UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

Disciplina: Projeto e Engenharia de Software
Professor: Eduardo de Lucena Falcão
Tópico: Introdução à ES e Processos de Desenvolvimento de Software
Aluno: Thiago Theiry de Oliveira

1. **Diferencie requisitos funcionais de requisitos não-funcionais.**

   R= Requisitos funcionais tem a funcionalidade de definir o que um sistema deve fazer, fornecer, como deve reagir a determinadas entradas e como se comportar em determinadas situações. Enquanto os requisitos não funcionais irão definir as restrições em que o sistema deve operar, seja aos serviços ou funções oferecidas pelo sistema, incluindo também restrição de tempo, no processo de desenvolvimento ou restrição imposta por alguma norma.

2. **Explique porque testes podem ser considerados tanto uma atividade de verificação como de validação de software. Qual tipo de teste é mais adequado se o objetivo for verificação? Qual tipo de teste é mais adequado se o objetivo for validar um sistema de software?**

   R= As atividades de verificação e validação (V&V) servem para assegurar que o software funcione de acordo com o que foi especificado e os testes são o meio usado para ter esse seguro, com isso podemos afirmar que os testes são a principal técnica de V&V fazendo parte do seu processo de verificação e validação.

   Para o objetivo de verificação, usamos os testes de verificação que se tem o objetivo de verificar se o software atende aos requisitos funcionais e não funcionais especificados, a verificação inclui da realização de testes para encontrar erros e tem como pergunta principal “Estamos construindo o produto corretamente?”

   Para o objetivo de validação, usamos os testes de validação que se tem o objetivo de procurar assegurar que o sistema atenda as expectativas e necessidades do cliente, a inexistência de erros não mostra a adequação operacional do sistema, logo isso deve ser feita a validação com o cliente. E tem como pergunta principal “Estamos construindo o produto correto?”

3. **Por que testes não conseguem provar a ausência de bugs?**

   R= Não conseguem afirmar a ausência de bugs, porque os testes não contemplam todos os possíveis casos, geralmente pode haver condições que não foram previstas, logo não testadas, o bug pode esta presente quando tal condição não prevista acontecer, Um dos exemplos mais famosos é a explosão do foguete francês Ariane 5, lançado em 1996, de Kourou, na Guiana Francesa, mencionado na aula. Além também, a codificação está propensa a deixar falhas ou bugs ocultos, como no exemplo do calculo da área do circulo mostrado em aula que era um bug oculto, pois pode ser que o código defeituoso nunca seja executado.

4. **Suponha um programa que tenha uma única entrada: um inteiro de 64 bits. Em um teste exaustivo, temos que testar esse programa com todos os possíveis inteiros (logo, $2^{64}$). Se cada teste levar 1 nanossegundo, quanto tempo levará esse teste exaustivo?**

   R= Levará um tempo de $2^{64} \times 1$ nanossegundo = $2^{64} \times 10^{-9}$ segundos = $1{,}8446744073^{10}$ segundos

5. **Refactoring é uma transformação de código que preserva o comportamento. Qual o significado da expressão preservar comportamento? Na prática, qual restrição ela impõe a uma operação de refactoring?**

   R= O significado da expressão é que será preservado o comportamento ou a semântica do sistema, mesmo que realize novos incrementos no código, sem comprometer seu funcionamento. Logo a restrição que ela impõe ao refactoring é que se deve entregar o sistema funcionando exatamente como antes das transformações, ou seja, deixar a funcionalidade inalterada.

6. **Dê exemplos de sistemas A (Acute, ou críticos) e B (Business, ou comerciais) com os quais já tenha interagido. Dê exemplos de sistemas C (casuais) que você já tenha desenvolvido.**

   R= sistemas A são sistemas para controlar um carro autônomo, uma usina nuclear, um avião, os equipamentos de uma UTI, um trem de metrô, etc. Sistemas B são sistemas que incluem as mais variadas aplicações corporativas (financeiras, recursos humanos, logística, vendas, contabilidade, etc.) e sistema Web dos mais variados tipos desde pequenas aplicações até as grandes aplicações, na qual tive interações, porém com os outros exemplos citados do sistema B e sistema A não tive interação, foram apenas exemplos pedidos. Sistemas C são aqueles que não sofrem pressão para terem níveis altos de qualidade em geral, são desenvolvidos por 1-2 programadores, ou seja, são sistemas pequenos e não críticos. Os sistemas que já desenvolvi foram apenas os projetos e jogos passados pelas disciplinas da graduação.

7. **Se considerarmos o contexto histórico, por que foi natural que os primeiros processos de desenvolvimento de software tivessem características sequenciais e fossem baseados em planejamento e documentação detalhados?**

   R= sempre que se precisou da criação de um software exigiu qualidade, produtividade e uma grande organização. Em períodos antigos que não se tinham tantas informações, ter uma sequencia, planejamento e documentação detalhada era de muita importância para que houvesse sucesso no desenvolvimento do software, as atividades do processo são planejadas com antecedência e o progresso é medido em relação a esse plano. Tem como exemplo Modelo Cascata, Desenvolvimento Incremental e orientada a reuso.

8. **Explique por que o desenvolvimento incremental é o método mais eficaz para o desenvolvimento de sistemas de software de negócios. Por que esse modelo é menos adequado para a engenharia de sistemas de tempo real?**

   R= É um método mais eficaz por ter uma metodologia de desenvolver um programa, expor ao seu cliente uma versão intermediaria e continuar por meio de manutenção e atualizações de versões até que seja atingido o programa necessário ao cliente, assim o custo de acomodar as mudanças nos requisitos é reduzido já que se tem um feedback sobre o que já foi implementado. Já em sistemas de tempo real seria um modelo que poderia gerar problemas, já que esses tipos de sistemas necessitam estarem completos e serem executados com o máximo de exatidão possível para que suas tarefas sejam feitas de maneira segura e eficiente, com poucos erros e falhas. Desse modo não se encaixando com o desenvolvimento incremental, que é um modelo que vai ganhando modo e forma com o tempo.

9. **Explique por que os sistemas desenvolvidos como protótipos normalmente não devem ser usados como sistemas de produção.**

   R= porque a Prototipação é o desenvolvimento rápido de um sistema, com a finalidade apenas de demonstração em que ainda precise de melhorias, mais incrementos e correção de falhas. Dessa forma não deve ser tomado como produto final.

10. **Quais são as vantagens de proporcionar visões estáticas e dinâmicas do processo de software, assim como no Rational Unified Process?**

    R= a utilização das visões estáticas que mostra as atividades realizadas no processo e das visões dinâmicas que mostra as fases do modelo ao longo do tempo permite melhorar significativamente a qualidade do código e obter várias vantagens, como Redução dos custos de desenvolvimento, Maior controlo relativamente à qualidade do código desenvolvido em outsourcing, Maior produtividade e outros.

11. **Por que métodos como o Processo Unificado (UP) e Espiral não são considerados ágeis? E qual a diferença deles para o Modelo Waterfall?**

    R= Porque eles apenas possuem algumas das naturezas ágeis em suas técnicas e suas práticas, mas não são é uma metodologia ágil. Como por exemplo, as atividades são planejadas com antecedências, e o progresso está sempre em avaliação, já nos métodos ágeis o planejamento é gradativo.

    A diferença é que eles não são estritamente sequenciais, como em Waterfall. Este método é bastante engessado que tem as etapas bem definidas restringindo que o desenvolvimento volte para uma etapa que já foi concluída dificultando as alterações que podem ocorrer posteriormente.