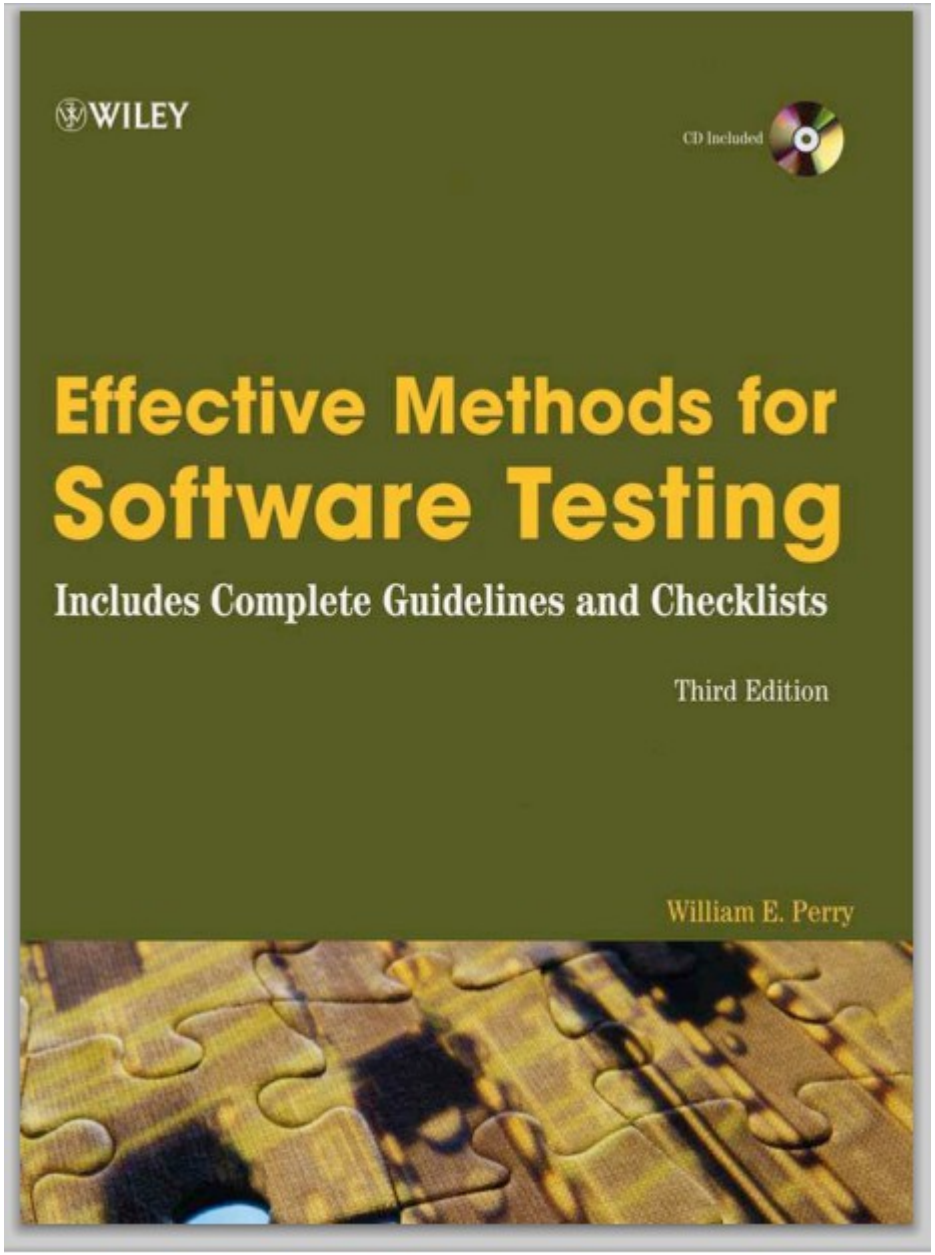


Apresentação capítulos 1, 2 e 3

Breno de Almeida Pereira

Métodos Eficazes para Teste de Software

Capítulo 1

Capacidade de análise, competência pessoal e satisfação do usuário

Teste de software é uma parte integral do processo de desenvolvimento de software, onde compreende 4 componentes:

1- Elaborar um plano: você deve elaborar um plano com objetivo definido, uma estratégia e métodos de suporte para alcançá-lo.

