LISTA DE EXERCICIOS DE ESW(337X)
LISTA 03

1) As seguintes afirmações dizem respeito ao modelo de desenvolvimento em Espiral - proposto por Barry Boehm na década de 70:

I)	suas atividades do desenvolvimento são conduzidas por riscos ;
II)	cada ciclo da espiral inclui 4 passos: passo 1 - identificação dos objetivos ; passo 2 – avaliação das alternativas tendo em vista os objetivos e os riscos (incertezas, restrições) do desenvolvimento; passo 3 - desenvolvimento de estratégias (simulação, prototipagem) p/ resolver riscos; e passo 4 - planejamento do próximo passo e continuidade do processo determinada pelos riscos restantes;
III)	é um modelo evolutivo em que cada passo pode ser representado por um quadrante num diagrama cartesiano: assim na dimensão radical da espiral tem-se o custo acumulado dos vários passos do desenvolvimento enquanto na dimensão angular tem-se o progresso do projeto.

Levando-se em conta as três afirmações I, II e III acima, identifique a única alternativa válida:
a) Apenas a I e a II estão corretas;
b) Apenas a II e a III estão corretas;
c) Apenas a I e a III estão corretas;
d) As afirmações I, II e III estão corretas;
e) Apenas a III está correta.

2) Considere que você trabalhe em uma empresa de desenvolvimento de *software* e que a empresa tenha decidido desenvolver um novo editor de texto para colocar no mercado. Esse editor deve ser um *software* que forneça recursos adicionais de apoio à autoria, embasado no estilo de escrita do usuário, o que o torna um *software* de funcionalidade mais complexa. Considere que a empresa deseje disponibilizar o produto no mercado em versões que agreguem esse suporte de forma gradativa, fazendo análise de risco para avaliar a viabilidade de desenvolvimento de uma nova versão. Tendo de escolher um modelo de processo para desenvolver esse editor, e conhecendo as características dos modelos existentes, entre os modelos abaixo, qual é o modelo apropriado para esse caso?
   a) Cascata.
   b) Espiral.
   c) RAD (*Rapid Application Development*).
   d) Prototipação.
   e) *Cleanroom.*

3) No processo de desenvolvimento de software, todo software passa pelas fases de análise e projeto, associadas, respectivamente, com o que deve ser feito e como deve ser feito. A partir dessa informação, avalie a opções correta.
   a) Na fase de análise, três modelos que devem ser considerados são: do domínio da informação, o funcional e o comportamental.
   b) Na fase de projeto, dois níveis de projeto devem ser considerados: o projeto detalhado, que se preocupa com uma transformação dos requisitos em um projeto de dados e arquitetural; e o projeto preliminar, que se preocupa em aprimorar o projeto detalhado para que a implementação possa ser realizada em seguida.
   c) O objetivo do projeto arquitetural é desenvolver uma estrutura de programa e representar os diversos fluxos de dados entre os módulos.
   d) O projeto arquitetural independe do paradigma de desenvolvimento.
   e) Para lidar com a complexidade do *software*, pode-se aplicar o princípio do particionamento, quebrando o problema em problemas menores. Esse princípio não é aplicado nas outras fases de desenvolvimento e ele não causa impacto nos custos de desenvolvimento.

4) Sobre Ciclo de Vida de Desenvolvimento de Software, é correto afirmar:

I)	O desenvolvimento em cascata tem como base a idéia de desenvolver uma implementação inicial, mostrar e discutir tal implementação com o usuário e fazer seu aprimoramento por meio de versões subsequentes, até que um sistema adequado tenha sido desenvolvido.
II)	No modelo de processo de desenvolvimento em espiral, cada loop na espiral representa uma fase do processo de software. Este modelo exige a consideração direta dos riscos técnicos em todos os estágios do projeto e, se aplicado adequadamente, deve reduzir os riscos antes que eles se tornem problemáticos.
III)	O Rapid Application Development (Desenvolvimento Rápido de Aplicação) é um modelo de processo de software incremental que enfatiza um ciclo de desenvolvimento rápido. Este modelo é uma adaptação de modelo cascata, no qual o desenvolvimento rápido é conseguido com o uso de uma abordagem de construção baseada em componentes.
IV)	O modelo incremental combina elementos do modelo em cascata aplicado de maneira iterativa. Em um processo de desenvolvimento incremental, os clientes identificam (esboçam) as funções a serem fornecidas pelo sistema e a importância das mesmas. Em seguida, é definida uma série de estágios de entrega, com cada estágio fornecendo um subconjunto das funcionalidades do sistema.

