Engenharia de Software

Um processo de software é um conjunto de atividades relacionadas, que levam à produção de um produto de software.

4 etapas fundamentais:

- Especificação de software (validação de requisitos).
- Projeto e implementação de software (projeto de arquitetura).
- Validação de software (testes unitários).
- Evolução de software (manutenção de software).

- **Produtos (ou artefatos):** São o resultado de determinada atividade do processo.
- **Papéis:** Trabalho dos envolvidos em uma ou mais etapas do processo.
- **Condições preliminares e as condições posteriores aos resultados das etapas:** Uma etapa prevê a criação de uma lista de funcionalidades do software a ser construído, o que constitui a condição preliminar. Após a criação dessa lista, apura-se e descreve-se o resultado como condição posterior.

Modelos de Processos

Entendemos que o modelo em cascata é um processo de software, pois se utiliza de uma sequência de etapas (ao invés de uma única ação) para o atingimento de seu objetivo. Os processos contêm divisões em sua estrutura e, a fim de entendermos melhor um processo de software, convém analisarmos duas delas, de acordo com Wazlawick.

Fases

Um conjunto de atividades afins e com objetivos bem definidos são realizados em uma fase do processo. O modelo em cascata, por exemplo, apresenta fases bem definidas, quais sejam a fase dos requisitos, a fase do projeto, a da implementação e assim por diante.

Atividades ou Tarefas

Em suas regras processuais, a organização pode determinar que seja adotado um documento oque descreva a atividade. Por meio dele, a equipe tomará conhecimento da tarefa, de seus responsáveis, dos objetivos, dos recursos a serem utilizados e de tudo o que a caracteriza por completo.

Sabemos até o momento que um processo é um conjunto disciplinado e articulado de tarefas que serve para sistematizar o desenvolvimento de um software.

Há certos modelos de processos ditos prescritivos, que contêm descrições de como as atividades são realizadas. O modelo cascata, também conhecido como modelo tradicional, é o mais conhecido e ainda bastante utilizado para desenvolvimento de produtos de software. Ele descreve, por meio de etapas bem definidas, o ciclo que o software cumprirá durante o período compreendido entre sua concepção e sua descontinuidade.

Ian Sommerville é um acadêmico britânico e autor de um livo que se tornou referência em Engenharia de Software. Nessa obra chamada simplesmente Engenharia de Software, ele coloca o modelo em cascata sob a perspectiva de um modelo de processo de software e apresenta uma visão interessante sobre aspectos da evolução do software e da relação dos requisitos com esse modelo.

Projeto Scrum

Em meados dos anos 1980, Hirotaka Takeuchi e Ikujiro Nonaka definiram uma estratégia flexível e abrangente de criação de um produto, na qual a equipe de desenvolvimento trabalhava como uma unidade para atingir um objetivo comum. Eles descreveram uma abordagem inovadora para o desenvolvimento de artefatos, a qual eles chamaram de “abordagem rugby”. A partir dela, uma equipe tenta percorrer a distância como uma unidade, passando a bola para frente e para trás.

Os pesquisadores japoneses propuseram que o desenvolvimento do produto não deveria ser como uma corrida de revezamento sequencial, mas análogo ao jogo de rúgbi, no qual a equipe trabalha junto, passando a bola para frente e para trás enquanto se move como uma unidade pelo campo. O conceito de rugby de um “Scrum” (momento em que um grupo de jogadores se forma para reiniciar o jogo) foi introduzido para descrever a proposta feita pelos pesquisadores.

Ken Schwaber e Jeff Sutherland desenvolveram o conceito Scrum e sua aplicabilidade no desenvolvimento de software em uma apresentação na conferência Programação Orientada a Objetos, Sistemas, Linguagens e Aplicações (OOPSLA) realizada em 1995, em Austin, Texas. Desde então, vários praticantes, especialistas e autores de Scrum continuaram a refinar seu conceito e sua estrutura. Nos últimos anos, o Scrum cresceu em popularidade e agora é a abordagem de desenvolvimento de projeto preferida para muitas organizações em todo o mundo.

A seguir veja uma síntese sobre o fluxo de um projeto Scrum.