2 - Executar o plano: crie condições e realize treinamento necessário para executar o plano, tenha certeza de que todos estão entendendo completamente os objetivos e o plano.

3 - Verificar os resultados: verifique para determinar se o trabalho está processando de acordo com o plano e se os resultados esperados foram obtidos.

4 - Tomar as medidas necessárias: se você perceber que tem alguma coisa errada e que o trabalho saiu for a dos planos, conceba medidas para fazer o que for necessário.

Três passos principais são citados no livro para se tornar uma organização de testes de classe mundial:

1: Definir ou adotar uma classe mundial de modelo de teste de software;

2: Determinar o nível atual da capacidade de teste de software de sua organização ;

3: Desenvolver e implementar um plano para melhorar a capacidade de teste de software, tomando por base atual competências e a satisfação daqueles na classe mundial do modelo de teste de software.

Passo 1 – Definindo classe mundial do modelo de teste de software:

A missão, estratégia e o ambiente sepre tem que ser focados na satisfação das partes interessadas:

Missão: define os objetivos de teste.

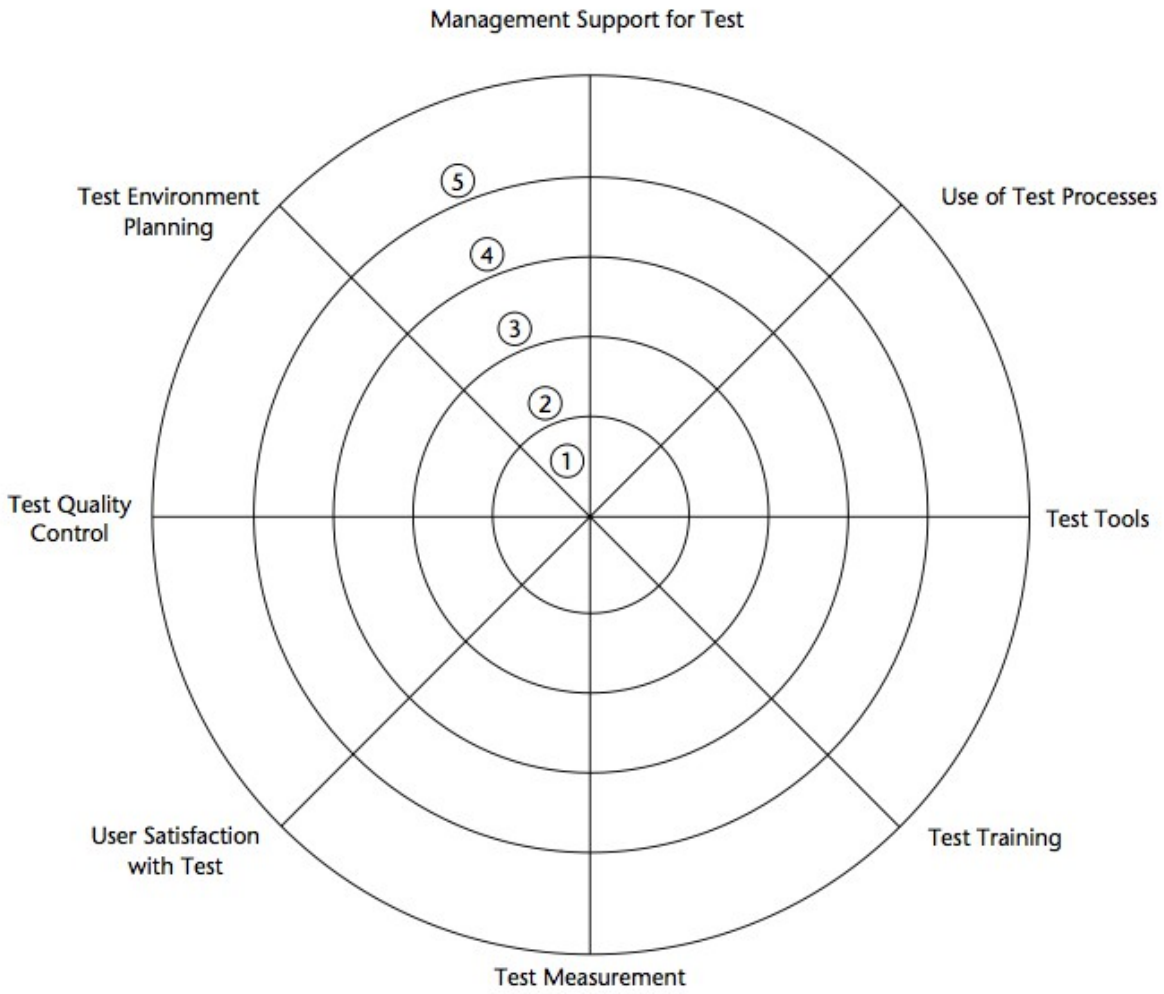
Estratégia: define como a missão será cumprida.