Assinale a alternativa *CORRETA:*

a) Somente as afirmativas I e II são corretas.
b) Somente as afirmativas I e III são corretas.
c) Somente as afirmativas III e IV são corretas.
d) Somente as afirmativas I, II e IV são corretas.
e) Somente as afirmativas II, III e IV são corretas.

5) Considere as seguintes afirmativas sobre os modelos prescritivos de processos de desenvolvimento de software

I)	Uma das vantagens do modelo de prototipação é servir como base para entendimento dos requisitos do sistema.
II)	Um dos problemas do modelo RAD (Rapid Application Development) é a necessidade de conseguir recursos suficientes para a montagem de vários grupos operando em paralelo
III)	O caso negócio (Business Case) é um dos produtos da fase de Concepção do Processo Unificado (Unified Process).

Assinale a alternativa *CORRETA:*

a) Apenas a afirmativa I é verdadeira.
b) Apenas a afirmativa II é verdadeira.
c) Apenas a afirmativa III é verdadeira.
d) Apenas as afirmativas I e II são verdadeiras.
e) Todas as afirmativas são verdadeiras.

6) A construção de sistemas é difícil devido à sua complexidade. Um fator crucial para gerenciar esta complexidade é o processo adotado para o desenvolvimento. O conjunto básico de atividades e a ordem em que são realizadas neste processo definem o que é também denominado de ciclo de vida do software. Analise as seguintes afirmações sobre processos de software:

I)	Um modelo de processo de software é uma representação abstrata de um processo; Exemplos de modelo de processos de software genéricos são o modelo waterfall (cascata) e o spiral (espiral);
II)	O modelo de processo waterfall ainda é hoje em dia um dos mais difundidos e tem por característica principal a codificação de uma versão executável do sistema desde as fases iniciais do desenvolvimento, de modo que o sistema final é incrementalmente construído, daí a alusão à idéia de “cascata” (waterfall);
III)	Em um processo de software incremental, o desenvolvimento do sistema é iterativo e partes de suas funcionalidades (denominadas “incrementos”) são entregues na medida em que são

	desenvolvidas; assim, estas entregas parciais tentam priorizar as necessidades mais urgentes do usuário e podem auxiliar a revisão e a uma melhor definição das partes ainda não entregues;

Levando-se em conta as três afirmações I, II e III acima, identifique a única alternativa válida:

a) Apenas a I e a II estão corretas;
b) Apenas a II e a III estão corretas;
c) Apenas a I e a III estão corretas;
d) As afirmações I, II e III estão corretas;
e) Apenas a III está correta.

7) O Processo Unificado (RUP – *Rational Unified Process*) é um moderno processo de desenvolvimento de *software* constituído de quatro fases. Assinale a opção que apresenta as quatro fases do RUP, na ordem em que elas devem ser executadas.

a) Concepção, elaboração, construção, teste.
b) Elaboração, transição, concepção, construção.
c) Elaboração, concepção, teste, transição.
d) Elaboração, concepção, transição, construção.
e) Concepção, elaboração, construção, transição.

8) Um modelo de processo de *software* pode ser visto como uma representação, ou abstração dos objetos e atividades envolvidas no processo. São modelos de processo de *software*, EXCETO:

a) RAD (*Rapid Application Development*).
b) Processo Unificado (UP – *Unified Process*).
c) Formal.
d) V-Model.
e) Espiral.

9) Analise as seguintes afirmativas sobre Engenharia de *Software*:

I)	Os modelos de maturidade têm o objetivo de avaliar a qualidade dos processos de *software* aplicados em uma organização (empresa ou instituição). Um exemplo de modelo de maturidade muito conhecido é o *Capability Maturity Model Integration* (CMMI) do *Software Engineering Institute* (SEI).
II)	*Refactoring* é o processo de modificar um sistema de *software* para melhorar seu comportamento externo, minimizando alterações na estrutura interna do código.
III)	Programação extrema (*eXtreme Programming*), ou simplesmente XP, é uma metodologia ágil para equipes pequenas e médias que irão desenvolver *software* com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de *software*.