1. **Ciclo Scrum:** O ciclo Scrum começa com a compreensão do caso que deve ser resolvido pelo projeto e passa, na etapa seguinte, para uma reunião das partes interessadas, durante a qual a visão do projeto é criada.
2. **Product Owner (PO):** O Product Owner então desenvolve um Backlog Priorizado do Produto, que contém uma lista priorizada de requisitos de negócios e projetos escritos na forma de Estórias de Usuário.
3. **Backlog do Produto:** Priorização das funcionalidades do produto desejadas pelo cliente.
4. **Planning:** Cada Sprint começa com uma reunião de planejamento, durante a qual as Estórias de Usuário de alta prioridade são consideradas para inclusão nela. Uma Sprint geralmente dura entre uma e seis semanas e envolve a equipe do Scrum trabalhando para criar artefatos entregáveis ou incrementos de produtos.
5. **Backlog da Sprint:** Funcionalidade(s) atribuída(s) à Sprint.
6. **Sprint:** 30 dias.
7. **Daily:** Ao longo da Sprint, são realizadas reuniões diárias curtas e com alto nível de foco, nas quais os membros da equipe discutem o progresso diário do projeto.
8. **Daily (Scrum):** Reuniões diárias de 15 minutos. Os membros da equipe respondem às questões básicas: O que você realizou desde a última Scrum? Você está tendo alguma dificuldade? O que você vai fazer antes da próxima reunião?
9. **Review:** Perto do final da Sprint, uma Reunião de Revisão da Sprint é realizada. Nela o Product Owner e as partes interessadas relevantes recebem uma demonstração dos entregáveis. O product Owner aceita os entregáveis somente se eles atenderem aos critérios de aceitação predefinidos.
10. **Retrospectiva:** O ciclo da Sprint termina com uma Reunião de Retrospectiva da Sprint, na qual a equipe discute maneiras de melhorar os processo e o desempenho à medida que avançam para a Sprint subsequente.
11. **Incremento do Produto:** A nova funcionalidade é demonstrada no final da Sprint.

O Framework Scrum representa o conjunto de princípios e de práticas do Scrum voltado a promover o desenvolvimento de um produto de software de forma ágil e unitária. O Product Owner, o Scrum Master e os eventos do Scrum são exemplos de partes integrantes desse Framework. A Part II da obra de Fábio Cruz (2018) detalha todos os elementos do Scrum.

Outra abordagem importante do Framework Scrum pode ser encontrada no primeiro capítulo da obra de Edson Silva chamada Scrum e FTS: Uma Abordagem Prática (Silva, 2017).

Por fim, o capítulo 4 da obra Engenharia de Software: Conceitos e Práticas (Wazlawick, 2013), apresenta abordagem bastante sucinta sobre o Scrum, no contexto dos métodos ágeis.

Branches

Controle de Versão com GitHub

Um branch representa uma linha independente de desenvolvimento em um projeto com seu próprio histórico de confirmações (ou commits).

No GitHub é possível visualizar os branches do repositório local ao executar a sequência:

git branch

git branch –remote

Remote

A opção –remote mostra a cópia do repositório local no repositório remoto do GitHub.

Branch Master

Todos os projetos possuem o branch Master por padrão e se considerarmos um projeto como uma árvore, ele seria o tronco, do qual se pode criar linhas independentes de desenvolvimento.

As linhas que representam as ramificações.

Os círculos que representam confirmações (commits) individuais em cada branch.

A seta simboliza o ponteiro que chamamos head (cabeça), o qual representa a ramificação atualmente em uso.

Versões de um Arquivo em um Projeto Git

- A versão do diretório de trabalho corresponde àquele que está sendo usado para edição.
- A versão staged corresponde à versão alterada, mas que não foi incluída de forma definitiva no repositório. Trata-se, portanto, de uma versão transitória e que está aguardando o próximo commit para se tornar efetiva.
- As versões permanentes, ou seja, aquelas que receberam o comando commit.

Operações Aplicadas nos Branches

A utilização dos branches não seria completa se não pudéssemos aplicar operações em suas instâncias por meio de comandos específicos.

Defeito de Software

Sabe aquele programa que você baixou e que prometia resolver determinado problema, mas que sequer funcionou? E aquele lanche que na foto parecia ser muito bom, mas que não tinha sabor de nada? Ou, ainda, sabe aquele serviço que você contratou, mas que só foi perda de tempo e dinheiro?

Certamente, um ou mais exemplos iguais a esses já aconteceram com a maioria de nós.

Vamos observar alguns tipos de ocorrências que degradam os serviços.

No exemplo a seguir, ao efetuar determinada operação, a tela de erros retornou.

