pvh( );

INSTITUTO FEDERAL
GOIÁS

Instituto Federal de Educação, Ciência e Tecnologia
Campus Inhumas

TESTES DE SOFTWARE

Introdução, histórico e questões básicas

Prof. Me. Victor Hugo Lázaro Lopes

AGENDA

- **Introdução, histórico e questões básicas;**
  - 1.1 O mercado de trabalho e certificações;
  - 1.2 Qualidade de software em perspectiva;
  - 1.3 Software Quality Assurance;
  - 1.4 Capability Maturity Model - CMM;
  - 1.5 A nova qualidade de software;
  - 1.6 Evolução da qualidade de software: mundial e nacional;
  - 1.7 Tendências da qualidade de software.

Introdução

Imaginemos um distinto senhor de meia idade que raramente, ou nunca, foi ao médico e, de repente, a sua esposa pede (manda!) que ele vá fazer um check-up completo, mesmo sabendo que aparentemente ele não apresenta nenhuma doença.

Depois de muita insistência ele vai, após os exames o médico constata:

- Pressão arterial constantemente alta;
- Fuma;
- Consome bebidas alcoólicas diariamente;
- Colesterol alto;
- Baixa capacidade pulmonar;
- Sono ruim e irregular;
- Paciente sedentário;
- Alto índice de estresse no trabalho;
- Alta carga de trabalho diário;
- Insatisfação com o salário.

Introdução

Sob a óptica dos sistemas e softwares, o fato deste senhor ter vivido por mais de 50 anos não deveria ser informação suficiente para atestar sua “qualidade” e “efetividade”??

Percebemos que ele está à beira de um ataque de nervos ou problema mais complexo.

Mas o que isso tem a ver com um software??

Um software precisa ser testado desde o seu “nascimento” e durante toda a sua vida, pois os “bugs” podem estar lá, Mesmo que você ache que os eliminou.

O software pode ter uma “doença” grave e ninguém saber!!!!

Introdução

Apenas 10% das empresas entrevistadas realizam algum teste de software no Brasil (Ministério da Ciência e Tecnologia):

- São realizados de forma correta e adequada?
- Quais as motivações destes testes?
- Cultura de qualidade de software reativa, e não pró-ativa.
- Maior parte do tempo gasto desenvolvendo, e o tempo que sobra é usado pra testar: cultura!
- O “teste” é o resto.
- E a culpa recai sobre quem testou: “não testou direito”.

Introdução

Erros comuns:

- Erro de dimensionamento: prazos inadequados e recursos escassos;
- Erros de escopo: o que deveria ser testado de forma prioritária??
- Erros de profundidade: quantos casos de testes foram realizados??

Introdução

Teste é qualidade de software?

- Teste é uma grande arma no que tange a ajudar a melhoria da qualidade de software e o produto final.
- Porém: não é um “salvador da pátria”, pois não resolve tudo sozinho.
- Teste deve ser visto como um impulsionador da melhoria da qualidade do software.
- Teste: um tópico importante dentro da área de qualidade de software.

Introdução

Portanto, o que é um teste de software, afinal?

- São testes controlados realizados em um software com a intenção de descobrir erros e defeitos (MYRES, 2004);
- Teste a que um software é submetido para mostrar a presença de defeitos, mas nunca para mostrar a ausência deles (DIJKSTRA, 1972);
- Se presta a verificar se o software está fazendo o que deveria, de acordo com os seus requisitos (RIOS, 2006);
- Qualquer atividade que a partir da avaliação de um atributo ou capacidade de um programa ou sistema seja capaz de determinar se ele alcança os resultados desejados (GELPERIN e HETZEL, 1988);

Introdução

Portanto, em essência:

- Teste de software é o processo que visa a sua execução de forma controlada, com o objetivo de avaliar o seu comportamento baseado no que foi especificado.

- A execução dos testes é considerada um tipo de ***validação***.

- Na prática, não se pode testar um programa por completo e garantir que ele ficará livre de bugs: é quase impossível testar todas as formas e alternativas de entrada de dados, bem como testar as diversas possibilidades e condições criadas pela cabeça do programador.

- Software 100% correto???

Introdução

Por que testar, se já consigo avaliar sozinho meus códigos e entendi corretamente todos os requisitos.

- Para assegurar que as necessidades dos usuários estejam sendo atendidas.
- Porque é provável que o software possua defeitos: feito por humanos!
- Desenvolvedor já alocado para outro projeto teria que resolver muitos bugs de projetos anteriores em produção.
- Porque falhas podem custar muito caro.
- Para avaliar a qualidade do software.

Introdução

Erro != Defeito != Falha

- Erro: é uma ação humana que produz um resultado incorreto;
- Defeito: A manifestação de um erro no software (bug);
- Falha: quando o sistema se comporta de forma inesperada devido ao defeito.
- Um erro não necessariamente leva a um defeito, que por sua vez não necessariamente leva a uma falha.
- Nível de severidade e emergência pode (deve) ser mensurado.

Introdução

IEEE e a padronização:

- **IEEE 610.12-1990 diferencia os termos:**
  - **Defeito (fault): passo, processo ou definição de dados incorreto;**
  - **Engano (mistake): ação humana que produz um resultado incorreto;**
  - **Erro (error): diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa;**
  - **Falha (failure): produção de uma saída incorreta com relação à especificação.**

Segundo MALDONADO (2004), defeito, engano e erro são causas, e falha é consequência.

Introdução

Tipos de erros:

- **Erros computacionais:** o erro provoca uma computação incorreta, mas o caminho executado (sequência de comandos) é igual ao caminho esperado;

- **Erros de domínio:** o caminho efetivamente executado é diferente do caminho esperado, ou seja, um caminho incorreto é selecionado.

Diz-se que um programa ***P*** com domínio de entrada ***D*** é correto com respeito a uma especificação ***S*** se:

$$S(d)=P^*(d), \quad \forall \quad d \in D,$$

ou seja, se o comportamento do programa está de acordo com o comportamento esperado para todos os dados de entrada.

Histórico

Desastres causados por erros de software

- Em 1996 - Um software com uma exceção não tratada foi responsável pela explosão do foguete Ariane-5, quando a 39 segundos após a iniciação da sequência de vôo, o foguete se desviou de sua rota, partiu e explodiu, tendo um prejuízo de 370 milhões de dólares.

- (...)
- Não foi uma falha mecânica nem um ato de sabotagem. O desastre foi causado por um simples bug em um software, que fez cálculos errados ao se tornar sobrecarregado com números mais longos do que era capaz de suportar.
- Tecnicamente, o problema é chamado de "estouro de inteiros", o que significa que os números são muito longos para serem armazenados em um sistema computacional, às vezes provocando seu mau funcionamento.
- Esse tipo de falha ocorre com uma frequência surpreendente. Acredita-se que um dos motivos pelo qual a Nasa perdeu o contato com a sonda espacial Deep Impact, em 2013, foi o fato de seu software ter alcançado um inteiro.
- (...)

Fonte: reportagem “As falhas numéricas que podem causar desastres”, disponível em: http://www.bbc.com/portuguese/noticias/2015/05/150513_vert_fut_bug_digital_ml

Histórico

Desastres causados por erros de software

- Em 2000 - Erro de cálculo no sistema de radioterapia, que era utilizado para controlar a emissão de radiação em tratamentos de câncer matou 8 pessoas e causou queimaduras graves em outras 20.