São VERDADEIRAS as afirmativas:

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

10) Ordene as atividades abaixo segundo o ciclo de vida clássico de engenharia de software, também chamado modelo em cascata.

1. Manutenção
2. Teste
3. Projeto
4. Análise
5. Codificação

Assinale a alternativa *CORRETA:*

a) 1 – 2 – 3 – 4 – 5.
b) 4 – 3 – 5 – 2 – 1.
c) 2 – 5 – 4 – 1 – 3.
d) 3 – 5 – 4 – 1 – 2.
e) 3 – 4 – 5 – 2 – 1.

11) O conjunto básico de atividades e a ordem em que são realizadas no processo de construção de um software definem o que é habitualmente denominado de ciclo de vida do software. O ciclo de vida tradicional (também denominado *waterfall*) ainda é hoje em dia um dos mais difundidos e tem por característica principal:
    a) O uso de formalização rigorosa em todas as etapas de desenvolvimento;
    b) A abordagem sistemática para realização das atividades do desenvolvimento de software de modo que elas seguem um fluxo sequencial;
    c) A codificação de uma versão executável do sistema desde as fases iniciais do desenvolvimento, de modo que o sistema final é incrementalmente construído, daí a alusão à idéia de "cascata"(*waterfall*);
    d) A priorização da análise dos riscos do desenvolvimento;
    e) A avaliação constante dos resultados intermediários feita pelo cliente.

12) No desenvolvimento em espiral, cada loop representa uma fase do processo de software. Identifique abaixo a opção que contém os quatro setores que compõem cada loop do desenvolvimento em espiral:
    a) Definição dos requisitos, análise, projeto e testes.
    b) Definição dos objetivos, planejamento, identificação dos riscos e testes.
    c) Requisitos, desenvolvimento, validação e evolução.
    d) Identificação dos riscos, projeto, implementação e testes.
    e) Definição de objetivos, avaliação e redução dos riscos, desenvolvimento e validação, e planejamento.

13) Os modelos de ciclo de vida surgidos na área de Interação Humano-computador apresentam uma tradição mais forte de foco no usuário, quando comparados aos modelos de ciclo de vida da Engenharia de Software.
    Assinale a alternativa incorreta:
    a) O desenvolvimento de protótipos é parte integral do design iterativo centrado no usuário porque possibilita que designers testem suas idéias com usuários.
    b) O modelo de ciclo de vida Estrela surgiu de um trabalho empírico de observação de como os designers de interface de usuário trabalhavam.
    c) O modelo de ciclo de vida Estrela não especifica a ordem em que as atividades devem ser realizadas.
    d) O modelo de ciclo de vida Estrela é centrado na avaliação; sempre que uma atividade é completada, seu resultado deve ser avaliado.
    e) No modelo de ciclo de vida Estrela o projeto deve iniciar com a avaliação de uma situação existente.

14) Documentos de projeto de software servem principalmente para ajudar o projetista a tomar boas decisões e para explicar o projeto para os outros envolvidos.
    Levando em consideração o conteúdo de um documento de projeto, assinale a alternativa abaixo que contém tópicos de um modelo de guia para o documento de projeto.

    a) Objetivo, escopo, requisitos, principais características do projeto e detalhes do código.

b) Objetivo, prioridades gerais, visão geral do projeto, principais características do projeto e detalhes do projeto.
c) Visão geral do projeto, escopo, objetivo, principais características do projeto e detalhes do código.
d) Objetivo, prioridades gerais, requisitos, escopo e detalhes do projeto.
e) Visão geral do projeto, principais características do projeto, escopo e detalhes do projeto.

15) A Engenharia de Requisitos é um processo que envolve todas as atividades exigidas para criar e manter o documento de requisitos de sistema.
Sobre a Engenharia de Requisitos, considere as afirmativas a seguir.

I)	A Engenharia de Requisitos, como todas as outras atividades de Engenharia de Software, precisa ser adaptada às necessidades do processo, do projeto, do produto e do pessoal que está fazendo o trabalho
II)	No estágio de levantamento e análise dos requisitos, os membros da equipe técnica de desenvolvimento do software trabalham com o cliente e os usuários finais do sistema para descobrir mais informações sobre o domínio da aplicação, que serviços o sistema deve oferecer, o desempenho exigido do sistema, as restrições de hardware, entre outras informações.
III)	Na medida em que a informação de vários pontos de vista é coletada, os requisitos emergentes são consistentes.
IV)	A validação de requisitos se ocupa de mostrar que estes realmente definem o sistema que o cliente deseja. Ela é importante porque a ocorrência de erros em um documento de requisitos pode levar a grandes custos relacionados ao retrabalho.