Ambiente: fornece a cultura, processos e ferramentas que propiciem a efetiva e efeciente teste de software.

Passo 1.1 – Customizando o modelo de classe mundial para sua organização:

Você pode customizar o modelo de classe mundial de teste de software definindo o atributo para cada componente ou você pode customizar esses atributos na missão da sua organização.

WORK PAPER 1-2 Test Environment Assessment Footprint Chart



Passo 2 – Desenvolvendo linhas de base para sua organização:

1 – Avaliar o ambiente de teste: fazendo a avalização de acordo com esses oito critérios, você vai desenvolver um Footprint Chart que mostra onde a melhoria é necessária.

2 – Avaliar as capacidades dos seus processos de teste existentes: do mesmo jeito que foi feito com o teste de ambiente, porém subsituindo os questionários de auto-avaliação.

3 – Analizando a competência de seus testes: essa prática vai permitir você avaliar suas competências de teste contra as dez categorias de habilidades em CBOK(Commom Body of Knowledge) para o certificado CSTE(Certified Software Tester).

Processos de execução:

Construir a equipe de avaliação: deve combinar pessoas que no total possuam o conhecimento de como sua organização gerencia testes de software.

Avaliar os resultados: para avaliar o estado de cada critério versus o que os critérios que os critérios que devem ser no ambiente de testes de classe mundial, você deve olhar o número de respostas “sim” versus a organização de classe mundial, onde tem cinco respostas “sim”.

