DOCUMENTAÇÃO E TESTES

André Feitoza de Mendonça

INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

DOCUMENTAÇÃO E TESTES

André Feitoza de Mendonça

INFORMAÇÃO E COMUNICAÇÃO

NT EDITORA

Autor

André Feitoza de Mendonça

Mestre em Ciência da Computação pela Universidade Federal de Pernambuco (2012) e laureado em Engenharia da Computação pela mesma Instituição (2010). Coordenador de desenvolvimento de sistemas voltados à automação do processo de aplicação de exames nacionais (ENEM, ENCCEJA, ENADE, por exemplo) no Instituto Nacional de Estudos e Pesquisas Anísio Teixeira. Foi Analista de processos com vistas à automatização do Processo Decisório do Conselho Deliberativo Nacional do SEBRAE.

Design Instrucional
Pedro Meneses
Sarah Rresende

Revisão
Filipe Lopes
Ricardo Moura

Editoração Eletrônica
Bruno Carneiro

Projeto Gráfico
NT Editora

Capa
NT Editora

Ilustração
Thiago Ferreira

NT Editora, uma empresa do Grupo NT
SCS Quadra 2 – Bl. C – 4° andar – Ed. Cedro II
CEP 70.302-914 – Brasília – DF
Fone: (61) 3421-9200
sac@grupont.com.br
www.nteditora.com.br e www.grupont.com.br

Mendonça, André Feitoza de.

Documentação e testes / André Feitoza de Mendonça – 1. ed. reimpr. – Brasília: NT Editora, 2018.

150 p. il. ; 21,0 X 29,7 cm.

ISBN 978-85-8416-286-4

1. Documentação. 2. Caso de uso. 3. História de usuário.

I. Título

LEGENDA

ÍCONES

Prezado(a) aluno(a),
Ao longo dos seus estudos, você encontrará alguns ícones na coluna lateral do material didático. A presença desses ícones o(a) ajudará a compreender melhor o conteúdo abordado e a fazer os exercícios propostos. Conheça os ícones logo abaixo:

Saiba mais
Esse ícone apontará para informações complementares sobre o assunto que você está estudando. Serão curiosidades, temas afins ou exemplos do cotidiano que o ajudarão a fixar o conteúdo estudado.

Importante
O conteúdo indicado com esse ícone tem bastante importância para seus estudos. Leia com atenção e, tendo dúvida, pergunte ao seu tutor.

Dicas
Esse ícone apresenta dicas de estudo.

Exercícios
Toda vez que você vir o ícone de exercícios, responda às questões propostas.

Exercícios
Ao final das lições, você deverá responder aos exercícios no seu livro.

Bons estudos!

Sumário

APRESENTAÇÃO

Caro(a) estudante,

Seja bem-vindo(a) a **Documentação e Testes**!

Os sistemas da informação têm se tornado cada vez mais importantes na vida das empresas e dos indivíduos. E os *softwares*, como você já deve saber, estão sendo empregados em áreas críticas, como a medicina. Nesse contexto, é essencial que tais sistemas estejam devidamente documentados e, consequentemente, testados para minimizar eventuais falhas.

Com efeito, o processo de desenvolvimento de *software*, durante muito tempo, não deu a devida importância para as atividades vinculadas à documentação e aos testes. A documentação garante o correto entendimento do funcionamento de um determinado sistema, e os artefatos denominados de documentos de requisitos detalham o comportamento do *software*. Se, por exemplo, esse artefato for mal escrito ou apresentar informações incorretas, projetos inteiros podem ser arruinados, uma vez que ajustes durante a codificação podem ser proibitivos. Por outro lado, um *software* não devidamente testado pode gerar diversos inconvenientes para os seus usuários e, até mesmo, danos financeiros para corporações.

Dessa forma, com o passar do tempo, a maneira de documentar os chamados requisitos funcionais de um *software* vem sofrendo diversas evoluções. Primeiramente, vale a pena destacar o uso de protótipos para a realização do levantamento das necessidades de negócio que serão automatizadas por meio de um sistema da informação. Essa prática aumentou a completude e a corretude das informações necessárias para que determinado *software* seja desenvolvido. Com relação à documentação propriamente dita, pode-se verificar o estabelecimento de um padrão, denominado caso de uso, para descrever as funcionalidades de um sistema.

Caso de uso: função do *software* que é realizada por um ou mais atores.

A documentação também deve se preocupar em descrever os requisitos não funcionais. Esses requisitos se relacionam com atributos de qualidade de um *software*, como o desempenho, a usabilidade, a disponibilidade, a segurança etc. Nesse tipo de situação, a documentação deve trazer de maneira clara qual métrica deve ser analisada para verificar se o requisito não funcional foi atendido ou não.

Os testes são realizados com base na documentação produzida para descrever os mais diversos aspectos de um *software*. Geralmente, essa atividade é executada por uma equipe diferente da de desenvolvimento. Isso se faz necessário para que pessoas não envolvidas no processo de codificação e, consequentemente, com uma visão não viciada sejam inseridas na verificação do sistema.

Em sistemas compostos por diversos módulos, é uma boa prática que a equipe de testes, inicialmente, concentre esforços para garantir o funcionamento correto desses componentes individualmente. Logo após, quando os módulos estiverem integrados, novos testes devem ser feitos de forma a detectar problemas ainda não encontrados.

Ultimamente, até mesmo os próprios desenvolvedores foram envolvidos no processo de garantia de qualidade por meio dos denominados testes unitários. Esse tipo de teste certifica que elementos básicos de codificação (como métodos e funções) estejam funcionando apropriadamente. Dessa forma, pode-se constatar que testes são empregados para verificar desde trechos de código até sistemas totalmente integrados.

Além de aspectos funcionais, testes podem ser utilizados para simular comportamentos que potencialmente podem ocorrer quando um *software* é disponibilizado para o seu efetivo uso. Por exemplo, existem os chamados testes de carga. Nessa abordagem, simula-se uma grande quantidade de acessos simultâneos a determinada parte do sistema e se verifica se o sistema responde corretamente.

Bons estudos!

André Feitoza de Mendonça

1 MODELOS DE DESENVOLVIMENTO DE *SOFTWARE*

Nesta lição, será apresentada uma descrição completa dos mais diversos modelos de desenvolvimento de *software*. Será explicitado também que a escolha de um modelo específico modifica consideravelmente a forma como a documentação de um sistema será confeccionada e o escopo da realização dos testes.

Objetivos

Ao final desta lição, você deverá ser capaz de:

- conceituar o modelo de desenvolvimento de *software*;
- diferenciar o modelo cascata e o modelo iterativo e incremental ;
- explicar qual é o processo básico de desenvolvimento de *software*;
- determinar o esforço relacionado à documentação e aos testes de um sistema, considerando abordagens distintas de modelos de processo de construção de *software*.

