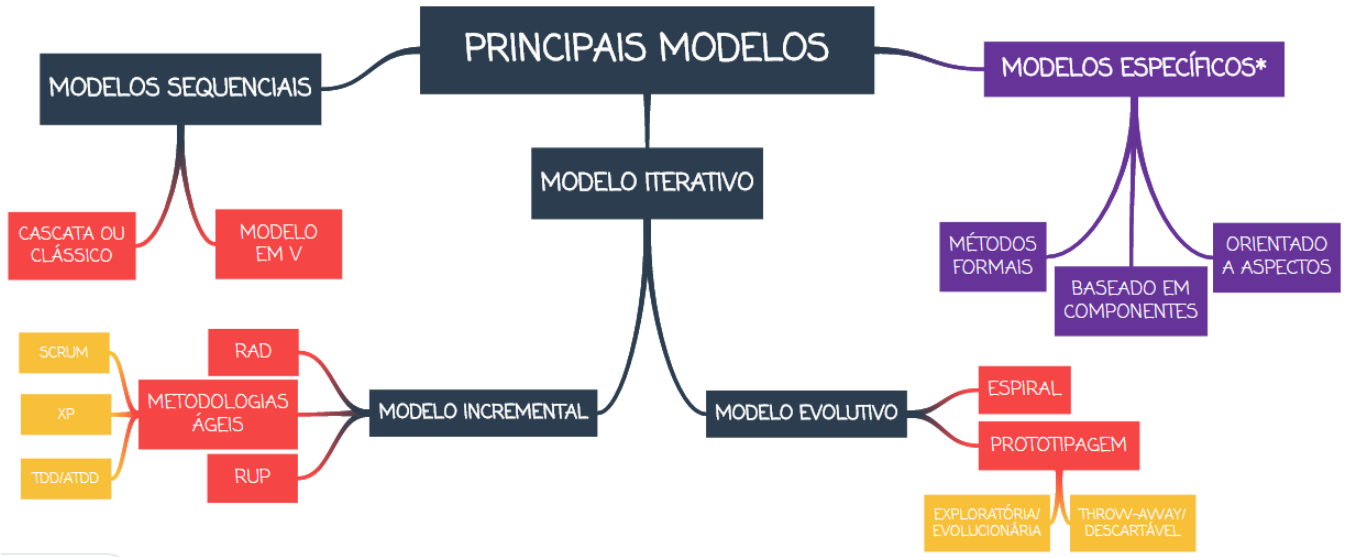
*Professor, mas e se cair essa diferença em prova?* Ora, caso caia em prova, saibam que a diferença é que, **no modelo incremental**, há várias equipes desenvolvendo uma parte do software a serem integradas ao final do desenvolvimento. Já **no modelo iterativo**, lança-se a versão 1.0, adicionam-se algumas funcionalidades; lança uma versão 2.0, adicionam-se mais algumas funcionalidades; e assim por diante. Vejamos isso mais claramente utilizando o clássico exemplo da Monalisa...

**Modelo Incremental:** observem que a imagem mostra um artista com uma ideia completa já sobre o quadro em sua cabeça, mas ele desenvolve cada parte do quadro separadamente até integrar as partes em uma imagem completa. É como se fosse um quebra-cabeças em que cada parte é entregue funcionando e depois integrada. **Ele produz *builds*, isto é, partes do software**.


**Modelo Iterativo:** observem que a imagem mostra um artista com um esboço do quadro, sendo que ele desenvolve várias versões até chegar ao resultado final que ele deseja. É como se fosse uma visão abstrata da imagem, que em seguida vai sendo melhorada até chegar a uma visão mais concreta. **Ele produz *releases*, isto é, versões constantemente melhoradas do software**.

Uma das vantagens do modelo iterativo e incremental é que **o cliente pode receber e avaliar as entregas do produto mais cedo, já no início do desenvolvimento do software**. Além disso, há maior tolerância a mudanças com consequência direta de redução do risco de falha do projeto, isto é, ele acomoda melhor mudanças – aumentando o reúso e a qualidade. *Lembram do exemplo do pedreiro e da casa?* Vale o mesmo aqui...

## 1.4 – Modelos Evolucionários




Antes de começarmos, eu acho oportuno dizer que a maioria dos autores consideram o **Modelo Evolucionário (Evolutivo) como um tipo de Modelo Iterativo** conforme mostra a imagem acima! Um dos nossos guias, Pressman diz: *Os modelos evolucionários são iterativos, apresentando características que possibilitam desenvolver versões cada vez mais completas do software.* Agora é hora de ver isso melhor...

Vocês sabem que mudanças são extremamente comuns, isto é, as necessidades de negócio e de produto mudam frequentemente, tornando inadequado seguir um planejamento rigoroso em linha reta de um produto final. Hoje em dia, nós vivemos em função do tempo! **É raro um cliente exigir um produto de software com prazos folgados e tranquilos – em geral, o cliente quer o produto para ontem!**

Todas essas variáveis determinadas pelo mercado tornam impossível terminar integralmente um produto de software mais abrangente, no entanto uma versão limitada pode de ser introduzida para aliviar e/ou atender às pressões comerciais ou da concorrência. **E é nessa hora que entra o conceito de Modelos Evolucionários e suas implementações.**

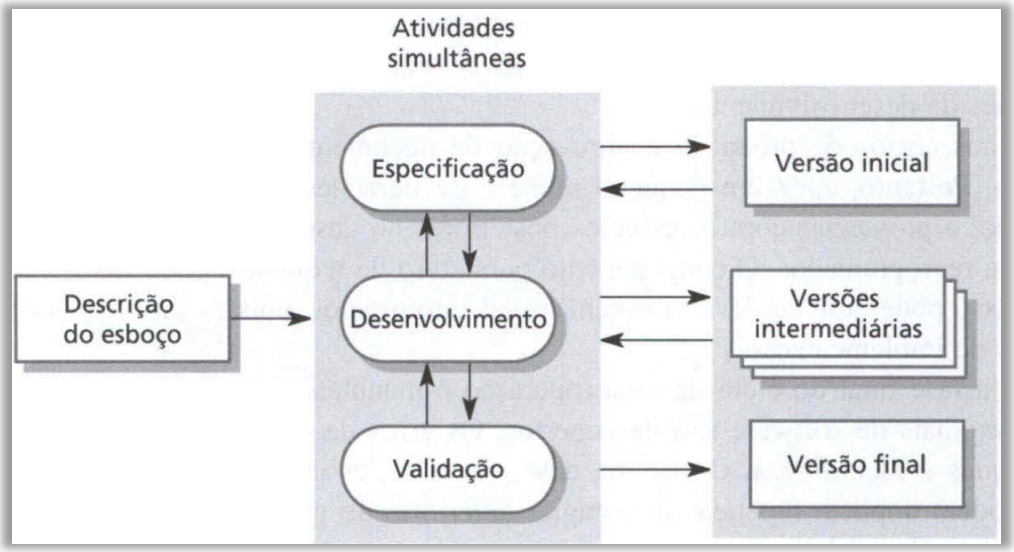
O Modelo Evolucionário **se baseia na ideia de desenvolvimento de uma implementação inicial**, expondo o resultado aos comentários do usuário e refinando esse resultado por meio de várias versões até que seja desenvolvido um sistema adequado às necessidades do cliente. As implementações mais famosas do Modelo Evolucionário são: Prototipagem e Espiral!

Em geral, **a abordagem evolucionária é mais eficaz que abordagem em cascata**. *Por que?* Porque a especificação pode ser desenvolvida de forma incremental e, não, integralmente como naquele modelo. À medida que os usuários compreendem melhor seu problema, isso pode ser refletido no sistema de software. No entanto, essa abordagem possui dois problemas (que, inclusive, já caíram no Senado Federal):

a) **O processo não é visível:** se os sistemas são desenvolvidos rapidamente, não é viável economicamente produzir documentos para cada versão do sistema.

b) **Os sistemas são frequentemente mal estruturados:** mudanças contínuas tendem a corromper a estrutura do software e tornar mudanças difíceis e caras.

Roger Pressman recomenda o Desenvolvimento Evolucionário **para sistemas de pequeno e médio porte** (até 500 mil linhas de código). Já para sistemas maiores e mais complexos, esses problemas supracitados tornam-se mais graves, uma vez que é difícil estabelecer uma arquitetura estável do software, tornando complexo integrar as contribuições das equipes.




Um esboço simples do processo de desenvolvimento evolucionário é apresentado acima. As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de separadas, **com feedback rápido que permeia as atividades**. Desenvolve-se rapidamente uma implementação inicial do software (protótipo) a partir de especificações bastante abstratas e são feitas modificações de acordo com sua avaliação.

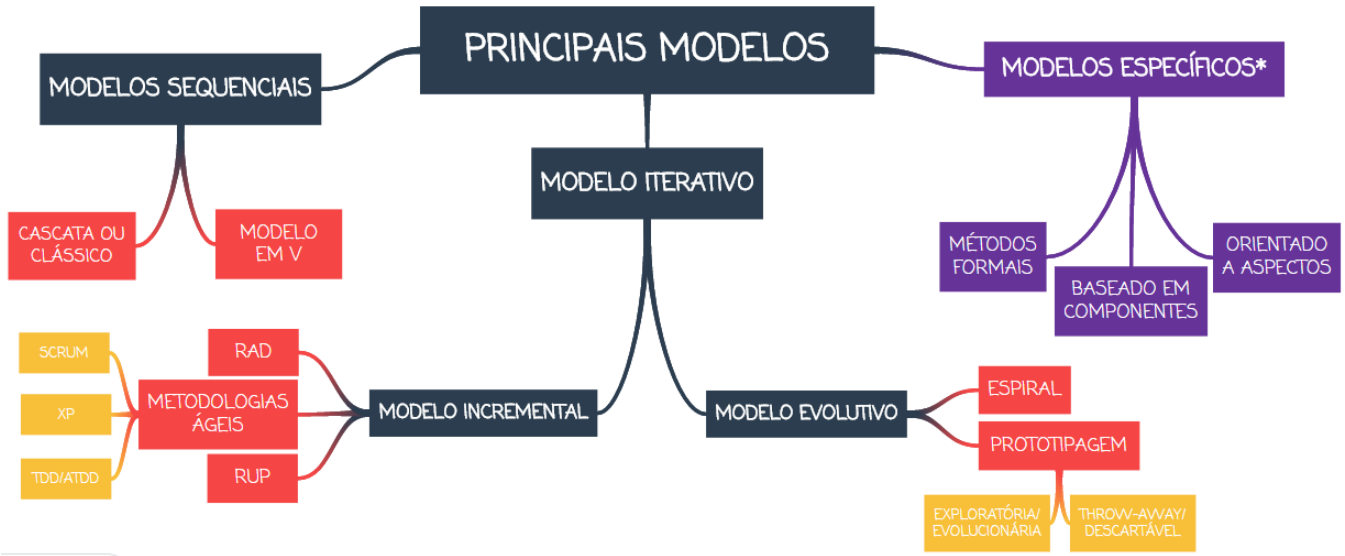
Cada versão do programa herda as melhores características das versões anteriores. **Cada versão é refinada com base no feedback recebido dos clientes para produzir um sistema que satisfaça suas necessidades**. Neste ponto, o sistema pode ser entregue ou pode ser reimplementado utilizando uma abordagem mais estruturada para aumentar a robustez e a facilidade de manutenções – veremos mais à frente!

As atividades de especificação, desenvolvimento e validação são concorrentes e apresentam um forte feedback entre si, como mostra a imagem apresentada anteriormente. **Por fim, é bom esclarecer a diferença entre Modelo Incremental e Evolucionário!** Pressman afirma que o primeiro tem o objetivo de apresentar um produto de trabalho ou uma funcionalidade operacional a cada iteração.

Já o segundo, durante as primeiras iterações, pode gerar versões compostas apenas por modelos em papel ou protótipos. **É beeeem sutil!** Em suma, um Modelo Evolucionário não vai necessariamente liberar funcionalidades prontas para o cliente em todas as iterações como ocorre no Modelo Incremental. É possível, por exemplo, ter uma iteração utilizada só para estudar melhor o produto. *Isso é uma funcionalidade útil para o cliente?* Não!

*Vamos resumir?* **O Modelo Evolucionário é quase idêntico ao Modelo Iterativo Incremental**! Ele funciona também com base em iterações ou repetições com o intuito de refinar o software a partir do feedback do cliente. A ideia é utilizar um esboço inicial e ir incrementando, melhorando, aperfeiçoando o software de acordo com as necessidades dos clientes até chegar a uma versão. Ao fim, o sistema pode ser entregue ao cliente ou ser refeito de forma mais estruturada.

## 1.5 – Modelos em Prototipagem




A Prototipagem é utilizada quando não se conhece bem os requisitos. **Trata-se de uma forma de entendê-los melhor para posteriormente desenvolver o software.** Ela se configura como um processo iterativo, interativo e rápido de desenvolvimento e o protótipo serve como um mecanismo de identificação dos requisitos do software, que servirão para uma futura especificação ou serão refinados e entregues ao cliente.

Um protótipo é uma versão inicial de um sistema de software utilizado para demonstrar conceitos, experimentar opções de projeto e, geralmente, conhecer mais sobre o problema e suas possíveis soluções. **Assim, os custos são controlados e as partes interessadas do sistema podem experimentar o protótipo mais cedo no processo de desenvolvimento**. A Prototipagem pode ocorrer de duas maneiras:

a) **Desenvolvimento Exploratório ou Evolucionária:** o objetivo do processo de desenvolvimento exploratório é trabalhar com o cliente para explorar os requisitos e entregar um sistema final. O desenvolvimento começa com as partes do sistema compreendidas. O sistema evolui por meio da adição de novas características propostas pelo cliente.

b) **Prototipação Throwaway ou Descartável:** o objetivo do processo de prototipação throwaway é compreender os requisitos do cliente e, a partir disso, desenvolver melhor definição de requisitos para o sistema. O protótipo se concentra na experimentação dos requisitos mal compreendidos do cliente, mas é posteriormente descartado.

### IMPORTANTE

QUANDO UMA QUESTÃO NÃO ESPECIFICA O TIPO DE PROTOTIPAÇÃO, GERALMENTE SE TRATA DE PROTOTIPAÇÃO THROW-AWAY/DESCARTÁVEL E, NÃO, EVOLUCIONÁRIA/EXPLORATÓRIA. SOMMERVILLE DECLARA: "O USO O TERMO PROTOTIPAÇÃO NO SENTIDO DE PROCESSO ITERATIVO DE DESENVOLVIMENTO DE UM SISTEMA EXPERIMENTAL QUE NÃO É DESTINADO À DISPONIBILIZAÇÃO AO CLIENTE".

Um protótipo de software pode ser utilizado em um processo de desenvolvimento de diversas maneiras! Veremos três modos possíveis de utilização:

a) **No processo de engenharia de requisitos,** um protótipo pode ajudar na descoberta e validação dos requisitos do sistema.

b) **Processo de projeto do sistema**, um protótipo pode ser usado para explorar soluções específicas de software e apoiar o projeto de interface com o usuário.

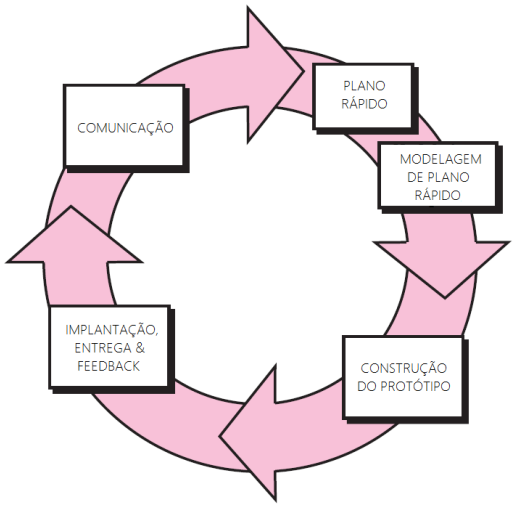
c) **No processo de teste,** um protótipo pode ser usado para realizar testes completos com o sistema que será entregue para o cliente.

A metodologia de Prototipagem Evolucionária é uma abordagem que visualiza o desenvolvimento de concepções do sistema conforme o andamento do projeto até chegar ao resultado final. **Esta metodologia baseia-se na utilização de prototipagem visual ou modelos do sistema final**. Estes modelos podem ser simples desenhos ou imagens do sistema.

**Protótipos permitem que os usuários introduzam novas ideias para os requisitos e encontrem pontos fortes e fracos no software**. Ademais, protótipos podem revelar erros e omissões nos requisitos propostos, além de reduzirem o tempo e esforço de desenvolvimento, treinamento e documentação o sistema. *Professor, só há vantagens?* Não, há desvantagens também! Olha só...

**Análise insuficiente, devido a desenvolvedores que se focam só no protótipo e esquecem de analisar o problema global**; usuários confundem protótipo com o sistema final, sendo que serão descartados posteriormente; documentação pode ser prejudicada; falta de visibilidade do progresso quando o sistema evolui, mas nunca termina; tempo excessivo para desenvolver o protótipo; etc.

Como apresenta a imagem ao lado, o paradigma de prototipação começa com a comunicação. Encontra-se com as partes interessadas para definir os objetivos gerais do software, **identificar os requisitos conhecidos e definir áreas em que uma definição mais profunda é obrigatória**. Uma iteração de prototipação é planejada rapidamente e, então, ocorre a modelagem dessa iteração. **Um plano rápido se foca na representação daqueles aspectos do software que serão visíveis aos usuários finais e leva à construção do protótipo.** O protótipo é implantado e avaliado pelas partes interessadas, que fornecem feedbacks que serão utilizados para refinar os requisitos. A



iteração ocorre à medida que o protótipo é ajustado para satisfazer as necessidades dos diversos stakeholders.

