Testes de software

Testar significa executar um programa ou sistema com a intenção de encontrar defeitos (Myer, 1979)
Demonstrar que o software atende aos requisitos;
Descobrir situações em que o software se comporta de maneira incorreta;

Modelo V & V (Validação e verificação)
Verificação -> O software está sendo construído de forma correta? (PADRÕES DA EMPRESA)
Validação -> Estamos construindo o produto certo? (ATENDE OS REQUISITOS?)

Falhas: Forma que os defeitos se manifestam**.** Elas podem ser:
– uma especificação errada ou falta de requisito,
– de um requisito impossível de implementar considerando o hardware e software estabelecidos,
– o projeto pode conter defeitos ou o código pode estar errado.

Frases bonitas pra lembrar:
“Testes podem mostrar a presença, mas não a ausência de defeitos”
“Qualidade melhora a produtividade”
“O objetivo do processo de teste é minimizar os riscos causados por defeitos do processo de desenvolvimento.”
“O planejamento dos testes deve iniciar com o projeto de construção do software (parte do plano de projeto)”

Mitos dos testes:
-O testador é inimigo do desenvolvedor
-A equipe de testes pode ser montada com os desenvolvedores menos qualificados
-Quando o software estiver pronto deverá ser testado pela equipe de testes

No **Modelo Cascata** a fase de testes é a penúltima, antes da operação, aí fica o desafio né, como desenvolver software e integrar os testes ao longo de todo o ciclo de desenvolvimento? aí os cara inventaram usar o modelo em V, dividindo o processo em **Validação** (testes de sistema e aceitação, nível alto) e **Verificação** (testes unitários e de integração, nível code)

Quando testar?

Testes Unitários: explorar a menor unidade do projeto procurando provocar falhas por defeitos de lógica e de implementação em cada módulo, separadamente. Normalmente é feita pelo programador.
Testes de integração: apontar as falhas decorrentes da integração entre as unidades. Para que esta fase seja executada, os módulos já devem ter passado pelos testes unitários.
Testes de Sistema: Verificação de cada uma das funcionalidades do sistema (funcionais e não funcionais). É teste de UTILIZAÇÃO como se fosse um usuário, nos mesmos ambientes, mesmo hardware.
Testes de Aceitação: Testes feitos pelo usuário, busca encontrar inconformidades dos requisitos.

Técnicas de Teste:
Caixa Branca: Avalia o comportamento interno do componente de software, código fonte, testa todas as estruturas internas
Caixa Preta: testar o software externamente.Entrada é fornecida e a saída é verificada. Não precisa conhecer o código.

Tipos de Teste:
Funcionais: o que foi especificado foi implementado? É as mesmas técnicas do caixa-preta.
Regressão e Recuperação de Versões: Sempre q tem manutenção verifica erros de atual. de versão. Fase de integr. e sist.
Segurança: proteção do software contra agentes nao autorizados. Fase de integração e sistema.
Carga: Avalia os limites do software, grandes entradas de dados, acessos etc
Usabilidade: avaliar o sistema do ponto de vista do usuário final, estética, acessibilidade, ergonomia.
Desempenho/Performance: Atende o tempo de resposta esperado? Fase de sistema.
Instalação/Desinstalação: Verifica a correta inst./desinstalação do sistema para diferentes plataformas
Configuração: se o software está apto a funcionar em diferentes versões ou configurações de ambientes
Recuperação após falha: capacidade de funcionamento do software após falhas
Interoperabilidade: Integração com outros softwares e ambientes.
Sobrevivência: Habilidade de continuar funcionando mesmo com falha em algum componente.

Equipe SQA:
Gerente de Testes: Planejamento e controle dos testes
Analista/projetista de testes: Projeto dos testes, seleção de técnicas de teste, definição de casos e procedimentos de teste
Testador: Executa testes conforme o planejado, revela falhas no produto, registra incidentes ocorridos na execução dos testes

Bom Teste: Alta probab. de revelar falhas, abrangente.

Planejar > Projetar > Executar > Analisar resultados

Métricas de software

Métricas de software possibilitam realizar uma das atividades mais importantes do gerenciamento de projetos: o planejamento.

Medida: indicação quantitativa da extensão, quantidade, dimensão, capacidade ou tamanho de algum atributo de um produto ou processo.

Medição: ato de determinação de uma medida.

Métrica: medida quantitativa do grau em que um sistema se encontra em relação a um determinado atributo.

Indicadores: métrica ou combinação de métricas que fornece uma compreensão de um processo, projeto, ou produto.

