Estudo dirigido preparatório para o Trabalho 2 e para a Prova.

1. Comente 3 mitos e 3 realidades relacionadas à atividade de Engenharia de Software.

2. Indique 3 dificuldades encontradas na atividade de engenharia de Software.

3. Descreva com as suas palavras o princípio de abstração.

4. Descreva com suas palavras o princípio de dividir para conquistar.

5. Descreva com as suas palavras o princípio de completeza.

6. Comente, com as suas palavras, como funciona o desenvolvimento Top-Down?

7. Comente, com as suas palavras, como funciona o desenvolvimento Bottom-up?

8. Qual o papel das normas e modelos no contexto da Engenharia de Software?

9. No contexto da engenharia de software, o que significa o termo processo?

10. Quais são as atividades normalmente existentes num processo de software?

11. O que é um ciclo de vida de um software?

12. Qual a diferença do modelo cascata do incremental?

13. Quando podemos optar por um modelo cascata, incremental, prototipação ao espiral? Que critérios podemos considerar para escolher o melhor modelo para o produto de software a ser desenvolvido?

14. Por que um processo de software é importante?

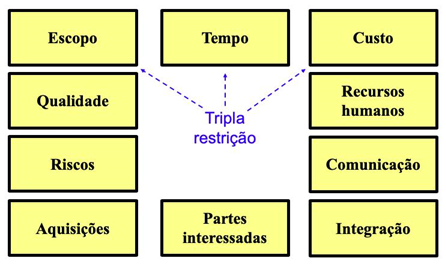
15. Em que situações podemos considerar atividades de adaptação ou de melhoria de um processo?

16. O que são o MPS.BR e o CMMI?

17. No contexto de gerenciamento de projetos, segundo o PMBOK, por que um projeto “Um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado único”?

18. Quais são geralmente as atividades envolvidas no gerenciamento de um projeto de software?

19. Discuta as áreas de conhecimento abaixo indicadas para o gerenciamento de um projeto:



20. Qual a importância da definição de escopo no gerenciamento de um projeto de software

21. O que são requisitos de um software?

Requisitos são os objetivos, a funcionalidade de um código - o que precisa funcionar.

22. O que são requisitos funcionais?

São o “o que?” de um sistema, quais as necessidades, serviços por serem implementados.

23. O que são requisitos não-funcionais?

São o como de um sistema; de que maneira os requisitos funcionais serão aplicados.

24. De um exemplo de um requisito funcional e um não funcional relacionado a app da PUCRS.

Requisito funcional: Consultar as notas das provas

Requisito não funcional: Ser um app mobile

25. O que define um requisito de sistema e um requisito de usuário?

Requisito de sistema é mais técnico e requer, normalmente, um desenvolvedor.

Requisito de usuário é mais natural e não requer muito conhecimento.

26. Qual o impacto de um requisito nào-funcional em um produto de software?

27. Explique o ciclo de engenharia de requisitos abaixo indicado:



28. Qual o papel do modelo de casos de uso no processo da engenharia de requisitos?

29. Qual o papel da gerencia de configuração no processo de desenvolvimento de um software?

30. Explique no contexto de um projeto de software em java ou qualquer outra linguagem, os conceitos abaixo relacionados:

• Item de configuração

• Versão

• Baseline

• Release

• Workspace

• Build

• Branch

• Merge

31. Por que é necessário definir a arquitetura de um software? Quando definida a arquitetura da torre de macarrão, quais foram as preocupações para que ela desse certo? E quais destas preocupações varem para a arquitetura de um software?

32. Qual o impacto dos requisitos não funcionais em uma arquitetura de software?

33. O que são padrões de arquitetura de software?

Estes padrões são soluções já consolidadas que auxiliam na solução de problemas e em todo o projeto de software.

34. Por que é importante definirmos o projeto de um software antes de desenvolvê-lo propriamente dito?

É importante pois afeta toda a organização e agilidade na hora do desenvolvimento.

35. Qual o papel do diagrama de classes na definição de um projeto de software?

O diagrama de classes é útil pois esta parte da arquitetura permite a melhor estruturação do programa e métodos e organização geral.

36. Comente as diretrizes que orientam a atividades de implementação de um software?

O que, quando, quem e como.

37. O que significa implementar para reutilização de código e implementar com a reutilização de código. Dê um exemplo para cada um.

Implementar para reutilização de código é quando cria-se um componente “genérico” para o qual diversas funções podem ser atribuidas em diversos outros sistemas. Um exemplo seria um método/função somaDois que recebe como parametro dois números e retorna a soma entre eles. Implementar com a reutilização de código é quando, por exemplo, utilizamos um import - como o Math.random() - que existe na maioria das linguagens para realizar uma atividade que precisa de um número aleatório, ao invés de gerá-lo por meio de um código próprio.

38. Qual o papel da IDES na atividade de implementação de um produto de software?

Papel importantíssimo, visto que auxiliam em muito na vida do desenvolvedor, facilitando testes e aumentando a produtividade geral.

39. Qual a diferença dos conceitos de verificação e validação de software?

Verificação de Software é uma análise mais técnica e funcional enquanto a validação é a parte mais "humana" de verificar se as expectativas e necessidades do cliente são atendidas.

40. O que difere Erro, Defeito e Falha?

Uma falha é algo que dá errado

Um defeito é quando um erro gera uma falha

Um erro é a ausência do resultado correto

41. O que significa o conceito de Depuração (debugging)?

O debugging é o processo de análise do software pelo programador buscando identificar problemas

42. Quando se faz Teste de unidade, Teste de integração e Teste de sistema?

Teste de Unidade - Teste simples e básico que testa retornos inesperados

Teste de Integração - Testa a unidade de grupos de componentes do sistema, chamadas externas por exemplo

Teste de sistema - Teste final de todas as funcionalidades e componentes baseado no usuário final.

43. O que e o teste de aceitação?

Similar ao teste de sistema, extensão deste.

44. Qual o papel das ferramentas de automação de testes?

Auxiliam na hora dos testes, de executá-los e gerenciá-los

45. Quando se desenvolve atividade de manutenção de um software

Rotineiramente; antes de problemas acontecerem