# Verificação e Validação

Prof. Pedro Henrique Dias Valle

Adaptado de:
Prof. Ellen Francine Barbosa

# Roteiro

- Casos de Falhas em Sistemas
- Atividades de V&V
- Revisões de Software
- Teste de Software

# Casos de Falhas em Sistemas I

- Software
- Fator chave que diferencia produtos modernos
  - Sistemas de transportes, medicos, telecomunicacoes, militares, processos industriais, entretenimento, ...
- Presente no dia-a-dia das pessoas
  - Importante que esteja correto!!!

# Casos de Falhas em Sistemas

- Por que devemos nos preocupar com a **qualidade** do software?

# Trem “Fantasma”

- Em 1995 registraram-se falhas nos sensores das linhas férreas situadas em túneis nos canais ingleses
- Os trens paravam devido à suspeita de outro trem na linha
- A causa disso era a névoa de água salgada que “confundia” os sensores
- Motivo da falha:
  - Provavelmente os requisitos do sistema não levavam em conta a névoa de água salgada como um possível evento.
  - Testes e validacao apropriados poderiam detectar o problema.

# Desastre no Metrô de Nova Iorque

- Em 1995 um trem bateu atrás de outro, matando o maquinista e ferindo 54 pessoas.
- A distância entre os sinais (projetado em 1918) era menor que a distância de parada necessária para os trens atuais (maiores, mais pesados e mais rápidos). Os trens foram atualizados sem modificação no sistema de controle.
- Motivo da falha:
    - Atualização de uma parte do sistema sem validar o sistema como um todo.

# O Foguete Espacial Challenger

- A explosão a bordo do Challenger é uma das falhas mais notáveis da tecnologia moderna.
- Em 1986, após 73 segundos do seu lançamento, uma explosão envolveu o foguete matando 7 astronautas.
- As investigações concluíram que o problema estava em algumas juntas do motor que não estavam projetadas para a temperatura e pressão ocorrida.
- Motivo da falha:
  - A especificação da pressão não estava de acordo com os requisitos do sistema.
  - Os testes realizados foram inapropriados para detectar a falha.

# O Veículo Espacial Ariane 5

- Em 1996, o veículo espacial Ariane 5 saiu do curso e explodiu segundos após o seu lançamento.
- Levou uma década de desenvolvimento e custou 7 bilhões de dólares.
- Os testes insuficientes em componentes reutilizados do veículo Ariane 4 foram a causa do acidente.
- Motivo da falha:
    - Erro de software no cálculo da velocidade horizontal do foguete.

# O Veículo Espacial Ariane 5

- Motivo da falha:
  - Erro de software no cálculo da velocidade horizontal do foguete.
    - A variável que armazenava este valor tinha 64 bits (floating point) e foi erroneamente modificada para 16 bits (signed integer).
    - O valor era maior que 32.767 (maior inteiro), gerando uma falha de conversão!!!

# Casos de Falhas em Sistemas

- Por que é difícil construir software com qualidade?
- Existe algum mecanismo para garantir a qualidade do software? Como utilizá-lo eficientemente?

# Qualidade de Software

- Engenharia de Software
  - Conjunto de princípios, métodos e técnicas que tratam o software como produto de engenharia que requer planejamento, projeto, implementação e manutenção
- Objetivo Principal
  - Produzir software de alta qualidade, com um baixo custo

# Qualidade de Software

- Qualidade é a totalidade de características e critérios de um produto ou serviço que exercem suas habilidades para satisfazer às necessidades declaradas ou envolvidas.

- Visões de Qualidade de Software

# Garantia de Qualidade (SQS - Software Quality Assurance)

- Conjunto de atividades técnicas aplicadas durante todo o processo de desenvolvimento.
- Atividades de SQA
  - Aplicação de métodos técnicos.
    - Ajudar o analista a conseguir uma especificação de qualidade.
    - Ajudar o projetista a desenvolver um projeto de qualidade.

- Realização de revisões.
  - Avaliar a qualidade da especificação, do projeto, do código, ...

# Atividades de V&V

- Dentre as atividades de SQA estão as atividades de verificação e validação de software

- O objetivo é minimizar a ocorrência de erros e riscos associados
  - Detectar a presença de erros nos produtos de software

# Verificação

- Assegurar consistência, completude e corretude do produto em cada fase e entre as fases consecutivas do ciclo de vida

- Estamos construindo corretamente o produto?