1.1 O que é um modelo de desenvolvimento de *software*?

Antes de começarmos a entrar nos detalhes sobre modelos de desenvolvimento de *software*, vale a pena destacar a diferença entre os termos modelo e metodologia. No decorrer do desenvolvimento de *software*, em diversas situações esses termos são usados indistintamente.

Dicas

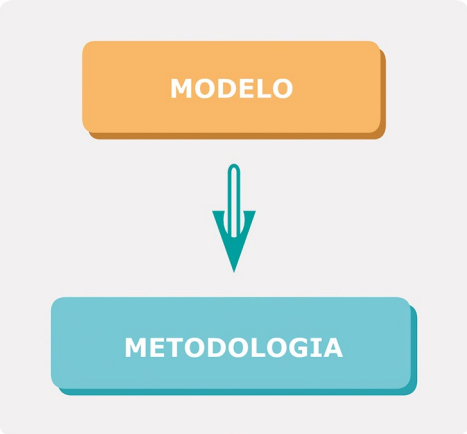
Um modelo de desenvolvimento determina o ciclo de vida do desenvolvimento de um *software*. Já um ciclo de vida de um projeto de sistema, por sua vez, indica a maneira com a qual as diversas fases de desenvolvimento são sequenciadas.

Saiba mais

É importante destacar a diferença entre ciclo de vida de desenvolvimento de *software* e ciclo de vida de um sistema da informação. O primeiro modelo abrange todas as etapas necessárias para que um *software* possa ser concebido, codificado, testado e, por fim, disponibilizado para os usuários finais para a sua efetiva utilização. Já o chamado ciclo de vida de sistema da informação engloba todas as fases elencadas anteriormente e, adicionalmente, traz também fases relacionadas à manutenção e à evolução do *software* já construído. Quando um sistema começa a ser usado, é comum que sugestões de melhoria ou, até mesmo, erros sejam encontrados. Nessas situações, esses ajustes não são considerados parte do projeto de efetiva construção do *software* e, consequentemente, não estão no ciclo de vida de desenvolvimento de um sistema da informação. Outro aspecto também observado no ciclo de vida de sistema é que, a longo prazo, é comum que o *software* apresente uma curva de declínio em sua utilização. Na maior parte das vezes, isso é provocado por uma defasagem tecnológica que faz com que o *software* tenha de ser substituído por outro.

Nesse contexto, uma metodologia apresenta os conceitos básicos e os principais valores que devem ser seguidos no desenvolvimento de um sistema. De uma forma geral, a metodologia apresenta boas práticas para que um *software* seja concebido corretamente. A figura a seguir ilustra a relação entre os dois conceitos apresentados.

Relacionamento entre modelo e metodologia.



Dessa forma, com base na figura, tem-se que cada modelo apresenta uma série de metodologias que seguem o seu paradigma. Para fins de exemplificação, por exemplo, podemos citar modelo iterativo e incremental. A metodologia RUP (que será detalhada no decorrer deste livro) segue a forma de sequenciamento das fases de desenvolvimento de *software* preconizada por esse modelo em questão.

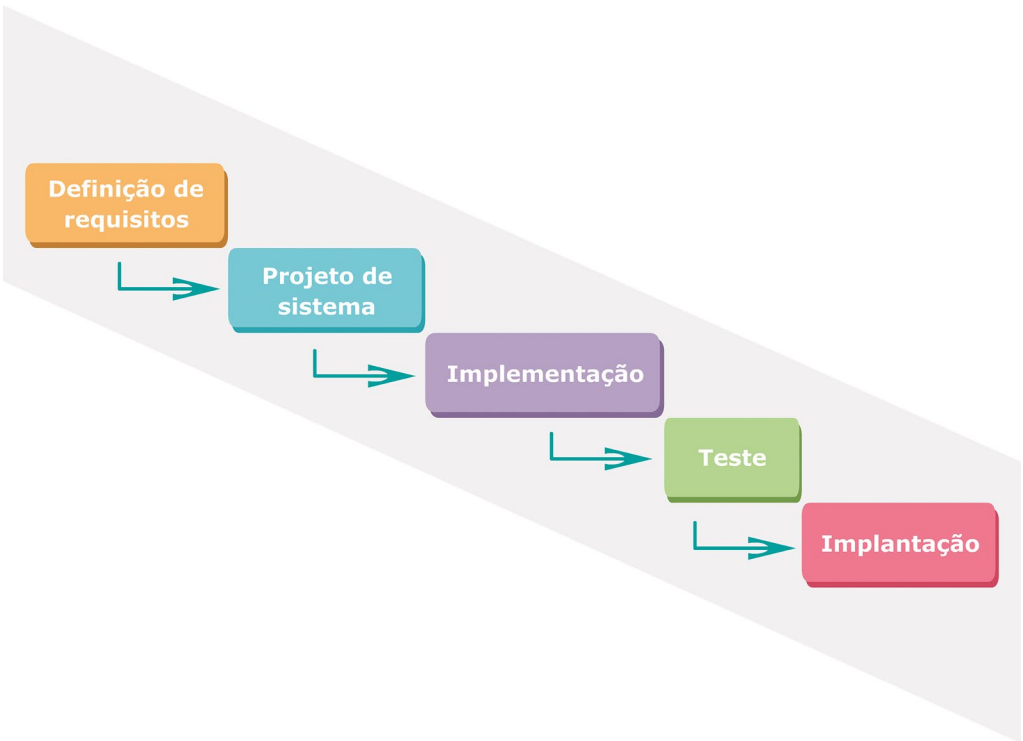
Agora, vamos analisar um fluxo básico de desenvolvimento de *software* e as principais fases necessárias para esse fim. Essas fases podem ser encontradas na maior parte dos modelos, embora sejam comuns alterações de nomenclatura dependendo de cada autor. Além disso, modelos diferentes também podem apresentar fases adicionais. A seguir, estão listadas essas fases e as mais importantes atividades realizadas em cada uma delas.

- Definição requisitos: estabelecimento do comportamento do sistema do ponto de vista de seus usuários. Nessa fase, são confeccionados documentos que nortearão todas as fases decorrentes.
- Projeto de sistema: criação de arquitetura do *software*. Ou seja, a partir dos requisitos, determinam-se quais módulos serão necessários para o seu correto funcionamento. Dependendo da metodologia escolhida, diagramas UML podem ser gerados nessa fase.
- Implementação: é a fase em que a codificação do sistema é realizada.
- Teste: uma vez que o *software* foi codificado, essa fase é responsável por garantir qualidade e satisfação do cliente.
- Implantação: são as atividades necessárias para que uma versão de *software* possa ser disponibilizada para a sua utilização pelos usuários.

Fluxo básico de desenvolvimento de *software*: etapas necessárias para que um incremento de *software* possa ser desenvolvido.

