1.1) Explique quais são as etapas do ciclo de vida de um software segundo rezende em seu livro engenharia de software e sistemas de informação. Faça uma breve correlação entre as etapas, destacando artefatos que podem ser gerados em cada uma.

Estudo de viabilidade, análise de sistemas, projeto, implementação, geração de testes de aceite, garantia da qualidade, descrição de procedimentos, conversão de banco de dados, instalação.

Todo o ciclo se inicia com o estudo de viabilidade, pois sem este não se pode avançar para as etapas seguintes. Tendo o estudo de viabilidade aprovado, inicia-se a análise do sistema, realizando o estudo do domínio nebuloso, um grande artefato desse processo é o modelo de caso de uso e a especificação de requisitos. Após esse processo, inicia-se o projeto do sistema, elaborando a arquitetura adequada, as ferramentas necessárias e a equipe que irá construir o projeto. Após todo o projeto planejado, pode-se iniciar a implementação do software levando em consideração as etapas de análise de sistemas e projeto. Com o fim da implementação, elabora-se os testes de aceite junto com o cliente, que avalia se a implementação está de acordo com o esperado. Com o cliente aceitando o sistema, pensa-se em meios de facilitar o uso do sistema, sendo a descrição dos procedimentos. A conversão do banco de dados deve ser levada em consideração caso exista sistema pré existente. A garantia da qualidade envolve todas as etapas, pois considera que o sistema irá satisfazer o cliente. A etapa de instalação é a última do ciclo, pois é o momento em que o sistema realmente é disponibilizado no ambiente dos usuários.

1.2) Explique os tipos de correções que podem existir em um sistema de software segundo rezende em seu livro engenharia de software e sistemas de informação.

Manutenção por legislação: quando a manutenção é decorrente de mudanças na lei, que implica em uma manutenção impositiva que precisa ser feita para estar em “compliance” com o negócio., manutenção por melhoria ou implementação, que podem ser para alterações no domínio ou por solicitações dos stakeholders, também pode ocorrer por conta de melhoria necessárias para ter uma melhor performance no sistema, como alteração de uma tecnologia ou aumento de recursos computacionais. Por fim, existe a manutenção por correções de erros, quando existe uma

discrepância entre o que o software faz, e o que era esperado pelos clientes, mostrando-se como o pior tipo de manutenção.

1.3) Conceitue erro, defeito e falha, e apresente a diferença entre os termos.

Defeito é uma anomalia introduzida na implementação de um software, erro é um resultado anômalo feito por um software devido a um defeito ou uma causa externa. Falha é a anomalia percebida pelo usuário do software.

2.1 Justificando sua resposta com base no tipo de sistema a ser desenvolvido, sugira o modelo genérico de processo de software mais adequado para ser usado como base para a gerência do desenvolvimento dos sistemas a seguir:

Um sistema para controlar o antibloqueio de frenagem de um carro.

Um modelo cascata, pois o sistema é embarcado e precisa ter os requisitos bem definidos antes de entregar o software.

Um sistema de realidade virtual para dar apoio à manutenção de software.

um modelo incremental, pois o tema é complexo e precisa de muita análise e auxílio dos stakeholders para a construção

Um sistema de contabilidade para uma universidade, que substitua um sistema já existente.

Um modelo incremental com reuso, pois pode ser entregue por módulos, além disso pode-se reutilizar componentes do sistema antigo.

Um sistema interativo de planejamento de viagens que ajude os usuários a planejar viagens com menor impacto ambiental.

Um modelo incremental, pois precisa de muita análise, precisa-se do stakeholder para dar feedbacks constantes, e também pode ser entregue por partes.

2.2 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?

Sistemas de negócio tem requisitos que mudam com muita frequência e por isso é mais indicado o incremental. Sistemas de tempo real são mais complexos e não podem para, por isso precisam ter os requisitos bem elicitados, para depois implementar o código, por isso é melhor o cascata.

