MATÉRIA - GESTÃO E QUALIDADE DE SOFTWARE – GQS
PROFESSOR - *Calvetti*
ALUNO - João Luiz da Silva – RA 82420546

Atividade 1:

Dê outros exemplos, no mínimo 5 (cinco), de aplicações dos conteúdos de base que serão estudados na UC Gestão e Qualidade de Software – GQS, explicando cada um deles.

1 Análise de falhas e causas
2 Aplicação da ISO/IEC 9126
3 Visão geral de métodos de desenvolvimento
4 Testes unitários
5 Versionamento

1. Análise de Falhas e Causas

A análise de falhas e causas é uma prática essencial para identificar os motivos pelos quais um sistema apresentou defeitos em produção ou testes. Por exemplo, quando um software bancário registra valores incorretos em uma transação, é preciso investigar se a causa está em uma falha de cálculo, erro de conversão numérica ou problema de integração entre módulos.
Essa análise permite não apenas corrigir o erro, mas também **criar mecanismos de prevenção**, como validações adicionais ou testes automatizados. Ferramentas como **FMEA (Failure Mode and Effects Analysis)** e **RCA (Root Cause Analysis)** são usadas para registrar e rastrear causas raiz, contribuindo para a melhoria contínua da qualidade do software.

2. Aplicação da ISO/IEC 9126 na Avaliação do Software

A norma **ISO/IEC 9126** (substituída posteriormente pela ISO/IEC 25010) define **características de qualidade de software**, como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.
Na prática, uma equipe de qualidade pode aplicar a norma para **avaliar um sistema web corporativo** antes de sua entrega, verificando, por exemplo:

- **Usabilidade:** se o sistema é fácil de usar e aprender.
- **Confiabilidade:** se o sistema mantém o desempenho sob carga.
- **Eficiência:** se o tempo de resposta é aceitável.
  Com base nessa avaliação, o time gera **relatórios de qualidade**, propondo melhorias nos módulos que não atingem os critérios definidos, garantindo que o produto final atenda aos padrões internacionais de qualidade.

3. Visão Geral de Métodos de Desenvolvimento de Software

Os métodos de desenvolvimento (como **Cascata, RUP, Scrum, Kanban e XP**) definem a forma como um software será planejado, construído, testado e entregue.

Por exemplo, em uma empresa que utiliza **Scrum**, as práticas de gestão de qualidade são aplicadas de forma iterativa: cada sprint inclui etapas de planejamento, desenvolvimento, testes e revisão de qualidade. Já em um projeto mais tradicional, com **modelo em cascata**, a fase de garantia de qualidade ocorre após o desenvolvimento completo.
Compreender esses métodos permite ao gestor de qualidade **adaptar as práticas de auditoria, controle de testes e métricas de qualidade** conforme o tipo de processo adotado pela equipe.

4. Testes Unitários

Os testes unitários são utilizados para verificar se **pequenas partes do código (funções, classes ou módulos)** funcionam conforme o esperado.
Por exemplo, um desenvolvedor pode criar testes unitários para uma função de cálculo de juros em um sistema financeiro, garantindo que ela retorne valores corretos em diferentes condições.
Esses testes ajudam a:

- Detectar erros logo no início do desenvolvimento;
- Garantir que alterações futuras não causem regressões;
- Aumentar a confiabilidade e a manutenibilidade do código.
  Ferramentas como **JUnit (Java)**, **PyTest (Python)** e **xUnit (.NET)** são amplamente utilizadas. Em projetos que seguem metodologias ágeis, os testes unitários são parte fundamental da integração contínua e da entrega contínua (**CI/CD**).

5. Versionamento de Software

O versionamento de software é o processo de **controlar e registrar as alterações realizadas no código-fonte ao longo do tempo**. Ferramentas como **Git** e plataformas como **GitHub**, **GitLab** e **Bitbucket** são usadas para manter o histórico de versões, permitir o trabalho colaborativo entre desenvolvedores e garantir que o código possa ser revertido caso uma falha seja introduzida.
Por exemplo, ao liberar uma nova versão (v2.1.0) de um sistema ERP, o controle de versionamento permite comparar mudanças, revisar commits e aplicar correções de forma segura.
Além disso, o versionamento contribui para a **gestão de configuração**, um dos pilares da qualidade de processo conforme modelos como **CMMI e MPS.BR**, garantindo rastreabilidade e controle sobre cada modificação feita no produto.

Conclusão:
Os conteúdos estudados na UC Gestão e Qualidade de Software (GQS) — desde a análise de falhas até o versionamento — são aplicados em todas as etapas do ciclo de vida do software. Eles garantem que o produto seja desenvolvido com qualidade, confiabilidade e segurança, reduzindo riscos e custos de manutenção a longo prazo.