Ao mesmo tempo, ele ajuda a entender melhor o que precisa ser feito! Idealmente, o protótipo serve como um mecanismo para identificar requisitos de software. **Se um protótipo será construído, pode-se utilizar partes existentes de programas ou aplicar ferramentas que permitem que programas sejam gerados rapidamente.** *E o que fazer com o protótipo construído?*

**Na maioria dos casos, o protótipo é inviável de ser utilizado, por ser muito lento e/ou muito grande e/ou difícil de utilizar**. Em geral, os protótipos são descartados assim que os objetivos de levantamento de requisitos são alcançados. No entanto, alguns preferem refiná-los iterativamente até evoluir ao sistema final requisitado pelo usuário. *Captaram?*

Em suma, o que vocês precisam saber é que o Modelo de Prototipação ou Prototipagem se baseia na ideia de construção de um protótipo (esboços, desenhos ou imagens) que auxiliem a construção do produto de duas maneiras: ou para levantar os requisitos do sistema junto aos usuários e depois ser descartado; ou para ser refinado, refinado e refinado até chegar ao sistema final desejado pelos usuários.

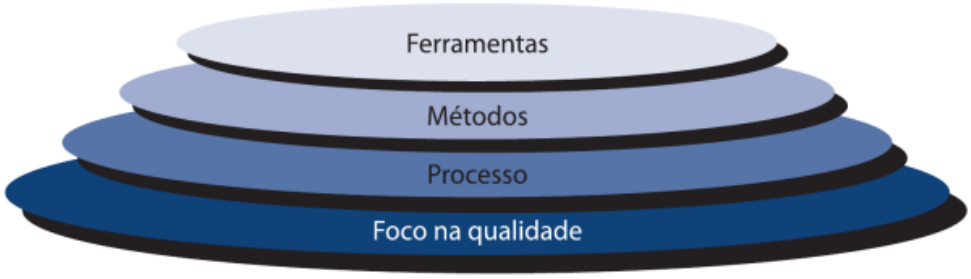
# 2 – EXERCÍCIOS COMENTADOS

ENGENHARIA DE SOFTWARE

1. **(CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Analista Judiciário - Tecnologia da Informação)** A engenharia de software engloba processos, métodos e ferramentas. Um de seus focos é a produção de software de alta qualidade a custos adequados.

**Comentários:**

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade. Basta visualizar a imagem para responder à questão! Ela não menciona foco na qualidade, mas não restringe também somente a processos, métodos e ferramentas.




**Gabarito:** Correto

---

2. **(FCC - 2012 - TST - Analista Judiciário - Análise de Sistemas)** A Engenharia de Software:

a) é uma área da computação que visa abordar de modo sistemático as questões técnicas e não técnicas no projeto, implantação, operação e manutenção no desenvolvimento de um software.

b) consiste em uma disciplina da computação que aborda assuntos relacionados a técnicas para a otimização de algoritmos e elaboração de ambientes de desenvolvimento.

c) trata-se de um ramo da TI que discute os aspectos técnicos e empíricos nos processos de desenvolvimento de sistemas, tal como a definição de artefatos para a modelagem ágil.

d) envolve um conjunto de itens que abordam os aspectos de análise de mercado, concepção e projeto de software, sendo independente da engenharia de um sistema.

e) agrupa as melhores práticas para o concepção, projeto, operação e manutenção de artefatos que suportam a execução de programas de computador, tais como as técnicas de armazenamento e as estruturas em memória principal.

**Comentários:**

De acordo com Pressman: "*A Engenharia de Software ocorre como consequência de um processo chamado Engenharia de Sistemas. Em vez de se concentrar somente no software, a engenharia de sistemas focaliza diversos elementos, analisando, projetando, e os organizando em um sistema que pode ser um produto, um serviço ou uma tecnologia para transformação da informação ou controle*". Logo, vamos aos julgamentos:

(a) Perfeito, observem as palavras-chave: modo sistemático; questões técnicas e não técnicas; projeto, implantação, operação e manutenção de desenvolvimento de software.

(b) *Técnicas para otimização de algoritmos e elaboração de ambientes de desenvolvimento?* Não, isso não é Engenharia de Software.

(c) Pessoal, discordo do gabarito! Certa vez, um aluno me disse que talvez fosse porque aspectos empíricos são mais voltados para metodologias ágeis. Sim, é verdade! No entanto, a engenharia de software trata também de metodologias ágeis. Se alguém encontrar o erro, avise :-]

(d) A Análise de Mercado serve mais como uma técnica para Análise de Interfaces, mas pode ser vista como um dos aspectos que envolvem a Engenharia de Software. Pressman afirma que: *"A Análise de Mercado pode ser inestimável na definição de segmentos de mercado e no entendimento de como cada segmento poderia usar o software de modos sutilmente diferentes"*. De todo modo, a questão está errada porque a Engenharia de Software depende da Engenharia de Sistema (como é mostrado acima).

(e) *Suportam a execução?* Não, suportam o desenvolvimento de programas de computador.

**Gabarito:** Letra A

---

3. **(FCC - 2012 - TRT - 6ª Região (PE) - Técnico Judiciário - Tecnologia da Informação)** Considere: *é uma disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Seu principal objetivo é fornecer uma estrutura metodológica para a construção de software com alta qualidade.* A definição refere-se:

a) ao ciclo de vida do software.
b) à programação orientada a objetos.
c) à análise de sistemas.
d) à engenharia de requisitos.
e) à engenharia de software.

**Comentários:**

Engenharia de Software é uma disciplina de engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, após sua entrada em produção. A meta principal da Engenharia de Software é desenvolver sistemas de software com boa relação custo-benefício. Essa é a pura definição de Engenharia de Software!

**Gabarito:** Letra E

---

4. **(CESPE - 2011 - MEC - Gerente de Projetos)** A engenharia de software, disciplina relacionada aos aspectos da produção de software, abrange somente os processos técnicos do desenvolvimento de software.

**Comentários:**

Sommerville afirma que: "*A engenharia de software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software*". Logo, não são apenas processos técnicos!

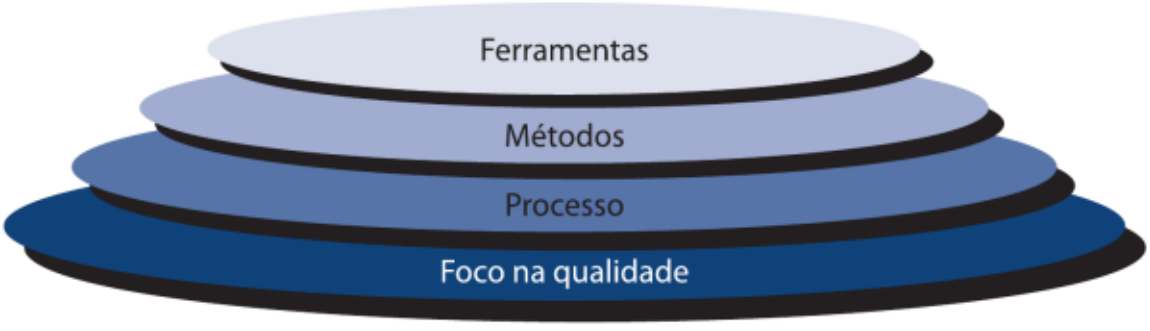
**Gabarito:** Errado

---

5. **(FGV - 2010 - BADESC - Analista de Sistemas - Desenvolvimento de Sistemas)** De acordo com Pressman, a engenharia de software é baseada em camadas, com foco na qualidade. Essas camadas são:

a) métodos, processo e teste.
b) ferramentas, métodos e processo.
c) métodos, construção, teste e implantação.
d) planejamento, modelagem, construção, validação e implantação.
e) comunicação, planejamento, modelagem, construção e implantação.

**Comentários:**



Bastava lembrar da imagem para responder à questão!

**Gabarito:** Letra B

---

6. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação)** Os princípios de engenharia de software definem a necessidade de formalidades para reduzir inconsistências e a decomposição para lidar com a complexidade.

**Comentários:**

A Engenharia de Software possui alguns princípios, dentre os quais: **Formalidade**, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; **Decomposição**, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica. Logo, são princípios: Formalidade e Decomposição.

**Gabarito:** Correto

---

7. **(FCC - 2010 - TRE-AM - Analista Judiciário - Tecnologia da Informação)** A Engenharia de Software:

   a) não tem como método a abordagem estruturada para o desenvolvimento de software, pois baseia-se exclusivamente nos modelos de software, notações, regras e técnicas de desenvolvimento.

   b) se confunde com a Ciência da Computação quando ambas tratam do desenvolvimento de teorias, fundamentações e práticas de desenvolvimento de software.

   c) tendo como foco apenas o tratamento dos aspectos de construção de software, subsidia a Engenharia de Sistemas no tratamento dos sistemas baseados em computadores, incluindo hardware e software.

   d) tem como foco principal estabelecer uma abordagem sistemática de desenvolvimento, através de ferramentas e técnicas apropriadas, dependendo do problema a ser abordado, considerando restrições e recursos disponíveis.

   e) segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

**Comentários:**

(a) Errado. Pelo contrário, ela se baseia em uma abordagem estruturada e sistemática! A IEEE define engenharia de software como a aplicação de uma abordagem sistemática, disciplinada e quantificável de desenvolvimento, operação e manutenção de software. Já Friedrich Bauer conceitua como a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais.

(b) Errado. Na verdade, Engenharia de Software é uma disciplina da Ciência da Computação. A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade.

(c) *Apenas o tratamento dos aspectos de construção de software? Só construção?* Não! (d) Correto, é exatamente isso! (e) *Composição?* Não, Decomposição! Divide-se o problema em partes para que cada uma possa ser resolvida de uma forma mais específica. Além disso, a abstração ignora detalhes!

**Gabarito:** Letra D

---

8. **(FCC - 2011 - INFRAERO - Analista de Sistemas - Gestão de TI)** Em relação à Engenharia de Software, é INCORRETO afirmar:

   a) O design de software, ao descrever os diversos aspectos que estarão presentes no sistema quando construído, permite que se faça a avaliação prévia para garantir que ele alcance os objetivos propostos pelos interessados.

   b) A representação de um design de software mais simples para representar apenas as suas características essenciais busca atender ao princípio da abstração.

   c) Iniciar a entrevista para obtenção dos requisitos de software com perguntas mais genéricas e finalizar com perguntas mais específicas sobre o sistema é o que caracteriza a técnica de entrevista estruturada em funil.

   d) No contexto de levantamento de requisitos, funcionalidade é um dos aspectos que deve ser levado em conta na abordagem dos requisitos funcionais.

   e) A representação é a linguagem do design, cujo único propósito é descrever um sistema de software que seja possível construir.

**Comentários:**

Galera, descrever um sistema de software que seja possível construir não é o único, mas um dos objetivos da representação. Ela auxilia a comunicação entre as partes interessadas e serve também como documentação.

**Gabarito:** Letra E

---

**9. (FCC – 2009 – AFR/SP - Analista de Sistemas)** A engenharia de software está inserida no contexto:

a) das engenharias de sistemas, de processo e de produto.
b) da engenharia de sistemas, apenas.
c) das engenharias de processo e de produto, apenas.
d) das engenharias de sistemas e de processo, apenas.
e) das engenharias de sistemas e de produto, apenas.

**Comentários:**

A engenharia de sistemas está preocupada com todos os aspectos do desenvolvimento de sistemas computacionais, **incluindo engenharia de hardware, engenharia de software e engenharia de processos**. Percebam, então, que a Engenharia de Sistemas está em um contexto com várias outras engenharias.

**Gabarito:** Letra A

---

**10. (CESPE – 2015 – STJ – Analista de Sistemas)** Embora os engenheiros de software geralmente utilizem uma abordagem sistemática, a abordagem criativa e menos formal pode ser eficiente em algumas circunstâncias, como, por exemplo, para o desenvolvimento de sistemas web, que requerem uma mistura de habilidades de software e de projeto.

**Comentários:**

Galera, basta pensar nas metodologias ágeis! *O que são metodologias ágeis?* São metodologias menos formais e são mais adequadas em determinadas circunstâncias. Notem que isso não vai de encontro ao princípio da formalidade, na medida em que a questão deixa claro que se trata de uma abordagem "*menos formal*" e, não, "*informal*".

Além disso, isso consta também do livro do Sommerville: *"No entanto, engenharia tem tudo a ver com selecionar o método mais adequado para um conjunto de circunstâncias, então uma abordagem mais criativa e menos formal pode ser eficiente em algumas circunstâncias. Desenvolvimento menos formal particularmente adequado para o desenvolvimento de sistemas Web, que requerem uma mistura de habilidades de software e de projeto".*

**Gabarito:** Correto

---

11. **(CESPE – 2015 – STJ – Analista de Sistemas)** O foco da engenharia de software inclui especificação do sistema, desenvolvimento de hardware, elaboração do projeto de componentes de hardware e software, definição dos processos e implantação do sistema.

**Comentários:**

Conforme vimos em aula, pode até tratar eventualmente de alguns aspectos de hardware; mas desenvolvimento de hardware, não.

**Gabarito:** Errado

---

12. **(FUNIVERSA – 2009 – IPHAN – Analista de Sistemas)** A Engenharia de Software resume-se em um conjunto de técnicas utilizadas para o desenvolvimento e manutenção de sistemas computadorizados, visando produzir e manter softwares de forma padronizada e com qualidade. Ela obedece a alguns princípios como (1) Formalidade, (2) Abstração, (3) Decomposição, (4) Generalização e (5) Flexibilização. Assinale a alternativa que apresenta conceito correto sobre os princípios da Engenharia de Software.

a) A flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

b) A formalidade é a maneira usada para resolver um problema, de forma genérica, com o intuito de poder reaproveitar essa solução em outras situações semelhantes.

c) A generalização preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

d) Pelo princípio da decomposição, o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva.

e) A abstração é a técnica de se dividir o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica.

**Comentários:**

A Engenharia de Software possui alguns princípios, tais como: **Formalidade**, em que o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva; **Abstração**, preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

Há a **Decomposição**, em que se divide o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica; **Generalização,** maneira usada para resolver um problema, de forma genérica, com o intuito de reaproveitar essa solução em outras situações; **Flexibilização** é o processo que permite que o software possa ser alterado, sem causar problemas

para sua execução. Logo, a flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

**Gabarito:** Letra A

---

**13. (CESPE – 2013 – TCE/RO – Analista de Sistemas)** Engenharia de software não está relacionada somente aos processos técnicos de desenvolvimento de softwares, mas também a atividades como gerenciamento de projeto e desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de softwares.

**Comentários:**

De acordo com Sommerville: *"A engenharia de software não está relacionada apenas com os processos técnicos de desenvolvimento de software, mas também com atividades como o gerenciamento de projeto de software e o desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de software".* Logo, a questão está perfeita!

**Gabarito:** Correto

---

**14. (CESPE – 2010 – DETRAN/ES – Analista de Sistemas)** Segundo princípio da engenharia de software, os vários artefatos produzidos ao longo do seu ciclo de vida apresentam, de forma geral, nível de abstração cada vez menor.

**Comentários:**

Galera, vamos pensar um pouco! A abstração é a subtração de detalhes – focar-se no que é mais relevante e ignorar detalhes menos relevantes. No início do ciclo de vida, os artefatos são bastante abstratos. Temos, por exemplo, uma modelagem do negócio, um documento de requisitos funcionais, a abstração vai diminuindo e temos a modelagem visual de análise, depois a modelagem do projeto, até chegar ao código-fonte e ao executável do software. Nesse ponto, temos o menor nível de abstração (com muitos detalhes!). *Compreenderam?*

**Gabarito:** Correto

---

**15. (CESPE – 2010 – TRE/BA – Analista de Sistemas)** Entre os desafios enfrentados pela engenharia de software estão lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto a prazos de entrega reduzidos.

**Comentários:**

Bem, essa questão é bastante intuitiva. De fato, a engenharia de software tem que lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto aos (curtos) prazos de entrega.

**Gabarito:** Correto

**16. (FCC – 2010 – DPE/SP – Analista de Sistemas)** A Engenharia de Software:

I. não visa o desenvolvimento de teorias e fundamentações, preocupando-se unicamente com as práticas de desenvolvimento de software.

II. tem como foco o tratamento dos aspectos de desenvolvimento de software, abstraindo-se dos sistemas baseados em computadores, incluindo hardware e software.

III. tem como métodos as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

IV. segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

É correto o que se afirma em:

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

**Comentários:**

(I) Errado, Sommerville diz: "*Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software*". No entanto, ele não diz que a engenharia de software se preocupa **unicamente** com as práticas de desenvolvimento de software.

(II) Errado, Pressman diz: *"System engineering is concerned with all aspects of computer-based systems development including hardware, software, and process engineering"* – a questão trata da Engenharia de Sistemas.

(III) Correto, de fato ela tem como métodos as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

(IV) Errado, o princípio da abstração ignora os detalhes; e o princípio da composição não existe – o que existe é o princípio da decomposição. E ele divide o problema em partes menores.