Assinale a alternativa correta.
a) Somente as afirmativas I e II são corretas
b) Somente as afirmativas I e III são corretas.
c) Somente as afirmativas III e IV são corretas.
d) Somente as afirmativas I, II e IV são corretas.
e) Somente as afirmativas II, III e IV são corretas.

16) Considere as afirmativas abaixo:

I)	Requisitos não-funcionais não são mensuráveis.
II)	Requisitos funcionais descrevem as funções que o software deverá executar.
III)	Requisitos não-funcionais expressam condições que o software deve atender ou qualidades específicas que o software deve ter.

Assinale a alternativa CORRETA:
a) Somente as afirmativas I e II são verdadeiras.
b) Somente as afirmativas II e III são verdadeiras.
c) Somente a afirmativa III é verdadeira.
d) As afirmativas I, II e III são falsas.
e) Todas as afirmativas são verdadeiras.

17) Requisitos são capacidades e condições para as quais um sistema deve ter conformidade.
Analise as afirmações a seguir:

I)	No Processo Unificado, requisitos são categorizados de acordo com o modelo FURPS+, onde o U do acrônimo representa requisitos de usabilidade.
II)	Casos de uso são documentos em forma de texto, não diagramas, e modelagem de casos de uso é basicamente um ato de escrever estórias de uso de um sistema.
III)	UML (Unified Modeling Language) provê notação para se construir o diagrama de casos de uso, que ilustra os nomes dos casos de uso, atores e seus relacionamentos.

Considerando-se as três afirmações (I), (II) e (III) acima, identifique a única alternativa válida:
a) Somente as afirmações (I) e (II) estão corretas.

b) Somente as afirmações (II) e (III) estão corretas.
c) Somente as afirmações (I) e (III) estão corretas.
d) As afirmações (I), (II) e (III) estão corretas.
e) Somente a afirmação (III) está correta.

18) Requisitos de um sistema são freqüentemente classificados como funcionais, não-funcionais e de domínio. Qual a definição que melhor descreve requisitos não-funcionais?
a) São ferramentas automatizadas de apoio ao processo de desenvolvimento de sistemas.
b) São requisitos que descrevem o que o sistema deve fazer, como deve reagir a determinadas entradas e como deve comportar-se em situações particulares.
c) São requisitos que derivam do domínio da aplicação e que refletem características e restrições desse domínio.
d) São requisitos que não estão diretamente relacionados com as funções específicas do sistema.
e) São requisitos que especificam como deve ser testada uma parte do sistema, incluindo-se as entradas, os resultados esperados e as condições sob as quais os testes devem ocorrer.

19) No processo de desenvolvimento de um sistema de controle de materiais (matérias-primas) para uma metalúrgica, a equipe de projeto, responsável pelo mapeamento dos requisitos, desenvolveu seus trabalhos seguindo os quatro subprocessos da engenharia de requisitos. Inicialmente, foram feitas a análise e a avaliação para se verificar se o sistema seria útil ao negócio. Em um segundo momento, os requisitos foram identificados e analisados e, logo em seguida, foram documentados. Finalmente, foi verificado se os requisitos identificados atendiam às demandas dos usuários. Tendo sido executado esse procedimento, uma empresa independente de auditoria, após análise, identificou dois problemas no processo: a documentação dos requisitos (formulários e padrões utilizados) estava inadequada e não possibilitava o entendimento correto dos requisitos; o processo de checagem entre as demandas dos usuários e as especificações relatadas não foi bem conduzido e seus resultados eram insatisfatórios.

Considerando o relatório da auditoria independente, quais foram as duas fases do processo de engenharia de requisitos que apresentaram problemas?
a) Entendimento do domínio e especificação.
b) Elicitação e validação.
c) Validação e entendimento do domínio.
d) Especificação e validação.
e) Validação e elicitação

20) Assinale a função correta de engenharia de requisitos:
a) Determinar o objetivo geral do sistema.
b) Definir um amplo conjunto de conceitos, princípios, métodos e ferramentas que se pode considerar à medida que o software é planejado e desenvolvido.
c) Ajudar os engenheiros de software a compreender melhor o problema que eles vão trabalhar para resolver.
d) Usar uma combinação de formas textuais e diagramáticas para mostrar os requisitos de dados, função e comportamento.
e) Especificar o conjunto de áreas que farão parte do projeto.

