Lista de Exercícios de Engenharia de Software – Implementação e Testes
Prof. Mariane Moreira de Souza
Semestre: 2024/2
Dupla: Fábio Borges Júnior e Isabella Cristina da Silveira.

1. Uma das atividades do fluxo de implementação é a construção de protótipos. Qual o objetivo dessa prática? Em que momento (fases: concepção, elaboração, construção e transição) de um processo tradicional iterativo incremental a prototipação pode ser aplicada?

A construção de protótipos tem como objetivo possibilitar a visualização das ideias, conceitos e requisitos antes da implementação final do projeto.

Pode-se aplicar a prototipação em diferentes fases, na fase de concepção, a prototipação faz com que os requisitos iniciais sejam esclarecidos permitindo ajustes nas expectativas antes de ir para a próxima etapa, na fase de elaboração a prototipação ajuda a refinar os requisitos, detalhando as funcionalidades específicas, assim entendendo melhor a necessidade do usuário. Na fase de construção, a prototipação vai garantir que as expectativas e requisitos sejam atendidos, validando as implementações conforme são feitas, já na fase de transição, a prototipação pode ser utilizada nos testes das funcionalidades e validar a experiência do usuário.

2. O que são boas práticas de codificação? Cite três exemplos e explique sua importância.

São regras para manter os desenvolvedores a escrever código de forma mais eficiente, legível e fácil de manter. Segundo Robert C. Martin, autor do livro "Código Limpo - Habilidades práticas do agile software", há tais boas práticas na codificação:

- Uso de comentários: Garantir que o código seja compreensível para outros desenvolvedores com explicações caso não seja explícito no trecho do código.
- Formatação: Serve como comunicação entre os membros da equipe, garantindo uma padronização de escrita, como espaçamentos verticais, maneiras de declarar variáveis.
- Tratamento de erro: Hoje em dia, as sintaxes das linguagens permitem maior compreensão do código para tratar erros.
-

3. O que é refatoração de código? Quais são os benefícios e quando deve ser aplicada?

A refatoração do código é o processo de reestruturar um código já existente sem alterar seu comportamento externo. Os benefícios são a otimização da legibilidade, da estruturação, a manutenção do sistema, a organização e a compreensão do código fonte. A refatoração deve ser aplicada quando é um código de difícil entendimento e manutenção, antes de adicionar novas funcionalidades, se tiver uma visão de melhoria para o sistema e após correção de bugs no sistema.

4. Como as ferramentas de controle de versão, como Git, ajudam na construção de software? Dê exemplos de práticas recomendadas.

Segundo DIAS, H. et al. versionamento de código é indispensável para fornecer ao analista a possibilidade de controlar os responsáveis por cada mudança para dividir as entregas do projeto. Algumas práticas no controle de versão são adequadas, como:

- Commits significativos
- Organizar estruturas de branches
- Merge e resolução de conflitos.

5. Explique, concordando ou discordando, a seguinte frase: "A definição e execução de testes são tarefas exclusivas do fluxo de teste".

Discordo da afirmação, pois embora no fluxo de teste ele trate essas etapas como as fundamentais do processo de garantia de qualidade, a colaboração entre equipes é essencial para a definição de requisitos e testes eficazes, assim como um feedback contínuo para que os testes possam ser ajustados ao longo do desenvolvimento. Então, a definição e execução de testes devem ser incluídas em todo o ciclo de desenvolvimento.

6. Descreva a importância da documentação na fase de construção do software. Que tipos de documentação você considera essenciais?

Segundo Simone Coelho: "A evolução de um software e seu sistema operacional devem ser claramente informados, estabelecendo procedimentos para registrar e resolver problemas, facilitando sua manutenção posterior e, assim, garantindo sua durabilidade". Em suma, a documentação tem como principal importância manter o código durável e facilitar suas posteriores manutenções. Os tipos de documentos que considero como essenciais comentários explicando no próprio código para maior entendimento do trecho e fluxogramas.

7. Qual a diferença entre atividades de inspeção de implementação e teste?

A inspeção é uma técnica que permite testar outros artefatos além do código fonte, principalmente no documento de definição de requisitos do software. O teste identifica falhas em funcionalidades, garantindo que o software se comporte conforme esperado sob diferentes cenários de uso. Então, enquanto a inspeção foca na revisão do código e conformidade, os testes se concentram em validar o comportamento do software em operação. São práticas que se complementam, trabalhando juntos na remoção de defeitos. A Inspeção encontra defeitos não encontrados em teste e o teste em si, encontra defeitos não encontrados na fase de inspeção ou que tenham sido inseridos depois da fase de Inspeção.

8. Explique a seguinte frase, concordando ou discordando: “O objetivo das metodologias de teste é maximizar a cobertura”.

Essa afirmação não é coerente, visto que os testes além de ser importante de ser aplicada em maior cobertura, seria de suma importância garantir a qualidade destes testes levando em consideração questões como segurança e qualidade.

9. Explique a seguinte frase, concordando ou discordando: “Assim como as atividades de desenho, as atividades de teste devem ser feitas por especialistas”.

Concordo, pois um especialista em teste vai trazer conhecimentos técnicos e metodológicos relevantes para um teste eficaz, pois são treinados para essa função, garantindo a qualidade do software e que ele funcione conforme esperado. Porém, a participação de outros profissionais pode oferecer um enriquecimento no processo de teste, com perspectivas diferentes sobre requisitos e cenários de uso, assim resultando em uma abordagem mais eficaz.

10. Dê um exemplo, no contexto de um sistema qualquer, de teste que utilize o método caixa branca e outro que utilize o método caixa preta.