Em suma, nenhuma das opções nos atende! *Vocês sabem qual opção a banca marcou como correta?* A Letra D!!! *E ela voltou atrás com os recursos?* Não!!! Pois é, galera! Acostumem-se com isso :(

**Gabarito:** Letra D

---

**17. (CESPE – 2013 – TCE/RO – Analista de Sistemas)** Assim como a Engenharia de Software, existe também na área de informática a chamada Ciência da Computação. Assinale a alternativa que melhor apresenta a diferença entre Engenharia de Software e Ciência da Computação.

a) A Ciência da Computação tem como objetivo o desenvolvimento de teorias e fundamentações. Já a Engenharia de Software se preocupa com as práticas de desenvolvimento de software.

b) A Engenharia de Software trata da criação dos sistemas de computação (softwares) enquanto a Ciência da Computação está ligada ao desenvolvimento e criação de componentes de hardware.

c) A Engenharia de Software trata dos sistemas com base em computadores, que inclui hardware e software, e a Ciência da Computação trata apenas dos aspectos de desenvolvimento de sistemas.

d) A Ciência da Computação trata dos sistemas com base em computadores, que inclui hardware e software, e a Engenharia de Software trata apenas dos aspectos de desenvolvimento de sistemas.

e) A Ciência da Computação destina-se ao estudo e solução para problemas genéricos das áreas de rede e banco de dados e a Engenharia de Software restringe- se ao desenvolvimento de sistemas.

**Comentários:**

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade.

Essa questão parece contradizer o que diz Sommerville, mas não! A Engenharia de Software coloca em prática a teoria e fundamentação trazida pela Ciência da Computação. A Engenharia de Software aplica a teoria, mas - grosso modo - ela não tem a finalidade principal de elaborá-la; a finalidade principal é de colocá-la em prática. Já a Ciência da Computação é exatamente o contrário. Enfim, a primeira opção foi retirada literalmente do Sommerville (Pág. 5, 8ª Ed.).

**Gabarito:** Letra A

---

18. **(CESPE – 2009 – ANAC – Analista de Sistemas)** O termo engenharia pretende indicar que o desenvolvimento de software submete-se a leis similares às que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos.

**Comentários:**

Perfeito! A Engenharia busca os princípios mais consolidados de outras engenharias mais tradicionais – engenharia mecânica, civil, elétrica, etc.

**Gabarito:** Correto

---

19. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – B)** A engenharia de software refere-se ao estudo das teorias e fundamentos da computação, ficando o desenvolvimento de software a cargo da ciência da computação.

**Comentários:**

A Engenharia de Software tem por objetivos a aplicação de teoria, modelos, formalismos, técnicas e ferramentas da ciência da computação e áreas afins para a desenvolvimento sistemático de software. Associado ao desenvolvimento, é preciso também aplicar processos, métodos e ferramentas sendo que a pedra fundamental que sustenta a engenharia de software é a qualidade. Conforme vimos em aula, não se trata do estudo – trata-se da aplicação. Além disso, o desenvolvimento também fica a cargo da engenharia de software. Em geral, a ciência da computação trata da teoria e a engenharia de software trata da prática.

**Gabarito:** Errado

---

20. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – E)** O conceito de software se restringe ao desenvolvimento do código em determinada linguagem e seu armazenamento em arquivos.

**Comentários:**

Em uma visão restritiva, muitas pessoas costumam associar o termo software aos programas de computador. Software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente. O software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente.

**Gabarito:** Errado

---

**21. (CESPE – 2009 – TCE/TO – Analista de Sistemas - A)** Quanto maior e mais complexo o projeto de software, mais simples deve ser o modelo de processo a ser adotado.

**Comentários:**

Galera, não existe essa relação! Em geral, quanto mais complexo o projeto mais complexo o modelo. No entanto, isso também não é uma regra. Hoje em dia, existem projetos megacomplexos feitos utilizando metodologias ágeis, por exemplo.

**Gabarito:** Errado

---

**22. (CESPE – 2009 – TCE/TO – Analista de Sistemas - B)** O modelo de ciclo de vida do software serve para delimitar o alvo do software. Nessa visão, não são consideradas as atividades necessárias e o relacionamento entre elas.

**Comentários:**

A escolha de um modelo de ciclo de vida (para concursos, sinônimo de modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas. Logo, o alvo do software serve para delimitar o modelo de ciclo de vida a ser escolhido. Primeiro, define-se o alvo do software para, só então, escolher o modelo de ciclo de vida mais adequado. Ademais, são consideradas as atividades necessárias e o relacionamento entre elas.

**Gabarito:** Errado

---

**23. (CESPE – 2009 – TCE/TO – Analista de Sistemas - C)** A escolha do modelo do ciclo de vida não depende de características específicas do projeto, pois o melhor modelo é sempre o mais usado pela equipe do projeto.

**Comentários:**

Um processo de software não pode ser definido de forma universal. Para ser eficaz e conduzir à construção de produtos de boa qualidade, um processo deve ser adequado às especificidades do projeto em questão. Deste modo, processos devem ser definidos caso a caso, considerando-se diversos aspectos específicos do projeto em questão. Dessa forma, não faz o menor sentido! A escolha depende das características do projeto; além disso, não existe um "*melhor*" modelo.

**Gabarito:** Errado

---

**24. (ESAF - 2012 – CGU – Analista de Sistemas)** A escolha de um modelo é fortemente dependente das características do projeto. Os principais modelos de ciclo de vida podem ser agrupados em três categorias principais:

a) sequenciais, cascata e evolutivos.
b) sequenciais, incrementais e ágeis.
c) sequenciais, incrementais e evolutivos.
d) sequenciais, ágeis e cascata.
e) cascata, ágeis e evolutivos.

**Comentários:**

A maioria dos processos de software é baseada em três modelos gerais: modelo em cascata; desenvolvimento iterativo e engenharia de software baseada em componentes. Isso entra em contradição com o que dizem outros autores, isto é, os principais modelos podem ser agrupados em três categorias: modelos sequenciais, modelos incrementais e modelos evolutivo.

**Gabarito:** Letra C

---

**25. (CESPE – 2011 – MEC – Analista de Sistemas)** O processo de desenvolvimento de software é uma caracterização descritiva ou prescritiva de como um produto de software deve ser desenvolvido.

**Comentários:**

Metodologia de Desenvolvimento de Software É uma caracterização prescritiva ou descritiva de como um produto de software deve ser desenvolvido, isto é, define o quê, como e quando fazer algo – um exemplo clássico é o RUP! Segundo Pressman, um modelo prescritivo consiste em um conjunto específico de atividades de arcabouço, como – por exemplo – a NBR ISSO/IEC 12207, que estabelece uma estrutura comum para os processos de ciclo de vida de software. Já um modelo descritivo apresenta o processo em detalhes, é como se fosse um guia para o desenvolvimento de software. Ambas as formas podem ser utilizadas.

**Gabarito:** Correto

---

**26. (CESPE – 2013 – TRT/10ª – Analista de Sistemas)** As atividades fundamentais relacionadas ao processo de construção de um software incluem a especificação, o desenvolvimento, a validação e a evolução do software.

**Comentários:**

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software. Logo, a questão está corretíssima!

**Gabarito:** Correto

---

**27. (CESPE – 2010 – TRE/BA – Analista de Sistemas)** Um modelo de processo de software consiste em uma representação complexa de um processo de software, apresentada a partir de uma perspectiva genérica.

**Comentários:**

Um modelo de processo é uma representação abstrata de um esqueleto de processo, incluindo tipicamente algumas atividades principais, a ordem de precedência entre elas e, opcionalmente, artefatos requeridos e produzidos. Em geral, um modelo de processo descreve uma filosofia de organização de atividades, estruturando-as em fases e definindo como essas fases estão relacionadas. Logo, um modelo de processo de software ou modelo de ciclo de vida trata da representação abstrata/simplificada de um processo de software, apresentada a partir de uma perspectiva genérica.

**Gabarito:** Errado

---

**28. (CESPE – 2011 – MEC – Analista de Sistemas)** Atividades comuns a todos os processos de software incluem a especificação, o projeto, a implementação e a validação.

**Comentários:**

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Notem que o examinador dividiu a atividade de Desenvolvimento em Projeto e Implementação, mas não invalida a questão. *Professor, ele não falou sobre a evolução!* Sim, mas a questão apenas afirma que essas são atividades comuns e, não, que são as únicas.

**Gabarito:** Correto

---

**29. (CESPE – 2015 – STJ – Analista de Sistemas)** As principais atividades de engenharia de software são especificação, desenvolvimento, validação e evolução.

**Comentários:**

Mais uma vez: Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são

elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software.

Conforme vimos em aula, a questão peca um pouco ao dizer que são atividades da engenharia de software. O ideal seria dizer que são as atividades principais do processo de software. Não sei se entraram com recurso e a banca recusou e se não entraram com recurso. Percebam também que é a terceira vez esse tipo de questão cai em prova.

**Gabarito:** Correto

---

**30. (FCC – 2012 – MPE/AP – Analista de Sistemas)** Um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software. Existem muitos processos de software diferentes, mas todos devem incluir quatro atividades fundamentais: especificação, projeto e implementação, validação e:

a) teste
b) evolução.
c) prototipação.
d) entrega.
e) modelagem.

**Comentários:**

As atividades são Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software. Logo, trata-se da evolução!

**Gabarito:** Letra B

---

**31. (CESPE – 2010 – EMBASA – Analista de Sistemas)** Ciclo de vida de um software resume-se em eventos utilizados para definir o status de um projeto.

**Comentários:**

O ciclo de vida de software é um conjunto de estágios (e, não, eventos) utilizados para definir o status de um software (e, não, projeto). Sommerville afirma, por exemplo, que um ciclo de vida é composto por estágios que demonstram atividades fundamentais de desenvolvimento. Questão errada!

**Gabarito:** Errado

---

**32. (CESPE – 2016 – TCE/PR – Analista de Sistemas)** As fases do ciclo de vida de um software são:

a) concepção, desenvolvimento, entrega e encerramento.
b) iniciação, elaboração, construção e manutenção.

c) escopo, estimativas, projeto e processo e gerência de riscos.
d) análise, desenvolvimento, teste, empacotamento e entrega.
e) planejamento, análise e especificação de requisitos, projeto, implementação, testes, entrega e implantação, operação e manutenção.

**Comentários:**

Alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção. Logo, a questão trata da última opção.

**Gabarito:** Letra E

---

**33. (MPE/RS – 2012 – MPE/RS – Analista de Sistemas)** O ciclo de vida básico de um software compreende:

a) a implementação, a implantação e o teste.
b) a análise, a segurança e o controle de usuários.
c) a implementação, a análise e o teste.
d) a implementação, a validação e as vendas.
e) a análise, o projeto, a implementação e o teste.

**Comentários:**

Alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção. Aqui temos que escolher a mais correta, na medida em que a primeira, terceira e última opções estão corretas. Nesse sentido, a última opção é a mais correta!

**Gabarito:** Letra E

---

**34. (CESGRANRIO – 2011 – PETROBRÁS – Analista de Sistemas)** A especificação de uma Metodologia de Desenvolvimento de Sistemas tem como pré-requisito indispensável, em relação ao que será adotado no processo de desenvolvimento, a definição do:

a) Engenheiro Responsável pelo Projeto
b) Documento de Controle de Sistemas
c) Software para Desenvolvimento
d) Ciclo de Vida do Software
e) Bloco de Atividades

**Comentários:**

A escolha de um modelo de ciclo de vida (para concursos, sinônimo de modelo de processo) é o ponto de partida para a definição de um processo de desenvolvimento de software. Um modelo de ciclo de vida, geralmente, organiza as macro-atividades básicas do processo, estabelecendo precedência e dependência entre as mesmas. Bem, essa questão foi bem escrita (finalmente)! Percebam que ele diferencia metodologia de desenvolvimento de software do ciclo de vida de software. Por exemplo: primeiro, eu defino que eu vou utilizar um ciclo de vida evolutivo e depois eu decido que eu vou utilizar a metodologia de desenvolvimento em espiral.

**Gabarito:** Letra D

---

**35. (CESPE – 2008 – INPE – Analista de Sistemas)** O ciclo de vida do software tem início na fase de projeto.

**Comentários:**

Mais uma vez: alguns autores afirmam que os modelos de ciclo de vida básicos, de maneira geral, contemplam pelo menos as fases de: Planejamento; Análise e Especificação de Requisitos; Projeto; Implementação; Testes; Entrega e Implantação; Operação; e Manutenção. Logo, ele não começa com projeto! *Como você começa com projeto sem sequer levantar os requisitos?* Era possível responder usando só lógica!

**Gabarito:** Errado

---

**36. (CESPE – 2013 – TRT/10 – Analista de Sistemas)** O ciclo de vida de um software, entre outras características, está relacionado aos estágios de concepção, projeto, criação e implementação.

**Comentários:**

Essa questão é péssima! Ela foi retirada do livro *Sistemas de Informação: Uma Abordagem Gerencial* (Steven R. Gordon, Judith R. Gordon). Nesse livro, os autores de fato tratam o ciclo de vida de software como concepção, projeto, criação e implementação. Essa é uma daquelas quase impossível de acertar! Esse não é um autor consagrado de Engenharia de Software.

**Gabarito:** Correto

---

**37. (CESPE – 2011 – TJ/ES – Analista de Sistemas)** Entre as etapas do ciclo de vida de software, as menos importantes incluem a garantia da qualidade, o projeto e o estudo de viabilidade. As demais atividades do ciclo, como a implementação e os testes, requerem maior dedicação da equipe e são essenciais.

**Comentários:**

Para a definição completa do processo, cada atividade descrita no modelo de ciclo de vida deve ser decomposta e para suas subatividades, devem ser associados métodos, técnicas, ferramentas e critérios de qualidade, entre outros, formando uma base sólida para o desenvolvimento. Adicionalmente, outras atividades, tipicamente de cunho gerencial, devem ser definidas, como controle e garantia da qualidade. Não existe uma hierarquia de etapas mais importantes e menos importantes!

**Gabarito:** Errado

---

**38. (CESPE - 2016 – TCE/PR – Analista de Sistemas – A)** A engenharia de software está relacionada aos diversos aspectos de produção de software e inclui as atividades de especificação, desenvolvimento, validação e evolução de software.

**Comentários:**

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software. Dessa forma, a questão está correta.

**Gabarito:** Correto

---

**39. (CESPE - 2016 – TCE/PR – Analista de Sistemas – D)** Um processo de software é composto por quatro atividades fundamentais: iniciação, desenvolvimento, entrega e encerramento.

**Comentários:**

Sommerville afirma que um processo de software é um conjunto de atividades e resultados associados que produz um produto de software. De acordo com ele, existem quatro atividades fundamentais de processo, que são comuns a todos os processos de software – são elas: Especificação de Software; Desenvolvimento de Software (Projeto e Implementação); Validação de Software; e Evolução de Software. Dessa forma, a questão está incorreta.

**Gabarito:** Errado

---

**40. (CESPE - 2016 – TCE/PR – Analista de Sistemas – B)** A metodologia de processos genérica para a engenharia de software é composta de seis etapas: comunicação, planejamento, modelagem, construção, emprego e aceitação.

**Comentários:**



Conforme vimos em aula, essas etapas parecem as fases do ciclo de vida de acordo com o Pressman, que são: Comunicação, Planejamento, Modelagem, Construção e Implantação. Percebam que ele troca Implantação por Emprego e Aceitação.

**Gabarito:** Errado

---

**41. (INSTITUTO CIDADE – 2012 – TCM/GO – Analista de Sistemas)** De acordo com a engenharia de software, como todo produto industrial, o software possui um ciclo de vida. Cada fase do ciclo de vida possui divisões e subdivisões. Em qual fase avaliamos a necessidade de evolução dos softwares em funcionamento para novas plataformas operacionais ou para a incorporação de novos requisitos?

a) Fase de operação;
b) Fase de retirada;
c) Fase de definição;
d) Fase de design.
e) Fase de desenvolvimento;

**Comentários:**


A fase retirada é um grande desafio para os tempos atuais. Diversos softwares que estão em funcionamento em empresas possuem excelente níveis de confiabilidade e de correção. No entanto, eles precisam evoluir para novas plataformas operacionais ou para a incorporação de novos requisitos.

A retirada desses softwares legados em uma empresa é sempre uma decisão difícil: *como abrir mão daquilo que é confiável e ao qual os funcionários estão acostumados, após anos de treinamento e utilização?* Portanto, processos de reengenharia podem ser aplicados para viabilizar a transição ou migração de um software legado para um novo software de forma a proporcionar uma retirada mais suave.

A avaliação da necessidade de evolução do software em funcionamento para novas plataformas operacionais ou para a incorporação de novos requisitos realmente ocorrem na fase de retirada.

**Gabarito:** Letra B

---

**42. (CESPE - 2010 – DETRAN/ES – Analista de Sistemas)** Quando um aplicativo de software desenvolvido em uma organização atinge, no fim do seu ciclo de vida, a fase denominada aposentadoria, descontinuação ou fim de vida, todos os dados por ele manipulados podem ser descartados.

**Comentários:**



Essa questão é um absurdo! Observem que ela afirma "*podem ser descartados*". Ora, é evidente que podem ser descartados. No entanto, o gabarito oficial é errado!

**Gabarito:** Errado

---

**MODELO EM CASCATA**

**43. (FCC - 2012 - TJ-RJ - Analista Judiciário - Análise de Sistemas - E)** Dos diferentes modelos para o ciclo de vida de desenvolvimento de um software é correto afirmar que o modelo em cascata é o mais recente e complexo.

**Comentários:**

Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento. *Mais recente?* Não, muito antigo! *Complexo?* Não, possui um encadeamento simples de fases.

**Gabarito:** Errado

---

**44. (FCC - 2009 - SEFAZ-SP - Agente Fiscal de Rendas - Tecnologia da Informação - Prova 3 - B)** O processo de engenharia de software denominado ciclo de vida clássico refere-se ao modelo incremental.