Falha de Software

No exemplo a seguir, as observações não foram carregadas. Ruim tanto do ponto de vista do usuário quanto do ponto de vista do desenvolvedor de software.

Mas como evitar que isso ocorra?

Existe uma área do conhecimento que trata somente dos aspectos relacionados à qualidade e que pode ser mobilizada para softwares, serviços, produtos, processos, etc.

Qualidade de Software

Mas, enfim, o que é qualidade?

Qualidade diz respeito aos métodos, às ferramentas, às metodologias e aos processos os quais garantirão que determinada entrega seja feita dentro dos padrões de qualidade estabelecidos entre as partes.

Todos esse processo se inicia no levantamento de requisitos, quando o cliente passa para a equipe de desenvolvimento todas as suas necessidades.

Requisitos Funcionais

Refletem a visão do usuário quanto ao funcionamento de determinada função dentro do software.

Requisitos Não Funcionais

Refletem a visão do desenvolvedor. A partir deles são determinados tanto o funcionamento técnico das funcionalidades quanto os mecanismos e os desempenhos esperados.

Em todo o ciclo de vida do projeto, podem ser utilizadas as ferramentas de garantia da qualidade, ou seja, nos processos de desenvolvimento, teste, validações, correções, enfim, em qualquer parte que venha a compor um projeto de desenvolvimento. É importante lembrar também que essas metodologias de garantia da qualidade necessitam de parâmetros, de métricas, que podem variar conforme métodos, necessidades, recursos, etc.

A Univesp disponibiliza uma sequência de aulas sobre qualidade de software, sendo muito interessante como material complementar, uma vez que são discutidos novos exemplos, os quais abrangem os conhecimentos com os cases em que as ferramentas da qualidade de software podem ser aplicadas. Confia o vídeo Gerência e Qualidade de Software – Apresentação da Disciplina, disponível no Youtube.

O artigo intitulado Contribuição dos Modelos de Qualidade e Maturidade na Melhoria dos Processos de Software (Tonini, Carvalho, Spinola, 2008) traz uma interessante discussão sobre a implementação de modelos de qualidade e de maturidade com base em um estudo de casos múltiplos. Essa análise proposta no artigo engloba três níveis: Desenvolvedores, grupos/equipes e organizacional. Vale a pena conferir.

Qualidade de Produto

ISO 9000

A ISO da família 9000 é de grande importância para as atividades relacionadas ao desenvolvimento de softwares. Temos, por exemplo:

- **ISO 9126-1 de 2001:** Trata das características, subcaracterísticas e métricas da qualidade de produto de software.
- **ISO 9126-2 de 2003:** Trata das métricas externas e de controle de falhas.
- **ISO 9126-3 de 2003:** O seu objetivo é verificar a quantidade de ocorrências de falhas e estimar o tempo de recuperação.
- **ISO 9126-4 de 2004:** Faz as tratativas de User Experience, produtividade, eficácia e segurança.

Compreender a sua trajetória evolutiva é muito interessante para entender como as novas necessidades foram moldando a estrutura que utilizamos hoje em dia. No site da ABNT, é possível ir acessando os links para verificar essa evolução histórica.

Conceitos de Testes de Software

Termos

Vamos ver os termos comuns no contexto dos testes de software frequentemente confundidos.

Que expressão você usa quando um programa simplesmente trava ou não produz o resultado que se espera dele?

Tudo que acontece de incomum em um programa pode ser chamado de erro?

- **Erro:** Ocorre quando o resultado obtido em um processamento e o que se esperava dele não são coincidentes. Um erro também está associado a uma violação nas próprias especificações do programa.
- **Defeito:** Trata-se de deficiência algorítmica que, se ativada, pode levar a uma falha.
- **Falha:** É tida como um não funcionamento do programa, provavelmente provocada por um defeito. No entanto, uma falha também pode ser atribuída a uma queda na comunicação ou a um erro na leitura do disco, por exemplo.

Segundo Wazlawick (2013), os termos a seguir podem ser tomados como sinônimos, mas não são:

- **Erro (Error):** Trata-se da diferença detectada entre o resultado de uma operação e o resultado correto que se pretendia obter com ela.
- **Defeito (Default):** Trata-se de uma linha de código, bloco ou conjunto de dados incorretos que provocam um erro.
- **Falha (Failure):** Trata-se de um não funcionamento do software, possivelmente provocada por um defeito, embora possa haver outras causas.
- **Engano (Mistake):** É a ação humana que produziu o defeito no código.