Caixa Branca: Verifica se todas as rotas possíveis HTTP são seguidas corretamente no código.

Caixa Preta: Em um formulário de preenchimento de nome completo do indivíduo, verificar se é apenas possível a utilização de dados válidos.

11. Existem diferentes categorias de teste. Qual a diferença entre um teste de aceitação e um teste de unidade? Dê um exemplo de teste de sistema e outro de integração. Para que servem os testes de regressão?

A diferença é que os testes de aceitação são testes formais executados para verificar se um sistema atende aos requisitos de negócios, tem o foco de replicar o comportamento do usuário, já um teste de unidade é feito em nível baixo, próximo ao código fonte, verificando partes específicas do código, como funções ou partes específicas do código isoladamente, garantindo um funcionamento correto e rápida identificação de erros.

Um exemplo de teste de sistema onde será testado todo o sistema de um aplicativo de gerenciamento de projetos, o testador verifica todas as funcionalidades, como criação de tarefas, atribuição de usuários e geração de relatórios, funcionam bem em conjunto. Um exemplo de teste de integração seria testar a integração entre um módulo de autenticação e um módulo de banco de dados, garantindo que, quando um novo usuário for inserido, as informações estejam corretamente salvas no banco, assim o sistema retornará que a operação foi realizada com sucesso.

O teste de regressão é uma técnica aplicável a cada alteração realizada no software, tendo o objetivo de garantir que as mudanças realizadas nessa nova versão não gerarão erros em componentes prontos e testados. Consiste em aplicar, antes e depois da alteração, todos os testes que já foram aplicados nas versões anteriores.

12. O que é um caso de teste? Dê um exemplo simples de um caso de teste de um sistema de informação típico.

É uma condição restrita que mantém uma forma de entrada com a finalidade de observar o resultado de saída.

Teste para função par(numeroDeEntrada) para verificar se o número é par.

- Caso o número de entrada for par, positivo:

Entrada: par(4)

Saída: True

- Caso o número de entrada for ímpar, positivo:

Entrada: par(3)

Saída: False

13. Explique a seguinte frase, concordando ou discordando: "Uma vez que testes são atividades exaustivas, não tem como saber em que momento é melhor parar de testar".

Discordo, pois existem critérios e métricas que ajudam a decidir quando parar os testes, como a cobertura de testes que são as porcentagem de requisitos e funcionalidades cobertas por teste, o histórico de defeitos onde há uma análise de defeitos encontrados e a taxa dos novos erros encontrados durante o teste, o risco aceitável que avalia os riscos associados ao software, permitindo decidir quando risco é aceitável para lançamento e o tempo e o recurso, onde pode limitar o processo, pois é necessário balancear qualidade e viabilidade do projeto.

14. Como a técnica de TDD (Test-Driven Development) pode influenciar a qualidade do software produzido? Descreva o processo.

A técnica TDD pode influenciar nas seguintes qualidades:

- Desenvolvimento guiado
- Código mais enxuto
- Confiança nos testes
- Detalhamento na documentação
- Redução de erros e retrabalhos

A técnica resulta em um código mais robusto, já que você tem a necessidade de escrever o código, fazer o teste passar, além de corrigi-lo na refatoração.

15. Quais são os principais desafios que podem ser enfrentados durante a fase de testes de um software? Como superá-los?

Na fase de testes podem aparecer alguns obstáculos, como a cobertura de testes insuficiente, a falta de documentação, mudanças de requisitos, ambientes de testes inadequados, falta de tempo e recurso, dificuldade em reproduzir defeitos e a comunicação entre equipe. Podemos superá-los implementando uma estratégia de teste baseada em risco, utilizar ferramentas de análise de cobertura de testes, criar e manter uma documentação atualizada sobre requisitos, casos de teste e resultados, revisar e atualizar os casos de teste conforme os requisitos mudam, criar um ambiente de teste o próximo possível do ambiente de produção, incluindo dados e configurações similares, planejar e priorizar testes com base no impacto e na probabilidade de falhas, implementar testes de regressão e registrar as informações sobre os ambientes que ocorreram defeitos e promover um ambiente com reuniões regulares para discutir progresso, desafios e soluções.

REFERÊNCIAS

- COELHO, S. Documentação De Software: Uma Necessidade. Universidade Federal de Minas Gerais.

- https://www.devmedia.com.br/ciclos-de-vida-do-software/21099

- https://logap.com.br/blog/o-que-e-refatoracao-software/

- DIAS, H. et al. Desenvolvimento Colaborativo e Integrado de Sistemas: Caso de Sucesso no CPD-UFRGS. Universidade Federal do Rio Grande do Sul.

- https://blog.vericode.com.br/testes-de-software/

- https://zup.com.br/blog/tipos-de-teste

- https://rederequisitos.com.br/qual-diferenca-entre-inspecao-e-teste-de-software-voce-conhece-o-processo-de-inspecao/

- https://tripleten.com.br/blog/diferentes-tipos-de-testes-de-software/#:~:text=Enquanto%20os%20testes%20de%20unidade,combinados%20funcionem%20harmoniosamente%2C%20evitando%20conflitos.

- https://www.atlassian.com/br/continuous-delivery/software-testing/types-of-software-testing

- https://www.agtic.ufpr.br/pds-ufpr/ProcessoDemoisellePlugin/guidances/supportingmaterials/tecnicasTestes_8AB32ED1.html#:~:text=O%20teste%20de%20regress%C3%A3o%20%C3%A9,foram%20aplicados%20nas%20vers%C3%B5es%20anteriores.

- https://www.batebyte.pr.gov.br/Pagina/atividade-de-teste-durante-o-ciclo-de-vida-do-software