Perceba que existe uma sequência lógica para que as fases sejam executadas, ou seja, um fluxo básico de desenvolvimento de *software*. Assim, primeiramente, requisitos são estabelecidos e, em seguida, o projeto de um sistema é definido. Com base nos requisitos e no projeto, a implementação pode ser iniciada. Por fim, os testes e a implantação podem ser realizados. Observe a figura a seguir, em que esse sequenciamento é representado.

Fases do fluxo básico de desenvolvimento de *software*



Conforme mostrou a figura acima, esse fluxo, na maioria das vezes, não ocorre estritamente de maneira sequencial. No decorrer do projeto do sistema, por exemplo, pode ser verificado algum requisito inconsistente. Nessa situação, a fase de definição de requisitos é executada novamente. Já em outras situações, a estratégia de construção de um sistema pode envolver o desenvolvimento em

paralelo de diversos módulos. Nesse caso, diversos fluxos básicos de desenvolvimento podem ser iniciados paralelamente.

Nesse contexto, a escolha de um modelo de desenvolvimento de *software*, em última instância, determina o grau de esforço, ao longo do tempo, despendido em cada uma das fases do fluxo básico de desenvolvimento.

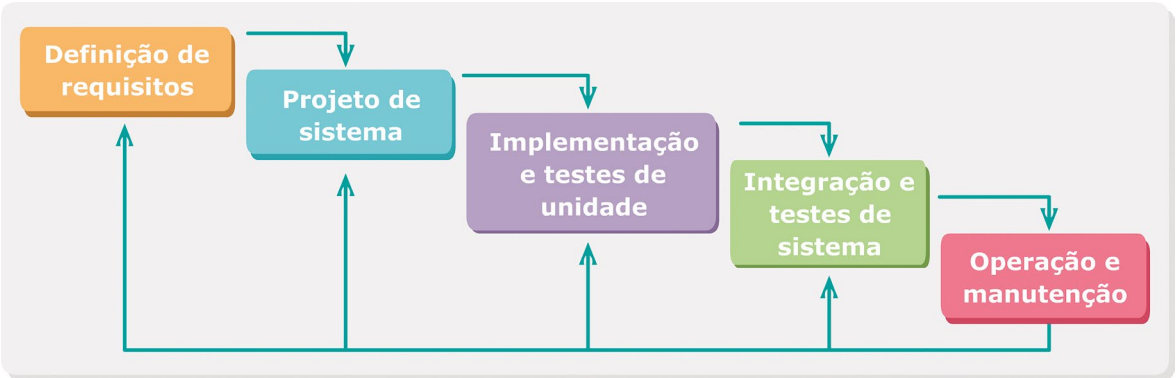
Modelo de desenvolvimento de *software*: ciclo de vida utilizado para realizar o desenvolvimento do *software*.

Modelo cascata: ciclo de vida em que todas as fases do fluxo básico de desenvolvimento são executadas apenas uma vez.

Vamos agora detalhar os mais utilizados modelos para conceber um sistema da informação. Nesta lição, realizaremos um estudo aprofundado dos modelos cascata, iterativo e incremental.

Primeiramente, o modelo cascata (também denominado ciclo de vida clássico ou tradicional) será analisado. Esse ciclo de vida preconiza que todas as fases de desenvolvimento de *software* são executadas apenas uma vez. Visualize, na ilustração a seguir, o fluxo de trabalho empregado por esse modelo, definido por Sommerville (2011).

Modelo cascata



Como você pôde observar, o modelo cascata indica um fluxo contínuo de trabalho. Ou seja, não é previsto que, por exemplo, durante a fase de implementação, requisitos sejam revisitados.

Importante

Ao fim do processo de desenvolvimento, esse modelo prevê uma fase responsável por manter o sistema operacional. Dessa forma, qualquer problema identificado nessa etapa pode desviar o processo para qualquer uma das fases anteriores. Se por acaso uma inconformidade nos requisitos for encontrada, toda a cascata é executada novamente, partindo da definição de requisitos.

Ao longo do tempo, muitos profissionais da tecnologia da informação encontraram problemas na utilização desse modelo. O principal deles é a alegação de que o processo de desenvolvimento de *software* é dinâmico e, na prática, apresenta diversas idas e vindas entre suas fases. Por exemplo, é extremamente comum que todos os requisitos de um sistema não sejam apropriadamente definidos e, no decorrer da implementação do código, ajustes tenham de ser realizados.

Outro ponto bastante discutido como uma desvantagem é que os usuários do sistema poderão utilizar o *software* por eles especificado apenas no fim do modelo cascata. Dessa forma, o risco de insucesso do projeto aumenta, uma vez que uma especificação mal compreendida só poderá ser verificada quando o sistema já estiver pronto.

Entretanto, cabe ressalta que o modelo cascata ainda hoje é utilizado para algumas situações. Dessa forma, para os casos de sistemas da informação críticos, esse modelo ainda é usado. Nesse tipo de *software*, falhas podem ser catastróficas, podendo gerar, até mesmo, mortes. Assim, cada etapa do modelo cascata é executada por meio de um processo rígido, que tem o objetivo de

detectar eventuais inconsistências. Uma fase só é finalizada quando se tem grande certeza de que todos os trabalhos foram concluídos com sucesso e sem erros. Por exemplo, o desenvolvimento de um *software* de controle de tráfego aéreo poderia se encaixar nesse modelo. Esse tipo de sistema pode perfeitamente ser desenvolvido pelo modelo cascata, tendo em vista que seus requisitos são estáveis e não são modificados com o tempo.

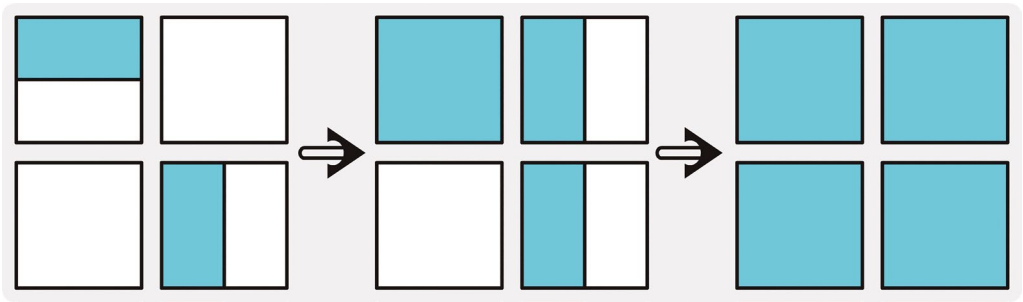
Saiba mais

No decorrer do desenvolvimento de *software*, requisitos podem ser modificados por alterações das necessidades do negócio, por requisitos legais ou, até mesmo, por restrições tecnológicas. Toda mudança de requisito sistema gera um custo de retrabalho do processo de desenvolvimento e, quanto mais tarde for a detecção dessa alteração, mais cara ela será. Dessa forma, se um requisito for alterado ainda no momento de sua definição, os custos adicionais para o projeto serão mínimos. Entretanto, se, em outra situação, um mesmo requisito for modificado mais à frente no momento dos testes, o desperdício de recursos financeiros serão consideravelmente maiores, tendo em vista que a especificação de requisitos terá de ser reescrita, eventualmente a arquitetura do sistema poderá ser modificada, o código sofrerá modificação e, por fim, novos testes deverão ser executados.