**Comentários:**

Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento. Logo, modelo clássico se refere a modelo em cascata, sequencial, linear, tradicional, waterfall, rígido ou monolítico.

**Gabarito:** Errado

---

**45. (CESPE – 2009 – INMETRO – Analista de Sistemas)** Em uma empresa que tenha adotado um processo de desenvolvimento de software em cascata, falhas no levantamento de requisitos têm maior possibilidade de gerar grandes prejuízos do que naquelas que tenham adotado desenvolvimento evolucionário.

**Comentários:**

| VANTAGENS | DESVANTAGENS |
|---|---|
| É simples de entender e fácil de aplicar, facilitando o planejamento. | Divisão inflexível do projeto em estágios distintos. |
| Fixa pontos específicos para a entrega de artefatos. | Dificuldade em incorporar mudanças de requisitos. |
| Funciona bem para equipes tecnicamente fracas. | Clientes só visualizam resultados próximos ao final do projeto. |
| É fácil de gerenciar, devido a sua rigidez. | Atrasa a redução de riscos. |

| | |
|---|---|
| Realiza documentação extensa por cada fase ou estágio. | Apenas a fase final produz um artefato de software entregável. |
| Possibilita boa aderência a outros modelos de processo. | Cliente deve saber todos os requisitos no início do projeto. |
| Funciona bem com projetos pequenos e com requisitos bem conhecidos. | Modelo inicial (Royce) não permitia feedback entre as fases do projeto. |
| | Pressupõe que os requisitos ficarão estáveis ao longo do tempo. |
| | Não funciona bem com projetos complexos e OO, apesar de compatível. |

O Modelo em Cascata, de fato, não reage bem a mudanças. Já Modelo evolucionário é mais adaptável a mudanças por se utilizar de iterações.

**Gabarito:** Correto

---

**46. (CESPE – 2011 – MEC – Analista de Sistemas)** O modelo Waterfall tem a vantagem de facilitar a realização de mudanças sem a necessidade de retrabalho em fases já completadas.

**Comentários:**

| VANTAGENS | DESVANTAGENS |
|---|---|
| É simples de entender e fácil de aplicar, facilitando o planejamento. | Divisão inflexível do projeto em estágios distintos. |
| Fixa pontos específicos para a entrega de artefatos. | Dificuldade em incorporar mudanças de requisitos. |
| Funciona bem para equipes tecnicamente fracas. | Clientes só visualizam resultados próximos ao final do projeto. |
| É fácil de gerenciar, devido a sua rigidez. | Atrasa a redução de riscos. |
| Realiza documentação extensa por cada fase ou estágio. | Apenas a fase final produz um artefato de software entregável. |
| Possibilita boa aderência a outros modelos de processo. | Cliente deve saber todos os requisitos no início do projeto. |
| Funciona bem com projetos pequenos e com requisitos bem conhecidos. | Modelo inicial (Royce) não permitia feedback entre as fases do projeto. |
| | Pressupõe que os requisitos ficarão estáveis ao longo do tempo. |
| | Não funciona bem com projetos complexos e OO, apesar de compatível. |

Pelo contrário, há dificuldade de lidar com requisitos voláteis, tendo em vista que dependendo do erro, é necessário refazê-lo desde seu início.

**Gabarito:** Errado

---

**47. (CESPE – 2008 – TST – Analista de Sistemas)** No modelo de desenvolvimento sequencial linear, a fase de codificação é a que gera erros de maior custo de correção.

**Comentários:**

Galera, erro na fase de requisitos que não foi corrigido e foi descoberto no final do processo de desenvolvimento, terá um custo de correção altíssimo, visto que provavelmente terá que se refazer tudo novamente. Ora, se eu peço a construção de um carro e você constrói uma moto, o custo para corrigir esse erro será altíssimo. Portanto não confundam essas duas coisas! Percebam o que eu disse: quanto mais tarde se descobre um erro, mais caro se torna sua correção.

Dizendo isso de outra forma: erros nas fases iniciais possuem custo de correção altíssimo. Uma coisa é o momento em que o erro ocorre (quanto mais cedo, mais caro); outra coisa é o momento em que um erro é identificado (quanto mais tarde, mais caro). Percebam que erros nas fases iniciais possuem custos de correção mais altos. *Logo, o maior custo está na fase de codificação?* Não, está na fase de requisitos que é a fase inicial!

**Gabarito:** Errado

---

**48. (CESPE – 2009 – INMETRO – Analista de Sistemas)** Em um processo de desenvolvimento em cascata, os testes de software são realizados todos em um mesmo estágio, que acontece após a finalização das fases de implementação.

**Comentários:**

| POR SOMMERVILLE | POR ROYCE | POR PRESSMAN (4ª ED) | POR PRESSMAN (6ª ED) |
|---|---|---|---|
| Análise e Definição de Requisitos | Requisitos de Sistema | Modelagem e Engenharia do Sistema/Informação | Comunicação |
| Projeto de Sistema e Software | Requisitos de Software | Análise de Requisitos de Software | Planejamento |
| Implementação e Teste de Unidade | Análise | Projeto | Modelagem |
| Integração e Teste de Sistema | Projeto | Geração de Código | Construção |
| Operação e Manutenção | Codificação | Teste e Manutenção | Implantação |
| | Teste | | |
| | Operação | | |

Todos em um mesmo estágio, não. A grande maioria dos testes ocorrem, de fato, após a finalização das fases de implementação. No entanto, podem ocorrer testes unitários durante a própria implementação.

**Gabarito:** Errado

---

**49. (CESPE – 2008 – SERPRO – Analista de Sistemas)** O modelo em cascata consiste de fases e atividades que devem ser realizadas em sequência, de forma que uma atividade é requisito da outra.

**Comentários:**

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4.

**Gabarito:** Correto

---

**50. (CESPE – 2005 – AL/ES – Analista de Sistemas - B)** O modelo de desenvolvimento em cascata descreve ciclos sequenciais, incrementais e iterativos, possuindo, entre outras, as fases de requisitos e implementação.

**Comentários:**

Não! Ele não descreve ciclos, muito menos ciclos iterativos. Na verdade, essa é a definição de Modelo Iterativo e Incremental.

**Gabarito:** Errado

---

**51. (CESPE – 2004 – STJ – Analista de Sistemas)** O modelo de desenvolvimento seqüencial linear, também chamado modelo clássico ou modelo em cascata, caracteriza-se por não acomodar adequadamente as incertezas que existem no início de um projeto de software, em especial as geradas pela dificuldade do cliente de explicitar todos os requerimentos que o programa deve contemplar.

**Comentários:**

Como uma fase só se inicia após o término da fase anterior, só é possível em geral verificar se houve erros nas últimas fases – como pode ser visto na imagem abaixo. Em outros modelos, os riscos são reduzidos desde as primeiras fases do processo de desenvolvimento. Logo, lembrem-se que ele acumula riscos e não lida bem com requisitos voláteis.

**Gabarito:** Correto

---

**52. (CESPE – 2009 – IPEA – Analista de Sistema)** No modelo em cascata de processo de desenvolvimento, os clientes devem definir os requisitos apenas durante a fase de projeto; e os projetistas definem as estratégias de projeto apenas durante a fase de implementação. As fases do ciclo de vida envolvem definição de requisitos, projeto, implementação, teste, integração, operação e manutenção. Em cada fase do ciclo de vida, podem ser produzidos diversos artefatos.

**Comentários:**

Essa questão não faz sentido! Os clientes definem os requisitos durante a fase de Definição de Requisitos. Já os projetistas definem as estratégias de projeto apenas durante a fase Projeto.

**Gabarito:** Errado

---

**53. (CESPE – 2008 – TCE/TO – Analista de Sistema – D)** No ciclo de vida em cascata, é possível realizar alternadamente e simultaneamente as atividades de desenvolvimento de software.

**Comentários:**

No Modelo em Cascata, **uma fase só se inicia após o término e aprovação da fase anterior**, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Logo, é sequencial e linear – não pode ser alternado e simultâneo!

**Gabarito:** Errado

---

**54. (CESPE – 2004 – TJ/PA – Analista de Sistema – D)** A abordagem sistemática estritamente linear para o desenvolvimento de software é denominada modelo em cascata ou modelo sequencial linear.

**Comentários:**

Citado inicialmente em 1970 por W. Royce, também designado Cascata ou Clássico ou Sequencial ou Linear ou Tradicional ou Waterfall ou Rígido ou Monolítico (todos esses nomes já caíram em prova!). Esse nome é devido ao encadeamento simples de uma fase com a outra. Os estágios do modelo demonstram as principais atividades de desenvolvimento.

**Gabarito:** Correto

---

**55. (CESPE – 2006 – TSE – Analista de Sistema – D)** O modelo em cascata organiza o desenvolvimento em fases. Esse modelo encoraja a definição dos requisitos antes do restante do desenvolvimento do sistema. Após a especificação e a análise dos requisitos, têm-se o projeto, a implementação e o teste.

**Comentários:**

| POR SOMMERVILLE | POR ROYCE | POR PRESSMAN (4ª ED) | POR PRESSMAN (6ª ED) |
|---|---|---|---|
| Análise e Definição de Requisitos | Requisitos de Sistema | Modelagem e Engenharia do Sistema/Informação | Comunicação |

| | | | |
|---|---|---|---|
| Projeto de Sistema e Software | Requisitos de Software | Análise de Requisitos de Software | Planejamento |
| Implementação e Teste de Unidade | Análise | Projeto | Modelagem |
| Integração e Teste de Sistema | Projeto | Geração de Código | Construção |
| Operação e Manutenção | Codificação | Teste e Manutenção | Implantação |
| | Teste | | |
| | Operação | | |

Perfeito! De fato, segue essa ordem!

**Gabarito:** Correto

---

**56. (CESPE – 2009 – INMTRO – Analista de Sistema)** No desenvolvimento de software, o modelo em cascata é estruturado de tal maneira que as fases que compõem o desenvolvimento são interligadas. Nessa situação, o final de uma fase implica o início de outra.

**Comentários:**

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Logo, está em conformidade com a definição.

**Gabarito:** Correto

---

**57. (CESPE – 2010 – BASA – Analista de Sistema)** No modelo em cascata, o projeto segue uma série de passos ordenados. Ao final de cada projeto, a equipe de projeto finaliza uma revisão. O desenvolvimento continua e, ao final, o cliente avalia a solução proposta.

**Comentários:**

De acordo com Vasconcelos (2006), no Modelo em Cascata, o projeto segue uma série passos ordenados, ao final de cada fase, a equipe de projeto finaliza uma revisão. Além disso, o desenvolvimento não continua até que o cliente esteja satisfeito com os resultados alcançados. A questão afirma que a equipe de projeto finaliza uma revisão ao final de cada projeto, mas na verdade é ao final de cada fase.

**Gabarito:** Errado

---

58. **(CESPE – 2009 – TRE/MT – Analista de Sistema - A)** O modelo em cascata é apropriado para software em que os requisitos ainda não foram bem compreendidos, pois é focado na criação de incrementos.

**Comentários:**

A utilização do Modelo em Cascata deve ocorrer preferencialmente quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudanças radicais durante o desenvolvimento do sistema. Logo, questão totalmente errada!

**Gabarito:** Errado

---

59. **(CESPE – 2009 – UNIPAMPA – Analista de Sistema – D)** O modelo em cascata sugere uma abordagem sistemática e sequencial para o desenvolvimento de software. Sua natureza linear leva a estados de bloqueio nos quais, para que nova etapa seja iniciada, é necessário que a documentação associada à fase anterior tenha sido aprovada.

**Comentários:**

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Não basta terminar uma fase, é necessário que ela tenha sido aprovada.

**Gabarito:** Correto

---

60. **(CESPE – 2004 – ABIN – Analista de Sistema)** O modelo de desenvolvimento seqüencial linear, também denominado modelo em cascata, é incompatível com o emprego de técnica de análise orientada a objetos no desenvolvimento de um sistema de informação.

**Comentários:**

| VANTAGENS | DESVANTAGENS |
|---|---|
| É simples de entender e fácil de aplicar, facilitando o planejamento. | Divisão inflexível do projeto em estágios distintos. |
| Fixa pontos específicos para a entrega de artefatos. | Dificuldade em incorporar mudanças de requisitos. |
| Funciona bem para equipes tecnicamente fracas. | Clientes só visualizam resultados próximos ao final do projeto. |
| É fácil de gerenciar, devido a sua rigidez. | Atrasa a redução de riscos. |
| Realiza documentação extensa por cada fase ou estágio. | Apenas a fase final produz um artefato de software entregável. |
| Possibilita boa aderência a outros modelos de processo. | Cliente deve saber todos os requisitos no início do projeto. |

| | |
|---|---|
| Funciona bem com projetos pequenos e com requisitos bem conhecidos. | Modelo inicial (Royce) não permitia feedback entre as fases do projeto. |
| | Pressupõe que os requisitos ficarão estáveis ao longo do tempo. |
| | Não funciona bem com projetos complexos e OO, apesar de compatível. |

Ele é compatível, mas não é recomendado! *Por que, não?* Imagina um projeto super complexo que utiliza uma análise orientada a objetos (que é um modelo mais sofisticado que a análise estruturada). Lembre-se que, no Modelo em Cascata, você não pode errar, porque se você errar, os riscos de o projeto falhar são enormes! Por essa razão, ele não é recomendável, apesar de compatível!

**Gabarito:** Errado

---

**61. (CESPE – 2004 – TRE/AL – Analista de Sistema)** O modelo cascata ou ciclo de vida clássico necessita de uma abordagem sistemática, que envolve, em primeiro lugar, o projeto e, em seguida, a análise, a codificação, os testes e a manutenção.

**Comentários:**

| POR SOMMERVILLE | POR ROYCE | POR PRESSMAN (4ª ED) | POR PRESSMAN (6ª ED) |
|---|---|---|---|
| Análise e Definição de Requisitos | Requisitos de Sistema | Modelagem e Engenharia do Sistema/Informação | Comunicação |
| Projeto de Sistema e Software | Requisitos de Software | Análise de Requisitos de Software | Planejamento |
| Implementação e Teste de Unidade | Análise | Projeto | Modelagem |
| Integração e Teste de Sistema | Projeto | Geração de Código | Construção |
| Operação e Manutenção | Codificação | Teste e Manutenção | Implantação |
| | Teste | | |
| | Operação | | |

*Primeiro Projeto e depois Análise?* Não, Análise vem antes do Projeto!

**Gabarito:** Errado

---

**62. (CESPE – 2008 – MPE/AM – Analista de Sistema)** O modelo de desenvolvimento sequencial linear tem como característica principal a produção de uma versão básica, mas funcional, do software desde as primeiras fases.

**Comentários:**

| VANTAGENS | DESVANTAGENS |
|---|---|
| É simples de entender e fácil de aplicar, facilitando o planejamento. | Divisão inflexível do projeto em estágios distintos. |
| Fixa pontos específicos para a entrega de artefatos. | Dificuldade em incorporar mudanças de requisitos. |
| Funciona bem para equipes tecnicamente fracas. | Clientes só visualizam resultados próximos ao final do projeto. |
| É fácil de gerenciar, devido a sua rigidez. | Atrasa a redução de riscos. |
| Realiza documentação extensa por cada fase ou estágio. | Apenas a fase final produz um artefato de software entregável. |
| Possibilita boa aderência a outros modelos de processo. | Cliente deve saber todos os requisitos no início do projeto. |
| Funciona bem com projetos pequenos e com requisitos bem conhecidos. | Modelo inicial (Royce) não permitia feedback entre as fases do projeto. |
| | Pressupõe que os requisitos ficarão estáveis ao longo do tempo. |
| | Não funciona bem com projetos complexos e OO, apesar de compatível. |

Pelo contrário, somente nas fases finais que se tem uma versão! Essa definição está mais com cara de modelo de desenvolvimento em prototipagem.

**Gabarito:** Errado

---

**63. (VUNESP - 2012 - SPTrans - Analista de Informática)** Uma das abordagens do processo de desenvolvimento da engenharia de software prevê a divisão em etapas, em que o fim de uma é a entrada para a próxima. Esse processo é conhecido como modelo:

a) Transformação.
b) Incremental.
c) Evolutivo.
d) Espiral.
e) Cascata.

**Comentários:**

No Modelo em Cascata, uma fase só se inicia após o término e aprovação da fase anterior, isto é, há uma sequência de desenvolvimento do projeto. Por exemplo, a Fase 4 só é iniciada após o término e aprovação da Fase 3. A Fase 5 só é iniciada após o término e aprovação da Fase 4. Logo, conforme vimos em aula, trata-se do Modelo em Cascata.

**Gabarito:** Letra E

---

64. **(CESGRANRIO – 2010 – PETROBRÁS – Analista de Sistemas – Processos de Negócio)** No Ciclo de Vida Clássico, também chamado de Modelo Sequencial Linear ou Modelo Cascata, é apresentada uma abordagem sistemática composta pelas seguintes atividades:

a) Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

b) Modelagem e Engenharia do Sistema/Informação, Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

c) Modelagem e Engenharia do Sistema/Informação, Projeto, Geração de Código, Teste e Manutenção.

d) Levantamento de Requisitos de Software, Projeto, Geração de Código e Manutenção e Análise de Requisitos de Software.

e) Levantamento de Requisitos de Software, Projeto, Geração de Código, Teste Progressivo e Manutenção.