- Assegurar que o produto, ou uma determinada função do mesmo, esteja sendo implementado corretamente
  - Verifica-se inclusive se os métodos e processos de desenvolvimento foram adequadamente aplicados

# Validação

- Assegurar que o produto sendo desenvolvido corresponde ao produto correto, conforme os requisitos do usuário

- Estamos construindo o produto certo?

# Atividades de V&V

- V&V abrangem um amplo conjunto de atividades de SQA:
  - Revisões técnicas formais
  - Auditoria de qualidade e configuração
  - Monitoramento de desempenho
  - Simulação
  - Estudo de viabilidade
  - Revisão da documentação
  - Revisão da base de dados
  - Testes

- V&V envolvem atividades de análise estática e da análise dinâmica

# Análise Estática

- Não requerem a execução propriamente dita do produto
- Podem ser aplicadas em qualquer produto intermediário do processo de desenvolvimento.
  - Documento de requisitos, diagramas de projeto, código-fonte, planos de teste, entre outros
- As revisões são o exemplo mais clássico de análise estática
  - Inspeção
  - Peer Review

# Análise Dinâmica

- Requerem a execução do produto
  - Código ou quaisquer outras representações executáveis do sistema

- Exemplos de atividades que constituem uma análise dinâmica do produto:
  - Teste de Software
  - Simulação

# Revisões de Software

- Meio efetivo para melhorar a qualidade de software
  - Filtro para o processo de Engenharia de Software

- Podem ser aplicadas em vários pontos durante o desenvolvimento do software

- Maneira de usar a diversidade de um grupo de pessoas para:
  - Apontar melhorias necessárias ao produto
  - Confirmar as partes de um produto em que uma melhoria não é desejada ou não é necessária
  - Realizar um trabalho técnico de qualidade mais uniforme de forma a torná-lo mais administrável

# Objetivos das Revisões de Software

- Encontrar erros durante o processo de desenvolvimento, de forma que eles não se transformem em defeitos depois da entrega do software

- Descoberta precoce dos erros
  - Melhoria da qualidade já nas primeiras fases do processo de desenvolvimento
  - Aumento da produtividade e diminuição dos custos
    - Erros são detectados quando sua correção é mais barata

# Objetivos das Revisões de Software

- É uma oportunidade de treinamento
  - Aprender por experiência
  - Participantes aprendem as razões e padrões em descobrir erros
  - Participantes aprendem bons padrões de desenvolvimento de software

- Com o decorrer do tempo
  - As revisões auxilia os participantes a desenvolver produtos mai fáceis de entender e manter

# Tipos de Revisões

- Discussão informal de um problema técnico
- Apresentação do projeto de software para uma audiência de clientes, administradores e pessoal técnico
- Revisões Técnicas Formais (RTF), as quais incluem avaliações técnicas do software realizadas em pequenos grupos

# Métodos de RTF

- Inspeção
- Walkthrough
- Peer-Review

# Inspeção de Software

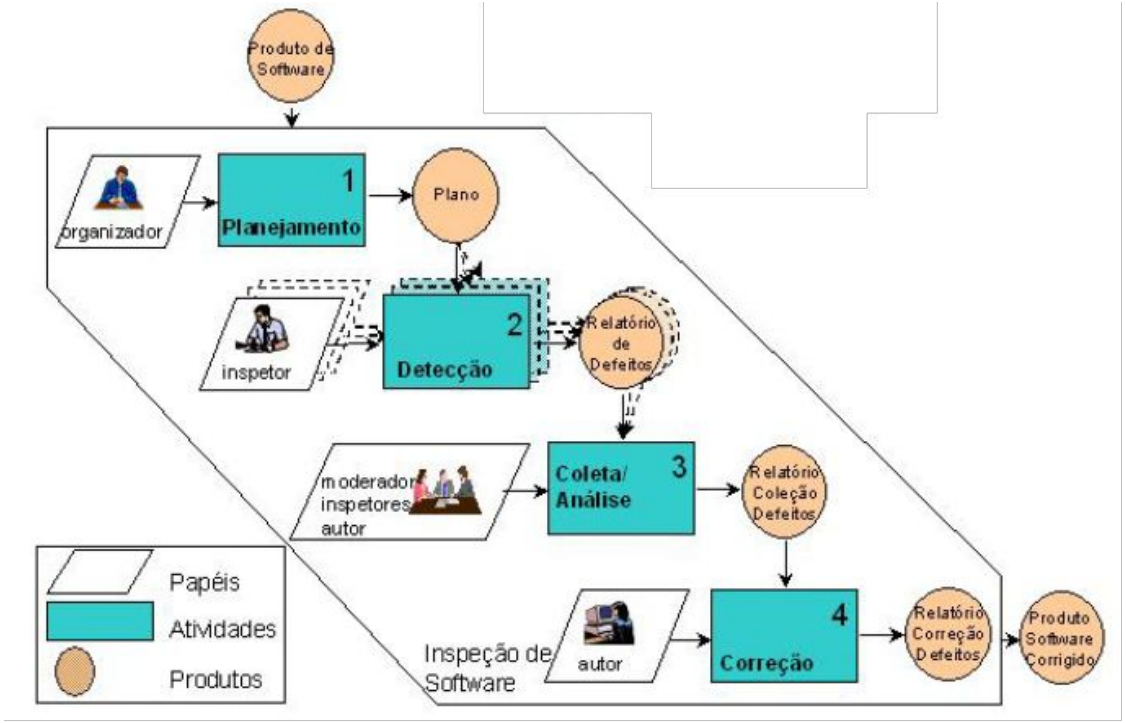
- Método de análise estática para verificar a qualidade de um produto de software