Nesse momento, o modelo iterativo e incremental será detalhado. Para uma melhor compreensão da dinâmica desse modelo, realizaremos a definição dos processos de desenvolvimento iterativos e incrementais isoladamente.

Processos iterativos possuem uma abordagem de realizar diversas repetições (iterações) do fluxo básico de desenvolvimento. A cada iteração, uma nova versão de sistema é entregue e, consequentemente, o *software* é progressivamente construído. Dessa forma, pode-se afirmar que, após uma iteração intermediária, um *software* não pronto para uso pode ser gerado. Para exemplificação, suponha que uma funcionalidade de um sistema preveja um relató rio com três filtros. É possível que, após uma iteração, esse relatório seja disponibilizado com apenas um filtro. Observe a figura a seguir em que o desenvolvimento iterativo é representado.

Representação da dinâmica do modelo iterativo



A figura acima apresenta três iterações de desenvolvimento de um *software* com quatro módulos (simbolizados pelos quatro quadrados). Para esse sistema em questão, cada módulo é independente dos demais e pode ser disponibilizado para os seus usuários isoladamente.

Repare que, após a primeira iteração, uma versão de sistema com apenas a metade das funcionalidades do primeiro e do quarto módulos é entregue. Em seguida, no segundo ciclo de desenvolvimento, o primeiro módulo é finalizado e o segundo é apenas iniciado. Por fim, na última iteração ilustrada, os quatro módulos são concluídos por completo.

Ainda com relação à figura anterior, verifique que não existiu uma ordem de início da construção de cada módulo. Logo na primeira iteração, o primeiro e o quarto módulo tiveram o seu desenvolvimento iniciado sem a devida conclusão. Enquanto, na segunda iteração, você pôde verificar que um novo módulo teve o seu fluxo de desenvolvimento posto em andamento sem a prévia finalização do módulo quarto.

Importante

Dessa forma, nota-se que, no modelo iterativo, não existe uma preocupação em finalizar a construção de partes de *software* que possam ser utilizados pelos usuários finais. O importante, nesse modelo, é dividir o esforço de desenvolvimento em iterações e, após cada uma delas, o código ser evoluído.

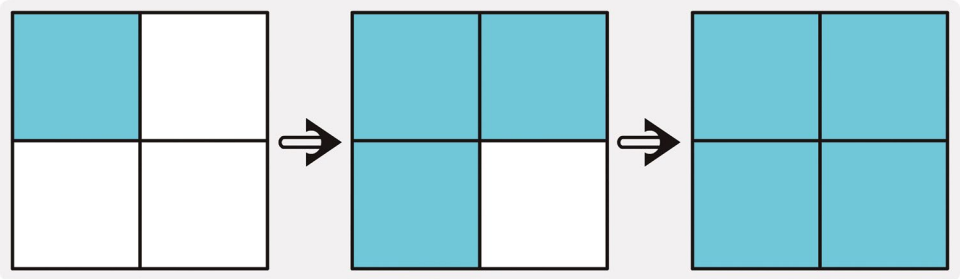
Após o detalhamento do modelo iterativo, você terá a oportunidade de conhecer os conceitos envolvidos no modelo incremental.

Saiba mais

De uma forma resumida, o modelo incremental, como o próprio nome diz, indica que o processo de desenvolvimento deve prever diversos incrementos de *software*. Um incremento, por sua vez, é uma versão de um sistema que possui um subconjunto de funcionalidades concluídas. Todo incremento pode ser disponibilizado para o usuário do sistema e, consequentemente, ser disponibilizado para o ambiente de produção sem problemas.

Agora, vejamos a figura que mostra um quadro de evolução de um processo de desenvolvimento seguindo o modelo incremental.

Representação da dinâmica do modelo incremental



Modelo iterativo: ciclo de vida em que são realizadas diversas repetições do fluxo básico de desenvolvimento no decorrer da construção do *software*.

Modelo incremental: ciclo de vida que prevê diversas entregas (ou incrementos) no decorrer da construção do *software*.

Note que, na figura acima, um incremento disponibilizado sinaliza uma entrega de uma parte do sistema de forma completa. A estratégia, nesse caso, não é dividir o *software* em módulos. Dessa forma, toda entrega no modelo incremental apresenta um subconjunto de funcionalidades que podem ser disponibilizadas para o uso do usuário final.

Testando o conhecimento

Assinale a alternativa que traz a informação correta sobre o modelo iterativo e o modelo incremental.

a) O modelo iterativo sempre disponibiliza uma versão que pode ser utilizada pelo usuário final.

b) No modelo incremental, iterações são realizadas e, ao fim de cada uma delas, diversas partes do sistema podem ter sido desenvolvidas e não concluídas.

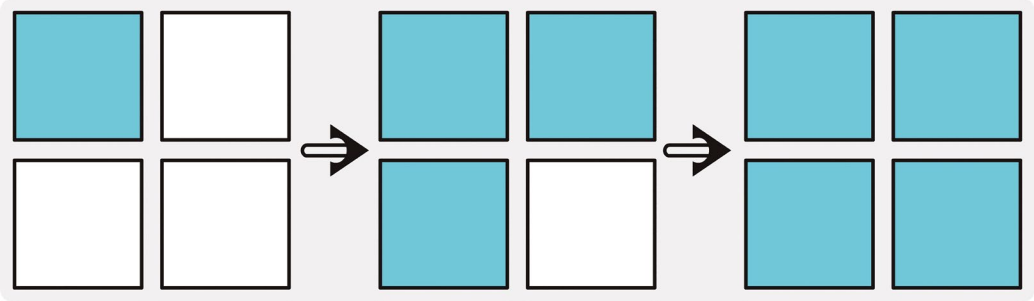
c) O modelo cascata e o modelo incremental são muito semelhantes, uma vez que, em ambos ciclos de desenvolvimento, um módulo só pode ser entregue após a conclusão dos demais que já estavam em andamento.

d) O modelo iterativo deixa livre que diversos módulos sejam iniciados na mesma iteração sem a devida conclusão ao fim do ciclo.