**Comentários:**

| POR SOMMERVILLE | POR ROYCE | POR PRESSMAN (4ª ED) | POR PRESSMAN (6ª ED) |
|---|---|---|---|
| Análise e Definição de Requisitos | Requisitos de Sistema | Modelagem e Engenharia do Sistema/Informação | Comunicação |
| Projeto de Sistema e Software | Requisitos de Software | Análise de Requisitos de Software | Planejamento |
| Implementação e Teste de Unidade | Análise | Projeto | Modelagem |
| Integração e Teste de Sistema | Projeto | Geração de Código | Construção |
| Operação e Manutenção | Codificação | Teste e Manutenção | Implantação |
| | Teste | | |
| | Operação | | |

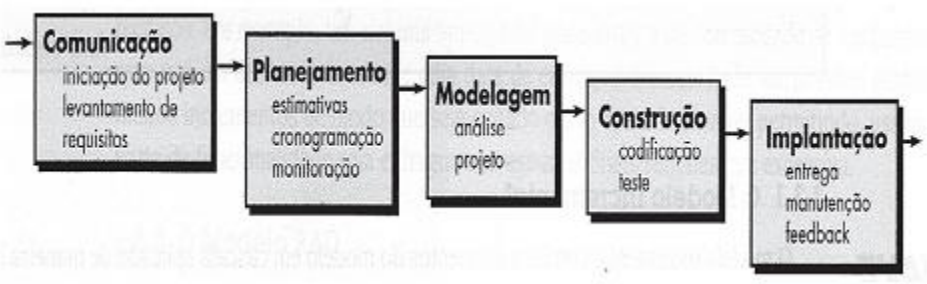
A Letra B está correta de acordo com o Pressman 4ª Edição, mas está errada de acordo com o Pressman 6ª Edição. Ademais, na questão ele sequer disse que era de acordo com o Pressman. Portanto, percebam que é um assunto polêmico e que as bancas deveriam ignorar, mas eventualmente elas cobram mesmo assim.

**Gabarito:** Letra B

---

65. **(FGV - 2015 – PGE/RO - Análise de Sistemas)** A figura abaixo ilustra um modelo de processo, que prescreve um conjunto de elementos de processo como atividades de arcabouço, ações

de engenharia de software, tarefas, produtos de trabalho, mecanismos de garantia de qualidade e de controle de modificações para cada projeto.




Esse modelo é conhecido como Modelo:

a) por funções.
b) em cascata.
c) incremental.
d) em pacotes.
e) por módulos.

**Comentários:**

| POR SOMMERVILLE | POR ROYCE | POR PRESSMAN (4ª ED) | POR PRESSMAN (6ª ED) |
|---|---|---|---|
| Análise e Definição de Requisitos | Requisitos de Sistema | Modelagem e Engenharia do Sistema/Informação | Comunicação |
| Projeto de Sistema e Software | Requisitos de Software | Análise de Requisitos de Software | Planejamento |
| Implementação e Teste de Unidade | Análise | Projeto | Modelagem |
| Integração e Teste de Sistema | Projeto | Geração de Código | Construção |
| Operação e Manutenção | Codificação | Teste e Manutenção | Implantação |
| | Teste | | |
| | Operação | | |

Conforme vimos em aula, trata-se das fases descritas pelo Pressman para o Modelo em Cascata.

**Gabarito**: Letra B

---

**66. (CESPE - 2016 – TCE/PR - Analista de Informática)** O modelo de desenvolvimento em cascata é utilizado em caso de divergência nos requisitos de um software, para permitir a evolução gradual do entendimento dos requisitos durante a implementação do software.

**Comentários:**

Essa questão trata, na verdade, do modelo iterativo e incremental e, não, em cascata.

**Gabarito:** Errado

---

**67. (CESGRANRIO – 2012 – Transpetro – Analista de Sistemas Júnior – Infra-estrutura)** Na engenharia de software, existem diversos modelos de desenvolvimento de software, e, dentre eles, o modelo em cascata, o qual, no contexto do desenvolvimento de sistemas de software complexos, recomenda:

a) distribuir a elicitação dos requisitos desde o início até o fim do desenvolvimento.
b) dividir o desenvolvimento do produto de software em fases lineares e sequenciais.
c) enfatizar a avaliação e mitigação de riscos durante o desenvolvimento.
d) realizar entregas incrementais do produto de software ao longo do desenvolvimento.
e) usar prototipagem rápida para estimular o envolvimento do usuário no desenvolvimento.

**Comentários:**

Recomenda dividir o desenvolvimento do produto de software em fases lineares e sequenciais. *Vocês se lembram que o Modelo em Cascata é um modelo sequencial linear?* Pois é!

**Gabarito:** Letra B

---

**68. (CESGRANRIO – 2012 – EPE – Analista de Gestão Corporativa – Tecnologia da Informação)** Uma das críticas feitas ao modelo do ciclo de vida do desenvolvimento de software em cascata refere-se a:

a) exigência de conhecimento de avaliação e gerenciamento de risco para evitar grandes surpresas no projeto.

b) comprometimentos na qualidade e nas possibilidades de manutenção a longo prazo, parecendo um protótipo.

c) pouca flexibilidade para mudanças futuras, exigindo compromissos nas fases iniciais do projeto.

d) pouca visibilidade das etapas do processo, tornando cara a documentação de todas as versões dos sistemas.

e) exigências de velocidade as quais levam o engenheiro de software a utilizar linguagens, algoritmos ou ferramentas ineficientes ao longo de todo o projeto.

**Comentários:**

É conhecida a pouca flexibilidade do Modelo em Cascata em se adaptar a mudanças conforme o projeto progride. Mudanças nas fases iniciais do projeto têm menor impacto do que as que são necessárias nas fases finais do projeto. Por isso, o Modelo em Cascata é usado preferencialmente em projetos com requisitos muito bem definidos e com pouca volatilidade.

**Gabarito:** Letra C

---

**MODELO EM CASCATA**

**69. (CESPE - 2011 – TJ/ES - Analista Judiciário - Análise de Sistemas - Específicos)** O modelo de processo incremental de desenvolvimento de software é iterativo, assim como o processo de prototipagem. Contudo, no processo incremental, diferentemente do que ocorre no de prototipagem, o objetivo consiste em apresentar um produto operacional a cada incremento.

**Comentários:**

De fato, no modelo iterativo e incremental, apresenta-se sempre um produto a cada incremento. Já na prototipação, não. Idealmente, ele serve apenas para identificar requisitos.

**Gabarito:** Correto

---

**70. (CESPE - 2008 – TJ/DF - Analista Judiciário - Análise de Sistemas)** No modelo de desenvolvimento incremental, embora haja defasagem entre os períodos de desenvolvimento de cada incremento, os incrementos são desenvolvidos em paralelo.

**Comentários:**

Questão perfeita! Os incrementos são codificados não exatamente em paralelo – há uma pequena defasagem.

**Gabarito:** Correto

---

**71. (CESPE - 2009 – UNIPAMPA - Análise de Sistemas)** No modelo de desenvolvimento incremental, a cada iteração são realizadas várias tarefas. Na fase de análise, pode ser feito o refinamento de requisitos e o refinamento do modelo conceitual.

**Comentários:**

Perfeito, é a fase seguinte à fase de requisitos e busca refiná-los.

**Gabarito:** Correto

---

**72. (CESPE - 2016 – TCE/PR – Analista de Sistemas – C)** No modelo iterativo de desenvolvimento de software, as atividades são dispostas em estágios sequenciais.

**Comentários:**

A questão trata do modelo em cascata de desenvolvimento de software. No modelo iterativo e incremental, as atividades são dispostas ao longo de diversas iterações

**Gabarito:** Errado

---

## MODELO EVOLUCIONÁRIO

**73. (FGV – 2008 – Senado Federal – Analista de Sistemas)** No modelo evolucionário, a mudança constante tende a corromper a estrutura do software.

**Comentários:**

Conforme vimos em aula, os sistemas são frequentemente mal estruturados: mudanças contínuas tendem a corromper a estrutura do software e tornar mudanças difíceis e caras.

**Gabarito:** Correto

---

**74. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** A prototipação evolucionária não gera problemas de manutenção de sistema porque o desenvolvimento é rápido e não sofre grandes mudanças.

**Comentários:**

Gera problemas, sim! Muitas mudanças tendem a corromper a estrutura do software e isso as tornam difíceis e caras. Observação: o ideal seria que a questão dissesse Modelo ou Abordagem Evolucionária e, não, Prototipação Evolucionária.

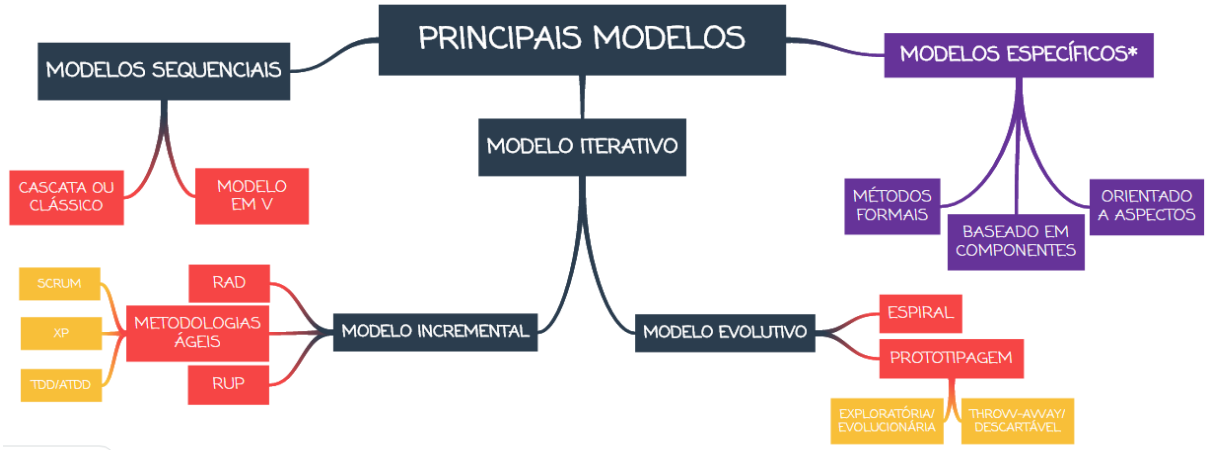
**Gabarito:** Errado

---

**75. (FCC – 2013 – DPE/SP – Analista de Sistemas)** No desenvolvimento de software, podem ser utilizados os chamados modelos evolucionários, cujo objetivo é lidar com produtos que possam evoluir ao longo do tempo. Assinale a alternativa que contém APENAS modelos evolucionários de desenvolvimento de software.

a) UML e de qualidade.
b) Componentes e arquétipo.
c) Prototipagem e espiral.
d) Redes de Petri e certificação.
e) Semântico e validação.

**Comentários:**




Galera, a imagem responde à pergunta: Prototipagem e Espiral.

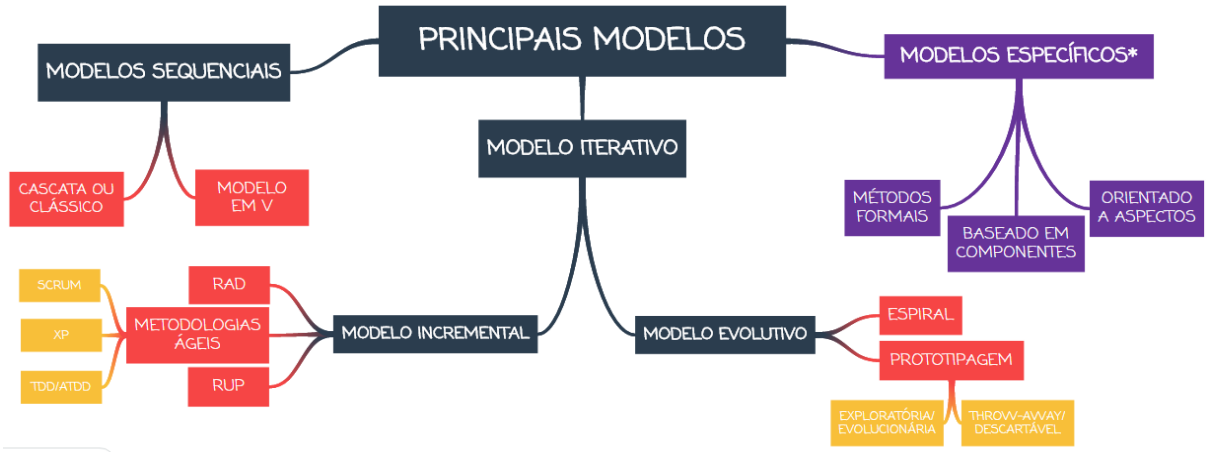
**Gabarito:** Letra C

---

**76. (IESES – 2017 – CEGÁS – Analista de Sistemas)** Para Sommerville (2007) modelos evolucionários se caracterizam por sua iteratividade e permitem o desenvolvimento de versões de software cada vez mais completas. Assinale a alternativa que caracteriza os dois tipos processos mais comuns destes modelos:

a) RUP e Cascata.
b) Cascata e incremental.
c) RAID e Cascata.
d) Espiral e Prototipação.

**Comentários:**




Mais uma vez: Espiral e Prototipação.

**Gabarito:** Letra D

---

**77. (CESPE – 2011 – FUB – Analista de Sistemas)** Os diversos modelos de processo de software disponíveis permitem a representação abstrata de um processo de software sob diferentes perspectivas. No modelo evolucionário, sob a perspectiva da arquitetura, a velocidade de desenvolvimento faz que a produção de documentos que reflitam cada versão do sistema seja economicamente inviável, gerando problemas na validação independente de sistemas.

**Comentários:**

Conforme vimos em aula, o processo não é visível: se os sistemas são desenvolvidos rapidamente, não é viável economicamente produzir documentos para cada versão do sistema.

**Gabarito:** Correto

---

**78. (CESPE – 2014 – MEC – Analista de Sistemas)** No desenvolvimento de software de grande porte, não se usam, em conjunto, os seguintes modelos de processo de software genéricos: modelo em cascata, desenvolvimento evolucionário e engenharia de software embasada em computador.

**Comentários:**

De acordo com Sommerville, os modelos genéricos de processos de software amplamente utilizados são o modelo em cascata, o modelo de desenvolvimento evolucionário e o modelo de desenvolvimento baseado em componentes. Estes, não são mutuamente exclusivos e comumente são utilizados em conjunto, especialmente para desenvolvimento de sistemas de grande porte.

Nós vimos em aula que o modelo evolucionário, por exemplo, não é recomendado para sistemas de grande porte. No entanto, a questão trata da utilização desses três modelos em conjunto – o que é possível! No entanto, não existe "*engenharia de software embasada em computador*" – o que existe é "*engenharia de software baseada em componentes*".

**Gabarito:** Errado

---

**79. (CESGRANRIO – 2014 – PETROBRÁS – Analista de Sistemas)** Um técnico de informática, com o objetivo de agilizar o desenvolvimento de um software, escolheu o desenvolvimento evolucionário, uma abordagem da área de Engenharia de Software, que:

a) facilita a produção de sistemas bem estruturados, privilegiando um processo de mudanças contínuas, cada vez mais fáceis e baratas.

b) se baseia na existência de um número significativo de componentes reusáveis, num processo de desenvolvimento que enfoca a integração desses componentes, em vez de desenvolvê-los a partir do zero

c) considera como atividades fundamentais do processo a especificação, o desenvolvimento, a validação e a evolução, representado-as como fases de processo separadas, sendo que para uma fase ser iniciada, a outra deve estar completamente terminada.

d) é mutuamente exclusiva com o modelo em cascata e de Engenharia de Software baseada em componentes, sendo que os subsistemas contidos em um sistema maior precisam ser desenvolvidos, usando a mesma abordagem ou modelo.

e) intercala as atividades de especificação, desenvolvimento e validação, permitindo que um sistema inicial seja desenvolvido rapidamente, baseado em especificações abstratas.

**Comentários:**

As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de separadas, com feedback rápido que permeia as atividades. Desenvolve-se rapidamente uma implementação inicial do software a partir de especificações bastante abstratas e são feitas modificações de acordo com sua avaliação. Ele – de fato – intercala atividades de especificação, desenvolvimento e validação, permitindo que um sistema inicial seja desenvolvido de maneira muito rápida baseado em especificações abstratas.

**Gabarito:** Letra E

---

**80. (ESAF – 2009 – ANA – Analista de Sistemas)** O modelo de processo de software caracterizado por intercalar as atividades de especificação, desenvolvimento e validação, denomina-se:

a) modelo de workflow.
b) modelo de fluxo de dados.
c) desenvolvimento evolucionário.
d) transformação formal.
e) modelo em cascata.

**Comentários:**

Um esboço simples do processo de desenvolvimento evolucionário é apresentado acima. As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de separadas, com feedback rápido que permeia as atividades. Desenvolve-se rapidamente uma implementação inicial do software a partir de especificações bastante abstratas e são feitas modificações de acordo com sua avaliação. Logo, só pode ser o modelo (ou desenvolvimento) evolucionário.

**Gabarito:** Letra C

---

**81. (FEPESE – 2014 – MPE/SC – Analista de Sistemas)** Assinale a alternativa abaixo que melhor identifica o modelo de processo de software no qual uma implementação inicial é exposta ao

usuário para que possam ser realizados refinamentos posteriores que representam novas versões do sistema. As atividades de especificação, desenvolvimento e validação são intercaladas.

a) Relational Unified Process (RUP)
b) Desenvolvimento Evolucionário
c) Método Ágil de Desenvolvimento
d) Modelo de Desenvolvimento em Cascata
e) Modelo de Engenharia de Software Baseado em Componentes

**Comentários:**

Um esboço simples do processo de desenvolvimento evolucionário é apresentado acima. As atividades de especificação, desenvolvimento e validação são intercaladas, em vez de separadas, com feedback rápido que permeia as atividades. Desenvolve-se rapidamente uma implementação inicial do software a partir de especificações bastante abstratas e são feitas modificações de acordo com sua avaliação. Essa é mais uma questão parecida e trata do modelo (ou desenvolvimento) evolucionário.