21) Assinale a alternativa falsa
a) A validação de requisitos é um processo da Engenharia de Requisito que trata, tal como o seu nome indica, da validação quanto à consistência, precisão, contextualização de requisitos levantados no processo de identificação e descoberta e de análise e negociação de requisitos.
b) A validação de requisitos envolve uma revisão de todos os requisitos levantados e negociados, assim como uma prototipagem e validação de modelos e teste de requisitos.
c) A validação de requisitos é um dos mais importantes processos na Engenharia de Requisitos. Isto porque tal como um documento de requisitos bem definido permite a correção de incoerências e

inconformidades no desenvolvimento de um produto de software, a validação permite maximizar o tempo gasto na detecção dessas incoerências e inconformidades devido à sua alta eficiência na sua descoberta.

d) Um erro encontrado numa fase tardia do desenvolvimento do projeto pode ser desastroso, pois a sua alteração poderá ser bastante custosa, se não impossível, em termos temporais.
e) O processo de análise e negociação de requisitos agrega um grande volume de informação pouco estruturada e informal cedida pelos Stakeholders e tenta ir ao encontro das reais necessidades destes, através da estruturação, organização e interpretação desta informação e através da negociação desta nova reformulação da informação com os StakeHolders

22) Assinale a alternativa falsa
a) Tal como definido na engenharia de requisitos, os requisitos funcionais especificam resultados particulares de um sistema. Isto deve ser contrastado com requisitos não-funcionais, os quais especificam características gerais, tais como custo e confiabilidade.
b) Os requisitos funcionais denotam a arquitetura técnica de um sistema, enquanto os requisitos não funcionais fazem parte da arquitetura do aplicativo de um sistema.
c) Em alguns casos, um analista de requisitos gera casos de uso após a coleta e validação de um conjunto de requisitos funcionais.
d) Cada caso de uso ilustra cenários de comportamento através de um ou mais requisitos funcionais.
e) Muitas vezes, um analista começará por evocar um conjunto de casos de uso, a partir do qual o analista pode derivar os requisitos funcionais, que devem ser implementados para permitir que um usuário possa realizar cada caso de uso.

23) Engenharia de Software inclui um grande número de teorias, conceitos, modelos, técnicas e métodos. Analise as seguintes definições.

I)	No planejamento de projetos de software, há várias técnicas que podem ser usadas para estimativa de custo e esforço. A técnica de Pontos por Função é uma técnica de estimativa que, embora não seja relacionada diretamente a linhas de código, é utilizada também para a obtenção de métricas de produtividade e qualidade do desenvolvimento de software;
II)	CMMI (Capability Maturity Model Integration) é um modelo estabelecido pelo Software Engineering Institute (SEI) que propõe níveis de competência organizacional relacionados à qualidade do processo de desenvolvimento de software;
III)	Engenharia Reversa é o processo de inferir ou reconstruir um modelo de mais alto nível (projeto ou especificação) a partir de um documento de mais baixo nível (tipicamente um código fonte);

Levando-se em conta as três afirmações I, II e III acima, identifique a única alternativa válida:
a) Apenas a I está correta;
b) Apenas a II está correta.
c) Apenas a II e a III estão corretas;
d) Apenas a I e a III estão corretas;
e) As afirmações I, II e III estão corretas.

24) O gerenciamento de configuração de software (GCS) é uma atividade que deve ser realizada para identificar, controlar, auditar e relatar as modificações que ocorrem durante todo o desenvolvimento ou mesmo durante a fase de manutenção, depois que o software for entregue ao cliente.
O GCS é embasado nos chamados itens de configuração, que são produzidos como resultado das atividades de engenharia de software e que ficam armazenados em um repositório.
Com relação ao GCS, analise as duas asserções apresentadas a seguir.

- No GCS, o processo de controle das modificações obedece ao seguinte fluxo: começa com um pedido de modificação de um item de configuração, que leva à aceitação ou não desse pedido e termina com a atualização controlada desse item no repositório porque o controle das

modificações dos itens de configuração baseia-se nos processos de check-in e check-out que fazem, respectivamente,
- a inserção de um item de configuração no repositório e a retirada de itens de configuração do repositório para efeito de realização das modificações.