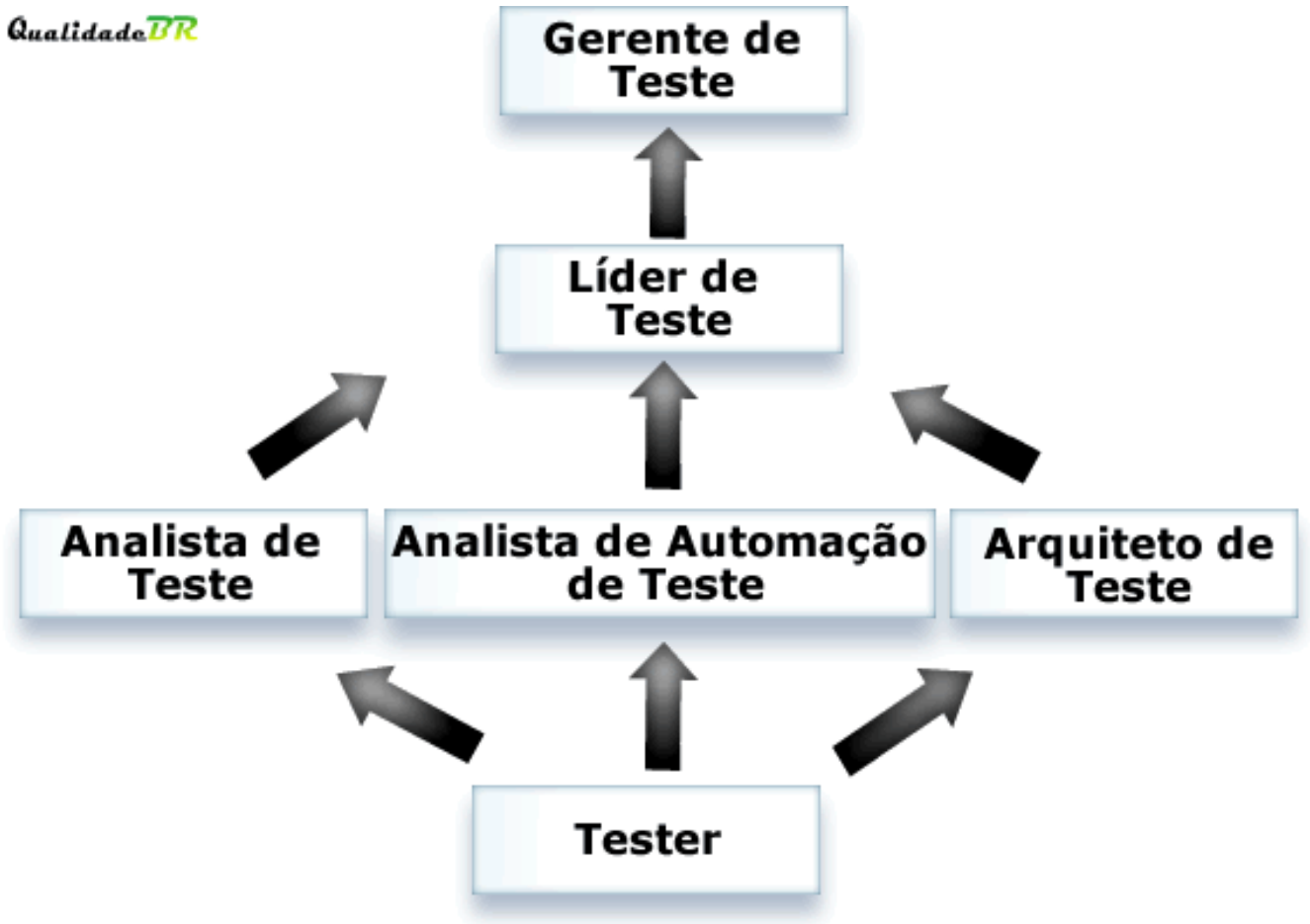
Processos de execução:

Interpretando o Footprint Chart: analisar o gráfico 1-2 que proporciona uma visão globa do seu teste de ambiente. Dada a análise, a equipe de avaliação deve tentar tirar algumas conclusões sobre o ambiente de teste.

Passo 3 – Desenvolvendo um plano de melhoria:

O Objetivo do plano de ação é mover o teste de software de onde é(linha de base) para onde deve ser(objetivo). Não há uma maneira de desenvolver esse plano. Algumas organizações querem implementar o plano por isso é um “com base no que você vai pagar”, outras organizações estão dispostas a investir no desenvolvimento de um processo de teste significativamente melhor sabendo que o retorno virá após o processo ser desenvolvido e implantado.

Hierarquia – área de testes:



Capitulo 2

Criando um ambiente favorável de teste de software

Métodos Eficazes para Teste de Software

Objetivo principal do teste de software é minimizar o risco operacional através da identificação de defeitos de software que está sendo colocado em operação.

Riscos associados a especificações de implementação:

Cronograma e orçamento insuficiente;

Processos inadequados de teste;

Competência inadequada.

Design de software defeituoso:

Projeto de software com incompletas ou critérios errados de tomada de decisão;

Omissão necessária para editar verificações para determinar integridade dos dados de saída.

Problemas de dados:

Dados imcompletos;

Dados incorretos;

Escrevendo uma política de teste de software:

A política de teste é importante, pois é a base para definir o que os “testers” de software vão incluir nos processos de teste, isso explica para partes externas, tal como a gestão organizacional, clientes e usuários, bem como o pessoal do projeto, o papael e as responsabilidades de teste de software.

Bons testes não acontecem sozinhos, eles devem ser planejados e uma política de teste deve ser a parte principal desse plano.

Métodos para estabelecer a política de teste:

Diretivas de administração: um ou mais gerente sênior de TI escreve a política determinando o que eles querem para os testes.

Informação política de consenso de serviços: gerente de TI convoca um grupo de sêniors para juntos desenvolverem a política.