Comentário: no modelo incremental, diversas versões de *software* funcionais são disponibilizadas como incrementos. Logo, a alternativa "b" está incorreta. Ao contrário do que afirma a alternativa "c", o modelo cascata não prevê entregas parciais de *software*. Por fim, pode se constatar que é possível que o modelo iterativo realize uma entrega intermediária do *software* não totalmente pronto para uso. Logo, a alternativa "a" está errada e, consequentemente, a resposta correta é a letra "d".

Por fim, vamos analisar o modelo de desenvolvimento de *software* que é, ao mesmo tempo, iterativo e incremental. Verifique, na figura a seguir, a evolução na construção de um sistema estabelecida por tipo esse modelo.

Representação da dinâmica do modelo iterativo e incrementa



Observe que as principais características dos ciclos iterativos e incrementais são preservadas. Se, por um lado, o desenvolvimento é realizado por meio de iterações; por outro, a cada uma dessas repetições, incrementos funcionais do *software* são entregues para os usuários finais.

Dicas

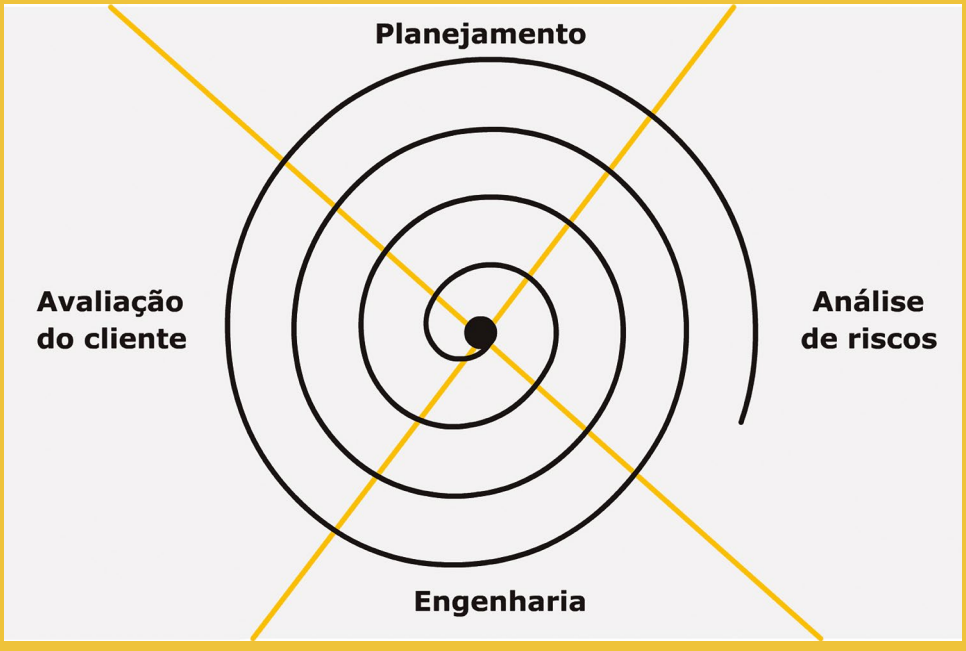
Atualmente, a maior parte dos *softwares* é desenvolvida sob o modelo iterativo e incremental. Requisitos não estáveis e a falta de conhecimento prévio de todo o escopo do sistema explicam a preferência por esse modelo. Dessa forma, a cada iteração, os clientes do sistema podem experimentar incrementos funcionais e relatar problemas antes do início da próxima iteração. Isso reduz o risco de falha do projeto, além de possibilitar o uso de partes do sistema antes de sua efetiva conclusão.

Vale a pena destacar também que o modelo iterativo e incremental é classificado como evolucionário. Modelos de desenvolvimento evolucionários são aqueles que preveem entregas no decorrer do tempo. A cada uma delas, o *software* é avaliado e eventuais melhorias ou erros são incorporados no escopo da próxima entrega.

Saiba mais

Além dos modelos apresentados nesta lição, vale a pena também analisarmos o ciclo de vida de desenvolvimento espiral. Assim como o modelo iterativo e incremental, o ciclo espiral também é classificado como evolucionário, ou seja, o *software* é entregue paulatinamente e, a cada disponibilização, os resultados são avaliados e levados em conta para a próxima entrega.

Nesse modelo, o fluxo de desenvolvimento se inicia no centro da espiral. Dessa forma, o processo de construção passa por vários *loops* que são compostos por quatro fases:



planejamento, análise de riscos, engenharia e avaliação do cliente. Assim, em cada volta na espiral, é realizado um planejamento da próxima entrega e, logo em seguida, os diversos riscos envolvidos são analisados. Após, a codificação se dá por meio da fase de engenharia e, por fim, os clientes avaliam os resultados. Esse *feedback* é um insumo para o planejamento do próximo *loop* da espiral. Ao percorrer toda a espiral, o *software* é entregue com todas as suas funcionalidades previstas.