Acerca dessas asserções, assinale a opção correta.
a) As duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
b) As duas asserções são proposições verdadeiras, e a segunda não é uma justificativa correta da primeira.
c) A primeira asserção é uma proposição verdadeira, e a segunda é uma proposição falsa.
d) A primeira asserção é uma proposição falsa, e a segunda é uma proposição verdadeira.
e) As duas asserções são proposições falsas.

25) Analise estas afirmativas relacionadas à gerência de configuração de software:

I)	Os artefatos que fazem parte de uma linha-base somente podem ser alterados mediante procedimentos formais de controle de modificação.
II)	A identificação dos itens de configuração é processo integrante da gerência de configuração.
III)	Controle de mudanças e controle de versões têm o mesmo significado no contexto da gerência de configurações.

A partir dessa análise, pode-se concluir que
a) apenas a afirmativa I está correta.
b) apenas a afirmativa II está correta.
c) apenas a afirmativa III está correta.
d) apenas as afirmativas I e II estão corretas.
e) apenas as afirmativas II e III estão corretas.

26) O Fluxo de Análise das ameaças e riscos, na ordem apresentada, consiste de
a) Diversificação das ameaças, minimização das probabilidades dos riscos, redução dos pesos dos riscos, controle do risco, eliminação dos riscos prioritários, adoção de medidas de proteção lógica.
b) Determinação das probabilidades dos riscos, quantificação dos riscos, avaliação do risco, proteção de ativos, eliminação dos riscos.
c) Restrição das ameaças, planejamento das probabilidades dos riscos, determinação da hierarquia dos riscos, aquisição de software, estabelecimento de propriedades, redimensionamento.
d) Identificação das medidas de proteção, determinação das probabilidades de ameaças, determinação das prioridades dos pesos dos riscos, vinculação de ameaças a riscos, realocação de pessoal.
e) Identificação das ameaças, determinação das probabilidades dos riscos, determinação dos pesos dos riscos, avaliação do risco, estabelecimento de prioridades de proteção, adoção de medidas de proteção.

27) A situação atual do desenvolvimento de software encontra-se aquém do ideal. Sistemas são invariavelmente entregues com atraso ou com o orçamento estourado, isto quando são efetivamente entregues... E o que é pior, freqüentemente eles não atendem os requisitos dos clientes. Existem várias alternativas de tentar enfrentar este desafio, entre as quais a adoção de métodos formais, a sistematização do desenvolvimento usando processos tais como o Unified Process (UP) e a integração de novas tecnologias. Uma outra abordagem que recentemente vem ganhando adeptos é o Desenvolvimento Ágil de software.

As seguintes afirmações dizem respeito a ele.

I)	Suas idéias principais estão divulgadas em um Manifesto para o Desenvolvimento Ágil de Software escrito pela Aliança Ágil (Agile Alliance), que reúne autores famosos como Martin Fowler, Alistair Cockburn, Scott Ambler, Ward Cunningham e Kent Beck;
II)	Desnvolvimento Ágil basicamente concentra-se em melhorias na comunicação (interna à equipe e com os clientes), na entrega incremental de várias versões funcionais do software continuamente até o fim do projeto e na maleabilidade e dinamicidade do desenvolvimento, facilitando as respostas às mudanças que aparecem durante este desenvolvimento.
III)	A técnica mais conhecida de Desenvolvimento Ágil é a Programação eXtrema (Extreme Programming - XP) que entre suas práticas possui programação em pares (*pair programming*), entregas pequenas (*small releases*) e frequentes, a propriedade coletiva do código (*collective ownership*), abolindo as práticas de teste e os padrões de codificação;

Levando-se em conta as três afirmações I, II III acima, identifique a única alternativa válida:

a) Apenas a I e a II estão corretas.
b) Apenas a II e a III estão corretas.
c) Apenas a I e a III estão corretas.
d) Apenas a II e III estão corretas.
e) Apenas a III está correta.