2.3 Considere o modelo de processo baseado em reúso da Figura 2.3. Explique por que, nesse processo, é essencial ter duas atividades distintas de engenharia de requisitos.

Porque os requisitos precisam se adaptar aos modulos que vai ser reutilizado.

2.4 Sugira por que é importante, no processo de engenharia de requisitos, fazer uma distinção entre desenvolvimento dos requisitos do usuário e desenvolvimento de requisitos de sistema.

Porque o requisito de usuário é muito incipiente, sendo necessário um refinamento e melhor entendimento, para transformá-lo em requisito de sistema.

2.5. O software pode ser do tipo tempo-real, embarcado, ou ser um software onde os requisitos são bem conhecidos.

3.1 Explique por que, para as empresas, a entrega rápida e implantação de novos sistemas frequentemente é mais importante do que a funcionalidade detalhada desses sistemas.

Porque a entrega de valor é mais rápida e o cliente pode dar o feedback mais rápido. O cliente fica mais confiante com a entrega do software pois observa as funcionalidades já em uso mais rapidamente.

3.2 Explique como os princípios básicos dos métodos ágeis levam ao desenvolvimento e implantação de software acelerados.

Levam a implantação acelerada, pois foca-se na implementação de um software com integração contínua, refatoração e testes automatizados, o cliente está sempre próximo da equipe para tirar dúvida, a equipe está em constante comunicação, pois existem reuniões todos os dias.

3.3 Quando você não recomendaria o uso de um método ágil para o desenvolvimento de um sistema de software?

Quando a equipe não está madura o suficiente, quando o software terá um ciclo de vida muito longo, quando o escopo do sistema é muito grande, quando não existem ferramentas case adequadas.

3.4 Extreme Programming expressa os requisitos dos usuários como histórias, com cada história escrita em um cartão. Discuta as vantagens e desvantagens dessa abordagem para a descrição de requisitos.

A vantagem é a rapidez para a elicitação de requisitos, pois o usuário precisa apenas escrever o que deseja no cartão, porém uma desvantagem é que o cliente, na maioria das vezes, não expressa o que realmente precisa e deve existir no sistema, sendo necessário utilizar outras técnicas para elicitar os requisitos.

3.5 Explique por que o desenvolvimento *test-first* ajuda o programador a desenvolver um melhor entendimento dos requisitos do sistema. Quais são as potenciais dificuldades com o desenvolvimento *test-first*?

Ajuda, pois para criar os casos de testes, pois antes de implementar, precisa criar os casos de testes. Os casos de testes só podem ser criados caso se conheça as regras de negócio do sistema. A dificuldade é não a equipe não ter experiência com o paradigma, outro problema é a cultura organizacional, que pode achar uma perda de tempo esse tipo de paradigma.

3.6 Sugira quatro razões pelas quais a taxa de produtividade de programadores que trabalham em pares pode ser mais que a metade da taxa de produtividade de dois programadores que trabalham individualmente.

- Redução de bugs;
- Código de melhor qualidade;
- Disseminação de conhecimento;
- Aprendizado com os pares.

3.7 Compare e contraste a abordagem Scrum para o gerenciamento de projetos com abordagens convencionais dirigida a planos, como discutido no Capítulo 23. As comparações devem ser baseadas na eficácia de cada abordagem para o planejamento da alocação das pessoas nos projetos, estimativa de custos de projetos, manutenção da coesão da equipe e gerenciamento de mudanças no quadro da equipe do projeto.

No scrum é mais fácil alocar pessoas na equipe, pois a cada sprint as mudanças podem ser realizadas, a equipe tem mais coesão pois existem reuniões constantes. As equipes são auto-organizáveis e cross organizáveis. Porém, no Scrum a estimativa de custos fica prejudicada, pois não se sabe a dimensão do projeto no início, tendo-se apenas uma

ideia do que será entregue. A abordagem dirigida a planos, é mais adequada na questão dos cursos, pois existe uma especificação detalhada dos requisitos, possibilitando uma estimativa clara. Sobre a alocação de pessoas, fica mais comprometida, pois o projeto é mais engessado. Além disso, a coesão pode ser prejudicada, pois não existe um foco na comunicação e em reuniões constantes.