- Não tão desastrosos assim: Em dezembro de 2014, o vídeo mais assistido na história do YouTube – Gangnam Style, do coreano Psy – "quebrou" o contador de visitas do site: aparentemente, o contador teria sido programado para ir, no máximo, até 2.147.483.647. O YouTube se beneficiou da propaganda grátis gerada pela notícia e atualizou seu contador para suportar mais de 9 quintilhões visitantes.

Histórico

Nas décadas de 1960 e 1970:

- 80% dos esforços despendidos nas atividades de codificação e nos testes unitários.
- **Uma parcela menor era dedicada à integração dos programas e nos testes dos sistemas.**
- **Teste era um mal necessário: provar aos usuários que os produtos funcionavam.**
- **Não eram tratados como um processo formal alinhado às atividades do processo de desenvolvimento do sistema (nem tratado como um processo era).**

Histórico

A partir dos anos 1980:

- Melhorias na engenharia de software:
  - maior importância à análise dos requisitos, ao desenho funcional e técnico dos novos sistemas;
  - **maior esforço empregado na integração das diversas peças que compunham os softwares e ao teste destes para funcionarem como um sistema;**

- Atividades de testes passaram a ser tratadas como um processo formal: surgem as metodologias de testes, em constante evolução.
- A crescente demanda atual por sistemas interoperáveis e em plataforma web (isso ainda importa???), cresce a complexidade de escalabilidade, segurança e performance: especialização do processo de testes.

Histórico

PENSANDO UM POUCO:

Quais impactos para uma empresa tendo o seu site fora do ar por defeitos?

Histórico

PENSANDO UM POUCO:

Quais impactos para uma empresa tendo o seu site fora do ar por defeitos?

Facebook caiu por 25 minutos em agosto de 2014:

De acordo com dados da consultoria VentureBeat, cada minuto fora do ar custa para o Facebook cerca de US$ 20 mil (R$ 80 mil, aprox.). Se levarmos em consideração tempo oficial fora do ar, chegamos ao significativo valor de US$ 500 mil. Isso levando-se em conta somente o faturamento por minuto com publicidade, sendo que os prejuízos intangíveis também são catastróficos.

Questões Básicas

Artefatos de Teste	• todo o conjunto de documentação gerado pelo processo de teste de software.
Caso de Teste	• é composto por um conjunto de entradas, por passos de execução e um resultado esperado.
Roteiro de Teste	• É composto por um conjunto de casos de teste definidos para uma determinada especificação.

Questões Básicas

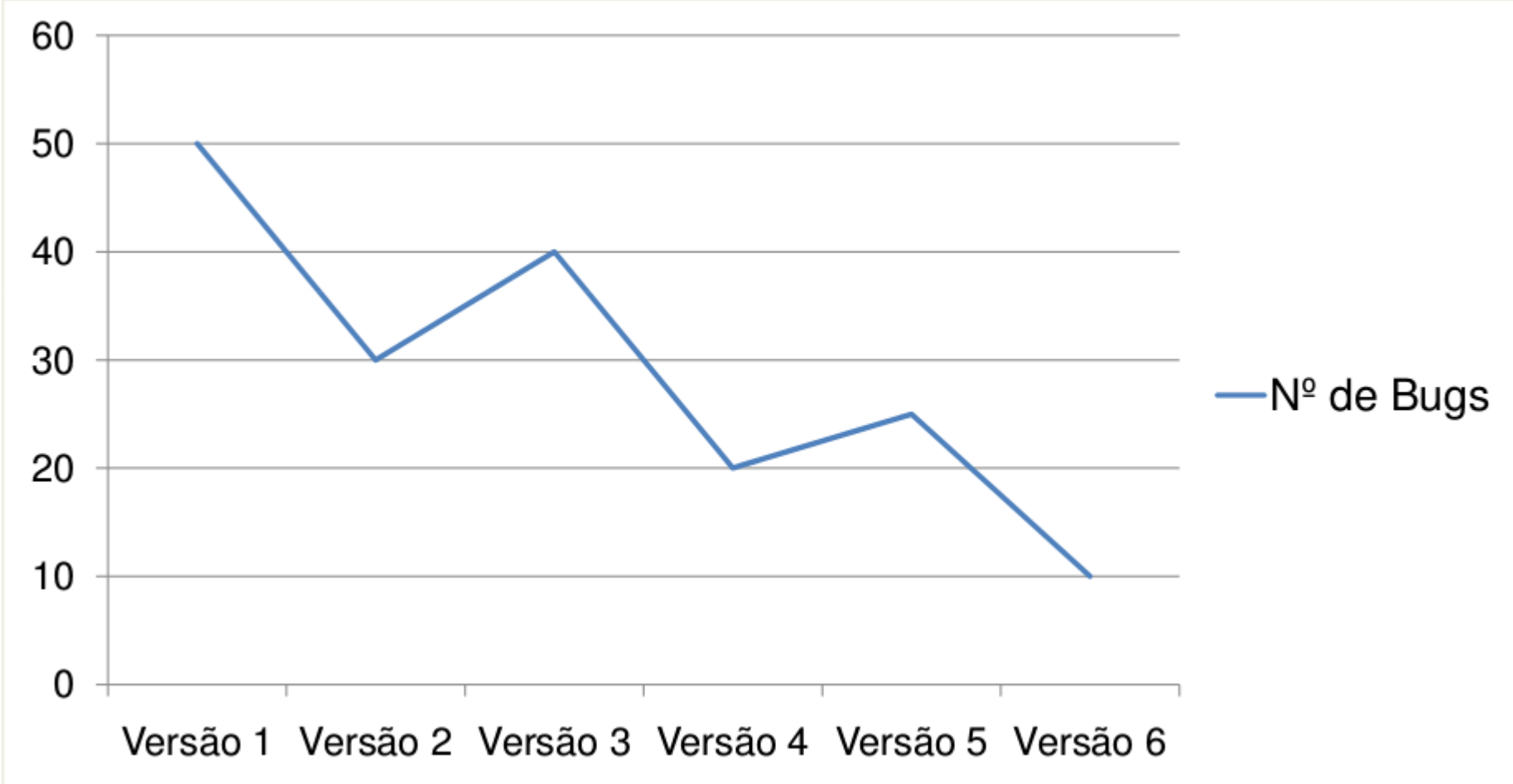
Requisitos	• regras de negócio do sistema.
Testar	• descobrir falhas através da execução do sistema.
Bug	• é um defeito encontrado no sistema em execução.

Questões Básicas

Confiabilidade do Software é a probabilidade que o software não causará uma falha no sistema por um tempo especificado, sob condições determinadas.

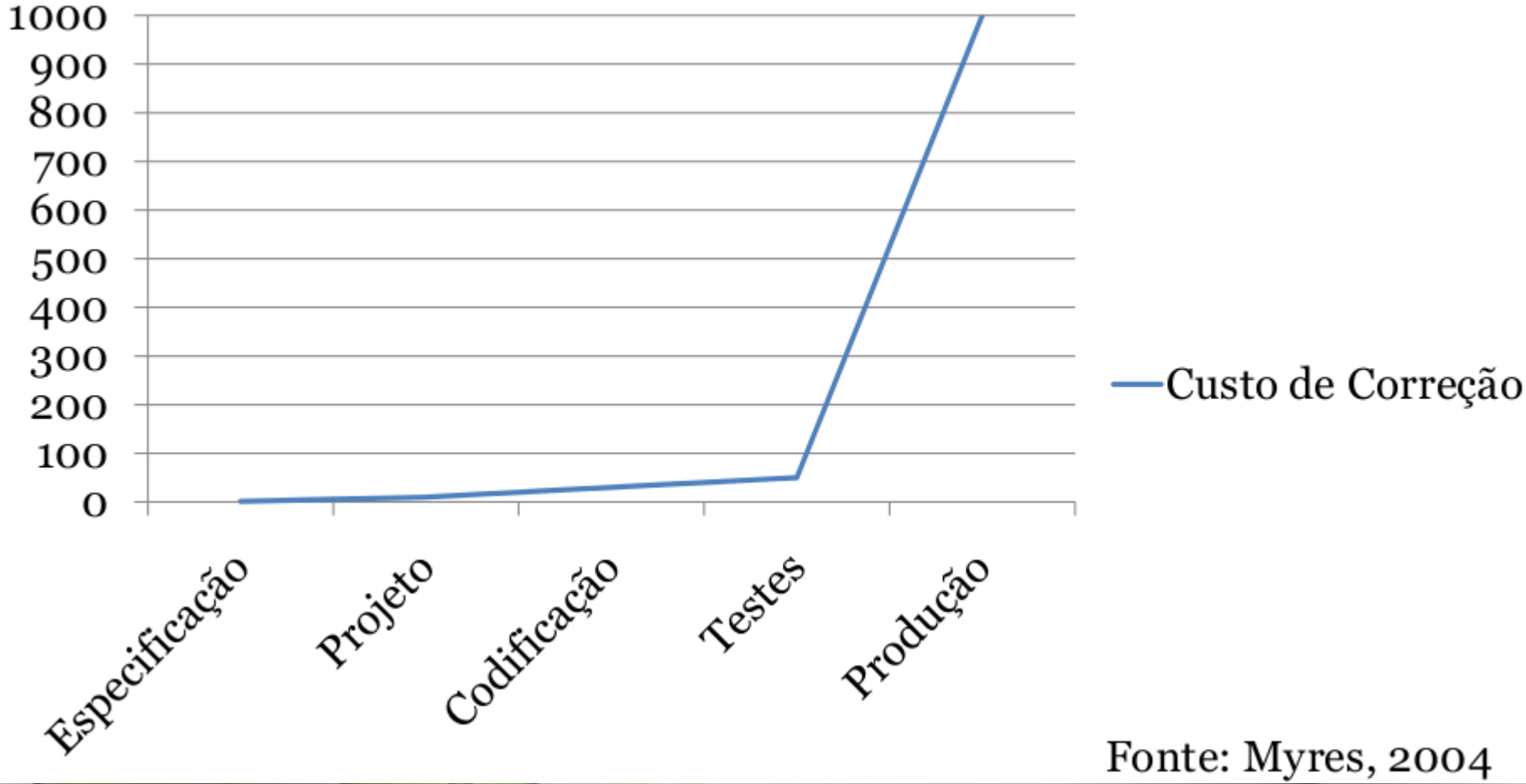
Questões Básicas

Confiabilidade de um software



Questões Básicas

O custo de correção de erros:



Fonte: Myres, 2004

Questões Básicas

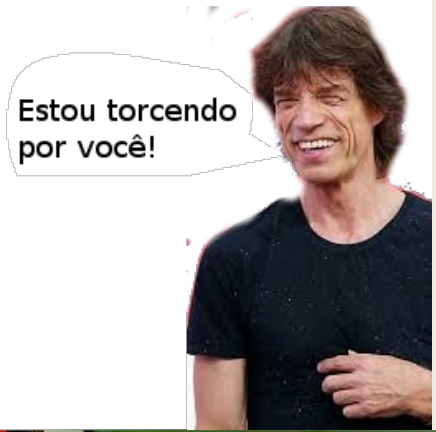
Mitos sobre testes de software

O testador é inimigo do desenvolvedor.

Os testadores devem ser os desenvolvedores menos qualificados.

O sistema está pronto quando o desenvolvedor termina de codificar.

Um programador consegue testar eficientemente o próprio código.



Glossário

A

Aceite: se um sistema satisfaz ou não os critérios de aceite e para permitir que o usuário, cliente ou outra entidade autorizada determinem se aceitam ou não o sistema.
Adaptabilidade: A capacidade que o sistema 1 tem de ser adaptado para diferentes ambientes especificados sem utilizar ações ou meios que não sejam aqueles fornecidos para esta finalidade.
Adequação: A capacidade que o sistema tem de fornecer um conjunto apropriado de funções para as tarefas e objetivos do usuário especificados.
Alterabilidade: A capacidade que o sistema tem de permitir que as alterações especificadas sejam implementadas.
Alvo do teste: Conjunto de critérios de saída.
Ambiente de teste: Um ambiente que contém hardware, instrumentação, simuladores, ferramentas de software e outros elementos de apoio necessários para conduzir os testes.
Ambiente operacional: Produtos de hardware e software instalados localmente nos usuários ou clientes onde o sistema em teste será usado. Os softwares podem incluir sistemas operacionais, sistemas de gerenciamento banco de dados e outras aplicações.

Uma verdade que se tornou mentira

“Das carreiras que compõe as entradas no mundo de TI, a de analista de teste talvez seja a que exija, em seus primeiros passos, o menor nível de conhecimento técnico, juntamente com o pessoal do Call Center.” Alberto Parada, 2013.

Até pouco tempo, o mercado aceitava com naturalidade encontrar erros em aplicações; hoje se tornou absolutamente inaceitável receber uma aplicação com defeito. O nível de exigência cresceu tanto que é muito comum encontrar em propostas comerciais (e até contratos) cláusulas de multas pesadas limitando as quantidades e tipos de erros que poderão ser encontrados nas aplicações.

CARGO	CIDADE	MÍNIMO	MÉDIO	MÁXIMO	VAGAS
Analista de Testes	Sao Paulo - SP	R$ 1.200,00	R$ 3.512,08	R$ 8.000,00	Vagas Analista de Testes
Analista de Testes	Manaus - AM	R$ 1.500,00	R$ 3.079,55	R$ 7.000,00	Vagas Analista de Testes
Analista de Testes	Rio de Janeiro - RJ	R$ 1.300,00	R$ 3.929,82	R$ 7.000,00	Vagas Analista de Testes
Analista de Testes	Brasilia - DF	R$ 1.500,00	R$ 3.662,62	R$ 6.500,00	Vagas Analista de Testes
Analista de Testes	Porto Alegre - RS	R$ 1.800,00	R$ 3.550,28	R$ 6.000,00	Vagas Analista de Testes
Analista de Testes	Belo Horizonte - MG	R$ 1.200,00	R$ 2.720,86	R$ 5.000,00	Vagas Analista de Testes
Analista de Testes	Curitiba - PR	R$ 1.900,00	R$ 3.279,12	R$ 5.000,00	Vagas Analista de Testes
Analista de Testes	Contagem - MG	R$ 2.000,00	R$ 3.900,89	R$ 5.000,00	Vagas Analista de Testes
Analista de Testes	Salvador - BA	R$ 2.000,00	R$ 3.647,92	R$ 5.000,00	Vagas Analista de Testes
Analista de Testes	Goiania - GO	R$ 2.000,00	R$ 2.856,41	R$ 4.000,00	Vagas Analista de Testes

https://www.ceviu.com.br/salario/tabela-pretensao-salarial/cargo/analista-testes

CERTIFICAÇÕES

Com o crescimento do Teste de Software, as **certificações** têm se tornado um diferencial cada vez mais interessante para os profissionais interessados numa ascensão na área. Certificações de diferentes níveis podem agregar vantagens tanto para o profissional quanto para a empresa, possibilitando desde o nivelamento dos conhecimentos técnicos da equipe até a melhoria do processo de teste como um todo.

CERTIFICAÇÕES

ISTQB: CTFL e CTAL

O International Software Testing Qualifications Board (ISTQB) é um selo de qualidade internacional voltado para os profissionais de teste de software fundado em Edinburgh em Novembro de 2002. O instituto atua em mais de 47 países, sendo que no Brasil, o BSTQB (Brazilian Software Testing Qualifications Board) é o seu representante oficial.

ISTQB: gerencia os representantes certificadores em cada país.

CERTIFICAÇÕES

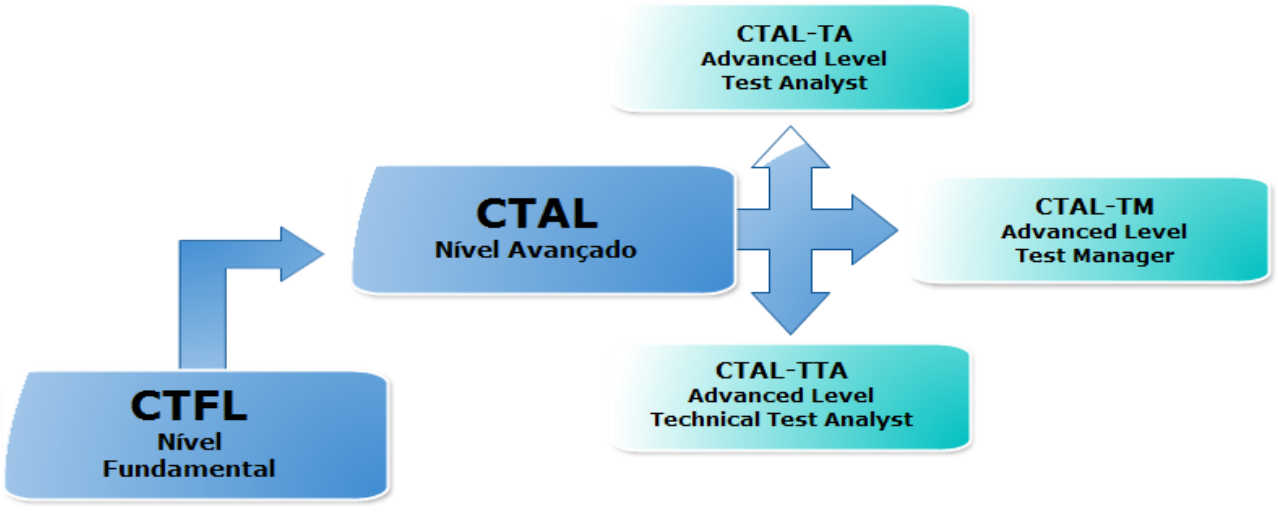
Missão do ISTQB:

- Aumentar o número de testadores qualificados ao redor do mundo;
- Promover o teste como uma profissão em mais países;
- Capacitar testadores a obter qualificação reconhecida na sua linguagem nativa;
- Permitir o compartilhamento de conhecimentos e recursos entre os países;
- Prover o reconhecimento internacional tanto dos testadores quanto da qualificação.

CERTIFICAÇÕES

Características das certificações ISTQB

- Não expiram;
- Reconhecimento e prestígio internacional;
- Dois níveis de certificação (Nível Expert em desenvolvimento):



CERTIFICAÇÕES

CTFL:

- Certified Tester Foundation Level;
- nível fundamental destinada a:
  - Testadores;
  - Analistas de Testes;
  - Engenheiros de Testes;
  - Consultores de Teste;
  - Gerentes de Teste;
  - Aceitação e Desenvolvedores de Software.
- Também serve aos interessados em maiores conhecimentos sobre testes, como gerentes de projetos, gerentes de qualidade, gerentes de desenvolvimento, analistas de negócios e gestores.
- http://www.bstqb.org.br/uploads/docs/guiadocandidato.pdf

CERTIFICAÇÕES

CTAL:

- Certified Tester Advanced Levell;
- nível avançado destinada aos profissionais que precisam de maiores conhecimentos, e já detém maior experiência em teste de Software.
- Pré-requisito:
  - CTFL.
  - Comprovar pelo menos um dos itens:
    - 2 anos de experiência prática em Teste de Software ou Qualidade em TI.
    - 2 anos dedicados à Pesquisa Acadêmica relacionadas à Qualidade de TI no nível de Pós Graduação, ou como instrutor de cursos ou disciplinas relacionadas à Qualidade de TI.
    - 3 anos de experiência prática em Desenvolvimento de Sistemas, Análise de Sistemas ou Engenharia de Software.

- http://www.bstqb.org.br/uploads/docs/guiadocandidato.pdf

Glossário

A

Analisabilidade: A capacidade que o sistema tem de ser diagnosticado para encontrar deficiências ou causas das falhas, ou para identificar quais partes que serão modificadas.
Analisador: Ferramenta que realiza a análise de um módulo ou software.
Analisador de código: Ferramenta que realiza a análise estática do código, verificando o código fonte em busca de determinadas propriedades, como conformidade a padrões de codificação, métricas de qualidade ou anomalias no fluxo dos dados.
Análise de árvore de falhas: Método usado para analisar as causas das falhas (defeitos).
Análise de cobertura: Mensuração da cobertura atingida para um determinado item de cobertura durante a execução dos testes, consultando critérios predeterminados para verificar se são necessários testes adicionais, e, se forem, quais casos de teste são necessários.
Análise de fluxo de dados: Uma forma de análise estática baseada na definição e uso de variáveis.

Qualidade de Software

- **É algo que todos querem:**
  - **Gerentes sabem que eles precisam ter alta qualidade em seus trabalhos;**
  - **Desenvolvedores sabem que desejam produzir um produto de alta qualidade;**
  - **Os usuários insistem que o seu trabalho através do uso do software deve ser confiável e consistente**

- **Dificuldade: organizações formam grupos de garantia de qualidade, melhorando e avaliando suas aplicações, porém, não existe aceitação geral (e real) prática de garantia de qualidade dentro dessas organizações.**
- **Sem um correto direcionamento de como gerir a qualidade, estamos em apuros!!!**

Qualidade de Software

- **Premissas importantes:**
- **Qualidade requer comprometimento, particularmente vindo do topo do gerenciamento: importante a cooperação entre a equipe e o gerente para “fazer acontecer”;**

- **Não existe o “zero-defeito”: é sensato definir níveis aceitáveis de defeitos. Um carro com o banco furado não o impede de andar, mas e se dois pneus furarem??**

- **Qualidade é frequentemente associada a custo, significando alta qualidade = alto custo. Nem sempre é verdade (Molinari diz que é FALSO!);**

Qualidade de Software

- **Premissas importantes:**
- **Qualidade demanda especificação de requisitos e suficiente detalhamento deles. Deve-se permitir a completa mensuração dos requisitos, permitindo medir o esforço a ser gasto na conformidade;**

- **Acredita-se que padrões inibem sua criatividade, e com isto padrões não são seguidos, entretanto, para a qualidade acontecer, padrões e procedimentos devem ser seguidos.**

Prevenção VS Detecção

Fato: uma falha, se não for prevenida, certamente será detectada.

Outro Fato: falhas levam a custos.

O gerenciamento da qualidade diminui os custos, visto que se cedo um defeito for localizado e corrigido, menor será o custo de defeitos encontrados ao longo do tempo.

O custo efetivo do gerenciamento de qualidade:

R$ prevenção + R$ inspeção + R$ falhas internas + R$ falhas externas

Prevenção VS Detecção

R$ prevenção + R$ inspeção + R$ falhas internas + R$ falhas externas

- Custos de falhas internas: corrigir defeitos antes de serem entregues;
- Custos de falhas externas: custos de falhas descobertas depos que os produtos foram entregues e liberados. Custos com correções e readequações, além dos custos intangíveis (reputação e imagem da fábrica de software).
- Maior será o ROI (return of investment) quanto maior for a prevenção.

Verificação VS Validação

- Verificação prova que o produto vai de encontro dos requisitos especificados;
- Validação checa se o sistema vai de encontro dos requisitos do consumidor;
- Um teste de um produto está muito mais perto de validação do que de verificação(MOLINARI, 2003);
- Tradicionalmente teste de software é considerado um processo de validação. Porém, quando a verificação é incorporada ao teste, o teste corre durante o desenvolvimento também;
- Verificação pode incrementar o custo, mas incrementa a qualidade.

Verificação VS Validação

- Verificação inclui procedimentos sistemáticos de revisão, análise e testes empregados durante o desenvolvimento, começando com as fases de requisitos de software e continuando através da codificação do produto.

- Afinal, o teste de software que alegamos fazer, na verdade, não passa de verificação básica!!!

- Quando procedimentos de verificação são usados, o gerenciamento pode assegurar que os desenvolvedores seguem um formal e sequencial processo de desenvolvimento, com um mínimo de atividades que garanta a qualidade do sistema.

Glossário

A

Análise de impacto: Avaliação das alterações a serem feitas nos documentos de desenvolvimento e teste, e nos sistemas, para implementar essas alterações nos requisitos especificados.

Análise de mudanças: Método que determina a completude da suíte de testes medindo o grau com que essa suíte consegue distinguir entre o programa original e o programa com ligeiras variações (mutações).

Análise de pontos de função (FPA): (Do inglês: Function Point Analysis). Método que tem como objetivo medir o tamanho da funcionalidade do sistema. Essa métrica é independente da tecnologia usada e pode ser usada como base para métricas de produtividade, para estimativa de recursos necessários e para controle do projeto.

Software Quality Assurance – SQA

Definição:

“Atividades sistemáticas que preveem evidência de melhoria do uso total do software enquanto produto” (MOLINARI, 2011).

É sedimentada através do uso de guias estabelecidas de controle de qualidade que garantam a integridade e o prolongamento da vida do software.

Conjunto de atividades necessárias para prover adequada confiança que o processo seja estabelecido e continuamente melhorado e organizado de forma a ir de encontro das especificações e seja adequado ao uso.

Software Quality Assurance – SQA

SQA = Garantia de qualidade de software (GQS).

Teste de software é uma parte integrante de SQA, isto é, teste de software é parte de uma estratégia, juntamente com outras técnicas, para SQA.

É uma área chave de um processo de software cujo objetivo é fornecer aos vários níveis de gerência a adequada visibilidade dos projetos, dos processos de desenvolvimento e dos produtos gerados. Ou seja, criado para avaliar o esforço de desenvolvimento de software.

Pode ser visto como área do CMM.

Software Quality Assurance – SQA

Pode ser descrito em 6 dimensões:

- 1.Métodos e ferramentas de construção;
- 2.Revisões formais;
- 3.Estratégia de teste;
- 4.Controle de documentação e sua história de mudanças;
- 5.Procedimentos para garantir a adequação aos padrões de desenvolvimento;
- 6.Mecanismos de medição e análise.

Software Quality Assurance – SQA

Pode-se dizer que possui como foco:
(bom, na verdade não é seu foco, mas pode contribuir!!)

- --Permitir o desenvolvimento de Software de alta qualidade;
  - –Garantir alta produtividade da equipe de desenvolvimento;
  - --Permitir o cumprimento do cronograma estabelecido;
  - --Não necessitar de recursos adicionais não previstos.

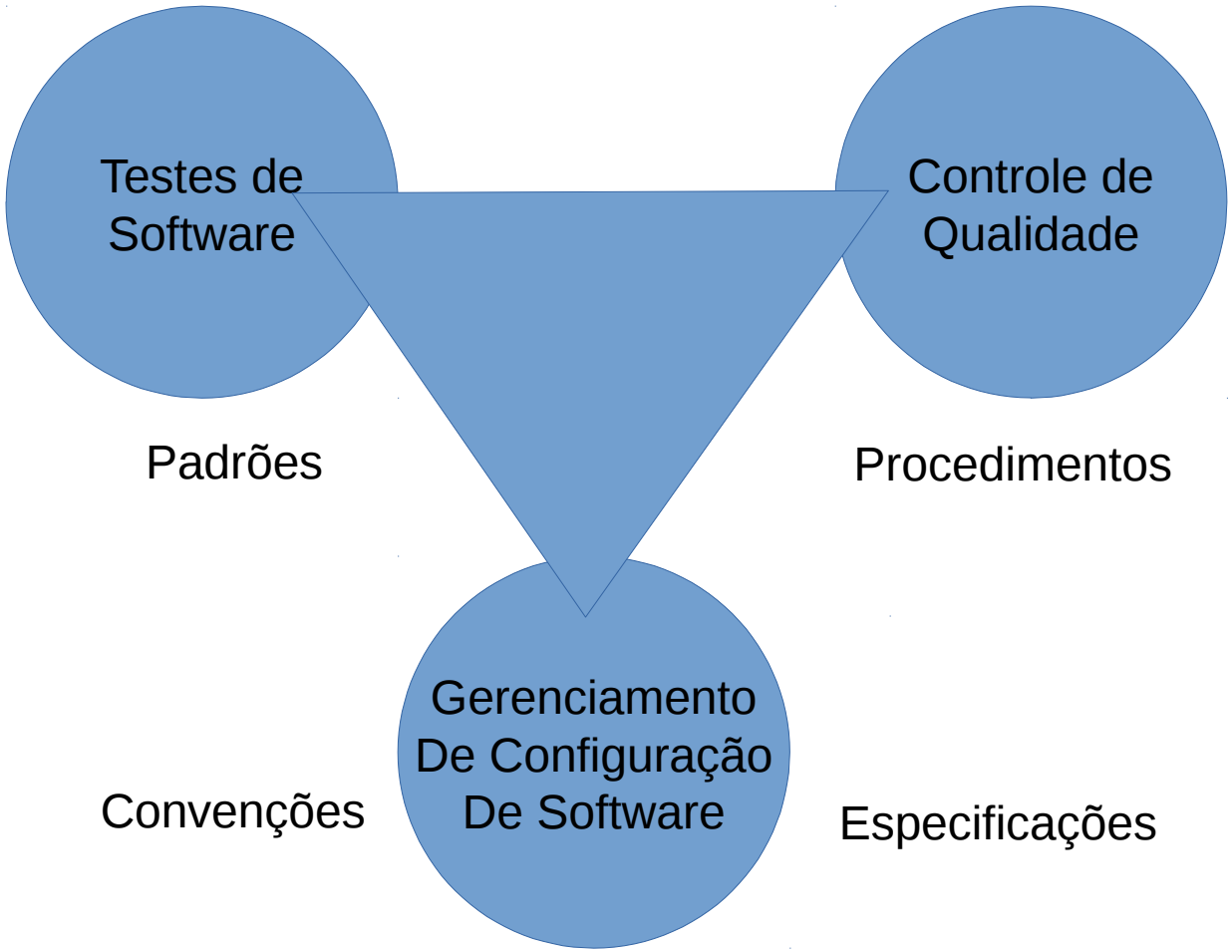
Software Quality Assurance – SQA

É **planejar** com o esforço de garantir que o software preencha todos os requisitos do produto, além de outros atributos, por ex.:

--**portabilidade**, eficiência, **reusabilidade** e flexibilidade.

Plano de SQA, que estabelece que os métodos utilizados no projeto sejam empregados para garantir que os documentos e produtos gerados e revisados (artefatos) sejam de alta qualidade.
Planejar a coleção de atividades e funções usadas para monitorar e controlar o projeto, de maneira que os objetivos sejam atingidos com o nível desejado.

Componentes da SQA



Glossário

A

Análise de pontos de teste (TPA): (do inglês: Test Point Analysis) Método de estimativa de esforço de teste baseado nas atividades relacionadas aos testes. Inspirado na análise por pontos de função.

Análise de riscos: Processo de avaliar os riscos identificados para estimar seu impacto e probabilidade de ocorrência.

Análise de valor limite: Técnica de projeto de teste caixa-preta onde os casos de teste são definidos com base no valor limite.

Análise dinâmica: Processo de avaliar o comportamento do sistema durante sua execução, como por exemplo, o desempenho da memória e uso da CPU.

Análise estática: Análise dos artefatos de software, como por exemplo, requisitos ou código, sem executá-los.

Capability Maturity Model – CMM
Modelo de Maturidade em Capacitação

O modelo CMM (SW-CMM) foi produzido por um grupo de profissionais de softwares (USA) – Primeira versão em 1991;

Necessidade de atender uma demanda do governo americano: avaliar a capacitação de seus fornecedores de software (licitações);

Define:maturidade
substantivo feminino

1. estado, condição (de estrutura, forma, função ou organismo) num estágio adulto; condição de plenitude em arte, saber ou habilidade adquirida. "m. intelectual"

2.termo último de desenvolvimento. "m. das ciências"

Antes de entender o conceito de maturidade:

Segundo o IEEE: **“Processo” é uma sequência de passos realizados para atingir um determinado objetivo.**

Segundo o CMM: **um “processo de software” é um conjunto de atividades, métodos, práticas e transformações que as pessoas usam para desenvolver e manter o software e seus produtos associados.**

CMM tem seu foco no processo de SW por entender que a qualidade de um sistema de SW é fortemente influenciada pela qualidade do processo utilizado para desenvolvê-lo e mantê-lo:

Foco no processo e no produto!!!

Antes de entender o conceito de maturidade:

A **Capacidade do Processo de SW** descreve o conjunto de resultados esperados que pode ser atingido quando se segue o processo estabelecido.

A **Maturidade do Processo de SW** é o quanto um processo específico é explicitamente definido, gerenciado, medido, controlado e efetivo.

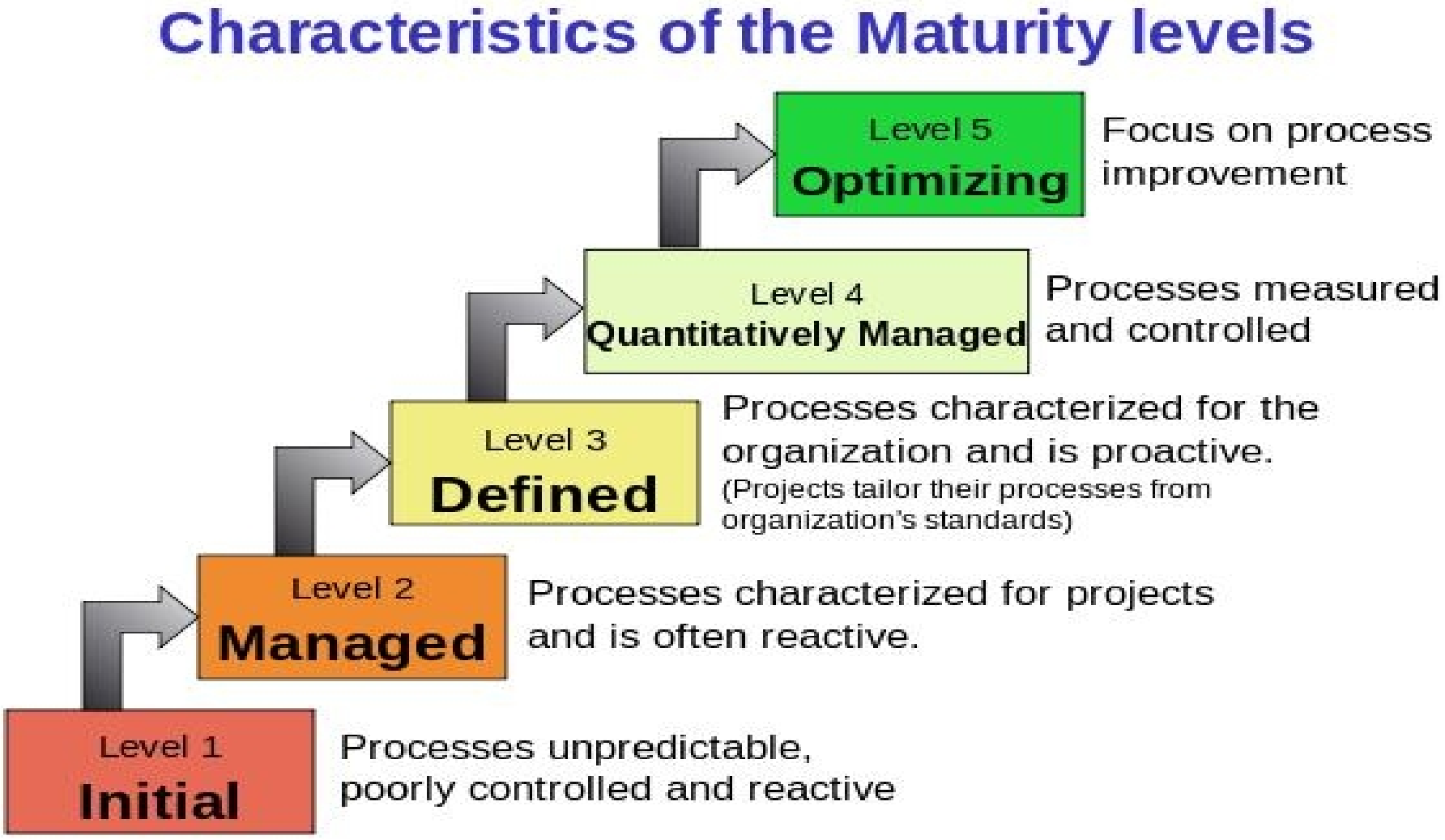
Portanto: **maturidade** implica num potencial de crescimento da capacidade e indica tanto a riqueza do processo de SW de uma organização, quanto a consistência na qual o processo é aplicado.

Uma empresa imatura (processo imaturo)

- Processos são improvisados ou não são seguidos;
- O trabalho é feito em regime de emergência;
- Compromissos de prazos e custo não são cumpridos;
- O planejamento não é feito com base em estimativas realistas;
- As iniciativas de melhoria do processo não se sustentam e não se perpetuam;
- Quando o projeto é pressionado por prazo, a qualidade e funcionalidade são sacrificadas;

Obs.: Mesmo assim podem produzir bons produtos...

Os níveis de maturidade



Fonte: http://istqbexamcertification.com/what-is-cmm-capability-maturity-model-what-are-cmm-levels/

Glossário

A

Análise estática do código: Análise do código fonte sem executá-lo.

Anomalia: Qualquer condição que desvia da expectativa (baseada nas especificações de requisitos, documentos de projeto, documentos de usuários, padrões, etc.) ou da percepção ou experiência de alguém. As anomalias podem ser encontradas durante (mas não limitada a) a revisão, testes, análise, compilação ou uso dos sistemas ou da documentação aplicável.

Atratividade: A capacidade que o sistema tem de ser atraente para o usuário. [ISO 9126] Veja também usabilidade.

Atributo de qualidade: Um atributo ou característica que afeta a qualidade de um item. [IEEE 610]

Auditoria: Uma avaliação independente do sistema ou processo de software para determinar a conformidade a padrões, diretrizes, especificações, e/ou procedimentos baseados em critérios objetivos, incluindo os documentos que especificam:
(1) A forma ou o conteúdo dos produtos que serão produzidos;
(2) O processo pelo qual os produtos devem ser produzidos;
(3) Como a conformidade a padrões ou diretrizes deve ser medida. [IEEE 1028]

A Nova Qualidade de Software

- **Nova tecnologia = novo bug;**
- **Questões novas surgidas pelo enfoque dado ao processo de desenvolvimento:**
  - **Qual a performance monetária do defeito que não podemos medir?**
  - **Como minimizar os custos da má qualidade do processo de desenvolvimento?**
  - **O que podemos fazer com poucos recursos para minimizar o custo do defeito e resolvê-lo em si?**

A Nova Qualidade de Software

- **Uma tendência?**
  - **A nova qualidade de software é antes de tudo a prevenção dos “bugs” (MOLINARI, 2011).**

- **Na prática, nota-se que os processos de desenvolvimento “fingem” prevenir defeitos, pois raramente vê-se um caso completo em que tudo o que ele realmente propõe foi testado, ou testado corretamente.**

- **Porém, a indústria (mercado?) almeja mais qualidade.**

1º item: Medição

Como prever a satisfação do usuário antes da liberação da primeira versão do sistema???

- **Ponha-se no lugar do usuário!!!**
  - **Será que o que foi desejado pelo usuário é, de fato, satisfatório?**
    - **“mas ele disse que poderia aguardar até 1 minuto pelo carregamento da lista!” -** Será mesmo que ficar parado 1 minuto na frente de um computador sem fazer nada é satisfatório, na prática?
    - Dica: medir antes das entregas parciais (marcos) os pontos fundamentais, como o tempo, segurança e facilidade de uso. Uma simples medição pode verificar constrangimentos reais de projeto.

1º item: Medição

Mas... como implantar um programa de métricas de qualidade de software?

- **Medir é algo que depende do método, do meio, de quem mede e do que se quer medir.**
  - Uma medição realizada por um observador imprudente pode baixar a qualidade da medição.
  - Saber o tempo de resposta da aplicação e sua disponibilidade são itens que compõem um processo de medição.
  - Métrica: SLA (System/Service Level Agreement – acordo de nível de sistema/serviço).

> é a especificação, em termos mensuráveis e claros, de todos os serviços que o contratante pode esperar do contratado na relação contratual, bem como termos de compromisso, metas de nível de serviço, suporte técnico, prazos contratuais, dentre outros aspectos. Em outras palavras, é um esclarecimento técnico do contrato.

1º item: Medição

Mas... como implantar um programa de métricas de qualidade de software?

- Para montar uma métrica, deve-se montar primeiro a meta, depois montar as perguntas que se complementam para atender à meta, e os tipos de medições distintas que darão ideia de como o processo está andando.

Por ex.:

- **Durante o teste é possível medir o quanto de requisitos já foi testado e o quanto ainda falta (métrica de cobertura).**

1º item: Medição

Como medir nessa nova qualidade de software?

- montar programas de medição de desenvolvimento do software em si:
  - de defeitos que apareçam antes e depois da implantação;
  - realizar testes automatizados de acompanhamento em produção da aplicação;
  - verificar a disponibilidade da aplicação e o seu atendimento ao SLA.

2º item: Gerenciamento dos defeitos encontrados

Defect Tracking ou bug Tracking

- Armazenar o histórico de defeitos encontrados durante os testes da aplicação em todo o ciclo de vida da mesma, indo desde a análise até a manutenção.
- **Um carro novo vem com um manual indicando a velocidade máxima do carro. Se no manual diz que o limite do carro é 100 km/h e você faz uma curva a 130 km/h e derrapa, de quem é a culpa? Não é do projeto do carro, pois o problema ocorreu numa situação extrema, que estava fora do estabelecido. Claro que o carro teve problemas, mas não defeitos.**
- Tecnicamente, se documentarmos um problema, ele deixa de ser candidato a se tornar defeito.
-

2º item: Gerenciamento dos defeitos encontrados

Defect Tracking ou bug Tracking

- **Dica: uso de ferramentas para gerenciar de forma macro os defeitos:**
  - Bugzilla
  - Eventum
  - FogBugz
  - Jira
  - Mantis
  - Trac
  - Redmine
  - Request Tracker
  - OTRS

3º item: Performance de Custos do Processo

Como medir a performance da aplicação em produção?

- Uma ferramenta automatizada para medir o SLA pode dar a noção monetária se a aplicação atinge ou não o percentual desejado de satisfação em termos de performance.
- **Se a aplicação não atinge o SLA, encontramos uma curva de prejuízos.**
- **O quanto de prejuízo somente saberemos quando for informado o custo do não atendimento a um SLA: isso varia pelo custo do erro para o cliente:**
  - ex.: a aplicação não atendeu as requisições corretamente em 10% do tempo num total de 100 horas de trabalho. Isso deu de prejuízos ao cliente R$ 10,00 por hora: R$ 1.000,00 de prejú!!!

3º item: Performance de Custos do Processo

Como medir o custo do defeito encontrado?

$$CMD = \frac{(Nº\ pessoas\ envolvidas * nº\ dias\ gastos) * custo\ por\ pessoa - dia}{nº\ de\ defeitos\ resolvidos}$$

- **Custo médio do defeito** pode ser usado para complementar os custos do sistema envolvido, e medir os custos de implementação.
- A efetiva performance de custos do processo pode ser calculada baseando-se no atendimento ao SLA e na quantidade de defeitos

Glossário

A

Auditoria de configuração: Verifica o conteúdo das bibliotecas dos itens de configuração, para verificar, por exemplo, a conformidade a padrões. [IEEE 610]

Automação da execução dos testes: Uso de software, como por exemplo, ferramentas de captura/reprodução, para controlar a execução dos testes, a comparação dos resultados obtidos e esperados, a configuração das pré-condições dos testes e outros controles dos testes e funções reportadas.

Automação dos testes: Uso do software para executar ou apoiar as atividades de testes, como por exemplo, gerenciamento, projeto e execução de testes, e verificação dos resultados.

Avaliação heurística: Técnica estática de teste de usabilidade para determinar a conformidade da interface do usuário com os princípios de usabilidade reconhecidos (também chamado de heurísticas).

Visão Mundial

Primórdios: SW produzido de maneira descompromissada com o que estava sendo feito – iniciativas isoladas procuravam melhorar o produto final;

Década de 80: caça às bruxas, em que o importante era descobrir “bugs” de software;

Década de 90: o enfoque volta-se para o negócio, em que o software deve suportar o negócio, sem exceções;

Agora: evolução; tornar o desenvolvimento coberto, garantido e assistido;

Testes de Software são o carro-chefe desse processo, sendo fundamental para muitas empresas. A bolha da internet teve como um dos fatores a falta de atendimento à demanda de cliente.

“Ah... que beleza uma página com erro 404 Not Found... se tivessem testado o site antes...” (autor desesperado, pronto pra pular da ponte!)

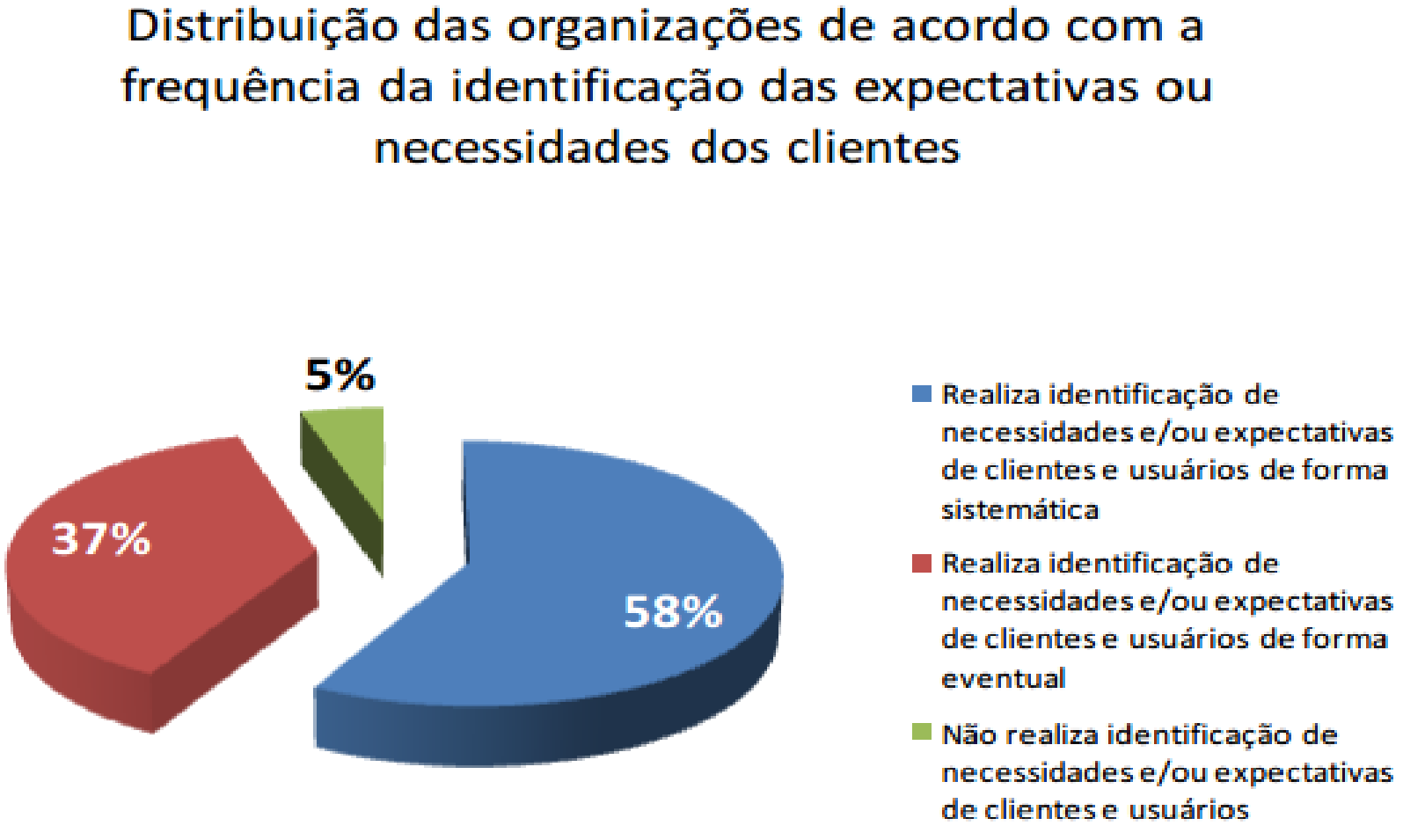
Visão Brasil

Primórdios: criado o PBQP-Software em 1990, pelo ministério da ciência e tecnologia, com foco na modernização do processo de produção através da promoção da qualidade e produtividade, com consequente ampliação da competitividade de bens e serviços produzidos no Brasil.

Realizou pesquisas que demonstram a maturidade do processo de qualidade de software no Brasil. (http://www.mct.gov.br/upd_blob/0210/210931.pdf)

- 84,7% das empresas de SW localizadas na região Sudeste;
- 12,4% na região Sul;
- 95,2% de pequeno porte, com até 9 profissionais, 3,9% com 10 a 49 profissionais e 1% com 50 ou mais profissionais;
- 89% não utilizam **nenhuma** norma para requisitos de qualidade de software;

Visão Brasil



Referências

Myres , G. F. “The Art of Software Testing”. Ed. John Wiley & Sons, Inc. New Jersey, 2004.

Dijkstra, E. W. "The Humble Programmer". Communications of the ACM 15 (10): 859–866, 1972.

Rios, Emerson. “Teste de software”. Alta Books Editora, 2006.

Gelperin, David, and Bill Hetzel. "The growth of software testing." Communications of the ACM 31.6 (1988): 687-695.

MOLINARI, Leonardo. “Testes de software: produzindo sistemas melhores e mais confiáveis: qualidade de software: soluções, técnicas e métodos”. Érica, 2003.