**Gabarito:** Letra B

---

**82. (FCC – 2014 – TRT/16 – Analista de Sistemas)** Os modelos de processo são uma representação abstrata de um processo de software, que podem ser usados para explicar diferentes abordagens para o desenvolvimento de sistemas. Analise as seguintes abordagens:

Desenvolvimento I : intercala as atividades de especificação, desenvolvimento e validação. Um sistema inicial é de- senvolvido rapidamente baseado em especificações abstratas e depois é refinado com as entradas do cliente para produzir um produto que o satisfaça.

Modelo II : considera as atividades fundamentais do processo, compreendendo especificação, desenvolvimento, validação e evolução e as representa como fases de processo separadas, tais como especificação de requisitos, projeto de software, implementação, teste etc.

III: baseia-se na existência de um número significativo de partes reusáveis. O processo de desenvolvimento do sistema enfoca a integração destas partes, ao invés de desenvolvê-las a partir do zero.

Os modelos de processo genéricos descritos em I, II e III são, correta e respectivamente, associados a:

a) em Espiral - Baseado em Componentes - RAD
b) Evolucionário - em Cascata - Baseado em Componentes
c) Baseado em Componentes - Sequencial - Refactoring
d) Ágil - Sequencial - Unified Process

e) em Cascata - Ágil - Refactoring

**Comentários:**

Na verdade, intercala atividades de especificação, desenvolvimento e validação só pode ser o Desenvolvimento Evolucionário. Apenas com isso, nós já podemos matar a questão.

**Gabarito:** Letra B

---

## MODELO EM PROTOTIPAGEM

**83. (CESPE – 2009 – UNIPAMPA – Analista de Sistemas)** No modelo de desenvolvimento prototipagem, um protótipo é desenvolvido para ajudar no entendimento dos requisitos do sistema.

**Comentários:**

A Prototipagem é utilizada quando não se conhece bem os requisitos. É uma forma de entendê-los melhor para posteriormente desenvolver o software. Ela se configura como um processo iterativo, interativo e rápido de desenvolvimento e o protótipo serve como um mecanismo de identificação dos requisitos do software, que servirão para uma futura especificação. Dessa forma, um protótipo é em geral desenvolvido para ajudar no entendimento dos requisitos do sistema.

**Gabarito:** Correto

---

**84. (CESPE – 2010 – TRE/MT – Analista de Sistemas - C)** A metodologia de prototipagem evolutiva é uma abordagem que visualiza o desenvolvimento de concepções do sistema conforme o andamento do projeto, por meio de protótipos visuais.

**Comentários:**

A metodologia de Prototipagem Evolutiva (ou Evolucionária) é uma abordagem que visualiza o desenvolvimento de concepções do sistema conforme o andamento do projeto até chegar ao resultado final. Esta metodologia baseia-se na utilização de prototipagem visual ou modelos do sistema final. Estes modelos podem ser simples desenhos ou imagens do sistema. Logo, na prototipação evolutiva, eu vou mostrando ao meu usuário funcionalidades por meio de protótipos visuais conforme o andamento do projeto.

**Gabarito:** Correto

---

**85. (CESPE – 2004 – TRE/AL – Analista de Sistemas)** No modelo de prototipação, o desenvolvedor cria inicialmente um modelo de software que será posteriormente implementado.

**Comentários:**

Cria-se, inicialmente, um modelo do sistema final! Além disso, o protótipo pode ser implementado e se tornar esse sistema final. Galera, essa questão foi retirada literalmente do Pressman (1995): *"Prototyping is a process that enables the construction of a model of the software which is to be built". Qual o problema dessa questão?* Sua tradução!

Pressman afirma que a Prototipação é um processo que permite ao desenvolvedor a construção de um modelo **do** software que será construído. Em outras palavras, a prototipação é um modelo/esboço do software que posteriormente será construído. Pensem em um software qualquer! Antes de construir o software, faremos um modelo/esboço dele (não importando se iremos descartá-lo ou não). A questão não afirma em nenhum momento que o protótipo será evoluído ou descartado.

**Gabarito:** Correto

---

**86. (CESPE – 2007 – TSE – Analista de Sistemas)** Um possível objetivo da prototipação é criar rapidamente um sistema experimental que possa ser avaliado por usuários finais. Um protótipo aprovado pelos usuários pode vir a ser usado como ponto de partida para a construção do sistema.

**Comentários:**

Na maioria dos casos, o protótipo é inviável de ser utilizado, por ser muito lento e/ou muito grande e/ou difícil de utilizar. Em geral, os protótipos são descartados assim que os objetivos de levantamento de requisitos são alcançados. No entanto, alguns preferem refiná-los iterativamente até evoluir ao sistema final requisitado pelo usuário. Logo, é uma alternativa, isto é, posso usá-lo como ponto de partida para construção do sistema, em vez de descartá-lo!

**Gabarito:** Correto

---

**87. (CESPE – 2008 – MPE/AM – Analista de Sistemas)** No modelo de prototipação, a especificação de requisitos tem pouca importância, pois o software é continuamente adaptado em função dos desejos do usuário.

**Comentários:**

A Prototipagem é utilizada quando não se conhece bem os requisitos. É uma forma de entendê-los melhor para posteriormente desenvolver o software. Ela se configura como um processo iterativo, interativo e rápido de desenvolvimento e o protótipo serve como um mecanismo de identificação dos requisitos do software, que servirão para uma futura especificação. Dessa forma, a prototipação serve não só para o levantamento de requisitos, mas também para sua especificação. *Galera, como não tem importância?* É para isso que ele serve!

**Gabarito:** Errado

---

**88. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** Uma vantagem da prototipação é promover a participação e o comprometimento do usuário em relação ao sistema em desenvolvimento.

**Comentários:**

Protótipos permitem que os usuários introduzam novas ideias para os requisitos e encontrem pontos fortes e fracos no software. Ademais, protótipos podem revelar erros e omissões nos requisitos propostos, além de reduzirem o tempo de desenvolvimento, treinamento e documentação o sistema. Logo, percebam que a prototipação promove uma grande participação dos usuários no processo de desenvolvimento.

**Gabarito:** Correto

---

**89. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** A prototipação de um software é uma técnica de desenvolvimento não-interativa porque o teste do sistema só ocorre na versão final.

**Comentários:**

A Prototipagem é utilizada quando não se conhece bem os requisitos. É uma forma de entendê-los melhor para posteriormente desenvolver o software. Ela se configura como um processo iterativo, interativo e rápido de desenvolvimento e o protótipo serve como um mecanismo de identificação dos requisitos do software, que servirão para uma futura especificação. Logo, ela é tanto iterativa (repete-se diversas vezes) quanto interativa (conta com a participação ativa dos usuários).

**Gabarito:** Errado

---

**90. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** Uma das finalidades da prototipação é reduzir o esforço de desenvolvimento de um software.

**Comentários:**

Protótipos permitem que os usuários introduzam novas ideias para os requisitos e encontrem pontos fortes e fracos no software. Ademais, protótipos podem revelar erros e omissões nos requisitos propostos, além de reduzirem o tempo e esforço de desenvolvimento, treinamento e documentação o sistema. Logo, ela realmente ajuda a reduzir o esforço! A prototipação (não importa qual tipo) ajuda a esclarecer requisitos. Caso eu não faça um protótipo, pode ocorrer de eu não compreender bem requisitos obscuros e, eventualmente, eu ter que refazer o desenvolvimento.

**Gabarito:** Correto

---

**91. (CESPE – 2010 – INMETRO – Analista de Sistemas – A)** Um dos benefícios da prototipação é a documentação normalmente gerada, que facilita a manutenção dos sistemas a longo prazo e a elaboração de casos de teste.

**Comentários:**

Na verdade, é justamente o inverso! A documentação geralmente é prejudicada quando se utiliza a prototipação! Imaginem só: você precisa entregar algo rápido para o usuário verificar se satisfaz seus requisitos, não é viável fazer uma documentação formal.

**Gabarito:** Errado

---

**92. (CESPE – 2010 – INMETRO – Analista de Sistemas – D)** Um dos riscos da prototipação é o usuário confundir o protótipo com o sistema verdadeiro e criar falsas expectativas com relação a prazos e recursos.

**Comentários:**

De fato, um dos riscos da prototipação realmente é o usuário confundir o protótipo com o sistema verdadeiro e criar falsas expectativas com relação a prazos e recursos.

**Gabarito:** Correto

---

**93. (CESPE – 2010 – INMETRO – Analista de Sistemas – E)** Na abordagem evolutiva para desenvolvimento de software, um protótipo do software é produzido e utilizado para identificar possíveis problemas com os requisitos, sendo descartado logo em seguida, e o desenvolvimento do software propriamente dito é, então, iniciado.

**Comentários:**

Se o protótipo é descartado, então não se trata de uma abordagem evolutiva! Cuidado, porque esses nomes confundem: existe modelo evolutivo e abordagem evolutiva – a Prototipação Throw-away não é uma abordagem evolutiva, mas é parte do Modelo Evolutivo.

**Gabarito:** Errado

---

**94. (CESPE – 2011 – MEC – Analista de Sistemas)** No modelo de prototipação, o processo de desenvolvimento de software é modelado como uma sequência linear de fases, enfatizando um ciclo de desenvolvimento de breve duração.

**Comentários:**

Não é linear, mas iterativo! A questão menciona uma sequência linear de fases, que lembra o Modelo em Cascata; e um ciclo de desenvolvimento de breve duração, que lembra o Desenvolvimento Rápido de Aplicações (RAD).

**Gabarito:** Errado

---

**95. (CESPE – 2013 – TRT/10 – Analista Judiciário – Tecnologia da Informação)** No modelo prototipação, a construção de software tem várias atividades que são executadas de forma sistemática e sequencial.

**Comentários:**

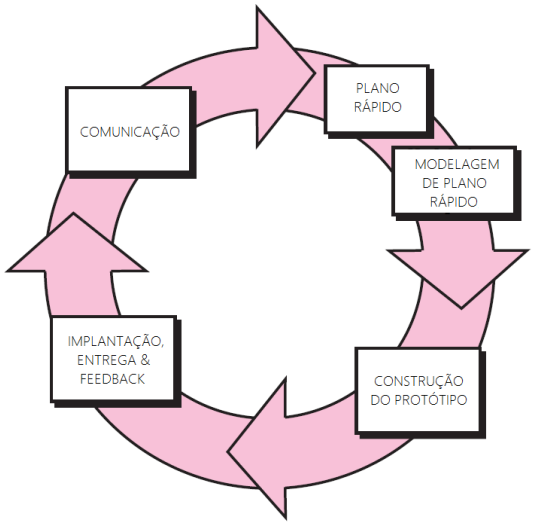
Na verdade, quem constrói software por meio de atividades executadas de forma sistemática e sequencial é o Modelo em Cascata; a prototipação é iterativa.

**Gabarito:** Errado

---

**96. (CESPE – 2009 – TCE/RN – Assessor Técnico de Informática)** A prototipação, uma abordagem para desenvolvimento de software na qual se cria um modelo do software que será implementado, é composta de quatro etapas: planejamento, análise de risco, engenharia e avaliação do cliente.

**Comentários:**




As etapas são Comunicação, Plano Rápido, Modelagem de Plano Rápido, Construção do Protótipo, e Implantação, Entrega e Feedback. As etapas mencionadas na questão são do Modelo em Espiral.

**Gabarito:** Errado

---

**97. (CESPE – 2009 – DETRAN/DF – Analista de Sistemas)** O modelo de processo de desenvolvimento de software evolucionário parte do desenvolvimento de uma implementação inicial cujos resultados são apresentados aos clientes e refinados por meio de várias versões até que se alcance o sistema adequado. A prototipação, como processo, tem por objetivo compreender as especificações do software para se chegar aos requisitos para o sistema.

**Comentários:**

Conforme vimos em aula, essa questão está errada. Ela afirma que a prototipação tem por objetivo (1) compreender as especificações do software para (2) se chegar aos requisitos para o sistema. Na verdade, é justamente o contrário! Primeiro, eu levanto os requisitos do sistema e, depois, eu faço a especificação. Cabe ressaltar que a especificação é o detalhamento dos requisitos, logo ele não pode vir antes do levantamento dos requisitos.

**Gabarito:** Errado

---

**98. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** A prototipação evolucionária permite que a versão inicial do protótipo seja desenvolvida e refinada em estágios sequenciados, até que se chegue à versão final do sistema.

**Comentários:**

Conforme vimos em aula, a questão está perfeita! Trata-se da prototipação evolucionária ou exploratória. Percebam que a questão afirma que a versão é desenvolvida em estágios sequenciados e, não, o processo de desenvolvimento. Esse é iterativo e, não, sequenciado. No entanto, as versões são sequenciais, no sentido de que as versões seguem uma ordem.

**Gabarito:** Correto

---

**99. (VUNESP - 2009 - CETESB - Analista de Tecnologia da Informação - Banco de Dados)** Considere um sistema cujos requisitos de interface são definidos apenas quando o cliente realiza um test-drive na aplicação e aprova essa interface. Assinale a alternativa que apresenta o modelo mais adequado para o desenvolvimento da interface desse sistema.

a) Ágil.
b) Cascata.
c) Iterativo incremental.
d) Prototipação.
e) Rapid Application Development.

**Comentários:**

A afirmativa diz que o sistema possui requisitos de interface que são definidos apenas quando o cliente realiza um test-drive, isto é, um teste inicial na aplicação e aprova essa interface. Em outras palavras, o cliente valida ou não esses requisitos a partir de uma primeira utilização do sistema. *Que metodologia de desenvolvimento de software pode ser utilizada para levantar e validar/aprovar requisitos?* Trata-se da Prototipação!

**Gabarito:** Letra D

---

**100. (FGV – 2010 – Fiocruz – Analista de Sistemas)** Como Modelo evolucionário do processo de software, uma característica da prototipagem é:

a) independer do estabelecimento e da definição de requisitos.
b) configurar um processo interativo e rápido de desenvolvimento.
c) iniciar o processo de desenvolvimento pela implantação e pelos testes.
d) gerar uma primeira versão do sistema completa e isenta de erros.
e) descartar a participação do cliente no processo de desenvolvimento e de implantação.

**Comentários:**

(a) Errado, claro que depende da definição de requisitos; (b) Correto, é um processo interativo, iterativo e rápido de desenvolvimento; (c) Errado. *Você vai implantar e testar o que?* Você primeiro tem que levantar, especificar e desenvolver; (d) Errado, se é uma primeira versão, não é completa. Além disso, não há sistemas sem erros; (e) Errado, você deve incentivar a participação do cliente.

**Gabarito:** Letra B

---

# 3 – LISTA DE EXERCÍCIOS

ENGENHARIA DE SOFTWARE

1. **(CESPE - 2013 - TRT - 10ª REGIÃO (DF e TO) - Analista Judiciário - Tecnologia da Informação)** A engenharia de software engloba processos, métodos e ferramentas. Um de seus focos é a produção de software de alta qualidade a custos adequados.

2. **(FCC - 2012 - TST - Analista Judiciário - Análise de Sistemas)** A Engenharia de Software:

   a) é uma área da computação que visa abordar de modo sistemático as questões técnicas e não técnicas no projeto, implantação, operação e manutenção no desenvolvimento de um software.

   b) consiste em uma disciplina da computação que aborda assuntos relacionados a técnicas para a otimização de algoritmos e elaboração de ambientes de desenvolvimento.

   c) trata-se de um ramo da TI que discute os aspectos técnicos e empíricos nos processos de desenvolvimento de sistemas, tal como a definição de artefatos para a modelagem ágil.

   d) envolve um conjunto de itens que abordam os aspectos de análise de mercado, concepção e projeto de software, sendo independente da engenharia de um sistema.

   e) agrupa as melhores práticas para o concepção, projeto, operação e manutenção de artefatos que suportam a execução de programas de computador, tais como as técnicas de armazenamento e as estruturas em memória principal.

3. **(FCC - 2012 - TRT - 6ª Região (PE) - Técnico Judiciário - Tecnologia da Informação)** Considere: *é uma disciplina que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Seu principal objetivo é fornecer uma estrutura metodológica para a construção de software com alta qualidade.* A definição refere-se:

   a) ao ciclo de vida do software.
   b) à programação orientada a objetos.
   c) à análise de sistemas.
   d) à engenharia de requisitos.
   e) à engenharia de software.

4. **(CESPE - 2011 - MEC - Gerente de Projetos)** A engenharia de software, disciplina relacionada aos aspectos da produção de software, abrange somente os processos técnicos do desenvolvimento de software.

5. **(FGV - 2010 - BADESC - Analista de Sistemas - Desenvolvimento de Sistemas)** De acordo com Pressman, a engenharia de software é baseada em camadas, com foco na qualidade.

   Essas camadas são:

   a) métodos, processo e teste.
   b) ferramentas, métodos e processo.
   c) métodos, construção, teste e implantação.
   d) planejamento, modelagem, construção, validação e implantação.
   e) comunicação, planejamento, modelagem, construção e implantação.

6. **(CESPE - 2010 - Banco da Amazônia - Técnico Científico - Tecnologia da Informação)** Os princípios de engenharia de software definem a necessidade de formalidades para reduzir inconsistências e a decomposição para lidar com a complexidade.

