# Introdução à Engenharia de Software

## Atividades de Aprendizagem e Avaliação

Lucas Daniel Batista, RA: 2159171

1. Considerando o conteúdo no link “Engenharia de Software – Preambulo”, complete
   a) Engenharia de Software é uma área da Computação dedicada a investigar os desafios e propor soluções que permitam desenvolver sistemas de software.

   b) Engenharia de Software trata da aplicação e abordagens sistemáticas, disciplinadas e quantificáveis para desenvolver, operar, manter e evoluir software.

   c) A Engenharia de Software surgiu da necessidade de que softwares fossem construídos com base em princípios práticos e teóricos, tal como ocorre em ramos tradicionais e bem estabelecidos da Engenharia.

   d) Segundo Brooks existem dois tipos de dificuldades em Desenvolvimento de Software

      1) dificuldades essenciais – relacionadas à área do problema

      2) dificuldades acidentais – relacionadas à tecnologia

   e) São dificuldades essenciais: a Complexidade; a Conformidade; a Facilidade de mudanças; e a Invisibilidade

   f) As 12 áreas de Engenharia de Software são:

      1) Engenharia de Requisitos

      2) Projeto de Software

      3) Construção de Software

      4) Testes de Software

      5) Manutenção de Software

      6) Gerência de Configuração

      7) Gerência de Projetos

      8) Processos de Software

9) Modelos de Software

10) Qualidade de Software

11) Prática Profissional

12) Aspectos Econômicos

g) Os requisitos funcionais de um sistema definem o que um sistema deve fazer; isto é, quais funcionalidades ou serviços ele deve implementar.

h) Os requisitos não funcionais de um sistema definem como um sistema deve operar, sob quais restrições e com qual qualidade de serviço.

i) O projeto de um Sistema de Software se inicia pela definição..... (módulos)

j) **Interfaces providas** se relacionam com o resto do sistema

k) **Interfaces requeridas** se relacionam com uma unidade de código, que depende dela para funcionar.

l) A **Arquitetura de Software** trata da organização de um sistema em um nível de abstração mais alto do que aquele que envolve classes ou construções semelhantes.

m) Testes de software mostram a presença de bugs mas não a sua ausência.

n) Testes de usabilidade objetivam verificar a usabilidade da interface do sistema.

o) Os **testes** podem ser usados para **verificação** com o objetivo de garantir que um sistema atende à sua especificação ou para **validação** com o objetivo de garantir que um sistema atende às necessidades de seus clientes

p) **Defeitos** são bugs, problemas no código já **falhas** ocorrem quando um código um código com defeito for executado.

q) Nem todo defeito resulta em uma falha pois pode acontecer que o código defeituoso nunca seja executado.

r) O defeito no código do foguete ‘Ariane 5’ estava relacionado com um trecho responsável pela conversão de um número real, em ponto flutuante, com 64 bits, para um número inteiro, com 16 bits, a falha ocorreu quando alguma situação nunca testada previamente exigiu a conversão de um número maior do que o maior inteiro que pode ser representado em 16 bits.

s) As manutenções de software podem ser classificadas em: corretiva, preventiva, adaptativa, refactoring e evolutiva

t) Manutenção adaptativa tem por objetivo adaptar um sistema a uma mudança em seu ambiente, incluindo tecnologia, legislação, regras de integração com outros sistemas ou demandas de novos clientes.

u) *Refactoring* é um tipo de manutenção que tem por objetivo a melhoria de seu código ou projeto.

v) Gerência de Configuração se relaciona com o conjunto de políticas para gerenciar as diversas versões de um sistema.

w) A **Lei de Brooks** é: A inclusão de novos desenvolvedores em um projeto que está atrasado contribui para torná-lo ainda mais atrasado.

x) A gerência de Projetos se ocupa de atividades tais como: prazos, contratos, Lei de Brooks, etc.

y) Um PROCESSO DE DESENVOLVIMENTO DE SOFTWARE define quais atividades e etapas devem ser seguidas para construir e entregar um sistema de software. .

z) Processos ***Waterfall*** (em cascata) foram inspirados nos processos usados em engenharias tradicionais e são largamente sequenciais

aa) As etapas de um processo de software em cascata são:

1) Levantamento de requisitos
2) Análise
3) Projeto
4) codificação
5) Testes
6) Implantação

bb) O **Manifesto Ágil** foi produzido em Utah, Estados Unidos no ano de 2001 por um grupo de 17 Engenheiros de Software

cc) A principal característica de um PROCESSO DE DESENVOLVIMENTO DE SOFTWARE ÁGIL é que um sistema deve ser construído de forma incremental e iterativa.

dd) XP, Scrum, Kanban e Lean Development. são exemplos de processos ágeis.

ee) A **Integração Contínua** recomenda que desenvolvedores integrem o código que produzem imediatamente, se possível todo dia.

ff) Modelos criados para entender um sistema já implementado são instrumentos de .....

gg) A **Qualidade de Software** pode ser avaliada em duas dimensões:

1) qualidade externa

2) qualidade interna.

hh) A qualidade externa considera fatores que podem ser aferidos sem analisar o código.

ii) Conceitue

1) Robustez o software continua funcionando mesmo quando ocorrem eventos anormais, como uma falha de comunicação ou de disco? Por exemplo, um software robusto não pode sofrer um crash (abortar) caso tais eventos anormais ocorram. Ele deve pelo menos avisar por qual motivo não está conseguindo funcionar conforme previsto.

2) Eficiência o software faz bom uso de recursos computacionais? Ou ele precisa de um hardware extremamente poderoso e caro para funcionar?

jj) A qualidade interna considera propriedades e características relacionadas com a implementação de um sistema.

kk) São exemplos de atributos da qualidade interna

1) modularidade

2) legibilidade

3) manutenibilidade

4) testabilidade.

ll) Cite um exemplo de métrica de processo: número de defeitos reportados em produção por usuários finais em um certo intervalo de tempo.

mm) Revisões de código tem por objetivo detectar bugs antecipadamente, antes de o sistema entrar em produção e disseminar oas práticas de Engenharia de Software entre os membros de um time de desenvolvimento.

nn) ***Over-engineering*** é uso de recursos mais sofisticados em um contexto que não demanda tanta preocupação.