3.8 Você é um gerente de software em uma empresa que desenvolve softwares críticos de controles para aeronaves. Você é responsável pelo desenvolvimento de um sistema de apoio ao projeto de software que dá suporte para a tradução de requisitos de software em uma especificação formal de software (discutido no Capítulo 13). Comente sobre as vantagens e desvantagens das estratégias de desenvolvimento a seguir:

a) Coletar dos engenheiros de software e *stakeholders* externos (como a autoridade regulatória de certificação) os requisitos para um sistema desse tipo e desenvolver o sistema usando uma abordagem dirigida a planos.

b) Desenvolver um protótipo usando uma linguagem de *script*, como Ruby ou Python, avaliar esse protótipo com os engenheiros de software e outros *stakeholders* e, em seguida, revisar os requisitos do sistema. Desenvolver novamente o sistema final, usando Java.

c) Desenvolver o sistema em Java, usando uma abordagem ágil com um usuário envolvido na equipe de desenvolvimento.

a-) a vantagem é que a especificação será bem planejada e os requisitos bem elicitados antes de implementar, o que vai garantir mais segurança para sistema. a desvantagem será o tempo gasto nesta etapa.

b-) a vantagem é ter um protótipo funcional, que se aproxima muito do que será realmente entregue. a desvantagem é perder um bom tempo desenvolvimento nas linguagens python e ruby para depois jogar fora e refazer em java.

c-) A vantagem será a rapidez na entrega, entretanto, como sistema precisa ter os requisitos muito bem entendidos, pode ser que a segurança fique comprometida.

desenvolvimento.

3.9 Tem-se sugerido que um dos problemas de se ter um usuário participando de uma equipe de desenvolvimento de software é que eles 'se tornam nativos', ou seja, adotam a perspectiva da equipe de desenvolvimento e perdem de vista as necessidades de seus colegas usuários. Sugira três maneiras de evitar esse problema e discuta as vantagens e desvantagens de cada abordagem.

Manter o P.O próximo da equipe apenas em um horário específico do dia, e no máximo por uma hora. Isso faz com que o P.O se foque nas necessidade do sistema e não tenha tempo para perceber as questões

entre a equipe. A desvantagem é que ele não vai estar dentro da equipe para tirar dúvidas a todo instante.

Mudar de P.O a cada sprint, fazendo que o mesmo não tenha tempo de se tornar “nativo” da equipe. Porém isso faz com que possam existir lacunas entre uma sprint e outro, pois um P.O pode priorizar coisas, e outro P.O, outras.

Manter contato com o P.O de maneira virtual, com reuniões online, sem que o mesmo esteja no mesmo ambiente físico da equipe, isso faz com que questões do dia a dia da equipe não sejam captadas pelo P.O. Porém, isso faz com que o mesmo não esteja vendo as dificuldades da equipe.

3.10 Como XP preconiza que devem ser os contratos de desenvolvimento de software?

com contratos de escopo aberto

3.11 Quais as diferenças entre XP e Scrum?

Xp é um método ágil com foco no desenvolvimento de software, pois traz práticas focadas nesse tipo de produto. Scrum é um método ágil que não traz práticas para desenvolver software, mas para desenvolver qualquer produto.

3.12 Times Scrum são ditos cross-funcionais e auto-organizáveis. Por quê? Defina esses termos.

cross funcionais, pois não existe a divisão dentro da equipe por áreas, todos são de um mesmo time. Auto-organizáveis, pois a cada sprint os membros da equipe podem desenvolver atividades diferentes.

3.13 Em Scrum, qual a diferença entre as histórias do topo e do fundo do Backlog do Produto?

As histórias do topo são as que têm maior prioridade de entrega. as do fundo são tem menor prioridade, e vão ser entregues no fim do projeto.

3.14 O que são e para que servem story points?

Story points são formas de mensurar a complexidade e tempo de implementação das funcionalidades de um software

3.15 Em Scrum, qual a diferença entre uma sprint review e uma retrospectiva?

Sprint review é feito com o P.O como entrega da sprint
Retrospectiva é uma reunião feita apenas com a equipe no fim da sprint.

3.16 Quais são as principais diferenças entre Scrum e Kanban?

Scrum tem papéis, reuniões e artefatos bem definidos, enquanto o kanban não tem. O kanban deve ser usados por equipes bem maduras.

4.1 Cite o nome de pelo menos cinco técnicas para elicitação de requisitos.

Prototipação, etnografia, entrevistas, caso de uso e cenários.

4.2 Explique os que são requisitos funcionais e não funcionais. Apresente quatro exemplos para cada tipo.

Requisitos não funcionais São restrições que um sistema deve ter. Ex:

- O sistema deve ter uma tela para surdos, com libras
- O sistema deve ter criptográfia do tipo EAS.
- O sistema deve ser desenvolvido na linguagem JAVA
- O sistema deve ter banco de dados postgres

Requisitos funcionais é o que sistema vai fazer:

realizar saque
realizar depósito
fazer transferência.
manter histórico de transações

4.3 Qual a diferença entre requisitos de usuário e de sistemas.

Requisitos de usuário são expressos na linguagem do usuário, mostra o desejo do mesmo sobre o que o sistema vai oferecer, porém podem ser utópicos. Requisitos de sistema é um refinamento dos requisitos de usuário com informações do que realmente o sistema irá oferecer levando em consideração regras de negócio, entradas e saídas a serem feitas.

4.4 Cite seis tipos de requisitos não funcionais e apresente um exemplo para cada um.

Segurança - sistema deverá possuir acesso por biometria
Confiabilidade - sistema deverá ter redundância em um servidor na nuvem
Usabilidade - sistema deverá possuir telas com mudança de contraste para pessoas com baixa visão
Portabilidade - sistema deverá funcionar em celular, computadores e tablets
Entrega: sistema deverá ser entregue usando o método ágil Scrum
Implementação - sistema deverá ser implementado em java

4.5 Explique brevemente a engenharia de requisitos e suas fases. Qual a fase mais importante é suscetível a enganos ?

Engenharia de requisitos tem o instituto de manter um documento de requisitos bem estruturado e completo. Suas fases são:
Elicitação - fase de descoberta dos requisitos com os stakeholders
Especificação - fase de modelar e documentar os requisitos.
Validação - Retirar as inconsistências dos requisitos, e deixá-los completos.
Gerenciamento - gerenciar os requisitos de forma a mapear durante todo o processo do ciclo de vida do software.

Entende-se que esse ciclo da engenharia de requisitos é cíclica, sendo que o gerenciamento poderá retornar a outras fases, pois o entendimento dos requisitos funcionais se tornaram melhores conforme os requisitos foram amadurecendo.

A fase mais suscetível a enganos é a elicitação, pois os stakeholders as vezes não sabem explicar o que querem, usam jargões, ou ainda, podem ter conflitos entre eles.

4.6 Porque a técnica de caso de uso pode ser usada para elicitar requisitos ?

Porque é um documento com diagrama e parte textual, que mostra a visão externa do sistema, sem dar detalhes técnicos da mesma. Ajuda a entender o uso do sistema pelos usuários externos

4.7 Porque métricas são importantes para especificar requisitos não funcionais ?

Porque permite testar o requisito não funcional. Sem métricas é difícil conseguir validar se o software atinge seus requisitos não funcionais, o que acaba se mostrando como a qualidade do software.