ATIVIDADE

1.Defina o que são requisitos e quais são os tipos com exemplos dos mesmos.

Mostram o que deve ter no sistema, o que ele deve fazer e as suas restrições. Os tipos são requisitos funcionais e não-funcionais. Requisitos funcionais são aqueles que ligados ao que o sistema faz, como cadastrar usuário, editar usuário, visualizar produtos, etc... Requisitos não-funcionais é inerente ao funcionamento do sistema no geral, como desempenho, segurança, usabilidade, etc...

2. Na fase de elicitação de requisitos é onde descobre e entende os principais requisitos do sistema que se pretende construir, dessa forma, apresente 5 técnicas que podem ser usadas na elicitação de requisitos.

Entrevistas, questionários, workshops, protótipos e estudos etnográficos.

3.Os profissionais da indústria que seguem os métodos ágeis utilizam uma técnica pragmática para solucioná-los os problemas com os requisitos tradicionais utilizados nos métodos Waterfall, que ficou conhecida pelo nome de **Histórias de Usuários**. Assim, uma história de usuário é composta por três partes, todas começando com a letra C, quais são e explique.

Cartão, Conversas e Confirmação.

Cartão: As histórias de usuários são escritas em cartões que serão colocadas num quadro, podendo ser Kanban. Estes cartões possuem dados sobre o que será feita, quem pediu, prioridades e outras informações. Resumindo, é onde é escrito algum requisito.

Conversa: Entre clientes e desenvolvedores, por meio das quais os clientes explicam e detalham o que escreveram em cada cartão.

Confirmação: Cenários serão usados pelo PO para aceitar a implementação da história.

4. Através de um exemplo de um sistema, escreva histórias de usuários para 3 tipos de usuários utilizando o seguinte formato: **Como um [papel de usuário], eu gostaria de [realizar algo com o sistema].**

Como um caixa, eu gostaria de registrar a compra do cliente e imprimir uma nota fiscal

Como um gerente geral, eu gostaria de visualizar o histórico de venda dos 10 produtos mais vendidos.

Como um gerente de estoque, eu gostaria de verificar quais produtos estão com baixa quantidade.

5. Quais são as características de boas histórias? Explique cada características.

INVEST. Independent, Negotiable, Valuable, Estimable, Succint, Testable.

* Independentes. Não depender de outras, existindo de forma completa por si só.
* Negociáveis. Histórias flexíveis, que podem ser discutidas durante as conversas e discussões.
* Valiosas. Cada história de usuário deve fornecer valor direto ao usuário final ou a outros stakeholders do projetos.
* Estimáveis. Para calcular o esforço necessário para a implementação de cada história.
* Sucintas. Histórias pequenas para que sejam implementadas em um tempo razoável.
* Testáveis. Deve ter critérios de aceitação claros para definir se a história foi implementada corretamente.

6.Explique o que é **Produto Mínimo Viável** (**MVP**). Esse método consiste em um ciclo com três passos, quais são e explique.

É um sistema funcional, muito básico, que serve para avaliar a viabilidade do projeto. Os passos são: construir, medir e aprender. Na construção, têm-se a ideia do produto e faz-se o MVP. Na medição, o MVP é usado por clientes para poder coletar dados de uso e analisar sua viabilidade. No aprendizado, é feita a análise dos dados para melhorar os próximos ciclos do MVP ou então abandonar o projeto se os dados mostrarem que ele não é viável.

7. (ENADE 2011) Uma equipe está realizando testes com o código-fonte de um sistema. Os testes envolvem a verificação de diversos componentes individualmente, bem como das interfaces entre eles. Essa equipe está realizando testes de:

1. unidade
2. aceitação
3. sistema e aceitação
4. integração e sistema
5. unidade e integração

8. Quando o desenvolvimento era em cascata, os testes ocorriam em uma fase separada, após as fases de levantamento de requisitos, análise, projeto e codificação. Além disso, existia uma equipe separada de testes, responsável por verificar se a implementação atendia aos requisitos do sistema. Para checar isso, frequentemente os testes eram manuais. Com métodos ágeis, a prática de testes de software foi reformulada, explique as diferenças.

Métodos ágeis usam muito testes automatizados, com todos os desenvolvedores envolvidos nos testes, ao invés de uma equipe dedicada a isso. Cada desenvolvedor testa o código que ele desenvolveu, ou seja, são responsáveis por testarem suas próprias criações.

9. Uma forma de classificar testes automatizados é por meio de uma **pirâmide de testes,** assim os testes são divididos em 3 grupos, classifique e explique cada grupo.

Teste de unidade: Pequenas partes do sistema, geralmente classes.

Testes de integração: Partes do sistema. Teste da interface entre diferentes classes.

Testes de sistema: Todo o sistema. A interação completa dum usuário real, do ponto de entrada, até a saída (testes de ponta-a-ponta).

10. Desenvolvimento Dirigido por Testes (Test Driven Development, TDD) é uma das práticas de programação propostas por Extreme Programming (XP). Assim, o desenvolvedor segue um ciclo composto por três estados, explique.

Vermelho, verde e refatorar. Inicialmente, o código de teste é criado (isso mesmo, o teste é criado antes do código que será testado). Ao executar o teste, ele fica vermelho, mostrando os erros. Ao implementar a classe “original” e atingir o comportamento esperado, o teste fica verde. Após o teste passar, o desenvolvedor pode refatorar para melhorar a manutenibilidade do código. Após o refatoramento, esse processo é repedido.