Tipos de Testes

Os testes de sistema e de usuário são executados nas etapas finais de seus respectivos níveis e são considerados de elevada importância na apuração consolidada de qualidade.

Testes de Sistema

O teste de sistema deve ser apresentado como uma etapa em que o sistema, já completamente integrado, passa pelo procedimento do teste.

Para que a equipe consiga chegar ao teste de sistema, todas as unidades e as integrações entre elas devem ter sido testadas previamente. Os erros encontrados devem ter sido também sanados.

Mesmo depois de testada, cada unidade isoladamente e a comunicação entre as unidades, é necessário ainda testar o funcionamento do sistema de modo global, adotando o ponto de vista de quem usará o sistema.

O objetivo do teste de sistema é o de apurar se o programa é capaz de executar processos completos, como se estivesse em uso pleno.

Testes de Usuário

Aplicado o teste de sistema, o produto final deve ser aceito pelo usuário. Como o nome sugere, esse teste é executado pelo usuário e não pela equipe de testadores ou desenvolvedores.

Em relação ao planejamento e à execução, o teste de usuário se equipara ao teste de sistema, com a diferença de que ele será executado pelo usuário. Se considerarmos que o teste de sistema ainda é parte da verificação do produto, o teste de aceitação serve para sua validação.

O teste de usuário tem objetivo e metodologia de aplicação semelhante ao teste de sistema, com a diferença de que ele é executado pelo usuário, como se estivesse operando-o em situação normal de uso. É aconselhada a elaboração de um roteiro de funções a serem testadas.

Desenvolvimento Orientado a Testes e Ferramentas Case

Testes de Unidade

- Um teste é apenas uma classe Java simples.
- A classe testada geralmente aparece como uma variável de instância da classe Teste.
- Vários métodos de teste, anotados com *@Test*, usarão as funcionalidades da classe testada e então confrontarão os resultados esperados com os resultados obtidos.

Vamos Ver como Escrever um Teste de Unidade com o JUnit?

Primeiro, precisamos de um exemplo de classe em teste, ou seja, uma classe implementando alguma funcionalidade que desejamos testar.

O exemplo a seguir é de uma classe calculadora, que retorna a soma entre dois valores inteiros:

```
class Calculator {
   int add(int a, int b) {
      return a + b;
   }
}
```

A seguir, a classe que testará a classe Calculator:

```
class CalculatorTest {
   private Calculator calc = new Calculator();

   @Test
   void testAdd() {
      int x = 2;
      int y = 1;
      int result = calc.add(x, y);

      Assertions.assertEquals(3, result);
   }
}
```

O funcionamento dessa classe inclui a inicialização de duas variáveis de entrada (x e y) e da chamada do método *add*, que desejamos testar com a entrada fornecida. O resultado da operação é armazenado em uma variável e, por fim, o resultado esperado é confrontado com o obtido (o valor 3, neste caso).

Anotações JUnit

Vamos ver agora algumas anotações do JUnit úteis para a organização e para a execução dos procedimentos de teste em classes Java.

- **@Test:** Usada para sinalizar que o método anotado é um método de teste.
- **@AfterAll:** Usada para sinalizar que o método anotado deve ser executado após todos os testes na classe de teste atual.
- **@AfterEach:** Para sinalizar que o método anotado deve ser executado após cada método *@Test*, *@RepeatedTest*, *@ParameterizedTest*, *@TestFactory* e *@TestTemplate* na classe de teste atual.
- **@BeforeAll:** Usada para sinalizar que o método anotado deve ser executado antes de todos os testes na classe de teste atual.
- **@BeforeEach:** Usada para sinalizar que o método anotado deve ser executado antes de cada método *@Test*, *@RepeatedTest*, *@ParameterizedTest*, *@TestFactory* e *@TestTemplate* na classe de teste atual.

- **@RepeatTest:** Usada para sinalizar que o método anotado é um método de modelo de teste que deve ser repetido um número específico de vezes com um nome de exibição configurável.

Visão de uma Auditoria de Sistemas em Uma Empresa de Desenvolvimento de Aplicativos para Mobile

Equipes de Desenvolvimento

- Front end e User Experience
- Banco de dados.
- Desenvolvimento back end Android.
- Desenvolvimento back end IOS.
- Desenvolvimento back end Híbrido.

Os gerentes de projetos entregam as atividades para as equipes e, ao longo das entregas, deve ocorrer a integração dos desenvolvimentos.

