Questão 3

Análise de Riscos no Gerenciamento de Projetos de Software

1. Identificação de Riscos:

- Técnicos: Problemas com tecnologia, arquitetura, desempenho, escalabilidade, etc.
- Gerenciais: Planejamento inadequado, falta de recursos, mudanças nos requisitos, etc.
- Operacionais: Falhas de comunicação, problemas de integração, dependências externas, etc.
- Externos: Mudanças no mercado, novas regulamentações, desastres naturais, etc.

2. Avaliação de Riscos:

- Probabilidade: Avaliar a probabilidade de ocorrência de cada risco.
- Impacto: Avaliar o impacto potencial de cada risco no projeto.
- Classificação: Classificar os riscos com base em sua probabilidade e impacto (alto, médio, baixo).

3. Desenvolvimento de Estratégias de Mitigação:

- Prevenção: Medidas para reduzir a probabilidade de ocorrência do risco.
- Minimização: Medidas para reduzir o impacto do risco caso ele ocorra.
- Contingência: Planos de ação para serem executados se o risco ocorrer.

4. Monitoramento e Controle de Riscos:

- Acompanhamento: Monitorar continuamente os riscos identificados.
- Revisão: Atualizar a lista de riscos e suas classificações conforme o projeto avança.
- Ação: Implementar ações de mitigação e contingência quando necessário.

Combinação de XP e Scrum no Desenvolvimento de Software

1. Scrum para Gerenciamento de Projetos:

- Papéis: Scrum define claramente os papéis de Product Owner, Scrum Master e equipe de

desenvolvimento, garantindo responsabilidade e foco.
- Sprints: O desenvolvimento é dividido em Sprints (iterações de tempo fixo), geralmente de 2 a 4 semanas, com entregas incrementais de software funcional.
- Reuniões: Inclui reuniões regulares como Sprint Planning, Daily Standup, Sprint Review e Sprint Retrospective, promovendo comunicação e feedback contínuos.

2. XP para Práticas de Engenharia:

- Programação em Par: Dois desenvolvedores trabalham juntos no mesmo código, melhorando a qualidade do código e facilitando a troca de conhecimento.
- Desenvolvimento Orientado a Testes (TDD): Escrever testes automatizados antes de escrever o código, garantindo que o código atenda aos requisitos e reduza bugs.
- Integração Contínua: O código é integrado e testado frequentemente, detectando erros rapidamente e garantindo que o software esteja sempre em um estado funcional.
- Refatoração: Melhorar continuamente o design do código sem alterar seu comportamento externo, mantendo a qualidade e a flexibilidade do software.

3. Integração das Duas Metodologias:

- Planejamento de Sprints com Práticas de XP: Durante o planejamento do Sprint, a equipe pode incorporar práticas de XP, como TDD e programação em par, nas tarefas definidas.
- Feedback Contínuo: As reuniões diárias do Scrum (Daily Standup) podem ser usadas para discutir o progresso das práticas de XP, como a cobertura de testes e a qualidade do código.
- Revisões de Código e Refatoração: As revisões de Sprint (Sprint Review) podem incluir a avaliação do código refatorado e a qualidade dos testes, garantindo que as práticas de XP sejam seguidas.
- Retrospectivas para Melhoria Contínua: As retrospectivas do Sprint (Sprint Retrospective) podem ser usadas para avaliar a eficácia das práticas de XP e fazer ajustes conforme necessário.