Estratégia Concursos

Aula 08

SERPRO (Analista - Especialização: Tecnologia) Bizu Estratégico - 2023 (Pós-Edital)

Autor:
Elizabeth Menezes de Pinho Alves, Leonardo Mathias, Paulo Júnior, Aline Calado Fernandes

23 de Maio de 2023

BIZU ESTRATÉGICO DE ENGENHARIA DE SOFTWARE – SERPRO

Olá, prezado aluno. Tudo certo?

Neste material, traremos uma seleção de *bizus* da disciplina de **Engenharia de software** para o concurso de **Analista – Especialidade: Tecnologia**.

O objetivo é proporcionar uma revisão rápida e de alta qualidade aos alunos por meio de tópicos que possuem as maiores chances de incidência em prova.

Todos os *bizus* destinam-se a alunos que já estejam na fase bem final de revisão (que já estudaram bastante o conteúdo teórico da disciplina e, nos últimos dias, precisam revisar por algum material bem curto e objetivo).

Além disso, utilizamos os materiais do professor Thiago Rodrigues Cavalcanti para elaborar esse Bizu.

Elizabeth Menezes

@elizabethmpalves

Leonardo Mathias

@profleomathias

Apresentação

Antes de começarmos, gostaria de me apresentar. Meu nome é **Elizabeth Menezes**, tenho 32 anos e sou natural do Pernambuco. Sou graduada em Administração pela UFPE e Pós-Graduada em Direito Administrativo e Constitucional.

Atualmente, exerço o cargo de Auditora de Controle Externo no Tribunal de Contas do Estado de São Paulo (TCE-SP). Também fui aprovada e nomeada para outros concursos da área fiscal (Auditor Fiscal Estadual e Municipal) e da área de controle.

Serei a responsável pelo **Bizu Estratégico de Engenharia de software** e, com ele, pretendo abordar os tópicos mais cobrados nessa disciplina, de maneira concisa e objetiva, por meio de uma linguagem bem clara!

Espero que gostem!

Um grande abraço e bons estudos!

ANÁLISE ESTATÍSTICA

Pessoal, segue abaixo uma análise estatística dos assuntos mais exigidos no âmbito da disciplina de Engenharia de software:

Assunto	% de cobrança
Scrum	34,48%
Engenharia de requisitos	17,24%
Qualidade. Análise estática de código. Teste unitário. Mock, stubs. Teste de integração. Teste de RNF (carga, estresse). Revisão e programação por pares. (Parte 1)	17,24%
XP	6,90%

Com essa análise, podemos verificar quais são os temas mais exigidos e, através disso, focaremos nos principais pontos em nossa revisão!

Engenharia de software – SERPRO		
Assunto	**Bizu**	**Caderno de Questões**
Scrum	1	http://questo.es/fwtnid
Engenharia de requisitos	2	http://questo.es/9ags59
Qualidade. Análise estática de código. Teste unitário. Mock, stubs. Teste de integração. Teste de RNF (carga, estresse). Revisão e programação por pares.	3	http://questo.es/hnbdy0

Engenharia de software

1. Scrum

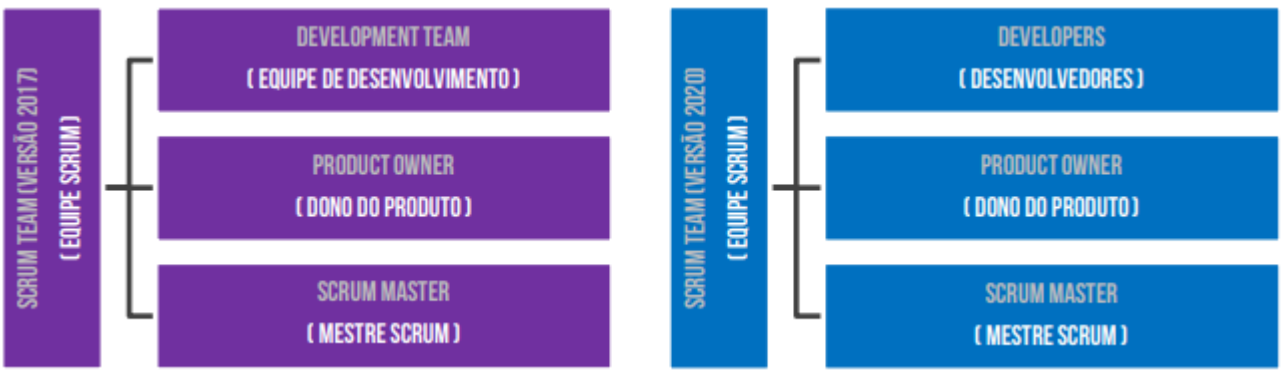
Scrum

- [Guia Scrum - Versão 2017] Scrum: um framework dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Scrum é: leve, simples de entender e difícil de dominar.