Processos de Integração Entre as Equipes

Para que sejam minimizados os erros e as falhas entre as equipes de desenvolvimento, existem processos bem definidos de coexistência entre as ilhas de produção de aplicativos mobile.

Esses processos são constantemente gerenciados e monitorado por cada gerente de projetos.

Erros de Processos de Integração Entre as Equipes

Ultimamente vêm ocorrendo erros de integração de projetos entre a equipe de front end e a de banco de dados e desenvolvimento back end.

São erros comuns, como tipagem de dados, nomes de variáveis, erros de requisitos, entre outros, embora simples de se ajustarem, têm consumido muito tempo das equipes.

Processo de Auditoria

- **Gerentes de Projetos:** Se reuniram com a direção da empresa e estabeleceram que seria contratada uma empresa para fazer a auditoria dos processos de desenvolvimento.
- **Auditores:** Passaram um tempo na empresa fazendo reuniões, análises documentais, entrevistas, observações, entre outras atividades. No dia a dia, os desenvolvedores notaram a mudança na rotina com a presença dos auditores.
- **Desenvolvedores:** No dia a dia, notaram a mudança na rotina com a presença dos auditores.

Devolutiva da Auditoria

Após um tempo, os auditores marcaram uma reunião com gerentes de projetos e colaboradores. Nela o relatório foi disponibilizado para todos e, em seguida, os resultados foram apresentados.

Mas o que fazer com essas devolutivas?

A auditoria aponta falhas, erros, melhorias e potencialidades dos processos. Com as devolutivas, os gerentes de projetos devem traças planos para que cada apontamento feito pela equipe de auditoria receba a devida tratativa.

A auditoria pode se iniciar no processo de contratação da empresa responsável por fazê0la ou, ainda, na escolha dos colaboradores internos que serão alocados para prover a auditoria.

Não necessariamente uma auditoria deve ser feita por uma empresa terceirizada, porém, em alguns casos, para que as observações não sejam tendenciosas, a contratação externa é uma boa saída.

Controles Gerais de Auditoria de Sistemas em uma Loja de Departamento

Os controles gerais tratam das estruturas internas das organizações, as políticas administrativas operacionalizadas e os procedimentos utilizados em atividades operacionais nas empresas. Ou seja, as políticas internas da empresa, que são um guia para o desenvolvimento do sistema.

Como Isso Funciona na Prática

Exemplo: Uma loja de departamento necessita de um software para gerenciar as vendas realizadas. Para isso, a empresa que vai desenvolver o software precisa entender as suas políticas internas, pois elas, de algum modo, interferem na forma de funcionamento dos sistemas ou em alguma funcionalidade específica.

Vamos analisar apenas uma funcionalidade do sistema em questão: A alteração do valor dos produtos.

Para elas existem duas possibilidades:

1. O gerente da loja tem autonomia de alterar os valores dos produtos, dessa forma o sistema deve ter uma funcionalidade para permitir essa alteração.
2. A alteração de valores é feita pela administração, logo o gerente de lojas não tem essa autonomia. Dessa forma, no sistema não haverá a funcionalidade de alteração do valor dos produtos.

Percebeu como a política da empresa guia como a funcionalidade deve ser desenvolvida dentro do sistema? Isso se chama controle geral de sistemas e faz parte dos diversos conhecimentos no qual o profissional deve ter competência para efetuar a análise correta da empresa, a fim de atender aos requisitos dela.

De que Forma a Auditoria Pode Agir Quanto aos Controles Gerais dos Sistemas

De maneira geral, a auditoria deve analisar se a construção do sistema está alinhada aos controles gerais e se atende aos requisitos. Isso é feito por meio de análise documental, análise de script, testes de uso, entre outros recursos.

Vamos supor que o gerente de loja não tenha autonomia para alterar o valor dos produtos.

A auditoria deve efetuar uma análise e gerar relatórios nos quais seja possível ver se, de fato, a funcionalidade de alteração de valor de produto não está acessível. Ainda é possível gerar relatórios

que apontem a origem das alterações de valores, a fim de gerar insights que comprovem que o sistema está funcionando de acordo com o esperado.

Qual Contribuição a Auditoria pode Gerar Quanto à Análise dos Controles Gerais dos Sistemas

O gerente da loja não tem autonomia para alterar os valores dos produtos, porém, ao utilizar o sistema, percebeu que consegue alterar o valor dos produtos via url. Como o gerente é uma pessoa mal-intencionada, ele altera os valores dos produtos, adicionando determinada porcentagem, depois volta ao valor normal e, no fim do dia, pega a diferença de caixa para si.

Certamente, se um processo de auditoria para verificação dos controles gerais tivesse sido feito, talvez o gerente não tivesse tido a oportunidade e não cometeria esse erro. E mesmo depois do gerente ter cometido uma irregularidade, uma auditoria poderia verificar essa vulnerabilidade e ainda comprovar que o gerente efetuou uma operação em desacordo com as políticas internas da loja de departamento.

Ferramentas de Manutenção – Indentação

Foi muito falado da classificação das ferramentas de manutenção e, dentre elas, pode-se destacar a indentação, que, quando bem-feita, auxilia a equipe de desenvolvimento no processo de compreensão das partes que compõe o sistema.

Encontrar um código-fonte indentado e comentado pode ser um grande diferencial no momento da manutenção. Esse processo pode ser encontrado tanto na reengenharia de software quanto nos processos de engenharia reversa, para entender a estrutura das aplicações.

Existem algumas situações que nos permitem entender o correto apenas quando visualizamos a forma incorreta de se fazer. Dito isso, a seguir um trecho de código sem nenhuma indentação. Para facilitar o entendimento da função, acompanhe o que está sendo executado linha a linha.

Independentemente se você conhece a linguagem, percebe como fica confuso entender a estrutura da aplicação? Existem uma brincadeira entre os desenvolvedores com relação a isso, na qual eles falam que um código nesse formato errado fica parecendo uma linguiça (comprida de reta).

A título de análise comparativa, observe a seguir o mesmo trecho de código, só que dessa vez com as devidas indentações.

Percebe a grande diferença entre um código sem indentação e um indentado?

Existem empresas nas quais o gerente de projetos só recebe a entrega de um código-fonte que contenha a estrutura de indentação e comentários de forma correta. Isso demonstra o padrão que é utilizado em desenvolvimento de sistemas, que visa colaborar e garantir a qualidade do software.

Atividades

No modelo de desenvolvimento Cascata, os sistemas são elaborados em fases ou etapas.

Esta etapa em questão, prevê a entrega de um modelo do sistema, expresso em diagramas e descrições estruturadas de funções, entre outros recursos. Nesta etapa, os profissionais envolvidos

estruturam a solução em termos do como o sistema executará as funções levantadas na etapa anterior.

Assinale a alternativa que contém a expressão que sintetiza a etapa do Modelo em Cascata descrita no texto dado

Alternativas:

- Projeto.

Quando uma equipe de desenvolvimento de software destina algumas horas com questionários, reuniões, apresentações e etc., o intuito é fazer um levantamento de requisitos que possibilite abstrair as necessidades do cliente.

Quanto aos requisitos, observe as afirmativas a seguir e assinale (RF) Requisitos Funcionais e (NF) para Requisitos Não Funcionais.

( ) O sistema deve enviar a solicitação de impressão para o equipamento mais próximo ao computador solicitante.
( ) O tempo de resposta da consulta de produtos deve ser dada por: quantidade * 0,001. Ou seja, a cada 1000 produtos um segundo.
( ) A foto do usuário deve ser anexada somente no formato .jpg.
( ) O aplicativo deve permitir que com no máximo 3 cliques, o usuário chegue a qualquer funcionalidade.
( ) A recuperação de senha deve ser por meio do link enviado no e-mail.

Assinale a alternativa com a sequência correta

Alternativas:

- NF – NF – NF – RF – RF.

Uma empresa diretamente ligada a banco de dados, está testando a performance da aplicação com banco de dados do tipo relacional e não relacional. O intuito é que se conheça o desempenho da aplicação. No nível atual já se utiliza os tratamentos estatísticos e a intenção nesse momento, é alcançar um novo patamar.

Com base na estrutura do CMM, assinale a alternativa que represente o nível de maturidade correspondente

Alternativas:

- Nível 5 – Otimização

Ao ingressar em uma empresa de desenvolvimento de software você tomou conhecimento do descontentamento de muitos clientes em relação à interpretação de seus requisitos feita pelos responsáveis por levantá-los e organizá-los. Com alguma frequência, o que era dito pelo cliente era interpretado incorretamente pelo profissional, o que acabava gerando funções inadequadas para as finalidades do sistema. A fim de diminuir as interpretações incorretas e aprimorar a comunicação entre equipe e cliente, você decidiu sugerir um procedimento adotado no paradigma ágil de desenvolvimento.

Com base no contexto apresentado, assinale a alternativa que contém a descrição da solução adequada para o caso, segundo o pensamento ágil de desenvolvimento

Alternativas:

- Adoção das histórias do cliente, prática pela qual o cliente descreve os requisitos desejados em uma ficha e os entrega à equipe.

Os métodos ágeis funcionam bem em certas situações porque possibilitam comunicação contínua entre o gerente do produto ou o cliente do sistema e o time de desenvolvimento.

Fonte: adaptado de SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson Prentice Hall, 2018.

Tomando como referência as funções de alguns membros das metodologias ágeis, julgue as afirmativas a seguir em (V) Verdadeiras ou (F) Falsas.

O trabalho do analista de testes no XP pressupõe a participação do cliente.
Ao desenvolvedor é dada apenas a função de codificar o sistema.
A execução das tarefas do Sprint Backlog é dever exclusivo do Scrum Master.
Assinale a alternativa que apresenta a sequencia CORRETA.

Alternativas:

- V - F - F.

Existem algumas regras, costumes e politicas administrativas que são peculiares de cada empresa. E esses pontos moldam a forma como as funcionalidades devem ser desenvolvidas. E com isso, são passiveis de análises por meio de auditoria.

Com base no exposto, assinale a alternativa da área que trata das estruturas internas das organizações.

Alternativas:

- Controle geral

As empresas de desenvolvimento de software possuem diferentes processos, características e peculiaridades, ainda que sejam da mesma área. Com isso, quando são necessários processos de auditoria, essas informações da empresa, são muito importantes para a equipe que irá conduzir os processos de auditoria.

Essas observações acerca das características das empresas, são conhecidas como controles gerais. Caso, nos processos de auditoria em sistemas da informação, o controle geral apresentar deficiências, assinale a alternativa com as consequências

Alternativas:

- Comprometimento da confiabilidade nos controles individuais.

Considerando conceitos e aplicações da depuração, avalie as afirmativas que seguem:

II. Um dos recursos da depuração inclui a inspeção de variáveis.

Considerando o contexto apresentado, assinale a alternativa correta.

Considerando o conceito e as aplicações do teste estrutural, avalie as afirmativas a seguir:

II. O teste estrutural é também chamado caixa branca por sua estrutura e seu código serem previamente conhecidos pelo testados.

Em um teste de caixa preta, o testador alimenta a entrada e observa as saídas produzidas, com o auxílio de uma ferramenta de teste. Nesse caso, o objetivo do teste é assegurar que para todo tipo de entrada apresentada, a saída observada corresponde àquela que é esperada. Existem vantagens e desvantagens quando se recorre a esse tipo de procedimento. A vantagem óbvia é que a caixa preta está livre das restrições impostas pela estrutura e lógica internas do objeto de teste. Todavia, nem sempre é possível executar um teste completo dessa maneira. Por exemplo, suponha que um componente simples aceite como entrada os três números a, b, c, e produza como saída as duas raízes da equação ax2 + bx + c = 0 ou a mensagem "Não há raízes reais".

Assinale a alternativa que contém a motivo de não ser possível executar um teste completo neste caso.

Alternativas:

- Não é possível realizar o teste com todo conjunto possível de valores para a, b, c.

A ISO 9126 é uma norma que tem como objetivo avaliar a qualidade do produto de software. Sendo esta dividida por especificidades, atendendo as necessidades de desenvolvimento de software em sua completude.

Com base no exposto, observe as afirmativas a seguir e assinale (V) verdadeiro ou (F) falso.

( ) A ISO 9126 é dividida em 5 partes, sendo 9126-0, 9126-1, 9126-2, 9126-3 e 9126-4.
( ) A ISO 9126-1 trata somente as características do produto de software.
( ) A ISO 9126-2 trata somente as subcaracterísticas do produto de software.
( ) A ISO 9126-3 trata quantificar erros e falhas quanto ao número de ocorrências.
( ) A ISO 9126-4 trata da experiência do usuário, eficácia e segurança do produto de software.

Assinale a alternativa com a sequência correta

Alternativas:

- F – F – F – V – V.

Os modelos de maturidade são ferramentas importantíssimas utilizadas por profissionais de tecnologia da informação, principalmente aqueles que trabalham com desenvolvimento de software. Com base nas características do CMM, observe as afirmativas a seguir.

II. O nível mais baixo (inicial), não existe controle de processos.

O CMMI é uma ferramenta que quando utilizada em conjunto com o desenvolvimento de software, visa contribuir com diversos benefícios, gerando assim, resultados muito interessantes. A sua estrutura é dividida em cinco níveis.

Com base no exposto, assinale (V) verdadeiro ou (F) nas afirmativas a seguir.

( ) No nível 0, não existem processos definidos.
( ) No nível 1, os processos satisfazem as metas.
( ) No nível 2, os processos são monitorados.
( ) No nível 3, os processos sofrem ajustes.
( ) No nível 4, os processos utilizam medições.

Assinale a alternativa com a sequência correta

Alternativas:

- F – V – V – F – F.

Uma das mais importante publicações na área de Engenharia de Software a define como uma abordagem sistemática, disciplinada e quantificável de desenvolvimento, operação e manutenção do software, além do estudo dessas abordagens (IEEE, 2004).

A parte da definição da Engenharia de Software que menciona "além do estudo dessas abordagens" revela

Alternativas:

- a intenção de manter a disciplina em constante evolução.

Considerando as funções de um controlador de versões e as ações tomadas pelos desenvolvedores em relação a ele, avalie as afirmativas a seguir:

II. Uma ferramenta de controle de versões previne o surgimento de eventuais discordâncias entre desenvolvedores, pois é capaz de registrar dados de auditoria relacionados a alterações feitas no código.

Um auditor em tecnologia da informação possuí algumas atribuições que visam atender as mais diversas demandas. Muitas vezes são necessários especialistas a fim de se conseguir fazer as devidas análises.

Quanto as atribuições dos auditores de desenvolvimento de sistemas, assinale (V) verdadeiro ou (F) falso nas afirmativas a seguir.

( ) O auditor tem atribuição de inspeção, onde deve fazer a análise de colaboradores de gestores de projetos.
( ) O auditor tem atribuição de controle, onde são verificados os processos e contratos.
( ) O auditor tem atribuição de análise de risco, onde são verificados os pontos que podem de alguma forma prejudicar o sistema.
( ) O auditor tem atribuição de análises pessoais, pautados em experiências anteriores.
( ) O auditor tem atribuição de consultor, pois é um profissional especializado e experiente.

Assinale a alternativa com a sequência correta.

Alternativas:

- F – F – V – F – F.

Quando uma empresa de desenvolvimento contrata uma auditoria, é iniciado um ciclo de vida, que trata de um período de convívio entre colaboradores da empresa e auditores, mas sempre com o intuito de promover melhorias.

Quanto as características do ciclo de vida da auditoria, observe as afirmativas a seguir:

I. Pode ter variação de tempo de um projeto para outro.
II. Possui 4 etapas em seu ciclo de vida.

Quando um sistema complexo é desenvolvido, são necessários muitos verões. Ainda que, após a entrega, com o encontro de inconformidades surgem novas versões. Para isso, foi convencionado um sistema de numeração.

Com base nas características do versionamento de sistemas, observe as afirmativas a seguir.

I. Em um software que tinha a versão 2.1.1.1 e passou para a versão 3.1.1.1, significa que foram feitos ajustes significativos.
II. Um software com a versão 2.1.10.11, passou para a versão 2.1.10.12, foi feito uma atualização de segurança.

Considerando recursos usados para descobrir erros em um programa, complete as lacunas da sentença que segue:

O fato de saber que parte do código não funciona não significa que necessariamente seja conhecido o trecho do código que provoca um ou mais erros. Assim, enquanto a atividade de _____ consiste em executar sistematicamente o software para encontrar erros _____, a _____ é a atividade que consiste em buscar a causa do erro e sua localização no código.

Assinale a alternativa que completa as lacunas corretamente

Alternativas:

- teste - desconhecidos - depuração.

O processo do Desenvolvimento Orientado a Testes e´ efetivado pelos seguintes passos: 1. Escrever um teste automatizado para a funcionalidade e implementá-lo. 2. Passar para implementação da próxima funcionalidade, depois de todos os testes passarem. 3. Definir o incremento de funcionalidade desejado. 4. Executar o teste, que sabidamente falhará. 5. Implementar a funcionalidade e novamente executar o teste, com o código refatorado. Assinale a opção que apresenta a ordem correta dos passos realizados.

Alternativas:

- 3 - 1 - 4 - 5 - 2.