28) Assinale a alternativa falsa

a) A medição é algo comum no mundo da engenharia. A engenharia de software está longe de ter uma medição padrão amplamente aceita e com resultados sem nenhum fator subjetivo.
b) Métricas de softwares nos possibilitam realizar uma das atividades mais fundamentais do processo de gerenciamento de projetos que é o planejamento.
c) As métricas de software, do ponto de vista de medição, podem ser divididas em duas categorias: medidas diretas e indiretas.
d) Podemos considerar como medidas diretas do processo de engenharia de software o custo e o esforço aplicados no desenvolvimento e manutenção do software e do produto, a quantidade de linhas de código produzidas e o total de defeitos registrados durante um determinado período de tempo. Porém, a qualidade e a funcionalidade do software ou a sua capacidade de manutenção são mais fáceis de serem avaliadas e só podem ser medidas de forma direta.
e) Podemos dividir as métricas de software, sob o ponto de vista de aplicação, em duas categorias: métricas de produtividade e de qualidade.

29) São axiomas em risco

a) É impossível testar um programa completamente. Teste de software é um exercício baseado em certezas. Quanto menos bugs forem encontrados, mais bugs existirão.
b) É possível testar um programa completamente. Teste de software não pode ter riscos. Quanto mais breaks forem encontrados, mais breaks existirão.
c) É impossível testar um programa completamente. Teste de software é um exercício baseado em risco. Quanto mais bugs forem encontrados, mais bugs existirão.
d) É impossível testar um programa que tenha riscos. Teste de software deve ser feito pelos seus desenvolvedores. Todos os bugs encontrados serão consertados.
e) É impossível testar um programa parcialmente. Teste de software aplica-se unicamente a ambientes sem risco. Quanto mais bugs forem encontrados, menos bugs existirão

30) Assinale a alternativa falsa

a) As métricas de produtividade se concentram na entrada do processo de engenharia de software e métricas de qualidade indicam o quanto o software atende aos requisitos definidos pelo usuário.
b) A medida de software mais familiar é a contagem de linhas de código. Esta métrica pode parecer simples, mas existe discordância sobre o que constitui uma linha de código.
c) Em vez de contar as linhas de código, a métrica orientada à função concentra-se na funcionalidade do software.
d) A medição é algo comum no mundo da engenharia. Mas para engenharia de software está longe se ter uma medição padrão amplamente aceita e com resultados sem nenhum fator subjetivo.
e) As métricas de software podem ser classificadas em medidas diretas (quantitativas) e medidas indiretas (qualitativas). As medidas diretas são aquelas que representam uma quantidade observada, tais como custo, esforço, número de linhas de código, tempo de execução e número de defeitos. Já as medidas indiretas são aquelas que exigem análise e estão relacionadas com a funcionalidade, qualidade, complexidade e manutenibilidade

31) Em relação à componentização e reuso, considere:

I. Se o componente sendo projetado é muito complicado, então, não é usável, por ser muito complexo ou apenas uma pequena porção desse componente é usada. Ao projetar um componente reusável, deve-se estar atento para que ele seja tão simples quanto possível.

II. Quando é projetada uma solução baseada em componentes, é possível obter um comportamento comum de modo que vários usuários possam utilizar. Uma outra forma para reuso de interfaces genéricas é o reuso da especificação. Uma vez que os componentes podem possuir múltiplas interfaces, é possível ter diferentes componentes.

III. No que concerne ao reuso dos componentes existentes, as interfaces podem ser projetadas para usar outras interfaces em tempo de design (desde que todas as implementações de componentes no sistema especificado suportem as interfaces) ou em tempo de implementação (usa os serviços de outras interfaces).

É correto o que consta em:

A) I, apenas.
B) II, apenas.
C) I, II e III.
D) III, apenas.
E) I e III, apenas.

32) Assinale a alternativa falsa
A) As métricas de software auxiliam diretamente no planejamento do projeto. Por exemplo, a métrica “LOC (Lines of Code)” é utilizada para dimensionar prazo e custo através da contagem de linhas de código
B) Diversas são as métricas existentes e as suas aplicações no ciclo de vida de um software. Cabe ao gerente de projeto coordenar as ações para determinar o padrão de qualidade requerido e definir quais elementos devem ser medidos e monitorados durante esse ciclo.
C) A coleta dessas informações permite não só um melhor acompanhamento do processo de desenvolvimento de um software, mas também a análise qualitativa desse software como um produto.
D) A base histórica das métricas permite que futuras propostas de mudança ou criação sejam mais precisas, visto que projetos similares tendem a passar pelos mesmos problemas e soluções.
E) Para manter ou elevar o nível de qualidade de um software não é essencial medir e monitorar durante todo o seu ciclo de desenvolvimento