- [Guia Scrum - Versão 2020] Scrum é um framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.

PILARES	DESCRIÇÃO
TRANSPARÊNCIA	Todo trabalho deve ser claramente definido e conhecido por todas as partes envolvidas no projeto.
INSPEÇÃO	Todo trabalho deve ser inspecionado com a frequência necessária para garantir a qualidade do produto.
ADAPTAÇÃO	O projeto deve ser capaz de se adaptar o projeto às necessidades de negócio.

VALORES	DESCRIÇÃO
CORAGEM	Os integrantes de um projeto precisam ter coragem para fazer a coisa certa e trabalharem juntos removendo impedimentos, buscando soluções.
FOCO	Os integrantes de um projeto precisam focar no trabalho durante a sprint e nas metas designadas – time disperso perde produtividade e não alcança os objetivos.
COMPROMETIMENTO	Os integrantes se comprometem com o trabalho que se responsabilizou em fazer, envolvendo-se e não abandonando pela metade ou entregando sem qualidade.
RESPEITO	Os integrantes se respeitam entre si a fim de manter a colaboração, a integração e o bom ambiente de trabalho.
ABERTURA	Os integrantes devem poder ser francos, expor ideias e propostas mesmo que elas não sejam proveitosas. Momentos de debates, discussões e sugestões são ideais.



RESPONSABILIDADES SCRUM MASTER (SM)	
	Responsável pela gestão de pessoas e gestão do processo.
	Ele deve garantir que o Scrum seja entendido e aplicado. O Scrum Master faz isso para garantir que a Equipe Scrum adere à teoria, práticas e regras do Scrum.
	O Scrum Master ajuda aqueles que estão fora da Equipe Scrum a entender quais as suas interações com a Equipe Scrum são úteis e quais não são.
	O Scrum Master ajuda todos a mudarem estas interações para maximizar o valor criado pela Equipe Scrum.
	Ele é responsável por orientar o Product Owner na criação e ordenação do Product Backlog.
	Ele é responsável por garantir que as regras do Scrum estejam sendo cumpridas e seus valores estejam sendo seguidos.
	Ele é responsável por ajudar a remover impedimentos que o time enfrente, fazendo isso sem o uso de qualquer autoridade.
	Ele utiliza técnicas de facilitação e coaching para que os membros do time consigam visualizar os problemas e encontrem a melhor solução.
	Durante eventos, ele é responsável por fazer com que a reunião flua adequadamente, utilizando técnicas de facilitação, embora não seja o responsável pela condução.
	Ele ajuda a treinar os desenvolvedores em autogerenciamento e interdisciplinaridade.
	Ele treina os desenvolvedores em ambientes organizacionais nos quais o Scrum não é totalmente adotado e compreendido.
	Ele ensina a Equipe Scrum a criar itens do Product Backlog de forma clara e concisa.

RESPONSABILIDADES DESENVOLVEDORES	
	Responsável pela micro-gestão e pela criação do produto.
	Eles são auto-organizados. Ninguém (nem mesmo o SM) diz aos desenvolvedores como transformar o Product Backlog em incrementos de funcionalidades potencialmente utilizáveis.
	Times de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto.
	O Scrum não reconhece títulos específicos para os desenvolvedores, independentemente do trabalho que está sendo realizado pela pessoa;
	Individualmente, os desenvolvedores podem ter habilidades especializadas, mas a responsabilidade pertence aos desenvolvedores como um todo.
	Os desenvolvedores não contêm sub-times dedicados a domínios específicos de conhecimento, tais como teste ou análise de negócios.
	Os desenvolvedores são estruturados e autorizados pela organização para organizar e gerenciar seu próprio trabalho.

RESPONSABILIDADES PRODUCT OWNER (PO)	Ele é responsável pela macro-gestão e pela gestão do produto.
	Ele é o responsável por maximizar o valor do produto e do trabalho dos desenvolvedores, sendo o único que pode gerenciar o Product Backlog.
	Ele pode até delegar as atividades de gerenciamento para os desenvolvedores, mas ainda será considerado o responsável pelos trabalhos.
	Ele é responsável por priorizar/ordenar os itens do Product Backlog e seleciona aqueles que serão implementados.
	Ele é responsável por garantir o ROI (Return On Investment ou Retorno sobre Investimento).
	Ele é responsável por expressar claramente os itens do Product Backlog.
	Ele é responsável por garantir que o Backlog do Produto seja visível, transparente, claro para todos, e mostrar o que a Equipe Scrum vai trabalhar a seguir.
	Ele é responsável por garantir que os desenvolvedores entendam os itens do Product Backlog no nível necessário.

PRODUCT BACKLOG	Trata-se de uma lista ordenada (por valor, risco, prioridade, entre outros) de requisitos ou funcionalidades que o produto deve conter criada pela Equipe Scrum e gerenciada pelo Product Owner.
SPRINT BACKLOG	Trata-se de conjunto de itens selecionados do Product Backlog, mais a meta da sprint e mais um plano de ação para entregar um incremento potencialmente usável – é criado e gerenciado pelos desenvolvedores.
SPRINT REVIEW	Trata-se da é da soma de todos os itens do Backlog do Produto completados durante a Sprint e o valor dos incrementos de todas as sprints anteriores – sendo validado como "pronto".
DEFINIÇÃO DE READY	Conjunto de critérios que indicam que já existem informações suficientes para um requisito começar a ser desenvolvido.
DEFINIÇÃO DE DONE	Conjunto de critérios que indicam que uma determinada história de usuário atende a todos os requisitos de aceitação para se tornar um incremento.

SPRINT PLANNING	Reunião dividida em duas partes que possui duração de até 8 horas. Na primeira parte, a equipe seleciona, alinha e detalha os itens que vão ser desenvolvidos na próxima sprint. Na segunda parte, cada item é estimado e decomposto nas tarefas necessárias para produzir as entregas.
DAILY SCRUM	Reunião diária para alinhar a comunicação do projeto, inspecionar o progresso para a meta, identificar impedimentos e adaptar o backlog da sprint, se necessário. Não pode ter mais que 15 minutos de duração, ocorrendo sempre no mesmo local e horário.
SPRINT REVIEW	Reunião de até 4h de duração realizada ao final de cada sprint para apresentar ao Product Owner as funcionalidades implementadas para que ele possa validá-las e eventualmente adaptar futuras modificações. Trata-se de um evento informal para apresentação do incremento e colaboração sobre os próprios passos.
SPRINT RETROSPECTIVE	Reunião de até 3h de duração realizada após a Sprint Review. No entanto, em vez de validar o produto, a equipe busca revisar e validar o processo executado para gerar as funcionalidades. A ideia é planejar maneiras de aumentar a qualidade e efetividade do processo.

2. Engenharia de requisitos

Engenharia de requisitos

CLASSIFICAÇÃO QUANTO AO NÍVEL DE ABSTRAÇÃO	
REQUISITOS DE USUÁRIO	Descrições, em linguagem natural e com diagramas, de quais serviços o sistema deve fornecer e as restrições sob as quais deve operar. São requisitos com alto nível de abstração e poucos detalhes, feitos para serem lidos por pessoas leigas – podem ser funcionais ou não funcionais.
REQUISITOS DE SISTEMA	Descrições detalhadas sobre as funções, operações e restrições de sistema que definem exatamente o que deve ser implementado. São requisitos com baixo nível de abstração e muitos detalhes, feitos para serem lidos por pessoas experientes – podem ser funcionais ou não funcionais.

CLASSIFICAÇÃO QUANTO À QUALIDADE	
REQUISITOS NORMAIS	Refletem os objetivos e metas estabelecidos para um produto ou sistema durante reuniões com o cliente. Se esses requisitos estiverem presentes, o cliente fica satisfeito. Exemplos de Requisitos Normais poderiam ser tipos de displays gráficos solicitados, funções de sistema específicas e níveis de desempenho definidos.
REQUISITOS ESPERADOS	Estão implícitos no produto ou sistema e podem ser tão fundamentais que o cliente não os declara explicitamente. Sua ausência será causa de grande insatisfação. Exemplos de Requisitos Esperados: facilidade na interação homem-máquina, confiabilidade e correção operacional global e facilidade na instalação do software.
REQUISITOS FASCINANTES	Esses recursos vão além da expectativa dos clientes e demonstram ser muito satisfatórios quando presentes. Por exemplo, o software para um novo celular vem com recursos-padrão, mas junto vem um conjunto de capacidades não esperadas. Exemplos de Requisitos Fascinantes: tecla multitoque e correio de voz visual.

CLASSIFICAÇÃO QUANTO À EVOLUÇÃO	
REQUISITOS PERMANENTES	Também chamados de Requisitos Estáveis, estão diretamente ligados a atividade principal da organização. São concebidos com a essência de um sistema e seu domínio da aplicação, e mudam mais lentamente que requisitos voláteis. Em geral, eles são derivados do Modelo de Domínio.
REQUISITOS VOLÁTEIS	Também chamados de Requisitos Instáveis, são específicos para a instanciação de um sistema em um ambiente ou um cliente particular e são mais propensos a mudança. Se modificam quando o sistema está em desenvolvimento ou em uso. Podem ser subclassificados em mutáveis, emergentes, consequentes ou de compatibilidade.

CLASSIFICAÇÃO QUANTO À FUNCIONALIDADE	
REQUISITOS FUNCIONAIS	São ações ou funcionalidades que o sistema deve fornecer para atingir seus objetivos. Eles dependem do tipo de software, dos usuários esperados e do tipo de sistema onde o software será implantado e fazem parte da arquitetura de um sistema. Grosso modo, pode-se dizer que eles tratam de o que o sistema deve fazer enquanto os requisitos não-funcionais tratam de como o sistema deve fazer.
REQUISITOS NÃO-FUNCIONAIS	São restrições ou condições estipuladas sobre as quais o sistema deve funcionar. Não estão diretamente relacionados às funções específicas do sistema, mas às gerais – e podem incluir restrições de tempo, restrições de processo de desenvolvimento, restrições impostas por padrões, entre outras. Podem ser mais críticos que os funcionais e sempre devem ser verificáveis. Eles fazem parte da arquitetura técnica de um sistema.
REQUISITOS DE DOMÍNIO	são requisitos derivados do domínio da aplicação e refletem características de sua área de negócio. Eles podem ser requisitos funcionais ou não-funcionais e, caso não sejam satisfeitos, o sistema pode não ser realizável. Por exemplo, um avião que não atende aos requisitos de confiabilidade, não será certificado para voo.

CLASSIFICAÇÃO QUANTO À ORIGEM	
REQUISITOS DE PRODUTO	Especificam o comportamento do produto. Entre os exemplos, estão requisitos de desempenho quanto à rapidez com que o sistema deve operar e quanto de memória ele requer, requisitos de confiabilidade que definem a taxa aceitável de falhas, requisitos de portabilidade e requisitos de usabilidade.
REQUISITOS ORGANIZACIONAIS	São derivados de políticas e procedimentos da organização do cliente e do desenvolvedor. Entre os exemplos, estão padrões de processo que devem ser usados, linguagem de programação ou o método de projeto usado, e requisitos de entrega que especificam quando o produto e a sua documentação devem ser entregues.
REQUISITOS EXTERNOS	Abrange todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento. Entre os exemplos, estão a interoperabilidade que define como o sistema interage com outros sistemas, requisitos legais que devem ser seguidos, requisitos éticos sistema para assegurar que ele será aceito por todos.

REQUISITOS NÃO-FUNCIONAIS	EXEMPLOS
REQUISITOS DE CONFIABILIDADE	O sistema não deve ficar fora do ar por mais de cinco segundos durante o dia.
REQUISITOS DE PROTEÇÃO	O sistema não deve permitir que os usuários modifiquem senhas de acesso que eles não criaram.
REQUISITOS DE DESEMPENHO	O sistema deverá ser capaz de processar oitocentas requisições por segundo.
REQUISITOS DE ESPAÇO	Também chamado de Requisitos de Armazenamento, o sistema deverá ocupar, no máximo, 80Mb da memória interna do dispositivo.
REQUISITOS DE USABILIDADE	Os usuários deverão operar todas as funcionalidades do sistema após 2 horas de treino.
REQUISITOS DE SEGURANÇA	O sistema não deve permitir a ativação simultânea de mais de três sinais de alarme.
REQUISITOS ETICOS	O sistema não apresentará aos usuários quaisquer dados de natureza confidencial de outrem.
REQUISITOS DE IMPLEMENTAÇÃO	A interface de usuário deve ser implementada em HTML e não se deve utilizar Applets de Java.

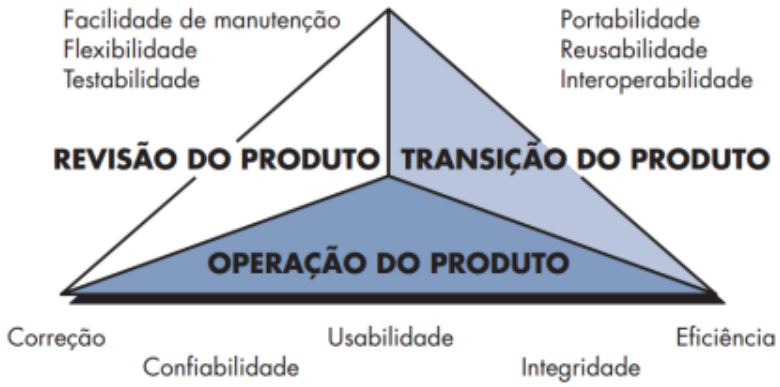
3. Qualidade. Análise estática de código. Teste unitário. Mock, stubs. Teste de integração. Teste de RNF (carga, estresse). Revisão e programação por pares.

Qualidade. Análise estática de código. Teste unitário. Mock, stubs. Teste de integração. Teste de RNF (carga, estresse). Revisão e programação por pares.

QUALIDADE DE SOFTWARE

Característica de ter demonstrado a realização da criação de um produto que atende ou excede os requisitos acordados, conforme avaliado por medidas e critérios acordados, e que é criado em um processo acordado.

CATEGORIAS DE FATORES DE QUALIDADE DE SOFTWARE (POR MCCALL, RICHARDS E WALTERS)

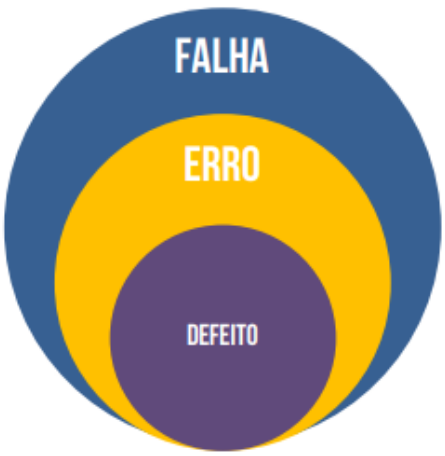


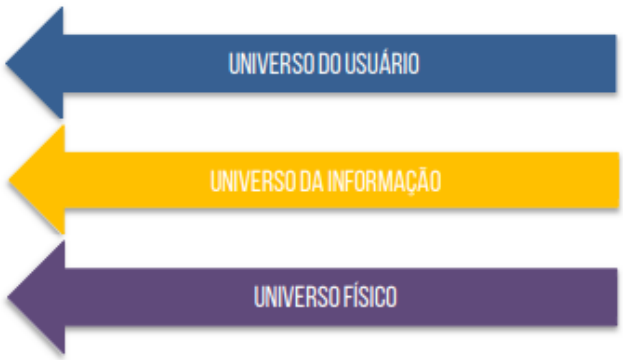
FATORES	DESCRIÇÃO
CORREÇÃO	O quanto um programa satisfaz a sua especificação e atende aos objetivos da missão do cliente.
CONFIABILIDADE	O quanto se pode esperar que um programa realize a função pretendida com a precisão exigida.
EFICIÊNCIA	A quantidade de recursos computacionais e código exigidos por um programa para desempenhar sua função.
INTEGRIDADE	O quanto o acesso ao software ou dados por pessoas não autorizadas pode ser controlado.
USABILIDADE	Esforço necessário para aprender, operar, preparar a entrada de dados e interpretar a saída de um programa.
FACILIDADE DE MANUTENÇÃO	Esforço necessário para localizar e corrigir um erro em um programa.
FLEXIBILIDADE	Esforço necessário para modificar um programa em operação.
TESTABILIDADE	Esforço necessário para testar um programa de modo a garantir que ele desempenhe a função destinada.
PORTABILIDADE	Esforço necessário para transferir o programa de um ambiente de hardware e/ou software para outro.
REUSABILIDADE	O quanto um programa [ou partes de um programa] pode ser reutilizado em outras aplicações.
INTEROPERABILIDADE	Esforço necessário para integrar um sistema a outro.