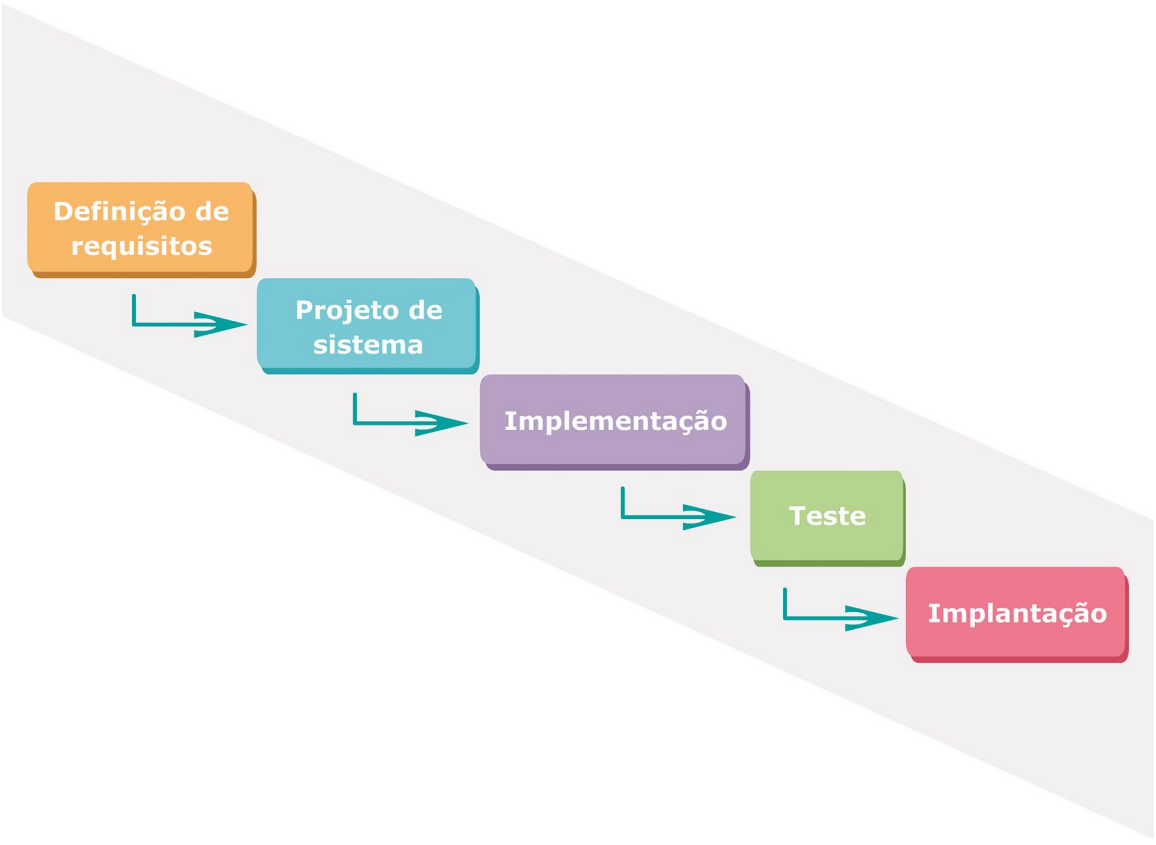
Testando o conhecimento

Assinale a alternativa que traz a informação correta sobre o modelo iterativo e incremental.
a) O modelo indica que, a cada iteração, um incremento de *software* é disponibilizado.
b) Não está previsto por esse modelo que, no decorrer do desenvolvimento de *software*, entregas sejam realizadas por meio de novos incrementos de sistema que podem ser utilizados.
c) O modelo iterativo e incremental, atualmente, está entrando em desuso por conta de sua estrutura rígida.
d) Nesse modelo, o fluxo básico de desenvolvimento é executado apenas uma vez.

Comentário: o modelo iterativo e incremental, ao contrário do que se afirma na alternativa "c", atualmente, está cada vez mais sendo utilizado como uma abordagem viável para a construção de sistemas. Nesse modelo, o fluxo básico de desenvolvimento é executado diversas vezes até que o *software* esteja pronto. Logo, a alternativa "d" está errada. O modelo iterativo e incremental estabelece que o *software* seja entregue em diversos incrementos por meio de iterações. Assim, a alternativa "b" está errada e, portanto, a resposta correta é a alternativa "a".

1.2 Documentação e testes x modelos de desenvolvimento

Iniciaremos este tópico com a ilustração a seguir, que apresentamos no início dos nosso estudos. Perceba que nela existe fluxo básico necessário para que um *software* (ou parte dele) seja desenvolvido.



Esse processo que você pôde analisar se inicia com o estabelecimento dos requisitos que definem as funcionalidades do sistema que está sendo projetado. Por sua vez, esses requisitos são formalizados por meio de uma documentação que pode ser extensa ou não. Por outro lado, no fim desse fluxo básico, testes são realizados de forma a garantir a qualidade do *software* entregue. Em diversas situações, dá-se pouca ênfase a essas etapas do processo de construção, mas traremos para você a oportunidade de aprender em detalhes os principais conceitos e técnicas relacionados à documentação e aos testes.

Entretanto, antes de iniciarmos um estudo mais detalhado sobre tais temas nas próximas lições, entenderemos como os modelos de desenvolvimento, explicitados até aqui, influenciam os trabalhos de documentar e testar um sistema.

Primeiramente, iremos detalhar como a documentação e os testes são abordados no modelo cascata. Como já descrito, esse ciclo de desenvolvimento preconiza que o fluxo básico de desenvolvimento ocorra apenas uma vez na construção de um sistema.

Importante

Só são admitidos retornos ao fluxo se, quando o *software* estiver em uso, forem encontradas falhas ou comportamentos não esperados.

Diante do exposto, a documentação de um sistema desenvolvido por um processo cascata deve ser elaborada sem a possibilidade de revisões solicitadas por outras fases do fluxo. Portanto, diante dessa falta de flexibilidade, algumas preocupações devem nortear a elaboração dessa documentação. Verifique cada uma delas a seguir:

- alto nível de detalhamento: a documentação deve contemplar a maior quantidade de detalhes sobre o funcionamento de um determinado sistema. Não só os fluxos principais, como também fluxos de execução alternativos e exceções devem ser previstos e especificados. Além disso, todos os usuários do sistema devem ser escutados e as suas respectivas opiniões sobre o comportamento do *software* devem ser levadas em conta;
- especificação de requisitos não funcionais: além do comportamento do sistema, a documentação deve prever com clareza quais são os seus requisitos não funcionais. Requisitos não funcionais influenciam a forma como as funcionalidades de um *software* são disponibilizadas para os seus usuários. Geralmente, tais requisitos se relacionam com desempenho, confiabilidade, segurança, usabilidade, entre outros. Para o modelo cascata, a documentação deve prever com clareza indicadores que sinalizem se um requisito não funcional foi contemplado com sucesso ou não;
- validação de documentação: antes que a fase de definição de requisitos seja finalizada no modelo cascata, toda a documentação deve passar por um processo completo de validação. Os futuros usuários do sistema devem ratificar o conteúdo descrito. Além disso, deve ser analisado se a especificação apresentada na documentação pode ser testada. Caso essa verificação seja positiva, isso indica que existe uma alta probabilidade de que a etapa de codificação ocorrerá sem maiores dúvidas.

No modelo cascata, os testes são realizados nas fases de implementação e de integração. Durante a implementação, são feitos os chamados testes unitários. Tais testes servem para garantir o funcionamento correto de unidades básicas de um determinado programa, como, por exemplo, métodos ou funções. Dessa forma, o teste unitário verifica se determinada saída de uma função está correta. Por outro lado, no decorrer da integração das diversas partes do software, teste de sistema ocorrem. Esse tipo de teste avalia o comportamento de todas as partes construídas isoladamente de forma integrada.

Importante

Assim como o processo de elaboração de documentação, os testes de sistema ficam concentrados em apenas uma fase da cascata. Isso aumenta os desafios e a complexidade do processo de garantia de qualidade do sistema. Dessa forma, uma documentação sem erros e imprecisões é fundamental para o sucesso dessa etapa. Caso eventuais falhas no *software* sejam críticas para os seus usuários, os testes de sistema podem ser demorados e muito custosos.

Adicionalmente, testes de sistema devem considerar também os denominados requisitos não funcionais. Assim, por exemplo, se houver um requisito de desempenho que sinalize que determinado relatório tem de ser emitido em, no máximo, cinco segundos, essa restrição temporal deve ser analisada.

Dicas

Nesse tipo de situação descrita acima, deve-se levar em conta a estimativa de usuários desse relatório. Caso exista a previsão de um grande número de acessos simultâneos, os denominados testes de carga devem ser empregados; e o desempenho do sistema, checado.

