**Introdução à Engenharia de Software**

**Atividades de Aprendizagem e Avaliação**

**Aluno\_André Luis Quiosi\_\_\_\_\_RA\_\_2369958**

Use esta cor em seu texto

1. Considerando o conteúdo no link “Engenharia de Software – Preambulo”, complete
   1. Engenharia de Software é uma área da Computação dedicada a propor e aplicar princípios de engenharia na construção de software.
   2. Engenharia de Software trata da aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para desenvolver, operar, manter e evoluir software.
   3. A Engenharia de Software surgiu da necessidade de .......
   4. 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
   5. São dificuldades essenciais: a complexidade; a conformidade; a facilidade de mudanças; e a Invisibilidade
   6. 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
   7. Os requisitos funcionais de um sistema definem o que um sistema deve fazer
   8. Os requisitos não funcionais de um sistema definem como um sistema deve operar, sob quais restrições e com qual qualidade de serviço
   9. O projeto de um Sistema de Software se inicia pela definição de interfaces, incluindo interfaces providas e interfaces requeridas (módulos)
   10. **Interfaces providas** se relacionam com aqueles serviços que uma unidade de código torna público para uso pelo resto do sistema.
   11. **Interfaces requeridas** se relacionam com interfaces das quais uma unidade de código depende para funcionar.
   12. 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.
   13. Testes de software mostram a presença de bugs, mas não a sua ausência.
   14. Testes de usabilidade objetivam verificar a usabilidade da interface do sistema.
   15. 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.
   16. **Defeitos** são erros de lógica já **falhas** ocorrem quando um código com defeito é executado.
   17. Nem todo defeito resulta em uma falha pois pode acontecer
   18. O defeito no código do foguete ‘Ariane 5’ estava relacionado com um número inteiro 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.
   19. As manutenções de software podem ser classificadas em: corretiva, preventiva, adaptativa, refactoring e evolutiva.
   20. 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.
   21. *Refactoring* é um tipo de manutenção que tem por objetivo a melhoria em um software preservando seu comportamento e visando exclusivamente a melhoria de seu código ou projeto.
   22. Gerência de Configuração se relaciona com o conjunto de políticas para
   23. A **Lei de Brooks** é: A inclusão de novos desenvolvedores em um projeto que está atrasado contribui para torná-lo ainda mais atrasado.
   24. A gerência de Projetos se ocupa de atividades tais como: prazos, contratos, Lei de Brooks, etc.
   25. Um PROCESSO DE DESENVOLVIMENTO DE SOFTWARE define quais atividades e etapas devem ser seguidas para construir e entregar um sistema de software.
   26. Processos ***Waterfall*** (em cascata) foram inspirados nos processos usados em engenharias tradicionais e são largamente sequenciais, como ilustrado no exemplo do prédio, usado no parágrafo inicial desta seção.
   27. As etapas de um processo de software em cascata são:
2. levantamento de requisitos.
3. análise (ou projeto de alto nível).
4. projeto detalhado.
5. codificação.
6. testes.
   1. O **Manifesto Ágil** foi produzido em Utah, Estados Unidos no ano de 2001 por um grupo de 17 Engenheiros de Software.
   2. A principal característica de um PROCESSO DE DESENVOLVIMENTO DE SOFTWARE ÁGIL é que sistema deve ser construído de forma incremental e iterativa.
   3. XP, Scrum, Kanban e Lean Development são exemplos de processos ágeis.
   4. A **Integração Contínua** recomenda que desenvolvedores integrem o código que produzem imediatamente, se possível todo dia.
   5. Modelos criados para entender um sistema já implementado são instrumentos de Engenharia Reversa
   6. A **Qualidade de Software** pode ser avaliada em duas dimensões:
      1. Qualidade interna
      2. Qualidade externa
   7. A qualidade externa considera fatores que podem ser aferidos sem analisar o código.
   8. Conceitue
      1. Robustez: um software robusto não pode sofrer um crash (abortar) caso 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?
   9. A qualidade interna considera propriedades e características relacionadas com a implementação de um sistema.
   10. São exemplos de atributos da qualidade interna
       1. Modularidade
       2. Legibilidade
       3. Manutenibilidade
       4. Testabilidade
   11. Cite um exemplo de métrica de processo: número de defeitos reportados em produção por usuários finais em um certo intervalo de tempo.
   12. Revisões de código tem por objetivo detectar bugs antecipadamente e disseminar boas práticas de Engenharia de Software entre os membros de um time de desenvolvimento.
   13. ***Over-engineering*** é o uso de recursos mais sofisticados em um contexto que não demanda tanta preocupação.