FATORES (ISO 9126)	DESCRIÇÃO
FUNCIONALIDADE	Trata-se do grau com que o software satisfaz às necessidades declaradas conforme indicado pelos seguintes subatributos: adequabilidade, exatidão, interoperabilidade, conformidade e segurança.
CONFIABILIDADE	Trata-se da quantidade de tempo que o software fica disponível para uso conforme indicado pelos seguintes subatributos: maturidade, tolerância a falhas, facilidade de recuperação.
USABILIDADE	Trata-se do grau de facilidade de utilização do software conforme indicado pelos seguintes subatributos: facilidade de compreensão, facilidade de aprendizagem, operabilidade.
EFICIÊNCIA	Trata-se do grau de otimização do uso, pelo software, dos recursos do sistema conforme indicado pelos seguintes subatributos: comportamento em relação ao tempo, comportamento em relação aos recursos.
MANUTENIBILIDADE	Trata-se da facilidade com a qual uma correção pode ser realizada no software conforme indicado pelos seguintes subatributos: facilidade de análise, facilidade de realização de mudanças, estabilidade, testabilidade.
PORTABILIDADE	Trata-se da facilidade com a qual um software pode ser transposto de um ambiente a outro conforme indicado pelos seguintes subatributos: adaptabilidade, facilidade de instalação, conformidade, facilidade de substituição.

GARANTIA DE QUALIDADE	CONTROLE DE QUALIDADE
Garantia da qualidade garante que o processo é definido e apropriado.	As atividades de controle da qualidade focam na descoberta de defeitos específicos.
Metodologia e padrões de desenvolvimento são exemplos de garantia da qualidade.	Um exemplo de controle da qualidade poderia ser: *"Os requisitos definidos são os requisitos certos?"*
Garantia da qualidade é orientada a processo.	Controle da qualidade é orientado a produto.
Garantia da qualidade é orientada a prevenção.	Controle da qualidade é orientado a detecção.
Foco em monitoração e melhoria de processo.	Inspeções e garantia de que o produto de trabalho atenda aos requisitos especificados.
As atividades são focadas no início das fases no ciclo de vida de desenvolvimento de software.	As atividades são focadas no final das fases no ciclo de vida de desenvolvimento de software.
Garantia da qualidade garante que você está fazendo certo as coisas e da maneira correta.	Controle da qualidade garante que os resultados do seu trabalho são os esperados conforme requisitos.

DEFINIÇÃO DE ACORDO COM A IEEE 610	
DEFEITO	Também chamado de Falta, trata-se do ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Em outras palavras, trata-se de um passo, processo ou definição de dados incorretos (Ex: instrução ou comando incorreto) - pode ocasionar a manifestação de erros em um produto.
ERRO	Também chamado de Engano, trata-se da ação humana que produz um resultado incorreto (Ex: lógica incorreta escrita pelo programador). Em outras palavras, é a manifestação concreta de um defeito em um artefato de software. É a diferença entre o valor obtido e o valor esperado, isto é, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa.
FALHA	Trata-se do comportamento operacional do software diferente do esperado pelo usuário. Em outras palavras, trata-se da produção de uma saída incorreta em relação à especificação. Em geral, defeitos e erros são causas e falhas são consequências. Elas afetam diretamente o usuário final da aplicação e pode inviabilizar a utilização de um software.





Vamos ficando por aqui.

Esperamos que tenha gostado do nosso Bizu!

Bons estudos!

Elizabeth Menezes

@elizabethmpalves

Leonardo Mathias

@profleomathias

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.