Importância das medições:

Quanto tempo vou demorar para desenvolver esse projeto? Eu vou conseguir entregar esse projeto em tempo? Qual é o custo de cada atividade do processo? Qual é a produtividade da equipe? Qual é a qualidade do código gerado? Como medir a qualidade do produto? Onde estão os nossos principais problemas? O cliente está satisfeito com nosso produto?

Métricas Diretas: Mensuráveis, custo, tempo, linhas de código, nº de defeitos

Métricas Indiretas: Qualidade, complexidade, eficiência, confiabilidade,...

Goals Question Metrics (GQM): Define um conjunto de métricas a serem utilizadas pela empresa.

GOALS (objetivos)- Com base nos interessados, estabelecem-se os principais objetivos da medição para a organização, o projeto ou uma tarefa específica.

– Reduzir defeitos

– Aumentar produtividade, etc.

QUESTIONS (questões, perguntas): A partir dos objetivos, geram-se perguntas cujas respostas dirão se os objetivos foram ou não alcançados

– Qual a taxa de defeito atual?

– Qual a taxa de defeito após a implantação do novo processo?

METRICS (métricas) :A partir das perguntas são definidas as métricas que permitem responder cada uma das perguntas: Que dados serão necessários? Quais os formatos? Como coletar (fórmula e processo)? Onde armazenar? Como utilizar? – Número defeitos por produto – Número de defeitos por status

Gerencia de Configuração

identificar, organizar e controlar modificações ao software sendo construído

Gerencia de versões: Gerenciar artefatos através do tempo.

Perda de código fonte • Programas inesperadamente não funcionam mais • Impossível saber qual foi a evolução do desenvolvimento de um programa • Impossível saber quem, porque e quando foram realizadas as alterações no sistema • Uma classe simplesmente sumiu • Bugs corrigidos aparecem inesperadamente

Baseline: Uma versão formalmente aprovada de um item de configuração, formalmente definida e fixada em um determinado momento (de tempo) durante o ciclo de vida do item de configuração.

Item de configuração: Uma entidade dentro de uma configuração que satisfaz uma função de uso final e que pode ser única em uma determinada baseline. Código fonte, Scripts do banco de dados, Documentos de análise, Diagramas

Responsabilidades da GC

Definir o ambiente de desenvolvimento
Política para controle de versões
Garantir a consistência dos artefatos
Definir procedimentos para solicitação de mudança
Administrar o ambiente e auditar mudanças
Facilitar a integração entre as partes dos sistema

Gerência de Mudanças

Garante a evolução do sistema em um processo organizado
prioridade às mudanças mais urgentes e de custo-benefício.

Baseline: Configuração do software em um instante de tempo. É tipo uma marcação de bandeira.
Build: Versão incompleta do sistema, mas estável, versão menor, não é só o code, mas docs etc. podem ter problemas
Releases: Versão de um sistema validada após testes. É uma entrega.
Versão: Política de evolução do produto. Evoluiu tanto que se tornou uma nova.
VERSÃO. RELEASE. BUILD. 3.1.1

Qualidade de Software

– **Conformidade com requisitos:** requisitos são especificados e espera-se que sejam atendidos.

– **Satisfação do cliente:** requisitos são especificados por pessoas para satisfazer outras pessoas.
– **Produto sem defeitos:** desenvolvido corretamente
Melhorando a qualidade do processo de software, é possível melhorar a qualidade dos produtos resultantes.
Processos imaturos(improvisados, sem disciplina) implicam na qualidade.
Normas e modelos de qualidade:

CMMI
-Tem 5 níveis (Inicial, gerenciado, definido, gerenciado quantitativamente, em otimização) 12345
Internacional

MPS.BR
-Tem 7 níveis: (Em Otimização, Gerenciado, Quantitativamente Definido, Largamente Definido, Parcialmente Definido, Gerenciado, Parcialmente Gerenciado) ABCDEFG
Brasileiro
Melhorar para exportação de software.

PMBOK
Framework de gerencia e desenvolvimento de projetos.

SCRUM
Método ágil para GESTÃO de projetos.

Implementando maturidade e agilidade em uma fábrica de software através de Scrum e MPS.BR nível G

-Scrum não satisfaz totalmente o nível do G do MPS.BR
-combinação de estimativas por Story Points com a técnica por complexidade Planning Poker, que consiste em um jogo de cartas numeradas com a série de Fibonacci.
-O gerenciamento de requisitos através de abordagens ágeis proporcionou uma forma alternativa e eficaz de gerenciar requisitos e solicitações de mudanças durante o desenvolvimento do projeto evitando o retrabalho.
nível G: Gerência de Requisitos Gerência de Projeto