11.Qual a diferença entre defeito ou bug e falha.

Defeito = bug: Erro no código. Lógica ou sintaxe.

Falha: Resultado errado após a execução dum código defeituoso, divergindo dos requisitos.

12.Explique o que é teste funcional (caixa preta) e teste estrutural (caixa branca).

Funcional é sobre um processo completo feito por um usuário, como cadastrar um produto e ver se ele foi realmente cadastrado.

Estrutural é sobre a forma como o sistema realiza as funcionalidades. Como ele está escrito para realizá-los.

13. Quando estimamos o software alguns itens são levados em consideração no momento que vamos estimar o software no plano de projeto, explique.

* Tamanho do software, medido em pontos por função ou número de linhas de código.
* Complexidade do software, que diz se é difícil ou fácil, que irá impactar no tempo gasto.
* Experiência da equipe, que impacta na velocidade de desenvolvimento do software, aumentando ou diminuindo o tempo gasto, dependendo da expertise e do entrosamento da equipe.
* Histórico de softwares semelhantes podem trazer uma ideia de como foi implementado, podendo haver reuso dele.
* Produtividade mostra, mais ou menos, o número de códigos por minuto.
* Esforço mostra, mais ou menos, o número de códigos para implementar uma funcionalidade.
* Especialistas podem ser usados para estimar o software. Com vários especialistas, compara-se os resultados deles para chegar a um consenso.

14. Explique por que um sistema de software usado em um ambiente real deve mudar ou tornar-se progressivamente menos útil.

Porque os requisitos e o próprio ambiente mudam. Aparecem novos concorrentes, os clientes precisam de novas funcionalidades do sistema, e hardwares são modificados, o que pode causar incompatibilidades ao sistema.

15. Quais são as fases de um sistema de evolução de software, explique casa fase.

* Especificação: possui os requisitos, o que o sistema deve e não deve fazer
* Implementação: construção do sistema
* Validação: Verifica se a implementação está de acordo com as especificações e a avaliação com o cliente
* Operação: Uso do sistema por parte do usuário

16. Explique a diferença de um sistema em evolução e em serviço.

Em evolução é aquele que segue sendo modificado para se adequar a novos requisitos e melhorias.

Em serviço é aquele está ativo, sem novas funcionalidades, e as manutenções são apenas para garantir estabilidade e confiabilidade.

17. O processo de evolução de mudanças pode variar entre as organizações, ou seja, pode ser formal ou informal. Assim, no processo de evolução formal cada mudança deve ser solicitada e planejada, e para isso, passa por algumas fases para verificar a viabilidade das mudanças. Explique como é o processo desse ciclo de mudanças.

1. Solicitação de mudança: alguém pede uma mudança e documenta-a.
2. Avaliação inicial: é observada a justificativa, e são definidos estimativas, custos e prazos.
3. Análise de impacto: vê onde o sistema será afetado, dependências, entre outros.
4. Aprovação da mudança
5. Planejamento da mudança
6. Implementação da mudança
7. Testes e validação
8. Implantação e monitoramento: lançar as mudanças e obter feedback dos usuários
9. Avaliação pós-implantação
10. Encerramento do ciclo de mudanças
11. Registro de mudanças

18. Em algumas situações emergenciais, algumas solicitações de mudanças podem ser urgentes, apresente alguns exemplos.

Vulnerabilidade de segurança, mudança legislativa, ou quando ocorre algum erro. Então é necessária uma manutenção corretiva, para voltar o programa ao pleno funcionamento.

19. O objetivo das leis de lehman é definir uma teoria unificada para a evolução de software e ai resultou e oito leis que “governam” a evolução de sistemas, cite pelo menos 4 leis e explique.

* Mudança contínua, onde os softwares sempre estão mudando, sob pena de tornarem-se inúteis.
* Complexidade crescente, onde a estrutura do software se torna mais complicada à medida que manutenções e novas funcionalidades são feitas, crescem o programa, e o torna ilegível.
* Crescimento contínuo, onde novas funcionalidades devem ser lançadas frequentemente para que os usuários mantenham o interesse pelo sistema.
* Sistemas de feedback, para ouvir as reclamações e sugestões dos usuários e implementar as devidas correções.

20. Descreva brevemente os três principais tipos de manutenção de software. Por que às vezes é difícil distingui- los?

* Preventiva: Prever um potencial problema, e aplicar correções para que não ocorram.
* Corretiva: Consertar um problema que está prejudicando o funcionamento normal do sistema.
* Aditiva: Adição de novas funcionalidades.

Porque algumas aditivas podem causar problemas que necessitam de correções, aí mistura tudo.

21.Segundo Sommerville existe vários fatores estão associados aos elevados custos de manutenção, quais são?

* Estabilidade da equipe
* Responsabilidade contratual
* Habilidades da equipe
* Idade e estrutura do programa

22. A análise por pontos de função utiliza alguns domínios de informação para quantificar o produto software. Dentre tais domínios, incluem-se

a) o número de comandos do tipo case e a quantidade de memória utilizada.

b) o número de tabelas utilizadas no sistema e o número de comandos do tipo case.

c) o número de entradas e saídas externas e o número de consultas externas.

d) a quantidade de memória e o número de tabelas utilizadas no sistema.

e) a velocidade de processamento e o número de consultas externas.