Agora, as atividades envolvidas na documentação e nos testes serão analisadas considerando o modelo iterativo e incremental. Ao contrário do que ocorre no ciclo cascata, a documentação é evoluída a cada iteração. Ou seja, não existe a preocupação de que os requisitos de um sistema sejam totalmente especificados de uma vez só. A cada iteração, somente a documentação referente ao incremento esperado é confeccionada. Dessa forma, as especificações de *software* são escritas de forma evolutiva, possibilitando, até mesmo, que requisitos previamente elaborados e desenvolvidos possam ser revistos.

Em comparação com o modelo cascata, as preocupações elencados relacionadas à documentação são relativizadas. Por exemplo, não é necessário um alto nível de detalhamento na descrição dos requisitos. No modelo iterativo e incremental, em diversas situações, é preferível que a documentação seja disponibilizada o mais rápido possível para as demais fases do fluxo básico de desenvolvimento. Outro ponto bastante interessante é que o nível de detalhamento é aumentado a cada iteração. Dessa forma, à medida que incrementos de *software* são finalizados e os clientes avaliam essas entregas, um maior conhecimento do sistema é obtido, o que é refletido na documentação.

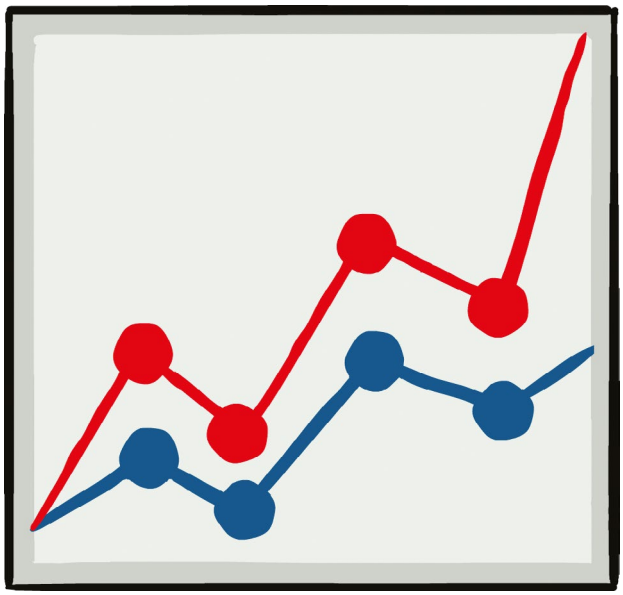
Dicas

Ainda com relação à documentação, nesse modelo evolucionário, não é obrigatório que os requisitos não funcionais sejam especificados de uma vez só.

Em diversas situações, os próprios usuários não conhecem ao certo quais são esses requisitos e, muitas vezes, somente após algumas iterações, restrições não funcionais podem ser melhores estabelecidas.

Da mesma forma, os testes realizados no ciclo de desenvolvimento iterativo e incremental são realizados progressivamente. Cada incremento de *software* entregue é testado após cada iteração. Portanto, ao contrário do que é preconizado no modelo cascata, os esforços para avaliação de qualidade são distribuídos ao longo do ciclo de construção do sistema.

Modelo evolucionário: tipo de modelo de desenvolvimento de *software* em que partes do *software* são disponibilizadas e, a cada entrega, o cliente avalia os resultados e esse *feedback* é levado em conta para as próximas entregas.



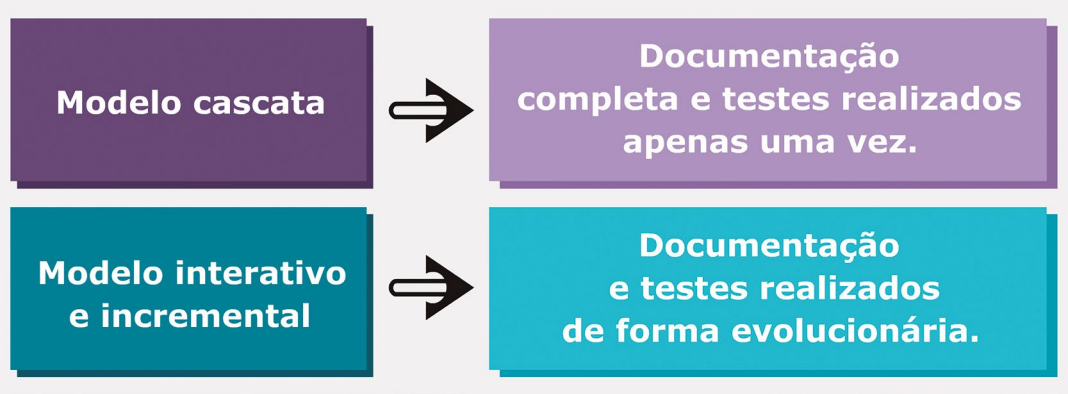
Além disso, no decorrer da etapa de codificação, existe uma preocupação de que uma nova iteração não introduza erros em funcionalidades já entregues anteriormente. Esse risco é minimizado pelo uso dos chamados testes unitários.

Importante

Cada incremento deve possuir uma série desses testes. Se, por acaso, um código que está sendo desenvolvido gerar efeitos colaterais negativos em outros módulos já finalizados, provavelmente, testes unitários irão falhar e sinalizar que correções precisam ser realizadas. Em algumas situações, nem mesmo é possível a criação de uma nova versão se ao menos um desses testes falharem.

Analise a figura a seguir,que resume bem os aspectos da documentação e dos testes considerando o modelo de desenvolvimento de *software*.

Característica da documentação e testes, de acordo com o modelo de desenvolvimento



O modelo iterativo e incremental utiliza-se do planejamento em ondas sucessivas. Esse tipo de planejamento indica que, nas primeiras iterações, é aceitável que detalhes do sistema que está sendo construído ainda não sejam conhecidos. Entretanto, à medida que as iterações são executadas, um maior conhecimento é obtido sobre o problema que está sendo resolvido pelo *software*. Ou seja, a cada iteração uma nova onda de planejamento é realizada, sempre buscando maiores informações.

Testando o conhecimento

Marque a alternativa que indica de forma correta como a documentação e os testes são tratados no modelo cascata.
a) A documentação é elaborada de forma evolucionária.
b) Apenas testes de sistema são realizados no modelo cascata. Testes unitários são empregados somente no modelo iterativo e incremental.
c) A documentação é elaborada de maneira ampla. Dessa forma, diversos atores são consultados, todos os fluxos são considerados e requisitos não funcionais são especificados em detalhes.
d) Testes de sistemas são aplicados a cada incremento disponibilizado.

Comentário: o modelo cascata estabelece que todo o escopo de determinado sistema seja considerado no fluxo básico de desenvolvimento. Portanto, testes de sistema são realizados sobre o conjunto total de funcionalidades do *software*. Logo, a alternativa "d" está errada. Além disso, os unitários também são empregados no modelo cascata, o que indica que a alternativa "b" está errada. No modelo cascata, a documentação é elaborada de uma vez só. Dessa forma, a alternativa "a" está incorreta. Por fim, é verdade que a documentação de sistema é elaborada de maneira ampla, levando em conta a maior quantidade de detalhes possível. Logo, a alternativa "c" está certa.

Resumindo

Nesta lição, você teve a oportunidade de aprender em detalhes o conceito de modelo de desenvolvimento de *software*. Assim, um modelo de desenvolvimento determina o ciclo de vida do desenvolvimento de um *software*. Já um ciclo de vida de um projeto de sistema, por sua vez, indica a maneira como as diversas fases de desenvolvimento serão sequenciadas.

Em seguida, o fluxo básico de desenvolvimento foi estabelecido. De uma forma geral, esse fluxo determina quais são as etapas necessárias para que um sistema (ou partes funcionais dele) seja disponibilizado para o efetivo uso. Assim, o modelo de desenvolvimento, em última instância, determina qual é o grau de esforço, ao longo do tempo, que cada fase desse fluxo despende.

No decorrer do nosso estudo, o modelo cascata e o modelo iterativo e incremental foram detalhados. De uma forma resumida, o ciclo de desenvolvimento em cascata prega que, após a finalização de uma etapa do processo, o fluxo segue adiante sem a possibilidade de retornos e, consequentemente, sem retrabalho. Já o modelo iterativo e incremental adota uma estratégia totalmente diferente. Nesse modelo, o *software* é concebido em forma de incrementos disponibilizados a cada iteração.

Por fim, foi realizada uma comparação dos esforços requeridos de documentação e testes, levando em conta o modelo de processo escolhido. Pode-se observar que a abordagem no modelo cascata é bem mais completa e, consequentemente, mais lenta. Toda a documentação e os testes devem ser finalizados no decorrer de um único fluxo de desenvolvimento, já que retrabalhados não são permitidos. Por outro lado, o modelo iterativo e incremental flexibiliza o processo, fazendo com que, a cada iteração, a documentação fique cada vez mais detalhada e os testes com uma cobertura cada vez maior.

Veja se você se sente apto a:

- explicar o conceito de modelo de desenvolvimento de *software*;
- diferenciar o modelo cascata e o modelo iterativo e incremental ;
- demonstrar qual é o processo básico de desenvolvimento de *software*;
- determinar o esforço relacionado à documentação e aos testes de um sistema, considerando abordagens distintas de modelos de processo de construção de *software*.

Parabéns, você finalizou esta lição!

Agora responda às questões ao lado.

Exercícios

Questão 1 – Assinale a alternativa que traz o conceito correto de modelo de desenvolvimento de *software*.

a) Uma lista de procedimentos que norteiam a construção de um sistema.

b) Atividades e boas práticas que devem ser utilizadas na fase de elaboração do código.

c) O modelo de desenvolvimento determina o ciclo de vida de construção de um *software*.

d) É a metodologia que o desenvolvimento de *software* segue.

Questão 2 – O que é um fluxo básico de desenvolvimento de *software*?

a) Nada mais é do que o modelo de desenvolvimento de *software*.

b) O fluxo básico determina quais são as etapas necessárias para que um *software* ou parte dele seja entregue.

c) A metodologia não tem nenhuma influência sobre a forma com que o fluxo básico é executado.

d) Conjunto de procedimentos de codificação.

Questão 3 – Assinale a alternativa que traz uma característica do modelo incremental.

a) Todo o desenvolvimento é realizado de uma vez só.

b) O desenvolvimento é dividido em iterações que podem gerar versões de sistema que podem não estar prontas para uso.

c) O desenvolvimento é dividido em repetições que podem gerar versões de sistema que sempre podem ser utilizadas pelos usuários finais.

d) Toda versão disponibilizada é um incremento de *software* pronto para uso.

Questão 4 – Marque a alternativa que traz a informação correta sobre testes unitários

a) Testes aplicados a módulos prontos.

b) São criados no momento da integração de módulos prontos.

c) Para cada cem linhas de código, um teste unitário deve ser elaborado.

d) São criados no momento da implementação e servem para avaliar se uma função ou método respondem corretamente.

Questão 5 – Marque a alternativa que traz um exemplo de metodologia de desenvolvimento.

a) Cascata.

b) Incremental.

c) Iterativo.

d) RUP.

Questão 6 – Assinale a alternativa que indica as etapas do modelo cascata (na ordem em que são executadas).

a) Definição de requisitos, projeto de sistema e testes unitários, implementação, integração e testes de sistema e implantação.

b) Definição de requisitos, projeto de sistema, implementação e testes de unidade, integração e testes de sistema e operação e manutenção.

c) Definição de requisitos, projeto de sistema, implementação e testes de unidade, integração e testes de sistema e implantação.

d) Definição de requisitos, projeto de sistema, implementação, integração e testes unitários, operação e manutenção.

Questão 7 – Assinale uma preocupação que deve ser tomada no momento da elaboração de documentação de requisitos no modelo cascata.

a) A documentação deve ser validada de forma a verificar se testes podem ser realizados.

b) As especificações devem ser escritas como histórias de usuário.

c) Os requisitos não funcionais somente devem ser especificados ao fim do desenvolvimento do sistema.

d) A documentação não deve ser detalhada para não aumentar a complexidade do desenvolvimento.

Questão 8 – Com relação ao modelo espiral, indique a alternativa que traz a informação correta.

a) Esse modelo é considerado não evolucionário e, portanto, todo sistema é entregue de uma vez só.

b) O ciclo de vida espiral é evolucionário e tem como característica a redução de riscos em longo do tempo.

c) Esse modelo não colhe *feedback* dos clientes.

d) O ciclo de vida espiral é evolucionário e tem como característica a produção de incrementos ao longo do tempo.

Questão 9 – Com relação ao modelo iterativo e incremental, indique a alternativa que traz a informação correta.

a) Esse modelo gera, a cada iteração, um incremento de *software* que é utilizado pelos clientes para fins de avaliação.

b) É um modelo semelhante ao modelo cascata.

c) Não um modelo evolucionário

d) É um modelo muito pouco flexível e, portanto, está entrando em desuso ultimamente.

Questão 10 – Assinale a alternativa que traz a informação correta sobre testes de sistema.

a) Esse tipo de teste é realizado no momento da integração de módulos de *software* finalizados. Assim, o sistema como um todo é analisado.

b) Testes de sistema são elaborados antes dos testes unitários.

c) Eles têm como principal objetivo garantir a qualidade de módulos de sistema isoladamente.

d) Apenas o *software* é avaliado sem a preocupação com aspectos de banco de dados.

Gabarito: 1c; 2b; 3d; 4d; 5d; 6b; 7a; 8b; 9a; 10a.