- Pode-se inspecionar tanto produtos de software como também projetos de software
  - Diferencial está na seleção dos aspectos que devem ser considerados durante a revisão

- Inspeção em Documentos de Requisitos

- Inspeção em Código-Fonte

# Como Conduzir uma Inspeção de Software



# Vantagens da Inspeção

- Detecção antecipada de defeitos (inspeção de requisitos)

- Aprende-se pela experiência
  - Participantes aprendem os padrões e o raciocínio utilizado na detecção de defeitos
  - Participantes aprendem bons padrões de desenvolvimento
- A longo prazo
  - A inspeção convence os participantes a desenvolverem produtos mais compreensíveis e mais fáceis de manter

As inspeções ajudam a integrar o processo de **prevenção** de defeitos com o processo de detecção de defeitos

# Técnicas de Leitura para Inspeção

- Como detectar defeitos?
  - Lendo o documento
  - Entendendo o que o documento descreve
  - Verificando as propriedades de qualidade requeridas

- Problema
  - Em geral não se sabe como fazer a leitura de um documento

- Razão:
  - Em geral, os desenvolvedores aprendem a escrever documento de requisitos, código, projeto, mas não aprendem a fazer uma leitura adequada dos mesmos

# Técnicas de Leitura para Inspeção

- Solução:
  - Fornecer técnicas de leitura bem definidas

- Benefícios
  - Aumenta a relação custo/benefício das inspeções.
  - Fornece modelos para escrever documentos com maior qualidade
  - Reduz a subjetividade nos resultados da inspeção

# Técnicas de Leitura para Inspeção

# Técnicas de Leitura para Inspeção

- O que é uma técnica de leitura?
  - Conjunto de instruções fornecido ao revisor dizendo como ler e o quê procurar no produto de software.

- Alguns técnicas de leitura
  - Ad-hoc
  - Checklist
  - Leitura Baseada em Perspectiva (PBR)

# Técnicas de Leitura Baseada em Perspectiva (PBR)

- Técnica de leitura proposta para detectar defeitos em especificações de requisitos

- Faz com que cada revisor se torne responsável por uma perspectiva em particular

- Possibilita que o revisor melhore sua experiência em diferentes aspectos do documento de requisitos