Reunião de usuário: membros chave do departamento de usuário se encontram com o departamento de TI para juntamente desenvolverem a política de teste._

Economia de teste:

“Muito poucos testes é um crime, mas muitos testes é um pecado”

Poucas organizações tem estabelecido a base para medir a eficácia dos testes. Isso torna difícil para os sistemas individuais para determinar o custo-eficácia dos testes. Sem padrões dos testes fica difícil avaliar em detalhes suficientes a eficácia para permitir que o processo seja medido e melhorado.

“Testers” de software necessitam ser treinados para melhorar as suas competências em teste de software, testes de processos e usar as ferramentas de testes.

Construção de uma abordagem estruturada para teste de software:

Requisitos: as atividades de verificação realizadas durante a fase de requisitos de desenvolvimento do software são extremamente significantes.

Design(projeto): durante a fase de design, a estratégia de testes gerias é formulada e um plano de teste é produzido. Se preciso, um time independente de teste é organizado.

Programa: muitas ferramentas de testes e técnicas existem para esse estágio de desenvolvimento do sistema. Passo a passo do código e inspeção do código são eficazes técnicas mauais.

Teste: durante o proceso de teste, controle cuidadoso e gestão de informação de teste é crítica. Conjunto, resultados e relatórios de testes devem ser catalogados e armazenados.

Instalação: o processo de colocação de programas testados em produção é uma fase importante normalmente executadas dentro de um período de tempo ilimitado.

Manutenção: mais de 50% do ciclo de vida de um software financeiramente falando é gasto em manutenção. Enquanto o sistema é usado, ele é modificado para corrigir alguns erros ou para melhorar o sistema original. A atividade de reanálise é chamada de teste de regressão. O objetivo do teste de regreção é minimizar o custo de revalidação do sistema.

Desenvolvendo uma estratégia de teste:

Selecionar e classificar os fatores de teste: os clientes chave e usuários do sistema em conjunto com a equipe de teste deve ser selecionado e os fatores de teste classificados.

Identificação das fases de desenvolvimento de sistemas: a equipe de desenvolvimento de projeto deve identificar as fases dos seus processos de desenvolvimento.

Identificar os riscos de negócio associados com o sitema em desenvolvimento: os desenvolvedores, usuários-chave, clientes e o pessoal de teste deve estender os riscos associados com o sistema de software.

Colocando os riscos na matriz:

WORK PAPER 2-2 Test Factors/Test Phase/Test Concerns

TEST PHASE / TEST FACTORS (RANKED HIGH TO LOW)	REQUIREMENTS	DESIGN	PROGRAM	TEST	INSTALLATION	MAINTAINENANCE
Factors or Risks			Test Concerns			

Permite que você faça os fatores mais importantes das especificações de testes. Esse Work Paper deve ser completado e no mínimo o fator importante na parte inferior da coluna de fatores de teste. Não listar todos os fatores de teste inconsequentes. Em seguida, listar as preocupações correspondentes na coluna apropriada de teste de fase

Capítulo 3

Construindo o processo de teste de software

Métodos Eficazes para Teste de Software

Orientações para teste de software:

A expericência tem mostrado, as seis diretrizes gerais de teste de software que, se seguidas, podem melhorar significativamente o teste de software:

1 - Teste de software deve reduzir o risco do desenvolvimento de software: executivos seniors de TI precisam desenvolver suas estratégias de TI. “Testers” devem entender que seu papel no mundo dos negócios é avaliar os riscos e relatar os resultados para a gestão.

2 - O teste deve ser realizado de forma eficaz: significa obter o máximo benefício a partir de recursos mínimos. O processo está bem definido. Deve haver pouca variação no custo de realizar a tarefa de “tester” para “tester”.

3 - Os testes devem revelar defeitos: todos os testes são focados em descobrir e eliminar defeitos ou variâncias do que é esperado. Um defeito encontrado no sistema a ser testado pode ser classificado como errado, ausente ou extra.

4 - O teste deve ser realizado utilizando a lógica de negócios: o custo de identificar e corrigir defeitos aumenta exponencialmente à medida que o projeto progride. O teste deve começar durante a primeira fase do ciclo de vida e continuar durante todo o ciclo de vida.