7. **(FCC - 2010 - TRE-AM - Analista Judiciário - Tecnologia da Informação)** A Engenharia de Software:

   a) não tem como método a abordagem estruturada para o desenvolvimento de software, pois baseia-se exclusivamente nos modelos de software, notações, regras e técnicas de desenvolvimento.

   b) se confunde com a Ciência da Computação quando ambas tratam do desenvolvimento de teorias, fundamentações e práticas de desenvolvimento de software.

   c) tendo como foco apenas o tratamento dos aspectos de construção de software, subsidia a Engenharia de Sistemas no tratamento dos sistemas baseados em computadores, incluindo hardware e software.

   d) tem como foco principal estabelecer uma abordagem sistemática de desenvolvimento, através de ferramentas e técnicas apropriadas, dependendo do problema a ser abordado, considerando restrições e recursos disponíveis.

   e) segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

8. **(FCC - 2011 - INFRAERO - Analista de Sistemas - Gestão de TI)** Em relação à Engenharia de Software, é INCORRETO afirmar:

   a) O design de software, ao descrever os diversos aspectos que estarão presentes no sistema quando construído, permite que se faça a avaliação prévia para garantir que ele alcance os objetivos propostos pelos interessados.

b) A representação de um design de software mais simples para representar apenas as suas características essenciais busca atender ao princípio da abstração.

c) Iniciar a entrevista para obtenção dos requisitos de software com perguntas mais genéricas e finalizar com perguntas mais específicas sobre o sistema é o que caracteriza a técnica de entrevista estruturada em funil.

d) No contexto de levantamento de requisitos, funcionalidade é um dos aspectos que deve ser levado em conta na abordagem dos requisitos funcionais.

e) A representação é a linguagem do design, cujo único propósito é descrever um sistema de software que seja possível construir.

9. **(FCC – 2009 – AFR/SP - Analista de Sistemas)** A engenharia de software está inserida no contexto:

a) das engenharias de sistemas, de processo e de produto.
b) da engenharia de sistemas, apenas.
c) das engenharias de processo e de produto, apenas.
d) das engenharias de sistemas e de processo, apenas.
e) das engenharias de sistemas e de produto, apenas.

10. **(CESPE – 2015 – STJ – Analista de Sistemas)** Embora os engenheiros de software geralmente utilizem uma abordagem sistemática, a abordagem criativa e menos formal pode ser eficiente em algumas circunstâncias, como, por exemplo, para o desenvolvimento de sistemas web, que requerem uma mistura de habilidades de software e de projeto.

11. **(CESPE – 2015 – STJ – Analista de Sistemas)** O foco da engenharia de software inclui especificação do sistema, desenvolvimento de hardware, elaboração do projeto de componentes de hardware e software, definição dos processos e implantação do sistema.

12. **(FUNIVERSA – 2009 – IPHAN – Analista de Sistemas)** A Engenharia de Software resume-se em um conjunto de técnicas utilizadas para o desenvolvimento e manutenção de sistemas computadorizados, visando produzir e manter softwares de forma padronizada e com qualidade. Ela obedece a alguns princípios como (1) Formalidade, (2) Abstração, (3) Decomposição, (4) Generalização e (5) Flexibilização. Assinale a alternativa que apresenta conceito correto sobre os princípios da Engenharia de Software.

a) A flexibilização é o processo que permite que o software possa ser alterado, sem causar problemas para sua execução.

b) A formalidade é a maneira usada para resolver um problema, de forma genérica, com o intuito de poder reaproveitar essa solução em outras situações semelhantes.

c) A generalização preocupa-se com a identificação de um determinado fenômeno da realidade, sem se preocupar com detalhes, considerando apenas os aspectos mais relevantes.

d) Pelo princípio da decomposição, o software deve ser desenvolvido de acordo com passos definidos com precisão e seguidos de maneira efetiva.

e) A abstração é a técnica de se dividir o problema em partes, de maneira que cada uma possa ser resolvida de uma forma mais específica.

13. **(CESPE – 2013 – TCE/RO – Analista de Sistemas)** Engenharia de software não está relacionada somente aos processos técnicos de desenvolvimento de softwares, mas também a atividades como gerenciamento de projeto e desenvolvimento de ferramentas, métodos e teorias que apoiem a produção de softwares.

14. **(CESPE – 2010 – DETRAN/ES – Analista de Sistemas)** Segundo princípio da engenharia de software, os vários artefatos produzidos ao longo do seu ciclo de vida apresentam, de forma geral, nível de abstração cada vez menor.

15. **(CESPE – 2010 – TRE/BA – Analista de Sistemas)** Entre os desafios enfrentados pela engenharia de software estão lidar com sistemas legados, atender à crescente diversidade e atender às exigências quanto a prazos de entrega reduzidos.

16. **(FCC – 2010 – DPE/SP – Analista de Sistemas)** A Engenharia de Software

I. não visa o desenvolvimento de teorias e fundamentações, preocupando-se unicamente com as práticas de desenvolvimento de software.

II. tem como foco o tratamento dos aspectos de desenvolvimento de software, abstraindo-se dos sistemas baseados em computadores, incluindo hardware e software.

III. tem como métodos as abordagens estruturadas para o desenvolvimento de software que incluem os modelos de software, notações, regras e maneiras de desenvolvimento.

IV. segue princípios, tais como, o da Abstração, que identifica os aspectos importantes sem ignorar os detalhes e o da Composição, que agrupa as atividades em um único processo para distribuição aos especialistas.

É correto o que se afirma em:

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

17. **(CESPE – 2013 – TCE/RO – Analista de Sistemas)** Assim como a Engenharia de Software, existe também na área de informática a chamada Ciência da Computação. Assinale a alternativa que melhor apresenta a diferença entre Engenharia de Software e Ciência da Computação.

a) A Ciência da Computação tem como objetivo o desenvolvimento de teorias e fundamentações. Já a Engenharia de Software se preocupa com as práticas de desenvolvimento de software.

b) A Engenharia de Software trata da criação dos sistemas de computação (softwares) enquanto a Ciência da Computação está ligada ao desenvolvimento e criação de componentes de hardware.

c) A Engenharia de Software trata dos sistemas com base em computadores, que inclui hardware e software, e a Ciência da Computação trata apenas dos aspectos de desenvolvimento de sistemas.

d) A Ciência da Computação trata dos sistemas com base em computadores, que inclui hardware e software, e a Engenharia de Software trata apenas dos aspectos de desenvolvimento de sistemas.

e) A Ciência da Computação destina-se ao estudo e solução para problemas genéricos das áreas de rede e banco de dados e a Engenharia de Software restringe- se ao desenvolvimento de sistemas.

18. **(CESPE – 2009 – ANAC – Analista de Sistemas)** O termo engenharia pretende indicar que o desenvolvimento de software submete-se a leis similares às que governam a manufatura de produtos industriais em engenharias tradicionais, pois ambos são metodológicos.

19. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – B)** A engenharia de software refere-se ao estudo das teorias e fundamentos da computação, ficando o desenvolvimento de software a cargo da ciência da computação.

20. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – E)** O conceito de software se restringe ao desenvolvimento do código em determinada linguagem e seu armazenamento em arquivos.

## PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

21. **(CESPE – 2009 – TCE/TO – Analista de Sistemas - A)** Quanto maior e mais complexo o projeto de software, mais simples deve ser o modelo de processo a ser adotado.

22. **(CESPE – 2009 – TCE/TO – Analista de Sistemas - B)** O modelo de ciclo de vida do software serve para delimitar o alvo do software. Nessa visão, não são consideradas as atividades necessárias e o relacionamento entre elas.

23. **(CESPE – 2009 – TCE/TO – Analista de Sistemas - C)** A escolha do modelo do ciclo de vida não depende de características específicas do projeto, pois o melhor modelo é sempre o mais usado pela equipe do projeto.

24. **(ESAF - 2012 – CGU – Analista de Sistemas)** A escolha de um modelo é fortemente dependente das características do projeto. Os principais modelos de ciclo de vida podem ser agrupados em três categorias principais:

a) sequenciais, cascata e evolutivos.
b) sequenciais, incrementais e ágeis.
c) sequenciais, incrementais e evolutivos.
d) sequenciais, ágeis e cascata.
e) cascata, ágeis e evolutivos.

25. **(CESPE – 2011 – MEC – Analista de Sistemas)** O processo de desenvolvimento de software é uma caracterização descritiva ou prescritiva de como um produto de software deve ser desenvolvido.

26. **(CESPE – 2013 – TRT/10ª – Analista de Sistemas)** As atividades fundamentais relacionadas ao processo de construção de um software incluem a especificação, o desenvolvimento, a validação e a evolução do software.

27. **CESPE – 2010 – TRE/BA – Analista de Sistemas)** Um modelo de processo de software consiste em uma representação complexa de um processo de software, apresentada a partir de uma perspectiva genérica.

28. **(CESPE – 2011 – MEC – Analista de Sistemas)** Atividades comuns a todos os processos de software incluem a especificação, o projeto, a implementação e a validação.

29. **(CESPE – 2015 – STJ – Analista de Sistemas)** As principais atividades de engenharia de software são especificação, desenvolvimento, validação e evolução.

30. **(FCC – 2012 – MPE/AP – Analista de Sistemas)** Um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software. Existem muitos processos de software diferentes, mas todos devem incluir quatro atividades fundamentais: especificação, projeto e implementação, validação e:

a) teste
b) evolução.
c) prototipação.
d) entrega.
e) modelagem.

31. **(CESPE – 2010 – EMBASA – Analista de Sistemas)** Ciclo de vida de um software resume-se em eventos utilizados para definir o status de um projeto.

32. **(CESPE – 2016 – TCE/PR – Analista de Sistemas)** As fases do ciclo de vida de um software são:

a) concepção, desenvolvimento, entrega e encerramento.
b) iniciação, elaboração, construção e manutenção.
c) escopo, estimativas, projeto e processo e gerência de riscos.
d) análise, desenvolvimento, teste, empacotamento e entrega.
e) planejamento, análise e especificação de requisitos, projeto, implementação, testes, entrega e implantação, operação e manutenção.

33. **(MPE/RS – 2012 – MPE/RS – Analista de Sistemas)** O ciclo de vida básico de um software compreende:

a) a implementação, a implantação e o teste.
b) a análise, a segurança e o controle de usuários.
c) a implementação, a análise e o teste.
d) a implementação, a validação e as vendas.
e) a análise, o projeto, a implementação e o teste.

34. **(CESGRANRIO – 2011 – PETROBRÁS – Analista de Sistemas)** A especificação de uma Metodologia de Desenvolvimento de Sistemas tem como pré-requisito indispensável, em relação ao que será adotado no processo de desenvolvimento, a definição do:

a) Engenheiro Responsável pelo Projeto
b) Documento de Controle de Sistemas
c) Software para Desenvolvimento
d) Ciclo de Vida do Software
e) Bloco de Atividades

35. **(CESPE – 2008 – INPE – Analista de Sistemas)** O ciclo de vida do software tem início na fase de projeto.
36. **(CESPE – 2013 – TRT/10 – Analista de Sistemas)** O ciclo de vida de um software, entre outras características, está relacionado aos estágios de concepção, projeto, criação e implementação.

37. **(CESPE – 2011 – TJ/ES – Analista de Sistemas)** Entre as etapas do ciclo de vida de software, as menos importantes incluem a garantia da qualidade, o projeto e o estudo de viabilidade. As demais atividades do ciclo, como a implementação e os testes, requerem maior dedicação da equipe e são essenciais.

38. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – A)** A engenharia de software está relacionada aos diversos aspectos de produção de software e inclui as atividades de especificação, desenvolvimento, validação e evolução de software.

39. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – D)** Um processo de software é composto por quatro atividades fundamentais: iniciação, desenvolvimento, entrega e encerramento.

40. **(CESPE - 2016 – TCE/PR – Analista de Sistemas – B)** A metodologia de processos genérica para a engenharia de software é composta de seis etapas: comunicação, planejamento, modelagem, construção, emprego e aceitação.

41. **(INSTITUTO CIDADE – 2012 – TCM/GO – Analista de Sistemas)** De acordo com a engenharia de software, como todo produto industrial, o software possui um ciclo de vida. Cada fase do ciclo de vida possui divisões e subdivisões. Em qual fase avaliamos a necessidade de evolução dos softwares em funcionamento para novas plataformas operacionais ou para a incorporação de novos requisitos?

    a) Fase de operação;
    b) Fase de retirada;
    c) Fase de definição;
    d) Fase de design.
    e) Fase de desenvolvimento;

42. **(CESPE - 2010 – DETRAN/ES – Analista de Sistemas)** Quando um aplicativo de software desenvolvido em uma organização atinge, no fim do seu ciclo de vida, a fase denominada aposentadoria, descontinuação ou fim de vida, todos os dados por ele manipulados podem ser descartados.

## MODELO EM CASCATA

43. **(FCC - 2012 - TJ-RJ - Analista Judiciário - Análise de Sistemas - E)** Dos diferentes modelos para o ciclo de vida de desenvolvimento de um software é correto afirmar que o modelo em cascata é o mais recente e complexo.

44. **(FCC - 2009 - SEFAZ-SP - Agente Fiscal de Rendas - Tecnologia da Informação - Prova 3 - B)** O processo de engenharia de software denominado ciclo de vida clássico refere-se ao modelo incremental.

45. **(CESPE – 2009 – INMETRO – Analista de Sistemas)** Em uma empresa que tenha adotado um processo de desenvolvimento de software em cascata, falhas no levantamento de requisitos têm maior possibilidade de gerar grandes prejuízos do que naquelas que tenham adotado desenvolvimento evolucionário.

46. **(CESPE – 2011 – MEC – Analista de Sistemas)** O modelo Waterfall tem a vantagem de facilitar a realização de mudanças sem a necessidade de retrabalho em fases já completadas.

47. **(CESPE – 2008 – TST – Analista de Sistemas)** No modelo de desenvolvimento sequencial linear, a fase de codificação é a que gera erros de maior custo de correção.

48. **(CESPE – 2009 – INMETRO – Analista de Sistemas)** Em um processo de desenvolvimento em cascata, os testes de software são realizados todos em um mesmo estágio, que acontece após a finalização das fases de implementação.

49. **(CESPE – 2008 – SERPRO – Analista de Sistemas)** O modelo em cascata consiste de fases e atividades que devem ser realizadas em sequência, de forma que uma atividade é requisito da outra.

50. **(CESPE – 2005 – AL/ES – Analista de Sistemas - B)** O modelo de desenvolvimento em cascata descreve ciclos sequenciais, incrementais e iterativos, possuindo, entre outras, as fases de requisitos e implementação.

51. **(CESPE – 2004 – STJ – Analista de Sistemas)** O modelo de desenvolvimento seqüencial linear, também chamado modelo clássico ou modelo em cascata, caracteriza-se por não acomodar adequadamente as incertezas que existem no início de um projeto de software, em especial as geradas pela dificuldade do cliente de explicitar todos os requerimentos que o programa deve contemplar.

52. **(CESPE – 2009 – IPEA – Analista de Sistema)** No modelo em cascata de processo de desenvolvimento, os clientes devem definir os requisitos apenas durante a fase de projeto; e os projetistas definem as estratégias de projeto apenas durante a fase de implementação. As fases do ciclo de vida envolvem definição de requisitos, projeto, implementação, teste, integração, operação e manutenção. Em cada fase do ciclo de vida, podem ser produzidos diversos artefatos.

53. **(CESPE – 2008 – TCE/TO – Analista de Sistema – D)** No ciclo de vida em cascata, é possível realizar alternadamente e simultaneamente as atividades de desenvolvimento de software.
54. **(CESPE – 2004 – TJ/PA – Analista de Sistema – D)** A abordagem sistemática estritamente linear para o desenvolvimento de software é denominada modelo em cascata ou modelo sequencial linear.

55. **(CESPE – 2006 – TSE – Analista de Sistema – D)** O modelo em cascata organiza o desenvolvimento em fases. Esse modelo encoraja a definição dos requisitos antes do restante do desenvolvimento do sistema. Após a especificação e a análise dos requisitos, têm-se o projeto, a implementação e o teste.

56. **(CESPE – 2009 – INMTRO – Analista de Sistema)** No desenvolvimento de software, o modelo em cascata é estruturado de tal maneira que as fases que compõem o desenvolvimento são interligadas. Nessa situação, o final de uma fase implica o início de outra.

57. **(CESPE – 2010 – BASA – Analista de Sistema)** No modelo em cascata, o projeto segue uma série de passos ordenados. Ao final de cada projeto, a equipe de projeto finaliza uma revisão. O desenvolvimento continua e, ao final, o cliente avalia a solução proposta.

58. **(CESPE – 2009 – TRE/MT – Analista de Sistema - A)** O modelo em cascata é apropriado para software em que os requisitos ainda não foram bem compreendidos, pois é focado na criação de incrementos.

59. **(CESPE – 2009 – UNIPAMPA – Analista de Sistema – D)** O modelo em cascata sugere uma abordagem sistemática e sequencial para o desenvolvimento de software. Sua natureza linear leva a estados de bloqueio nos quais, para que nova etapa seja iniciada, é necessário que a documentação associada à fase anterior tenha sido aprovada.

60. **(CESPE – 2004 – ABIN – Analista de Sistema)** O modelo de desenvolvimento seqüencial linear, também denominado modelo em cascata, é incompatível com o emprego de técnica de análise orientada a objetos no desenvolvimento de um sistema de informação.

61. **(CESPE – 2004 – TRE/AL – Analista de Sistema)** O modelo cascata ou ciclo de vida clássico necessita de uma abordagem sistemática, que envolve, em primeiro lugar, o projeto e, em seguida, a análise, a codificação, os testes e a manutenção.

62. **(CESPE – 2008 – MPE/AM – Analista de Sistema)** O modelo de desenvolvimento sequencial linear tem como característica principal a produção de uma versão básica, mas funcional, do software desde as primeiras fases.

63. **(VUNESP - 2012 - SPTrans - Analista de Informática)** Uma das abordagens do processo de desenvolvimento da engenharia de software prevê a divisão em etapas, em que o fim de uma é a entrada para a próxima. Esse processo é conhecido como modelo:

a) Transformação.
b) Incremental.
c) Evolutivo.
d) Espiral.
e) Cascata.

64. **(CESGRANRIO – 2010 – PETROBRÁS – Analista de Sistemas – Processos de Negócio)** No Ciclo de Vida Clássico, também chamado de Modelo Sequencial Linear ou Modelo Cascata, é apresentada uma abordagem sistemática composta pelas seguintes atividades:

a) Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

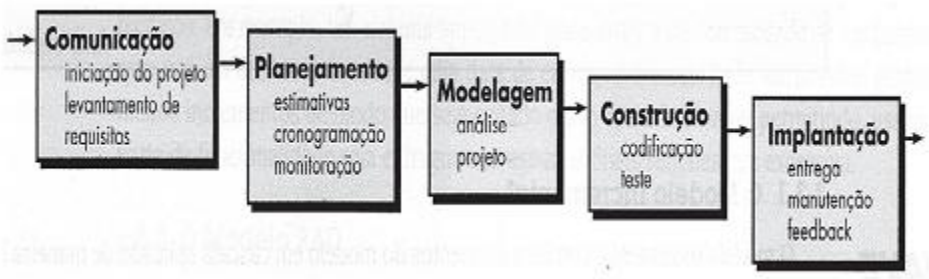
b) Modelagem e Engenharia do Sistema/Informação, Análise de Requisitos de Software, Projeto, Geração de Código, Teste e Manutenção.

c) Modelagem e Engenharia do Sistema/Informação, Projeto, Geração de Código, Teste e Manutenção.

d) Levantamento de Requisitos de Software, Projeto, Geração de Código e Manutenção e Análise de Requisitos de Software.

e) Levantamento de Requisitos de Software, Projeto, Geração de Código, Teste Progressivo e Manutenção.

65. **(FGV - 2015 – PGE/RO - Análise de Sistemas)** A figura abaixo ilustra um modelo de processo, que prescreve um conjunto de elementos de processo como atividades de arcabouço, ações de engenharia de software, tarefas, produtos de trabalho, mecanismos de garantia de qualidade e de controle de modificações para cada projeto.




Esse modelo é conhecido como Modelo:

a) por funções.
b) em cascata.
c) incremental.
d) em pacotes.
e) por módulos.

66. **(CESPE - 2016 – TCE/PR - Analista de Informática)** O modelo de desenvolvimento em cascata é utilizado em caso de divergência nos requisitos de um software, para permitir a evolução gradual do entendimento dos requisitos durante a implementação do software.

67. **(CESGRANRIO – 2012 – Transpetro – Analista de Sistemas Júnior – Infra-estrutura)** Na engenharia de software, existem diversos modelos de desenvolvimento de software, e, dentre eles, o modelo em cascata, o qual, no contexto do desenvolvimento de sistemas de software complexos, recomenda:

a) distribuir a elicitação dos requisitos desde o início até o fim do desenvolvimento.

b) dividir o desenvolvimento do produto de software em fases lineares e sequenciais.

c) enfatizar a avaliação e mitigação de riscos durante o desenvolvimento.

d) realizar entregas incrementais do produto de software ao longo do desenvolvimento.

e) usar prototipagem rápida para estimular o envolvimento do usuário no desenvolvimento.

68. **(CESGRANRIO – 2012 – EPE – Analista de Gestão Corporativa – Tecnologia da Informação)** Uma das críticas feitas ao modelo do ciclo de vida do desenvolvimento de software em cascata refere-se a:

a) exigência de conhecimento de avaliação e gerenciamento de risco para evitar grandes surpresas no projeto.

b) comprometimentos na qualidade e nas possibilidades de manutenção a longo prazo, parecendo um protótipo.

c) pouca flexibilidade para mudanças futuras, exigindo compromissos nas fases iniciais do projeto.

d) pouca visibilidade das etapas do processo, tornando cara a documentação de todas as versões dos sistemas.

e) exigências de velocidade as quais levam o engenheiro de software a utilizar linguagens, algoritmos ou ferramentas ineficientes ao longo de todo o projeto.

## MODELO EM CASCATA

69. **(CESPE - 2011 – TJ/ES - Analista Judiciário - Análise de Sistemas - Específicos)** O modelo de processo incremental de desenvolvimento de software é iterativo, assim como o processo de prototipagem. Contudo, no processo incremental, diferentemente do que ocorre no de prototipagem, o objetivo consiste em apresentar um produto operacional a cada incremento.

70. **(CESPE - 2008 – TJ/DF - Analista Judiciário - Análise de Sistemas)** No modelo de desenvolvimento incremental, embora haja defasagem entre os períodos de desenvolvimento de cada incremento, os incrementos são desenvolvidos em paralelo.

71. **(CESPE - 2009 – UNIPAMPA - Análise de Sistemas)** No modelo de desenvolvimento incremental, a cada iteração são realizadas várias tarefas. Na fase de análise, pode ser feito o refinamento de requisitos e o refinamento do modelo conceitual.

**72. (CESPE - 2016 – TCE/PR – Analista de Sistemas – C)** No modelo iterativo de desenvolvimento de software, as atividades são dispostas em estágios sequenciais.

## MODELO EVOLUCIONÁRIO

**73. (FGV – 2008 – Senado Federal – Analista de Sistemas)** No modelo evolucionário, a mudança constante tende a corromper a estrutura do software.

**74. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** A prototipação evolucionária não gera problemas de manutenção de sistema porque o desenvolvimento é rápido e não sofre grandes mudanças.

**75. (FCC – 2013 – DPE/SP – Analista de Sistemas)** No desenvolvimento de software, podem ser utilizados os chamados modelos evolucionários, cujo objetivo é lidar com produtos que possam evoluir ao longo do tempo. Assinale a alternativa que contém APENAS modelos evolucionários de desenvolvimento de software.

a) UML e de qualidade.
b) Componentes e arquétipo.
c) Prototipagem e espiral.
d) Redes de Petri e certificação.
e) Semântico e validação.

**76. (IESES – 2017 – CEGÁS – Analista de Sistemas)** Para Sommerville (2007) modelos evolucionários se caracterizam por sua iteratividade e permitem o desenvolvimento de versões de software cada vez mais completas. Assinale a alternativa que caracteriza os dois tipos processos mais comuns destes modelos:

a) RUP e Cascata.
b) Cascata e incremental.
c) RAID e Cascata.
d) Espiral e Prototipação.

**77. (CESPE – 2011 – FUB – Analista de Sistemas)** Os diversos modelos de processo de software disponíveis permitem a representação abstrata de um processo de software sob diferentes perspectivas. No modelo evolucionário, sob a perspectiva da arquitetura, a velocidade de desenvolvimento faz que a produção de documentos que reflitam cada versão do sistema seja economicamente inviável, gerando problemas na validação independente de sistemas.

**78. (CESPE – 2014 – MEC – Analista de Sistemas)** No desenvolvimento de software de grande porte, não se usam, em conjunto, os seguintes modelos de processo de software genéricos: modelo em cascata, desenvolvimento evolucionário e engenharia de software embasada em computador.

79. **(CESGRANRIO – 2014 – PETROBRÁS – Analista de Sistemas)** Um técnico de informática, com o objetivo de agilizar o desenvolvimento de um software, escolheu o desenvolvimento evolucionário, uma abordagem da área de Engenharia de Software, que:

a) facilita a produção de sistemas bem estruturados, privilegiando um processo de mudanças contínuas, cada vez mais fáceis e baratas.

b) se baseia na existência de um número significativo de componentes reusáveis, num processo de desenvolvimento que enfoca a integração desses componentes, em vez de desenvolvê-los a partir do zero

c) considera como atividades fundamentais do processo a especificação, o desenvolvimento, a validação e a evolução, representado-as como fases de processo separadas, sendo que para uma fase ser iniciada, a outra deve estar completamente terminada.

d) é mutuamente exclusiva com o modelo em cascata e de Engenharia de Software baseada em componentes, sendo que os subsistemas contidos em um sistema maior precisam ser desenvolvidos, usando a mesma abordagem ou modelo.

e) intercala as atividades de especificação, desenvolvimento e validação, permitindo que um sistema inicial seja desenvolvido rapidamente, baseado em especificações abstratas.

80. **(ESAF – 2009 – ANA – Analista de Sistemas)** O modelo de processo de software caracterizado por intercalar as atividades de especificação, desenvolvimento e validação, denomina-se:

a) modelo de workflow.
b) modelo de fluxo de dados.
c) desenvolvimento evolucionário.
d) transformação formal.
e) modelo em cascata.

81. **(FEPESE – 2014 – MPE/SC – Analista de Sistemas)** Assinale a alternativa abaixo que melhor identifica o modelo de processo de software no qual uma implementação inicial é exposta ao usuário para que possam ser realizados refinamentos posteriores que representam novas versões do sistema. As atividades de especificação, desenvolvimento e validação são intercaladas.

a) Relational Unified Process (RUP)
b) Desenvolvimento Evolucionário
c) Método Ágil de Desenvolvimento
d) Modelo de Desenvolvimento em Cascata
e) Modelo de Engenharia de Software Baseado em Componentes

**82. (FCC – 2014 – TRT/16 – Analista de Sistemas)** Os modelos de processo são uma representação abstrata de um processo de software, que podem ser usados para explicar diferentes abordagens para o desenvolvimento de sistemas. Analise as seguintes abordagens:

Desenvolvimento I : intercala as atividades de especificação, desenvolvimento e validação. Um sistema inicial é de- senvolvido rapidamente baseado em especificações abstratas e depois é refinado com as entradas do cliente para produzir um produto que o satisfaça.

Modelo II : considera as atividades fundamentais do processo, compreendendo especificação, desenvolvimento, validação e evolução e as representa como fases de processo separadas, tais como especificação de requisitos, projeto de software, implementação, teste etc.

III: baseia-se na existência de um número significativo de partes reusáveis. O processo de desenvolvimento do sistema enfoca a integração destas partes, ao invés de desenvolvê-las a partir do zero.

Os modelos de processo genéricos descritos em I, II e III são, correta e respectivamente, associados a:

a) em Espiral - Baseado em Componentes - RAD
b) Evolucionário - em Cascata - Baseado em Componentes
c) Baseado em Componentes - Sequencial - Refactoring
d) Ágil - Sequencial - Unified Process
e) em Cascata - Ágil - Refactoring

## MODELO EM PROTOTIPAGEM

**83. (CESPE – 2009 – UNIPAMPA – Analista de Sistemas)** No modelo de desenvolvimento prototipagem, um protótipo é desenvolvido para ajudar no entendimento dos requisitos do sistema.

**84. (CESPE – 2010 – TRE/MT – Analista de Sistemas - C)** A metodologia de prototipagem evolutiva é uma abordagem que visualiza o desenvolvimento de concepções do sistema conforme o andamento do projeto, por meio de protótipos visuais.

**85. (CESPE – 2004 – TRE/AL – Analista de Sistemas)** No modelo de prototipação, o desenvolvedor cria inicialmente um modelo de software que será posteriormente implementado.

**86. (CESPE – 2007 – TSE – Analista de Sistemas)** Um possível objetivo da prototipação é criar rapidamente um sistema experimental que possa ser avaliado por usuários finais. Um protótipo aprovado pelos usuários pode vir a ser usado como ponto de partida para a construção do sistema.

87. **(CESPE – 2008 – MPE/AM – Analista de Sistemas)** No modelo de prototipação, a especificação de requisitos tem pouca importância, pois o software é continuamente adaptado em função dos desejos do usuário.

88. **(CESPE – 2008 – TJ/DF – Analista de Sistemas)** Uma vantagem da prototipação é promover a participação e o comprometimento do usuário em relação ao sistema em desenvolvimento.

89. **(CESPE – 2008 – TJ/DF – Analista de Sistemas)** A prototipação de um software é uma técnica de desenvolvimento não-interativa porque o teste do sistema só ocorre na versão final.

90. **(CESPE – 2008 – TJ/DF – Analista de Sistemas)** Uma das finalidades da prototipação é reduzir o esforço de desenvolvimento de um software.

91. **(CESPE – 2010 – INMETRO – Analista de Sistemas – A)** Um dos benefícios da prototipação é a documentação normalmente gerada, que facilita a manutenção dos sistemas a longo prazo e a elaboração de casos de teste.

92. **(CESPE – 2010 – INMETRO – Analista de Sistemas – D)** Um dos riscos da prototipação é o usuário confundir o protótipo com o sistema verdadeiro e criar falsas expectativas com relação a prazos e recursos.

93. **(CESPE – 2010 – INMETRO – Analista de Sistemas – E)** Na abordagem evolutiva para desenvolvimento de software, um protótipo do software é produzido e utilizado para identificar possíveis problemas com os requisitos, sendo descartado logo em seguida, e o desenvolvimento do software propriamente dito é, então, iniciado.

94. **(CESPE – 2011 – MEC – Analista de Sistemas)** No modelo de prototipação, o processo de desenvolvimento de software é modelado como uma sequência linear de fases, enfatizando um ciclo de desenvolvimento de breve duração.

95. **(CESPE – 2013 – TRT/10 – Analista Judiciário – Tecnologia da Informação)** No modelo prototipação, a construção de software tem várias atividades que são executadas de forma sistemática e sequencial.

96. **(CESPE – 2009 – TCE/RN – Assessor Técnico de Informática)** A prototipação, uma abordagem para desenvolvimento de software na qual se cria um modelo do software que será implementado, é composta de quatro etapas: planejamento, análise de risco, engenharia e avaliação do cliente.

97. **(CESPE – 2009 – DETRAN/DF – Analista de Sistemas)** O modelo de processo de desenvolvimento de software evolucionário parte do desenvolvimento de uma implementação inicial cujos resultados são apresentados aos clientes e refinados por meio de várias versões até que se alcance o sistema adequado. A prototipação, como processo, tem

por objetivo compreender as especificações do software para se chegar aos requisitos para o sistema.

**98. (CESPE – 2008 – TJ/DF – Analista de Sistemas)** A prototipação evolucionária permite que a versão inicial do protótipo seja desenvolvida e refinada em estágios sequenciados, até que se chegue à versão final do sistema.

**99. (VUNESP - 2009 - CETESB - Analista de Tecnologia da Informação - Banco de Dados)** Considere um sistema cujos requisitos de interface são definidos apenas quando o cliente realiza um test-drive na aplicação e aprova essa interface. Assinale a alternativa que apresenta o modelo mais adequado para o desenvolvimento da interface desse sistema.

a) Ágil.
b) Cascata.
c) Iterativo incremental.
d) Prototipação.
e) Rapid Application Development.

**100. (FGV – 2010 – Fiocruz – Analista de Sistemas)** Como Modelo evolucionário do processo de software, uma característica da prototipagem é:

a) independer do estabelecimento e da definição de requisitos.
b) configurar um processo interativo e rápido de desenvolvimento.
c) iniciar o processo de desenvolvimento pela implantação e pelos testes.
d) gerar uma primeira versão do sistema completa e isenta de erros.
e) descartar a participação do cliente no processo de desenvolvimento e de implantação.

## 4 – GABARITO

1. CORRETO
2. LETRA A
3. LETRA E
4. ERRADO
5. LETRA B
6. CORRETO
7. LETRA D
8. LETRA E
9. LETRA A
10. CORRETO
11. ERRADO
12. LETRA A
13. CORRETO
14. CORRETO
15. CORRETO
16. LETRA D
17. LETRA A
18. CORRETO
19. ERRADO
20. ERADO
21. ERRADO
22. ERRADO
23. ERRADO
24. LETRA C
25. CORRETO
26. CORRETO
27. ERRADO
28. CORRETO
29. CORRETO
30. LETRA B
31. ERRADO
32. LETRA E
33. LETRA E
34. LETRA D
35. ERRADO
36. CORRETO
37. ERRADO
38. CORRETO
39. ERRADO
40. ERRADO
41. LETRA B
42. ERRADO
43. ERRADO
44. ERRADO
45. CORRETO
46. ERRADO
47. ERRADO
48. ERRADO
49. CORRETO
50. ERRADO
51. CORRETO
52. ERRADO
53. ERRADO
54. CORRETO
55. CORRETO
56. CORRETO
57. ERRADO
58. ERRADO
59. CORERTO
60. ERRADO
61. ERRADO
62. ERRADO
63. LETRA E
64. LETRA B
65. LETRA B
66. ERRADO
67. LETRA B
68. LETRA C
69. CORRETO
70. CORRETO
71. CORRETO
72. ERRADO
73. CORRETO
74. ERRADO
75. LETRA C
76. LETRA D
77. CORRETO
78. ERRADO
79. LETRA E
80. LETRA C
81. LETRA B
82. LETRA B
83. CORRETO
84. CORRETO
85. CORRETO
86. CORRETO
87. ERRADO
88. CORRETO
89. ERRADO
90. CORRETO
91. ERRADO
92. CORRTO
93. ERRADO
94. ERRADO
95. ERRADO
96. ERRADO
97. ERRADO
98. CORRETO
99. LETRA D
100. LETRA B

# ESSA LEI TODO MUNDO CONHECE:
# PIRATARIA É CRIME.

**Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.**

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

**Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.**