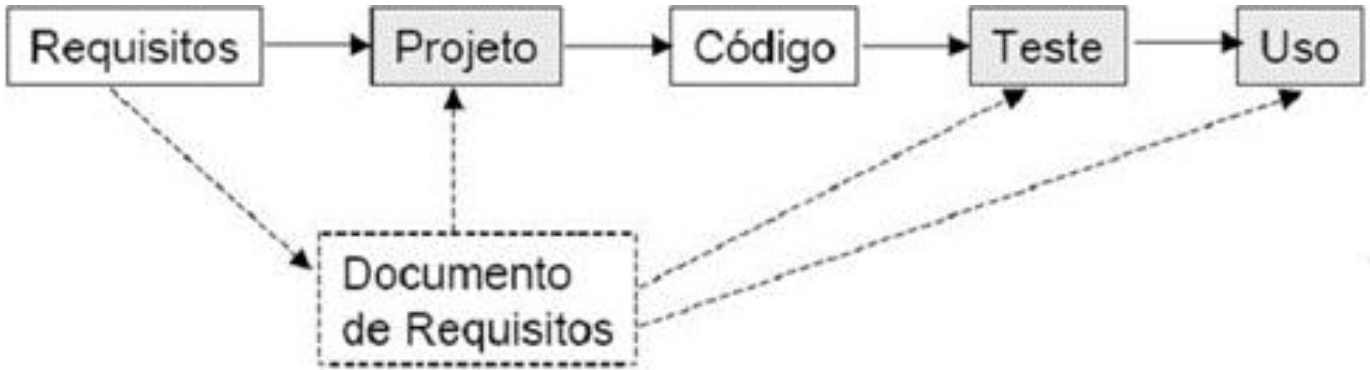
# Técnicas de Leitura Baseada em Perspectiva (PBR)

- Assegura que perspectivas importantes sejam contempladas.
- Cada revisor lê o Documento de Requisitos com olhos diferentes
- Benefícios:
  - Determina uma responsabilidade específica para cada revisor.
  - Melhora a cobertura de defeitos.

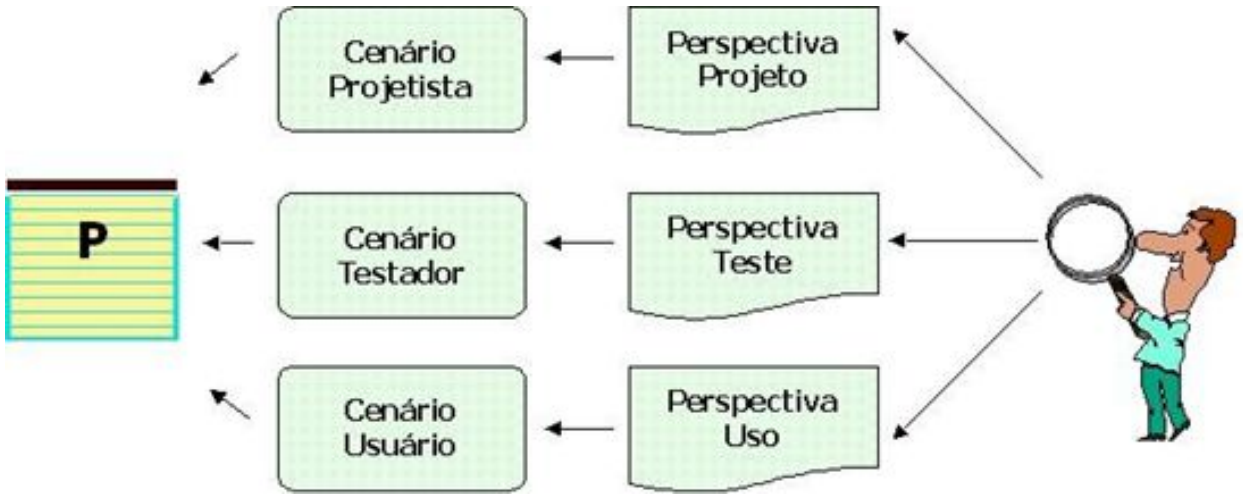
# Técnicas de Leitura Baseada em Perspectiva (PBR)

- Várias leituras podem ser feitas no Documento de Requisitos.
- O projetista que usa o DR para gerar o projeto do sistema.
- O testador que, com base no DR, deve gerar casos de teste para testar o sistema quando este estiver implementado.
- O usuário que verifica se o DR está capturando toda funcionalidade que ele deseja para o sistema.

# Técnicas de Leitura Baseada em Perspectiva (PBR)



# Técnicas de Leitura Baseada em Perspectiva (PBR)



# Walkthrough

- Procedimento similar ao procedimento para condução de uma inspeção.

- A diferença fundamental está na maneira como a sessão de revisão é conduzida.
  - Em vez de ler o programa ou checar os erros por meio de um checklist, os participantes simulam sua execução.
  - Papel adicional: testador.
    - Elaborar um pequeno conjunto de casos de teste (em papel).
    - Monitorar e controlar os resultados obtidos.

# Peer-Review

- Conduzida por pares de programadores.
  - Mesmo nível de conhecimento.

- Aplicada ao código.

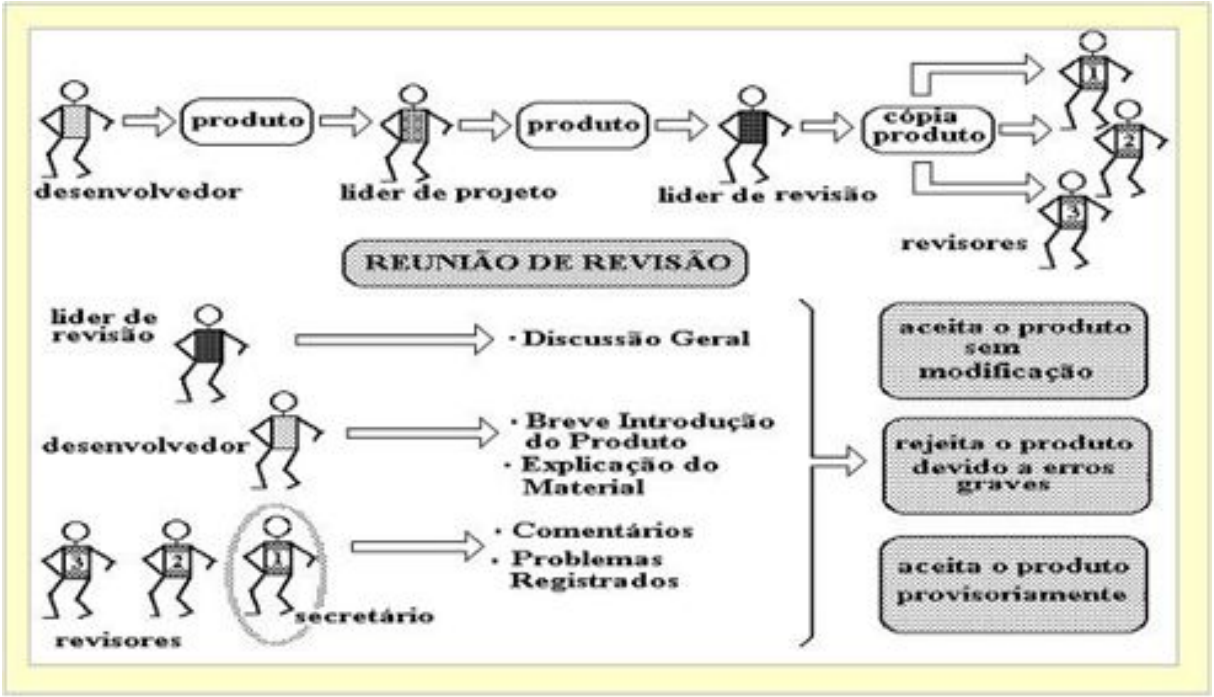
- Reuniões com duração de 1 a 2 horas.
  - Somente um programa ou parte dele (rotinas) deve ser revisado.

- Resultados são publicados em um relatório informal.
    - Não faz parte da documentação oficial do projeto.

# Reunião de Revisão Técnica

- Independentemente do formato da RTF, toda reunião de revisão deve seguir as seguintes recomendações:
  - Envolver de 3 a 5 pessoas.
  - Deve haver uma preparação para a reunião.
    - A preparação não deve exigir mais de 2 horas de trabalho de cada pessoa.
  - A reunião deve durar menos de 2 horas.
  - Deve-se focalizar uma parte específica do software.
    - Maior probabilidade de descobrir erros.

# Reunião de Revisão Técnica



# Diretrizes para Revisão

- Revise o produto, não o produtor.
- Fixe e mantenha uma agenda.
- Limite o debate e a refutação.
- Relacione as áreas problematicas
- Faça anotações por escrito
- Limite o número de participantes e insista em uma preparação antecipada
- Desenvolva uma lista de conferência (checklist) para cada produto que provavelmente será revisto
- Atribua recursos e uma programação de tempo para as revisões
- Realize um treinamento significativo para todos os revisores
- Reveja suas antigas revisões

# Teste de Software

- Análise dinâmica do produto de software
  - Processo de executar o software de modo controlado, observando seu comportamento em relação aos requisitos especificados

- Processo de executar um programa com a intenção de encontrar erros
  - O teste bem sucedido é aquele que consegue determinar casos de teste que resultem na falha do programa sendo analisado

# Verificação e Validação

Prof. Pedro Henrique Dias Valle

Adaptado de:
Prof. Ellen Francine Barbosa