5 - Os testes devem ocorrer durante todo o ciclo de vida: como você está testando a implementação, prepare uma série de testes que o seu departamaneto de TI pode executar periodicamente após a revisão do sistema. Testes não param uma vez que você o sistema esteja completamente implementando,
deve continuar até que você substitua ou atualize novamente.

6 - O testes devem testar a função e a estrutura: o teste funcional é às vezes chamado de teste de caixa preta, porque nenhum conhecimento de lógica interna do sistema é usado para desenvolver casos de teste. Por outro lado, teste estrutural é às vezes chamado de teste de caixa branca, pois o conhecimento da lógica interna do sistema é usado para desenvolver casos de testes hipotéticos.

Teste que é paralelo ao processo de desenvolvimento de software:

Quando os processos para o desenvolvimento de software e software de teste são mostrados em um único diagrama, eles são frequentemente apresentados como o que é conhecido como um diagrama de V.

Determinar os objetivos da estratégia de teste:

Estratégia de test normalmente desenvolvida por uma esquipe muito familiar com os riscos de negócio associados ao software; táticas são desenvolvidas pela esquipe de teste. Assim, a equipe de teste precisa adquirir e estudar a estratégia de teste.

Determinar o tipo de projeto de desenvolvimento:

O tipo de projeto de desenvolvimento refere-se ao ambiente/ metodologia na qual o software será desenvolvido. Como o ambiente muda, o mesmo acontece com o risco de testes. Os riscos associados com o esforço de desenvolvimento tradicional diferem dos riscos associados com o software comprado.

Determinar o tipo de sistema de software:

Refere-se ao tratamento que será realizado por esse sistema. Um único sistema de software pode incorporar mais do que um tipo. Identificando um específico tipo de software ajuda a construir um plano de teste eficaz.

Determinar o escopo do projeto:

Refere-se ao total de atividades que serão incorporadas dentro do sistema de software que será testado. O âmbito do desenvolvimento do novo sistema é diferente do âmbito da alteração de um sistema existente. Esse passo descreve algumas das características necessárias, mas esta lista deve ser expandido para englobar os requisitos do sistema de software específico a ser testado.

Identificar os riscos de software:

Os riscos estartégicos são os riscos comerciais de alto nível enfrentados pelo sistema de software. A finalidade de decompor os riscos estratégicos em riscos táticos é para ajudar a criar os cenários de teste que ira lidar com esses riscos.

Determinar quando os testes devem ocorrer:

As etapas anteriores identificaram o tipo de projeto de desenvolvimento, o tipo de sistema de software, o escopo do projeto e riscos técnicos. Usando essas informações, o ponto no desenvolvimento de processos quando o teste pode ocorrer deve ser determinado. E essa etapa vai dizer quando os testes devem ocorrer.

Definindo o sistema padrão de plano de teste:

Um plano de teste tático deve ser desenvolvido para descrever quando e como o teste irá ocorrer. Este plano de teste fornecerá informação de fundo sobre o software a ser testado, sobre os objetivos de teste e riscos, bem como sobre as funções comerciais a serem testadas e os testes específicos a serem realizados.

Definindo a unidade padrão de plano de teste:

Durante a concepção interna, o sistema é dividido em componentes ou unidades que executam o processamento detalhado. Cada uma dessas unidades deve ter o seu próprio plano de teste. Os planos podem ser simples ou complexos como a organização requer, baseado nas expectativas de qualidade.

Convertendo estratégias de testes para estratégias táticas:

Desenvolver táticas não é um componente para estabelecer um ambiente de teste. Contudo, entender as táticas que serão usadas para implementar a estratégia é importante na criação de processos de trabalho, selecionar ferramentas e certificar que sejam escolhidas as pessoas certas e treinadas.

Conceito de “workbench”:

Para entender a metodologia de teste, você deve entender o “workbench” que nada mais é uma maneira de ilustrar e documentar como atividade específica é para ser realizada. Pode ser usado para ilustrar um dos passos envolvidos na construção de sistemas.

Conclusão:

Testes eficientes e eficazes ocorrerão somente quando um processo bem definido existir. Seis diretrizes foram apresentadas para melhorar a eficácia e eficiência do processo de teste de software. Foi explicado o conceito de “workbench” para ser usado na construção do processo de teste de software.

Testes antigamente: