Visões sobre Teste de Software

Diferentes perspectivas da comunidade de teste brasileira discutidas no DFTestes

Indice

Sobre o eBook

Este eBook é um trabalho colaborativo e aberto a comunidade de teste de software.

Este ebook foi criado com o intuito de disseminar e facilitar o acesso a informação gerada pela comunidade de teste de software possuindo diversos autores e colaboradores que fizeram o trabalho de revisão e escrita das Mesas Redondas no grupo de e-mail DFTestes.

Distribua este eBook a qualquer pessoa que queira receber informação sobre a área de teste de software.

Como este conteúdo surgiu?

Em 2009, em uma lista de e-mails chamada DFTestes o Fabricio Ferrari lançou na lista um desafio colaborativo chamado de Mesa Redonda.

Depois de diversas mesas redondas e ótimas discussões resolvemos (como comunidade) elencar as 10 principais e trabalha-las a fim de criar um conteúdo que qualquer um possa consumir sem custo e com a visão de diversos profissionais da área de teste de software.

Eu (Elias Nogueira) estou chamando agora esta iniciativa criada lá em 2009, onde muitas continuam muito atuais, de **Visões sobre Teste de Software**. As 10 principais discussões estão também disponíveis no site http://dftestes.gershon.info/Conteudo.html

Este ebook foi criado a "quatro mãos" por profissionais de Teste de Software que deram seu tempo para disseminar este conhecimento. Todos eles são os autores deste ebook e merecem todo o nosso reconhecimento!

Introdução

Shmuel Gershon

Revisor: Marcelo Andrade

> *(...) a terra em si é de muito bons ares (...) Em tal maneira é graciosa que, querendo-a aproveitar, dar-se-á nela tudo!*

Em seu relatório de viagem sobre o Brasil, Pero Vaz de Caminha demonstrou bastante perspicácia para um turista em sua primeira visita. Caminha também notou algo mais sobre esta nova 'ilha', e afirmou que apesar de todas as terras e águas, o que o Brasil tem de melhor são seus habitantes: "o melhor fruto que dela se pode tirar parece-me (...) esta gente."

Eram os anos Mil e Quinhentos, e nos anos Dois Mil a situação pouco mudou. Trocam-se os 'mancebos' nativos pelos brasileiros modernos, e trocam-se os arcos e flechas por computadores, sistemas operacionais, Internet e aplicativos; o Brasil ainda está empenhado em avançar.

Esse avanço tecnológico cobre todas as áreas da informática moderna. Uma boa fração da população ainda não tem acesso a essas novidades, mas a indústria mantém-se atualizada e oferece serviços que competem no mercado internacional.

Um desses serviços, matéria que viu grande crescimento no Brasil nesta última década, é o Teste de *Software*.

Esta importante etapa no desenvolvimento de *software* tem como objetivo revelar informações sobre os aplicativos e diminuir a incerteza sobre sua qualidade e risco. A qualidade de um *software* é definida pelo valor que este tem para o cliente, e os testadores são os responsáveis por estudar o valor percebido pelo cliente e relacioná-lo às necessidades da empresa.

Hoje, mais de 1500 desses testadores de *software* se reúnem no fórum de discussão DFTestes, correspondendo-se através de mais de 300 mensagens ao mês. A que se deve tanto movimento?

Fóruns de discussão são o *habitat* natural dos testadores de *software*, pois "Testar é questionar um produto para descobrir informações sobre sua qualidade", como define James Bach. Um testador passa sua jornada de trabalho fazendo perguntas; ao diretor, ao cliente, ao programador, a um colega, até mesmo perguntas ao aplicativo. Mas as questões mais difíceis, as "cabeludas" e filosóficas, as que precisam de uma visão objetiva e descompromissada, essas vão para o fórum, para os colegas virtuais.

Para quem não conhece a indústria de testes, isso pode parecer estranho... Que perguntas cabeludas podem haver sobre Testes de *Software*? Testar o *software* consiste em executar um programa e seguir passos predefinidos, não?

Essa falta de segurança na natureza do trabalho é uma das grandes dúvidas de nossa área. Algumas respostas estão disponíveis no capítulo "Testar é tão fácil que até minha mãe testaria" deste livro.

Pois este é um dos objetivos deste livro: expor o mundo dos Testes de *Software* ao leitor de diferente área.

Ao abordar os dilemas de forma aberta e baseando-se em comentários da comunidade, este livro é um ótimo retrato do cenário atual de Teste de *Software* no Brasil. Isso é, um grande auxílio para colaboradores e diretores, pois empresas e equipes precisam dessa compreenção para posicionar seu time de testadores de maneira eficiente. Por exemplo, todo administrador procura as melhores soluções para os problemas cobertos nos capítulos "Melhoria do Processo de Teste" e "Estimativas de Testes". E em uma transição para metodologias ágeis, o papel do time de testes pode ser definido de diversas maneiras -- de modo que os capítulos "Teste ágil, como implementar?" e "Utilização de Processos Ágeis no Teste de *Software*" sao úteis ao apresentar algumas das práticas da indústria.

Mas a educação de parceiros não é o único objetivo deste livro. O principal propósito é guiar o testador irresoluto, e ajudá-lo a levantar novas perguntas. As dúvidas são inúmeras, sejam relacionadas a carreira (ver capitulos "Certificações, valem a pena?" e "O Testador também necessita saber programar?"), ou relacionadas a eficácia no trabalho ("Quando automatizar?", "Testar sem documentação é possível?" e "Quando documentar não é preciso?").

Denota-se que o livro não visa responder em definitivo a nenhuma dessas questões, mas sim oferecer comentários de colegas de profissão experientes e referências *online* -- na esperança que o leitor possa criar sua própria resposta que satisfaça seu contexto e necessidades.

Como escreve Michael Bolton, "Contrariamente ao folclore de nossa profissão, uma boa pergunta sobre testes não tem necessariamente uma resposta definitiva ou um resultado decisivo. Testes ou perguntas que apenas visam confirmar uma teoria predeterminada dificilmente revelam nova informação útil."

Finalmente, por que não, fica o leitor convidado a compartilhar conosco suas próprias respostas e opiniões sobre os assuntos na lista DFTestes. É dessa contribuição que nossa comunidade deriva sua força, e todos são bem vindos!

Autores e Revisores

Introdução

Autor: **Shmuel Gershon é** Líder Técnico no campus Israelense da Intel Corporation, Shmuel Gershon tem experiencia em testes de firmware e software, em treinamento de testadores e em ajuda a amigos. No passado ele foi programador, mas descobriu que testar provê diversão em dobro. Shmuel acredita que os fatores mais significativos na nossa busca pela qualidade são as pessoas, e não funções ou tecnologias. Ele escreve sobre testes em http://testing.gershon.info, e recentemente publicou "Rapid Reporter", um aplicativo para tomar notas em testes exploratórios.

Revisor: **Marcelo Andrade**

Capítulo 1

Autora: **Alice H. Tamashiro** é especialista em Gestão da Qualidade, Testes e Processos. Trabalha como professora no curso de pós-graduação em Engenharia de Software, Desenvolvimento Colaborativo - JAVA e no curso de extensão em Qualidade e Testes de Software na Universidade da Cidade de São Paulo - UNICID.

Revisor: Rodrigo Souza

Capítulo 2

Autor: **Gustavo Nascimento** é docente de graduação e de pós-graduação, ministra disciplinas relacionadas a qualidade de software, a testes, a gerência de configuração e ao modelo MPS.BR. Trabalha com melhoria de processos baseada no modelo MPS.BR e coordena equipes de testes, de gerência de configuração e de garantia da qualidade. Gustavo também trabalha e ministra cursos sobre ferramentas de integração de processo e sobre os temas citados.

Revisor: **Felipe da Silva** é Analista de Testes da IBM, atua em contato direto

com cliente DIRECTV dos E.U.A, atua na área de qualidade e melhoria de processos, desenvolve e aprimora ferramentas e metodologias visando à melhoria contínua de produtividade, presta suporte ao cliente e ao time da IBM na Índia, assim como leciona treinamentos e apoia demais membros do projeto, é ponto focal sênior na aplicação coração do sistema de vendas de seu cliente.

Capítulo 3

Autora: **Karine Birnfeld**

Revisora: **Anna Carolina Rocha**

Capítulo 4

Autora: **Sarah Pimentel**

Revisora: **Keite Moraes**

Capítulo 5

Autor: **Edwagney Luz**

Revisora: **Keite Moraes**

Capítulo 6

Autora: **Maíra Dutra**

Revisora: **Patrícia Silva Corrêa**

Capítulo 7

Autor: **Lucas Gonçalves Nadalete**

Revisora: **Juliana Kryszczun**

Capítulo 8

Autor: **Fabrício Ferrari de Campos** é apaixonado pelo que faz e tenta espalhar esse espírito para os outros profissionais, pois acredita que o trabalho é uma das formas de contribuir para o mundo e também para o crescimento pessoal e profissional. Atua na Vizir, onde é co-fundador, uma *start-up* focada em mídias sociais que desenvolve o Vizir, uma ferramenta para monitoração nas redes sociais, e também presta consultorias na área de desenvolvimento de software. Fabrício também é autor do QualidadeBR, um blog focado em Teste e Qualidade de Software, onde ele tenta escrever semanalmente sobre essa área tão divertida e desafiante.

Revisor: **Marcelo Andrade**

Capítulo 9

Autores: **Fabrício Ferrari de Campos** e **Karine Birnfeld**

Autor: **Fabrício Ferrari de Campos** é apaixonado pelo que faz e tenta espalhar esse espírito para os outros profissionais, pois acredita que o trabalho é uma das formas de contribuir para o mundo e também para o crescimento pessoal e profissional. Atua na Vizir, onde é co-fundador, uma *start-up* focada em mídias sociais que desenvolve o Vizir, uma ferramenta para monitoração nas redes sociais, e também presta consultorias na área de desenvolvimento de software. Fabrício também é autor do QualidadeBR, um blog focado em Teste e Qualidade de Software, onde ele tenta escrever semanalmente sobre essa área tão divertida e desafiante.
Revisora: **Patrícia Silva Corrêa**

Capítulo 10

Autor: Ueslei Aquino da Silva é Analista de Testes Sênior da RSI. Atuou como líder de equipe de testes na iTeste. Ocupou a posição Gestor do Dpto de Testes na MXM Sistemas. Pós-Graduado com MBA em Garantia da Qualidade de Software pela COPPE-UFRJ. Atuou como membro do Conselho Técnico do MPT pela ALATS com tarefas de ((i)Criação e aprimoramento contínuo do modelo MPT e seus Guias específicos;(ii)Validação das avaliações efetuadas; (iii) etc.). Atuou no Comitê de Inovação da ALATS na frente de trabalho do MPT para revisão e auxílio às necessidades do MPT. É Certificado como Implemetador MPT (Nível 1 e 2) e CBTS - Certificação Brasileira de Teste de Software. Consultor MPT.
Revisora: **Carla Regina Florentino Sampaio**

Diagramadores

Robson Agapito Corrêa

Fabrício Ferrari de Campos é apaixonado pelo que faz e tenta espalhar esse espírito para os outros profissionais, pois acredita que o trabalho é uma das formas de contribuir para o mundo e também para o crescimento pessoal e profissional. Atua na Vizir, onde é co-fundador, uma *start-up* focada em mídias sociais que desenvolve o Vizir, uma ferramenta para monitoração nas redes sociais, e também presta consultorias na área de desenvolvimento de software. Fabrício também é autor do QualidadeBR, um blog focado em Teste e Qualidade de Software, onde ele tenta escrever semanalmente sobre essa área tão divertida e desafiante.

Facilitadores

Daniel Goettenauer é Bacharel em Sistema de Informação pela Universidade Luterana do Brasil, Tecnólogo em Desenvolvimento de Software pelo Instituto Federal do Amazonas; Pós-graduado em Governança de TI pelo SENAC-RIO. Há 5 anos dedicando estudos a Engenharia de Software com foco em Testes. Membro da Associação Latino Americana de Teste de Software. Colaborou como Editor no Livro Conversando sobre Teste de Software. Atualmente cursando MBA de Gerenciamento de Projetos na Fundação Getúlio Vargas.
http://goette.com.br

Fabrício Ferrari de Campos é apaixonado pelo que faz e tenta espalhar esse espírito para os outros profissionais, pois acredita que o trabalho é uma das

formas de contribuir para o mundo e também para o crescimento pessoal e profissional. Atua na Vizir, onde é co-fundador, uma *start-up* focada em mídias sociais que desenvolve o Vizir, uma ferramenta para monitoração nas redes sociais, e também presta consultorias na área de desenvolvimento de software. Fabrício também é autor do QualidadeBR, um blog focado em Teste e Qualidade de Software, onde ele tenta escrever semanalmente sobre essa área tão divertida e desafiante.

Paulo César

Ricardo Franco

Editores

Andrea Cruz

Daniel Goettenauer

Elias Nogueira

Juliana Kryszczun

Capítulo 1: Testar é tão fácil, que até a minha mãe testaria!

Alice Tamashiro

Revisor: Rodrigo Souza

Introdução

Atualmente vivemos em um mundo onde os negócios possuem uma grande dependência pela TI (Tecnologia da Informação) e à medida que esse fator aumenta, também cresce a produção de *software* e complexidade dos sistemas. Um exemplo é quando um risco operacional da TI se propaga para os negócios, deixando de ser um risco da TI, passando a ser um risco do negócio, ou seja, se uma falha ocorrer nos sistemas, consequentemente ocorrerá uma falha nos negócios. Desta forma, torna-se fundamental o desenvolvimento de software com mais qualidade e confiabilidade. Neste capítulo será abordado o porquê se devem ter profissionais qualificados na elaboração e execução de testes.

Linha de Evolução entre TI e Testes

A tecnologia pode ser identificada em vários momentos da evolução humana, as descobertas de uma época que para a maioria da população eram novidades ou desconhecidas, eram consideradas como tecnologia.

A tecnologia relacionada a informações surgiu em meados dos anos 60 com a criação do Processamento de Dados. O Processamento de Dados era responsável por realizar a execução de atividades burocráticas como geração de relatórios de folhas de pagamento.

Nos anos 70, iniciou-se a era dos Sistemas de Informações onde as informações coletadas e reportadas pelo Processamento de Dados passaram a ser organizadas e refinadas com o auxilio da evolução do meio de armazenamento e processamento destas informações.

Nos anos 80, inseriu-se no contexto tecnológico a era da Inovação e Vantagem Competitiva, nessa época as empresas descobriram que podiam utilizar as informações de forma organizada e refinada com o auxilio da execução das tarefas de escritório e não apenas para agilizar o processo de trabalho, mas também tomar decisões de forma mais rápida e entender melhor os seus clientes. Nessa época também teve uma grande evolução do *hardware* e das redes de computadores, porém, ainda não havia uma boa integração.

Nos anos 90, com a necessidade constante de obter informações cada vez mais rápidas, distribuídas e com maior integração entre hardware, *software*, redes, surge a era da Integração e Reestruturação do Negócio, onde passa a ter maior flexibilidade e troca de informações em que as empresas se vêem na necessidade de utilizar cada vez mais sugestões tecnológicas para a tomada de decisão.

Assim como a tecnologia, os testes tiveram evolução ao longo do tempo. Essa evolução se deu devido à necessidade de melhorar a qualidade do *software* desenvolvido. Iniciou-se nos anos 70 devido ao aumento de informações e complexidade de armazenamento dos dados. No início, os testes de *software* eram realizados dentro do processo de desenvolvimento pelo próprio desenvolvedor realizando os atualmente chamados Testes Unitários.

A complexidade dos programas de computador dificultava muito a execução dos testes e como consequência os programas eram liberados com inúmeros defeitos. No final de 1979 o Teste de *Software* foi conceituado por Myers como sendo um processo no qual se executava um programa com a intenção de encontrar erros.

Nos anos 80 e 90, iniciou-se o movimento da melhoria dos Testes de*Software*, os resultados obtidos foram ótimos e empresas começaram a investir em ferramentas de automação, houve diminuição dos custos de correção dos defeitos, criação de área própria e aderente ao processo de desenvolvimento, as atividades de Testes de *Software* começaram a iniciar-se paralelamente e integradas com o desenvolvimento.

Hetzel conceituou o Teste de *Software* como sendo qualquer atividade que tem como objetivo mensurar a qualidade do *software* e avaliar um atributo de um programa ou sistema.

Em 2002, Graig conceituou o Teste de *Software* como um processo de ciclo de vida concorrente ao projeto e mantém o *testware* a fim de medir e melhorar a qualidade de *software* que está sendo testado.

Contudo, você deve estar se perguntando, porque diante de todo esse avanço da tecnologia e da qualidade de *software*, ainda ocorrem defeitos? Segundo Rios isso acontece devido:

- *As organizações não dão a devida importância à atividade, que é informal, sem metodologia, funções e responsabilidades*.
    - Em análise do contexto descrito, pode-se dizer que este fato ocorre devido à falta de conhecimento da empresa sobre seu próprio processo de desenvolvimento de *software*. Em caso de empresas em que a TI é um processo de apoio, não conseguem entender a importância das atividades de teste em conjunto com as atividades de desenvolvimento. Logo, não vêem a necessidade de se ter uma pessoa especialista responsável por esta atividade.

- *Os testes são incompletos durante o desenvolvimento, implicando em problemas que ocorrem após sua implantação.*
    - São incompletos devidos à visão de testes que toda a equipe de desenvolvimento possui (Gerente, Analista, Desenvolvedor e Testador, se existir). Em geral, cada papel citado possui uma preocupação diferente da execução de bons testes (isso deveria ser exceção para Testador). O Gerente de Projetos não quer ter muito custo com testes, os Analistas não procuram especificar de forma mais detalhada o possível para entendimento de qualquer pessoa que leia sua documentação, o Desenvolvedor busca testar o cenário feliz, ou seja, se o que ele desenvolveu está correto e o Testador acaba não possuindo base incentivo e metodologia de trabalho. Cenário que regride aos anos 70. O resultado disso é o alto custo para identificar e corrigir os problemas.

- *A abordagem dos testes não foi adequada para as novas tecnologias. Na*

realidade, pouco esforço foi feito nas organizações para adequar os procedimentos e reciclar o pessoal técnico de testes para tratar novas tecnologias.
    - O principal motivador é o custo que terá. Teste é visto como algo que não é caro, que deve ser mais barato que o desenvolvimento.

- *A estrutura organizacional para testes não tem se modificado. Quase todos os níveis de testes ainda são feitos pelos desenvolvedores que muitas vezes não gostam de testar o software. Além disso, não possui o perfil de Testador e/ou formalização especializada para executar tal atividade.*
    - Novamente custo.

- *Pouca utilização de ferramentas de automação de testes que são essenciais para certos tipos de testes.*
    - O investimento em automação é muito maior que o simples investimento em evolução de metodologia e conhecimento. Logo, torna-se algo ainda mais longe de se alcançar.

Concluindo, a TI era considerada armazenadora de informações e atualmente é considerada como ferramenta de extrema importância para a tomada de decisões de negócios. A disciplina de Testes de *Software* era considerada um meio de encontrar defeitos, atualmente é vista como ferramenta de melhoria e qualidade do *software*. A disciplina de Testes de *Software* teve uma grande evolução devido à necessidade de entregar produtos com mais qualidade e de acordo com as necessidades dos usuários que começou a surgir nos anos 70. Porém, as culturas das empresas ainda não se adequaram com a visão Melhoria e Qualidade de *Software*, considerando muitas vezes a disciplina de Testes como sendo apenas um meio de encontrar defeitos.

Atualmente, as especificações dos sistemas e regras de negócios não são adequadamente descritas e a atividade de Testes de *Software* é a última etapa no processo de desenvolvimento. Estas duas atividades levam à disponibilização de produtos e soluções que não operam de acordo com o especificado, levando à insatisfação do cliente e/ou usuários finais.

Desta forma, podemos afirmar que a quantidade de problemas encontrados na entrega e implantações de *software* são ocasionadas muito mais por falta de cultura que existem ainda nas organizações do que falta de evolução da tecnologia e metodologia para dar qualidade.

Mito ou Realidade?

Mesmo com o surgimento de padrões, modelos e metodologias reconhecidas internacionalmente a área de garantia da qualidade vem enfrentando diversos tipos de obstáculos o quais chamamos aqui de mitos, vejam alguns deles:

- Enquanto não tivermos um programa em funcionamento, não será possível avaliarmos a sua qualidade. Isso é um mito! Atualmente temos mecanismos mais efetivos para garantir a qualidade do *software*, tal mecanismo permite que seja possível aplicar a garantia da qualidade desde o início do projeto. Esse mecanismo é conhecido como revisão técnica formal.

- "Testar é tão fácil, que até a minha mãe testaria!", será isso verdade? Diante dos exemplos aqui apresentados podemos perceber que isso não é verdade. Além disso, os *software*s atuais são muito complexos e possuem contextos diferentes, ou seja, não é possível testarmos um *software* de missão crítica da mesma forma que um *software* de comércio eletrônico. Este cenário mostra o quanto é importante ter profissionais qualificados que entendam essas diferenças e saibam aplicá-la de forma que atendam as expectativas do negócio.

- "Não é necessário gastar tempo com testes, pois o programa já foi testado!", quem já não se deparou com uma frase dessas? Infelizmente, muitos programadores pensam dessa forma e aplicam testes sem critério algum. O intuito não é criticar os programadores e muito menos provar que 100% do que está descrito é verdadeiro, mas sim mostrar que a disciplina de testes de *software* existe e deve ser aplicada de forma profissional. Isso não significa que o desenvolvedor não deva testar, muito pelo contrário, existem testes específicos que seguem uma técnica que devem ser executados pelo desenvolvedor durante o desenvolvimento, os Testes Unitários.

- "Vamos pensar nos testes, após o desenvolvimento!" O planejamento dos testes é feito tardiamente, ou seja, após a fase de construção. Esta fase é super crítica, pois existe uma enorme pressão para entregar o projeto que muitas vezes pode estar atrasado e com custos elevados. Todos esses fatores comprometem a qualidade do produto a ser entregue ao cliente.

Qual é a realidade da disciplina de Testes de *Software*? Os mitos se tornaram

realidade e a nossa missão é mudar a visão do próprio profissional da área de Testes de *Software* e das demais áreas.

A seguir será apresentada uma compilação da discussão da primeira mesa redonda (Testar é tão fácil que até minha mãe testaria!) realizada no DFTestes. A discussão contou com 19 participantes, que deram as suas opiniões e compartilharam as suas experiências na área, gerando assim um total de 26 respostas.

Para facilitar a visualização e compreensão foi utilizado o Diagrama de Causa e Efeito, conhecido também como Diagrama de Ishikawa ou Fishbone. Este diagrama tem como objetivo expor as relações de um determinado efeito (conforme figura: Defeito no *Software* e Qualidade no *Software* Entregue) e as suas causas potenciais, estas por sua vez possuem um até dois níveis.

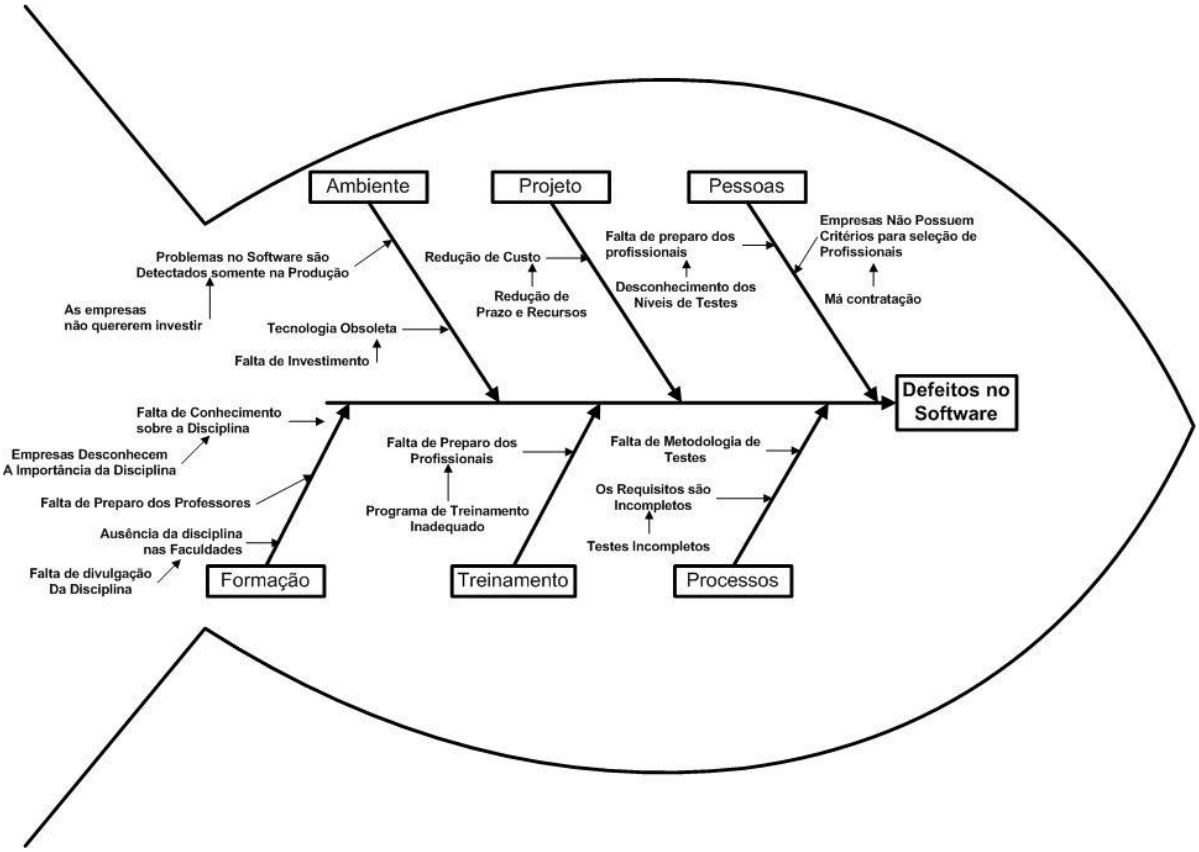


Figura 1.02 - Ishikawa - Qualidade no Software Entregue

Cases

A seguir serão apresentados alguns casos reais de problemas que ocorreram recentemente, devido à má qualidade do *software*, problemas de desempenho e usabilidade. Embora algumas empresas aqui citadas não tenham divulgado a perda financeira que tiveram, podemos concluir que essas falhas causaram prejuízos diretos e indiretos à organização, além de comprometer a imagem da organização e gerar impactos para futuros negócios.

Caso 1 - Fonte IDG NOW, Symantec compensou 50 mil vítimas de atualização com falha na China. Publicada em 25 de junho de 2007 às 09h18

A Symantec distribuiu uma atualização problemática para 50 mil computadores chineses, essa atualização classificou equivocadamente arquivos do sistema como *malwares' e os colocou em quarentena, afetando assim o funcionamento do computador, gerando uma onda de protestos na* web*.*

Para compensar os danos a companhia ofereceu uma extensão de 12 meses de licenças do Norton e uma cópia da ferramenta Norton Save & Restore 2.0, para os usuários em geral. Para os clientes corporativos, a empresa ofereceu licenças da Symantec Ghost Solution Suite.

Caso 2 - Fonte IDG NOW, Bolsa de Tóquio fechou mais cedo por pane em TI. Publicada em 18 de janeiro de 2006 às 11h07.

Aumento no volume de transações, resultante de um forte movimento de queda nos valores, levou o sistema da bolsa de Tókio (a segunda maior no mundo em valor de capitalização, depois de Nova York) ao limite de processamento. O índice Nikkei fechou em queda de 3% (15.341 pontos), a maior baixa em um ano.

Caso 3 – Fonte IDG NOW, Site do Submarino apresentou problemas de acessibilidade. Publicada em 25 de novembro de 2008 às 15h39.

O site de comércio eletrônico Submarino apresentou instabilidades, tornando-se inacessível aos internautas.

Os usuários que tentavam acessar a *home* do *site* se deparam com uma página incompleta com a seguinte mensagem “Página não encontrada”. Nesta página tinha um link, que às vezes encaminhava para a página correta, outras

não.

Concluindo, os problemas citados acima poderiam ser evitados, caso a empresa tivesse adotado um processo formal de garantia da qualidade.

A correta escolha da técnica de testes também é fundamental para o sucesso dos testes, por exemplo:

- 1º caso a aplicação de testes funcionais;
- 2º caso testes de desempenho, carga e volume.
- 3º caso testes de navegabilidade.

Além disso, podemos concluir a partir desses casos apresentados, que um teste mal sucedido pode significar um caminho aberto para outros tipos de problemas.

Conclusão

A disciplina de Testes de *Software* evoluiu muito durante esses anos, no entanto muitos profissionais da área sentem-se desvalorizados devido à falta de reconhecimento e visibilidade do papel deste profissional na indústria do *software*. A falta de conhecimento deste papel inicia-se:

- Nas faculdades de TI onde a ênfase é dada mais em linguagens de programação. Somente no último ano (quando tem) o aluno aprende algo sobre a disciplina de Testes de *Software*.

- Há mais de dez anos que tramitam no Congresso Nacional, diversos projetos que visam à regulamentação da área da TI. Contudo alguns desses projetos referem-se à regulamentação apenas da profissão de Analista de Sistemas, por exemplo . Em alguns desses projetos a disciplina de Testes de *Software* está inserida como uma atividade do desenvolvedor.

- As empresas preferem contratar desenvolvedores com várias habilidades, inclusive de garantia da qualidade.

A visão de um desenvolvedor é diferente de um Testador, geralmente desenvolvedores não gostam de testar e quando fazem não exercitam todas

as condições. Já o Testador, tem uma visão mais crítica e detalhista.

Concluindo, a disciplina de Testes de *Software* é essencial para o desenvolvimento, pois fornece evidências da confiabilidade de produtos e soluções, e garantia do atendimento aos requisitos de negócios. Além disso, não pode ser considerada como qualquer atividade.

Referências Bibliográficas

KENN, Peter G. W. Guia Gerencial para a tecnologia da informação: Conceitos essenciais e terminologia para empresas e gerentes. Rio de Janeiro: Campus, 1996.

CRAIG, Rick D.; JASKIEL, Stefan P. Systematic Software Testing. Artech House © 2002

RIOS, Emerson; MOREIRA, Filho. Testes de Software. Rio de Janeiro, Alta Books, 2003.

COMPUTERWORLD/EUA. As seis profissões da área de tecnologia mais valorizadas em 2010. Disponível em <http://idgnow.uol.com.br/carreira/2009/12/30/as-seis-profissoes-mais-valorizadas-em-2010/>. Acesso em 30/01/2010.

CAMPOS, Fabrício F. Testar é tão fácil, que até minha mãe testaria! Disponível em: <http://qualidadebr.wordpress.com/2009/11/08/testar-e-tao-facil-que-ate-a-minha-mae-testaria/>. Acessado em 30/01/2010.

HETZEL, Bill. The complete Guide to Software Testing. New York, John Wiley & Sons, 1998.

MYERS, G.J. The Art of Software Testing. 2 ed. New Jersey, John Wiley & Sons, 2004.

WIKIPEDIA. Malware. Disponível em <http://pt.wikipedia.org/wiki/Malware>. Acesso em 30/01/2010.

LEMON, Summer. Symantec Compensa 50 mil vítimas de atualização com falhas na China. Disponível em: <http://idgnow.uol.com.br/seguranca/2007/06/25/idgnoticia.2007-06-25.7174306115/>. Acesso em 30/01/2010

IDG Now!. Bolsa de Tóquio fecha mais cedo por pane em TI. Disponível em: . Acesso em 30/01/2010.

RODRIGUES, Nando. Site do Submarino apresenta problemas de acessibilidade. Disponível em: <http://idgnow.uol.com.br/seguranca/2008/11/25/site-do-submarino-esta-com-problemas-de-acessibilidade/>. Acesso em 30/01/2010.
Mayer, Roberto C. Regulamentação das profissões de TI: a quem interessa? Disponível em: <http://www.administradores.com.br/noticias/regulamentacao_das_profissoes_de_ti_a_quem_interessa/20340/>. Acesso em 30/01/2010.

Capítulo 2: Certificações, valem a pena?

Gustavo Nascimento

Revisor: Felipe da Silva

Introdução

A indústria de *software* vem apostando cada dia mais na qualidade de seus produtos. Um dos caminhos que as empresas estão seguindo é a profissionalização da atividade de teste. Ter uma equipe de testes já é uma tendência nas fábricas de *software*.

O intuito desta área, que pode ser considerada nova no mercado brasileiro, é detectar e corrigir falhas nos *software*s antes que esses sejam entregues aos clientes. No Brasil, os compradores de serviços de *software* (principalmente órgãos governamentais) e as empresas de *software* começam a valorizar esta área. Isso faz com que os profissionais especializados sejam mais valorizados e, neste caminho, as certificações começam a aparecer como um diferencial .

Com a área de testes em expansão, surgem questões relacionadas a certificações. Afinal, elas valem a pena?

O objetivo deste capítulo é apresentar ao profissional alguns aspectos que devem ser considerados na hora de traçar seu plano de carreira voltado para a área de testes e também esclarecer algumas dúvidas sobre o que a comunidade pensa sobre certificações em testes de *software*. O capítulo também traz um "guia" das principais certificações existentes atualmente e algumas orientações ao profissional que deseja seguir este caminho.

Porque certificar

Semelhantemente à outras áreas da engenharia de *software*, existem diversas certificações em Teste de *Software* que visam garantir, às empresas contratantes, que o profissional certificado possui determinado nível de conhecimento na área. Do outro lado, o profissional ganha visibilidade no mercado e aumenta suas chances de conseguir uma posição no mercado.
Embora as certificações comprovem certo nível de conhecimento e algumas exijam algum tempo de experiência na área de tecnologia da informação como pré-requisito, de uma forma geral, elas não conseguem medir a experiência do profissional na área de teste. Quando se fala em Testes de *Software*, é quase que um consenso (dentro da comunidade de testes DFTestes) que a experiência do profissional na área agrega muito valor e que esta experiência deve ser relevante em um processo de seleção ou em uma promoção.
O fato é que grande parte das empresas ainda enfrentam dificuldades em avaliar o conhecimento e experiência de um profissional da área e utilizam-se das certificações como instrumento de avaliação. Quando isso ocorre, é muito provável que um profissional experiente, que supostamente traria mais resultados positivos para a empresa contratante, seja descartado em um processo de seleção ou em uma promoção, antes mesmo de iniciá-lo, por não possuir uma certificação. Este é o principal motivo das críticas direcionadas às certificações em Teste de *Software*.
Em resumo, como veremos nas próximas seções, as diversas certificações em Testes de *Software* abordam, em níveis e formas diferentes, diversos conteúdos relacionados a testes. Isso faz com que uma certificação seja mais abrangente que outra e, por sua vez, mais valorizada. Algumas certificações ainda tem reconhecimento internacional e outras são reconhecidas apenas no Brasil. Portanto, é importante que o profissional que deseja se certificar saiba que a certificação não substituirá a experiência e, ainda, saiba escolher a certificação mais adequada a seus objetivos profissionais a curto, médio e longo prazo.

Certificações existentes na área de testes de software

Diversas são as certificações existentes em testes de *software*. Independente de qual certificação for escolhida, o profissional que deseja se certificar deve estar ciente de que só a certificação não é suficiente para garantir sucesso profissional.
As certificações são bem vindas, desde que acompanhadas de comprometimento, empenho e, com o passar do tempo, experiência. A experiência é o principal fator que pode resultar no sucesso profissional.
Nesta seção são apresentadas algumas das principais certificações em testes de *software* existentes atualmente.

QAI – Quality Assurance Institute

A QAI é a instituição que administra a *Software Certifications*, organização sem fins lucrativos, e é responsável pela criação e atualização da base de conhecimento de onde os exames QAI são gerados. A QAI oferece grupos distintos de certificação, que são voltadas para as áreas de Teste de *Software*, de garantia da qualidade e de gerenciamento de projetos .
As certificações do QAI foram projetadas para testar as habilidades de profissionais que atuam em uma das áreas cobertas pelo corpo comum de conhecimento relacionado ao assunto desejado, denominado CBOK (material base para estudo). Estas certificações são direcionadas aos profissionais que tenham experiência significativa e ampla para dominar os conceitos básicos desses assuntos .
De acordo com Fernando Scarazzato, as certificações da área de Testes de *Software* oferecidas pela QAI para o mercado brasileiro é a CSTE (*Certified Software Tester*):

- A CSTE é voltada para especialistas em Testes de*Software*, esta certificação demonstra o nível de conhecimento e habilidades que o profissional possui nos princípios, conceitos e práticas do Teste de *Software*. As pessoas com a habilitação CSTE conquistam o papel de conselheiros da alta gerência, auxiliam outros indivíduos na melhoria e evolução dos programas de Teste de *Software* da organização, motivam o pessoal responsável pelo Teste de *Software* a manter suas habilidades sempre atualizadas e são vistos como agentes de mudanças, alguém que possa mudar a cultura e os hábitos de trabalho dos indivíduos para fazer com que a qualidade no Teste de *Software* aconteça.

Para estar qualificado para se candidatar ao exame de certificação cada candidato deve ter trabalhado na área de serviços de informação, preferencialmente com Teste de *Software*, nos últimos 18 meses e possuir um dos quatro pré-requisitos abaixo:

- Formação de 4 anos em alguma instituição de nível universitário reconhecida e 2 anos de experiência na área de sistemas de informação; ou
- Formação de 3 anos em alguma instituição de nível universitário reconhecida e 3 anos de experiência na área de sistemas de informação; ou
- Formação de 2 anos em alguma instituição de nível universitário reconhecida e 4 anos de experiência na área de sistemas de informação; ou
- Seis anos de experiência na área de sistemas de informação.

A QAI oferece ainda para o mercado brasileiro as certificações CSQA - *Certified Software Quality Analyst* (voltada para especialistas em garantia da qualidade) e CSPM - C*ertified Software Project Manager* (voltada para especialistas em gerenciamento de projetos).

ISTQB – International Software Testing Qualifications Board

O ISTQB é um conselho internacional, sem fins lucrativos, formado em 2002, composto por representantes de mais de 40 países, denominados conselhos membros. Esse conselho, que cresce a cada ano, dedica-se à disciplina de Testes de *Software* e promove o profissionalismo na área através de um programa de certificações profissionais. No Brasil, o conselho membro é o BSTQB (formado em 2006).

Atualmente são mais de 120.000 certificados ISTQB no mundo e mais de 500 certificados no Brasil, sendo a maior certificadora em Testes de *Software* no Brasil e no mundo .

As certificações na área de testes oferecidas pelo ISTQB são:

- CTFL (*Certified Tester Foundation Level*): Esta certificação destina-se ao profissional de Testes de *Software* que tem como objetivo estar apto a comparar as práticas de testes entre diferentes países, a trocar conhecimentos na área de testes, a ter uma certificação reconhecida internacionalmente, a desenvolver um corpo comum internacional de compreensão e conhecimento sobre Teste de *Software* (este corpo de conhecimento é denominado *syllabus*, e é o material onde a certificação se baseia). O certificado CTFL não possui pré-requisitos obrigatórios, não expira e não precisa ser renovado.
- CTAL (*Certified Tester Advanced Level*): O objetivo desta certificação é assegurar, em níveis avançados, a compreensão em técnicas de teste, gestão e melhoria do processo de teste. Esta certificação destina-se ao especialista em Testes de *Software*, que tenha experiência em sua carreira de testes. Isto inclui pessoas em papéis como testador, analistas de testes, engenheiro de testes / arquiteto de teste, líder de teste / gerente de teste / coordenador de testes. Esta certificação também é adequada para quem deseja um entendimento mais profundo sobre Testes de *Software*, tais como Gerentes de Projeto, Gerentes de Qualidade, Gerentes de Desenvolvimento de *Software*, Analistas de Negócios, Diretores de TI e Gestores.

O CTAL possui como pré-requisitos que o candidato tenha a certificação CTFL e que tenha uma experiência mínima de três anos em Testes de *Software* ou

sistemas, desenvolvimento, garantia da qualidade ou áreas correlatas.

O CTAL é dividido em CTAL-TA (*Advanced Level Test Analyst*), CTAL-TM (*Advanced Level Test Manager*), CTAL-TTA (*Advanced Level Technical Test Analyst*).

ALATS – Associação Latino Americana de Testes de Software

A ALATS é uma instituição sem fins lucrativos, fundada em 2002, que tem o objetivo de reunir profissionais das áreas de teste e de qualidade de sistemas para servir de apoio a troca constante de informações. A participação na ALATS é aberta a todos os profissionais e empresas que tenham interesse nas áreas de teste e/ou qualidade de sistemas .

Para atender as exigências do mercado brasileiro, a ALATS criou a seguinte certificação:

- CBTS (Certificação Brasileira em Testes de *Software*): o objetivo desta certificação é estabelecer padrões de conhecimento na área de Testes de *Software*. Os exames de qualificação para a CBTS ocorrem duas vezes ao ano (maio e novembro) e tem validade de 3 anos. Ao longo desse período, o profissional deve acumular uma pontuação de 50 PDTS ou deve realizar novo exame para validar sua re-certificação.

ISEB – Information Systems Examination Board

O ISEB é uma instituição internacional criada para padronizar a competência e *performance* de profissionais da área de TI e suas diversas disciplinas.

As certificações ISEB são divididas em ISEB *Foundation Level* (aborda uma vasta introdução a disciplina desejada), ISEB *Practitioner Level* (aborda experiência prática para a disciplina desejada) e ISEB *Higher Level* (aborda um profundo conhecimento para a disciplina desejada e é voltada para especialistas ou gerentes) .

Para a disciplina específica de Testes de *Software*, a ISEB oferece as seguintes certificações:

ISEB Foundation Level

- *Foundation Certificate in Software Testing*: esta certificação é voltada para o profissional que tem interesse em entender os conceitos básicos em Testes de *Software*. Esta certificação também é acreditada pelo ISTQB, portanto, o profissional que obtém esta certificação recebe ambas as certificações: ISEB *Foundation Certificate in Software Testing* e ISTQB *Certified Tester Foundation Level*

- *Intermediate Certificate in Software Testing*: esta certificação é voltada para o profissional que está se especializando na área de Testes de *Software* e que deseja comprovar habilidades analíticas na área e que possui conhecimentos teórico e prático na área. Esta certificação é o próximo nível a ser obtido pelo profissional certificado no ISEB *Foundation Certificate in Software Testing*. Esta certificação é pré-requisito para a obtenção de certificações ISEB *Practitioner Level* voltadas para a área de testes

ISEB Practitioner Level

- *Practitioner Certificate in Test Analysis:* esta certificação é voltada para o profissional que deseja comprovar conhecimentos para se manter ativamente envolvido com análises da área de testes, em qualquer aspecto. Esta certificação estende os conhecimentos da certificação ISEB *Foundation Certificate in Software Testing* e cobre os aspectos técnicos da certificação ISEB I*ntermediate Certificate in Software Testing*. Um profissional com esta certificação possui alto nível de competência técnica na área.

- *ISEB Practitioner Certificate in Test Management*: esta certificação é voltada para o professional que trabalha com gerência de Testes de *Software*. Ela estende os conhecimentos da certificação *ISEB Foundation Certificate in Software Testing* e cobre os aspectos gerenciais da certificação *ISEB Intermediate Certificate in Software Testing*. Um profissional com esta certificação possui alto nível de competência gerencial na área.

IIST – International Institute of Software Testing

O IIST é uma instituição formada por um conselho, formado de especialistas e profissionais da área de testes, que tem o intuito de direcionar os esforços para desenvolver um plano de treinamentos baseado em certificações. O IIST acredita que as certificações comprovam o conhecimento do profissional naquele assunto específico.
As certificações IIST expiram em 3 anos e são divididas em:

- CSTP – *Certified Software Test Professional*: Voltado para o profissional iniciante na área de Testes de *Software* e que deseja se especializar na área. Os conhecimentos exigidos do profissional relaciona-se aos seguintes tópicos: princípios de testes, projeto de testes, gerenciamento de testes, execução e rastreamento de defeitos, definição, refinamento e verificação de requisitos, testes automatizados e testes estáticos.

- CTM - Certified Test Manager: voltado para o profissional com experiência mínima de 3 anos na área de Testes de *Software* e que pretende se tornar gerente de testes. Os conhecimentos exigidos do profissional relaciona-se aos seguintes tópicos: gerência do processo de testes, gerência de projetos de testes, melhoria e métricas relacionadas aos testes, gerenciamento organizacional da área de testes, gerência de riscos, estratégias e arquiteturas para automatização de testes, garantia da qualidade de *software*.

Qual o caminho a ser seguido por um profissional da área de testes de software?

O mercado de TI vem se tornando mais competitivo a cada dia. Participar de um processo seletivo com chances de ficar com a posição requer que o profissional tenha diferenciais em seu currículo. Neste contexto, as certificações podem representar este diferencial e, consequentemente, a contratação.
Acredita-se que a certificação tem maior importância para o profissional que está em início de carreira. As certificações asseguram a esses profissionais, apesar de algumas opiniões controversas, um determinado nível de conhecimento sobre a área e isso pode significar uma posição em uma empresa.

A certificação é um bom início para quem deseja iniciar na área de teste. No entanto, o profissional não deve ficar satisfeito com essa opção. Ele precisa

adquirir experiência e conhecimento na área se desejar alavancar a carreira. Existem algumas opções de plano de carreira na área de testes: testador, analistas de testes, engenheiro de testes / arquiteto de teste, líder de teste / gerente de teste / coordenador de testes. Este seria um possível caminho a ser percorrido por um profissional de testes, no entanto, este caminho não é o único e não deve ser considerado como uma sugestão deste autor. Os cargos e as remunerações variam entre as empresas e entre as diversas regiões do país e a carreira deve estar atrelada aos objetivos profissionais.
Esta variação pode ser observada através do resultado da pesquisa de cargos e salários organizada por Cristiano Caetano em 2007 . Nas Figura 1 e Figura 2 estão apresentados os resultados obtidos com a pesquisa, onde é possível observar a variação de remuneração praticada pelas empresas para diferentes cargos e regiões do país.

Função	Salário	
	Menor	Maior
Analista de Teste	800	5.500
Arquiteto de Teste	4.750	5.000
Auditor de Qualidade de Software (SQA)	1.060	8.000
Automatizador de Teste (Funcionais, Performance, etc)	1.450	2.500
Gerente de Teste	3.000	8.000
Líder de Teste	1.800	7.200
Testador	500	4.800
Outros	2.500	3.500

Figura 2.01 - Cargos e Salários

Estado	Função	Salário	
		Menor	Maior
AM	Analista de Teste	2.650	2.650
AM	Testador	1.400	1.400
CE	Analista de Teste	2.400	2.400
CE	Lider de Teste	5.000	5.000
DF	Analista de Teste	1.300	5.500
DF	Arquiteto de Teste	4.750	5.000
DF	Gerente de Teste	5.300	8.000
DF	Testador	2.700	2.700
GO	Analista de Teste	2.000	2.000
MG	Analista de Teste	1.200	2.000
MG	Auditor de Qualidade de Software (SQA)	3.300	3.300
MG	Gerente de Teste	3.000	3.000
PB	Analista de Teste	2.300	2.300
PB	Auditor de Qualidade de Software (SQA)	1.950	1.950
PE	Lider de Teste	7.200	7.200
PR	Analista de Teste	1.200	1.200
RJ	Analista de Teste	1.500	4.500
RJ	Arquiteto de Teste	5.000	5.000
RJ	Lider de Teste	6.500	6.500
RJ	Testador	1.500	1.500
RS	Analista de Teste	880	5.500
RS	Auditor de Qualidade de Software (SQA)	1.060	8.000
RS	Automatizador de Teste (Funcionais, Performance, etc)	1.450	2.500
RS	Gerente de Teste	3.900	6.300
RS	Lider de Teste	1.800	6.700
RS	Testador	500	4.800
RS	Outros	2.500	3.500
SC	Analista de Teste	1.700	1.700
SC	Testador	600	1.500
SP	Analista de Teste	800	5.000
SP	Auditor de Qualidade de Software (SQA)	1.500	1.700
SP	Gerente de Teste	8.000	8.000
SP	Líder de Teste	2.200	5.500
SP	Testador	3.300	3.300

Figura 2.02 - Cargos e salários distribuídos por estados

Para o profissional que deseja obter uma certificação e ingressar no mercado de Testes de *Software*, seguem algumas dicas de alguns profissionais da área:

- Escolher um bom curso preparatório. Herbert Maroni, diretor de desenvolvimento da 4sec Brasil e autor de livros sobre C#, VB.NET, AJAX e ASP.NET, faz um alerta: "geralmente o curso não é o suficiente". Maroni acredita que o candidato precisa se dedicar aos estudos independentemente do curso que tenha escolhido .

- Procurar o máximo de informações sobre a prova a ser realizada. Isso significa conhecer o conteúdo do exame, a duração, o custo e o local de realização .

- Adquirir e separar material de estudo e fazer simulados podem fazer a

diferença para o candidato. Recomenda-se fazer, no mínimo, dois simulados antes da prova e estudar as respostas das questões que se errou.

Além de conhecimentos e experiências na área de Testes de *Software*, pode ser importante que o profissional adquira conhecimentos e experiências em áreas correlatas. Isso dependerá da área e da carreira almejadas.
Para um profissional que deseja se especializar em automatização de testes, a aquisição de conhecimentos em linguagens de programação pode ser uma boa opção. O mesmo raciocínio vale para o profissional que deseja ocupar a posição de gerente de testes. Neste caso, uma certificação em gestão de projetos pode ser um diferencial.

Conclusão

A área de testes vem se expandindo no Brasil e acredita-se que a existência de uma área de testes nas fábricas de *software* já é uma tendência. Para tanto, espera-se que os profissionais da área acompanhem esta tendência e especializem-se cada dia mais.
Percebe-se ainda que há diferenças significativas de reconhecimento e remuneração nas diversas partes do país. Isso se deve ao fato da área ainda estar em expansão e das empresas ainda não estarem preparadas para acomodar esse tipo de profissional especializado. Pode-se dizer que esta é a realidade no Brasil de hoje.
No entanto, independentemente da região do país, as certificações em Teste de *Software* já se tornaram um assunto corriqueiro nas discussões que envolvem tais profissionais especializados.
Nestas horas, entra em ação sempre a mesma dúvida: Certificações, valem a pena?
A resposta é: Depende!
Várias perguntas devem ser respondidas pelo profissional antes que se chegue a alguma conclusão. Conforme vimos neste capítulo, a conclusão vai depender da expectativa e da experiência do profissional, da região do país onde ele atua (ou deseja atuar), da empresa contratante, dentre diversos outros fatores. Além disso, a escolha pela certificação também pode ser um fator determinante.

As diversas certificações existentes exigem diferentes níveis de conhecimento e dedicação do candidato e possuem diferentes níveis de reconhecimento

mercadológico. Sem dúvida, as certificações agregam valor ao candidato, porém, não espere que ela substitua a necessidade de experiência. A experiência ainda é mais relevante em uma possível contratação.

O profissional de TI deve ter, durante sua carreira, dois objetivos básicos: Conhecimento e Experiência.
Quando o profissional consegue perseguir estes dois objetivos com foco e determinação, as chances dele ser bem sucedido crescem significativamente. Se a certificação desejada estiver alinhada aos objetivos citados, as chances de sucesso aumentam ainda mais.

Referências Bibliográficas

Lima, L. À procura de testadores de software. Disponível em: <http://www.timaster.com.br/revista/materias/main_materia.asp?codigo=1560>. Acesso em: 19 fev. 2010.

Scarazzato, F. Entrevista com Fernando Scarazzato. Disponível em: <http://www.testexpert.com.br/?q=node/1086>. Acesso em 18 fev. 2010.

QAI Brasil - Quality Assurance Institute. Disponível em <http://www.qaibrasil.com.br/>. Acesso em 16 mar. 2010.Edit this page (if you have permission) |

BSTQB - Brasilian Software Testing Qualifications Board. Disponível em: <http://www.bstqb.org.br/>. Acesso em: 01 mar. 2010.

ALATS - Associação Latino-Americana de Testes de Software. Dispónível em: <http://alats.org.br>. Acesso em 01 mar. 2010.

THE CHARTERED INSTITUTE FOR IT. Disponível em <http://www.bcs.org/server.php?show=nav.5732>. Acesso em 01 mar. 2010.

Caetano, C. Cargos e Salários: Quanto ganha o profissional de teste e qualidade de software. Disponível em: <http://www.testexpert.com.br>.

Acesso em: 18 fev. 2010.

Ramos, T. O. Certificações: prepare-se para os exames das principais plataformas. Disponível em: <http://www.itweb.com.br>. Acesso em: 18 fev. 2010.

Ramos, T. O. Certificações: avaliar demanda do mercado ajuda na escolha. Disponível em: <http://www.itweb.com.br>. Acesso em: 18 fev. 2010.Ramos, T. O. Certificação pode fazer diferença no começo da carreira. Disponível em: <http://www.itweb.com.br>. Acesso em: 18 fev. 2010.

IIST - International Institute for Software Testing. Disponível em <http://www.testinginstitute.com>. Acesso em 01 mar. 2010.Google Docs -- Web word processing, presentations and spreadsheets.

Capítulo 3: O Testador também necessita saber programar?

Karine Birnfeld e Fabrício Ferrari de Campos

Revisora: Anna Carolina Rocha

Introdução

Uma visão comum do Teste de Software é de que ele consiste apenas na execução do sistema, e devido a isso, existe a dúvida se realmente é necessário que um testador de software possua um conhecimento aprofundado de programação para executar suas tarefas.

Quando procuramos na bibliografia pelas tarefas típicas de um profissional da área de teste de software, encontramos atividades que vão desde o planejamento dos testes, especificação, configuração do ambiente, criação da massa de dados, execução manual e automatizada até o registro dos resultados obtidos.

Na verdade o que acontece é que algumas empresas não possuem bem definidas as diferentes especialidades existentes na área de testes de software. Caso um mesmo profissional seja responsável pela execução de todas as atividades de testes, pode-se concluir que este profissional necessita ter bons conhecimentos de gestão de projetos, análise de requisitos, especificação de testes, banco de dados e inclusive bons conhecimentos de programação.

Especialidades da Área de Teste de Software

Este tópico tem como objetivo principal apresentar as diferentes divisões (especialidades) que existem na área de Teste de Software. Para cada especialidade são apresentadas as principais atividades bem como os conhecimentos necessários.

É possível encontrar cinco especialidades diferentes na área de Testes de Software, conforme segue:

- **Líder de Teste / Gerente de Teste / Coordenador de Teste:** Planejamento e controle de teste. Deve possuir conhecimento e experiência na área de teste de software, qualidade de software, gerenciamento de projetos e gestão de pessoas.
- **Engenheiro de Teste / Arquiteto de Teste:** Organização da infra-estrutura de teste. Necessário possuir conhecimento para instalar e operar o ambiente de teste, realizando a sua administração e suporte a rede.
- **Analista de Teste:** Modelagem e especificação de testes. Exigido conhecimento e experiência em técnicas de teste para realizar a especificação dos testes, além de conhecimento em banco de dados, sendo que também é de sua responsabilidade a preparação e aquisição da massa de dados de teste.
- **Automatizador de Teste:** Criação e execução de testes automatizados. Um automatizador de teste necessita de conhecimento básicos de testes, porém excelentes conhecimentos e experiência em programação com ferramentas de automação de teste.
- **Testador:** Execução dos casos de teste. Necessário conhecimentos para execução de testes e registro de resultados.

Conhecimentos de Programação

Este tópico visa tratar especificamente da necessidade de saber programar para cada especialista da área de testes.

Começando pelo líder de teste, este profissional deve ser capaz de auxiliar sua equipe na resolução de problemas. Para isso, é fortemente recomendável o conhecimento de linguagens de programação para auxiliar sua equipe. Além disso, este profissional precisa saber utilizar ferramentas de gerenciamento de projeto, bem como testwares e ainda ter boas habilidades na gestão de pessoas.

O engenheiro de teste necessita saber programar, pois ele precisa ser capaz de coletar informações no código fonte do programa para realizar as configurações e parametrizações necessárias no ambiente de teste, além disso, ele também necessita de conhecimentos de programação para entender a arquitetura do software e, com isso, conseguir disponibilizar um ambiente mais próximo possível do ambiente de produção.

Já o analista de testes não precisa ter excelentes conhecimentos de programação, mas necessita ter bons conhecimentos de lógica de programação para inferir sobre os testes que deverão ser realizados. Este conhecimento em lógica de programação auxiliará o analista de testes na identificação dos principais pontos em que uma falha pode ocorrer. Além disso, o analista de testes necessita saber analisar os requisitos do cliente, visando extrair o máximo de cenários de teste possíveis.

Para o automatizador de teste, sem sombra de dúvidas, é exigido excelente conhecimento em programação, visto que sua atividade gira em torno da criação e execução de scripts de teste em diferentes linguagens de programação, dependendo da ferramenta de automação utilizada.

Referente ao testador, cujas atribuições incluem a execução dos testes manuais e registro dos resultados obtidos, não é obrigatório possuir conhecimentos de programação. Este profissional pode tanto ser um profissional da TI como um profissional que tenha bons conhecimentos do negócio a ser validado no sistema. Porém, isso depende muito do nível em que o teste é especificado e do tipo de projeto. Caso existam profissionais que não sejam da área da TI, ou seja, não possuam conhecimentos em lógica de programação, banco de dados e sistemas operacionais, por exemplo, a especificação dos testes a serem executados deverá chegar ao nível de detalhe

dos procedimentos de teste, fornecendo a este testador cada passo que ele deverá realizar para executar os testes.

Como é trabalhoso realizar a especificação de teste até o nível dos procedimentos, é mais fácil para as empresas contratarem profissionais que já tenham essa base de conhecimento de TI, podendo ser utilizada uma especificação de testes num nível mais alto, visto que um profissional da área de TI saberá realizar uma validação em banco de dados ou realizar operações em diferentes sistemas operacionais.

Programação a serviço do Teste de Software

Como vimos, o conhecimento de programação se fará necessário dependendo da especialidade que o profissional pretende seguir, sendo que em algumas delas conhecimentos de programação são essenciais. Além disso, há outros fatores que poderão influenciar na necessidade do profissional de teste saber ou não programar, dentre os quais os principais são:

- Cultura da empresa: dependendo dela, pode haver uma forte tendência a execução de testes manuais, o que não demandaria conhecimentos de programação; por outro lado, poderia haver uma forte tendência na automação dos testes, onde aí sim, se faz necessário possuir conhecimentos de programação;
- Metodologia adotada: hoje em dia, com o crescimento das metodologias ágeis, acaba se tornando importante/diferencial o profissional de teste saber programar, se ele desejar atuar numa equipe ágil, uma vez que metodologias ágeis, incentivam a automação dos testes.

Além disso, o conhecimento de programação pode agregar muito ao profissional e tornar o seu trabalho mais efetivo, tanto no planejamento de um teste, uma vez que se ele tiver conhecimentos de programação, poderá ter *insights* mais específicos (por exemplo, fazer "injeções SQL"), como também poderá utilizar esse conhecimento para automatizar testes de regressão ou criar massa de dados, e assim poupar o seu tempo com tarefas repetitivas.Ou seja, mesmo se o profissional não for usar na prática o seu conhecimento sobre programação, ele poderá ser muito útil para a execução das tarefas habituais de um profissional de Teste de Software, uma vez que ele terá uma percepção mais afinada, sob uma perspectiva técnica, quanto ao comportamento do sistema. Deste modo, poderá fazer asserções mais pertinentes e elaborar casos de testes mais complexos.Quando falamos em automação na área de Teste de Software, dificilmente poderemos abdicar de

profissionais com conhecimento de programação, por mais que existam ferramentas que sejam fáceis de serem utilizadas para automatizar os testes, como por exemplo o Selenium IDE. Isso porque, geralmente, há certos cenários de testes mais complexos, que demandarão um uso mais avançado da ferramenta, o que muitas vezes, envolve o desenvolvimento de algum script.Dentre as vantagens que podemos ter quando sabemos programar, citadas pelo Shmuel Gershon, estão:

- O testador é capaz de entender os tipos de problemas que o aplicativo pode apresentar, pois o testador pode montar seu modelo mental de como o software funciona por dentro e testar os limites desse modelo.
- O testador é capaz de fazer testes automáticos quando necessário. Talvez até mais importante, a criação de pequenas ferramentas que facilitam a configuração rápida do sistema, ou que recolhem dados para relatórios podem fazer do testador um jogador importante com influência na equipe toda.
- Facilita a comunicação com os programadores.

Conclusão

O Teste de Software já passou por grandes evoluções e tende a continuar evoluindo. Caso o profissional de testes não acompanhe esta evolução, no sentido de se interar de novas tecnologias e aprender a arte da programação, será sempre apenas um testador de software, visto que para todas as demais especialidades é exigido ou fortemente recomendado este conhecimento.

Referências Bibliográficas

Spillner, A., Linz T., Schaefer H. (2007) Software Testing Foundations. Santa Barbara, Rocky Nook, 2nd Edition.

Capítulo 4: Testar sem documentação é possível?

Sarah Pimentel

Revisora: Keite Moraes

Introdução

De acordo com o RUP, caso de teste "é a definição (geralmente formal) de um conjunto específico de *inputs* de teste, condições de execução e resultados esperados, identificados com a finalidade de avaliar um determinado aspecto de um item de Teste-alvo". Duas disciplinas da Engenharia de *Software* provêem informações para criação dos casos de teste. São elas: Engenharia de Requisitos e Projeto.

De acordo com o IEEE, a Engenharia de Requisitos é o processo de aquisição, refinamento e verificação das necessidades do cliente. Dentro das diversas metodologias de desenvolvimento há maneiras distintas de documentar requisitos. No RUP, o principal documento gerado, que é utilizado pela equipe de teste, são os Modelos de Caso de Uso; já nas metodologias ágeis, um documento mais comum é o conjunto de *User Stories*. Na prática, muitas equipes não têm nem os Casos de Uso e nem as *User Stories*, mas sim, descrições em alto nível dos requisitos, sejam elas por escrito ou não.

A disciplina de Projeto, por sua vez gera documentos complementares às especificações de Requisitos. Nessa etapa, a estrutura interna do *software* é descrita através de diagramas. A utilização da linguagem UML é bastante recomendada na geração desses diagramas. Exemplos de diagramas de projeto são Diagramas de Estados, Diagramas de Seqüência e Diagrama de Atividades, entre outros.

Essa documentação serve como apoio para a elaboração dos testes. Todas elas são de grande valia para o analista no seu projeto de casos de teste. Mas e se o analista não as tiver? É possível, mesmo assim, testar? Essa é a pergunta tema da Nona Mesa Redonda do DFTestes.

Esse tema nos fez refletir sobre a necessidade da documentação para o teste. Primeiramente não vamos confundir necessidade com importância. Denotativamente o termo necessidade significa uma obrigação imprescindível, enquanto que importância ou relevância significa valor.

As participações nessa mesa redonda são freqüentemente utilizadas no texto que segue para expressar a opinião da comunidade sobre o assunto.

Testes Exploratórios

É possível sim testar sem *scripts* e isso já não se pode chamar de novidade. O termo "testes exploratórios" foi citado pela primeira vez por Cem Kaner em seu livro Testing Computer Software (1999).Segundo James Bach, testes exploratórios são simultaneamente um aprendizado, um projeto e uma execução de teste. Através da profunda exploração das habilidades de ouvir, ler, pensar e reportar eficazmente, os testes exploratórios podem trazer informações muito importantes de maneira tão ou mais produtiva que os testes baseados em casos de teste. A falta de um *script* a ser seguido tende a potencializar essas habilidades.

Escolas de Teste

Sobre a documentação em que iremos embasar a criação dos casos de teste, o Jorge Diz disse: *"não coloquemos todas as nossas fichas na documentação do sistema. Com certeza, é possível testar sem documentação formal: pode ser mais ou menos eficaz, dependendo do contexto. E sempre devemos lembrar que o documento mais atualizado é sempre o próprio sistema sendo testado."*

O contexto! Esse é o grande segredo dessa discussão. A eficácia do seu teste pode ou não ser impactada pela falta de documentação considerando diversos cenários. É papel do analista de testes sinalizar até que ponto ele consegue aferir a qualidade do sistema com as ferramentas (documentação, especialista, etc.) que lhe foram dadas.

Dentro das Escolas de Teste, veremos que de acordo com as características de cada escola (um contexto), o tema "testar sem documentação é possível?" é tratado de maneiras diferentes.De acordo com Pettichord (2007), uma escola é definida por afinidades intelectuais, interação social e objetivos comuns. São compostas por uma hierarquia de valores, técnicas, padrões de crítica, instituições organizadas e vocabulário comum.

Existem cinco escolas de teste e suas visões são:

- Escola Analítica: o teste deve ser rigoroso e técnico, com base na academia
- Escola Padrão: o teste é uma maneira de medir o progresso com ênfase no custo e padrões repetíveis
- Escola da Qualidade: enfatiza o processo e policia os desenvolvedores
- Escola baseada em Contexto: enfatiza as pessoas, procuram *bugs* com os quais os clientes se importam mais
- Escola Ágil: usa o teste para provar que o desenvolvimento está completo. Enfatiza o teste automatizado

Sobre os testes sem documentação, são a favor:

- Escola baseada em Contexto: "Faça o que puder para ser útil. Faça perguntas se necessário. Desencave especificações escondidas."
- Escola Ágil: "Conversas são mais importantes que documentação"

E são contra:

- Escola Analítica: "É impossível"
- Escola Padrão: "Algum tipo de documentação é necessária"
- Escola da Qualidade: "Force os desenvolvedores a seguir o processo"

Como cada escola vive um contexto e analisa o teste de acordo com os seus

cenários, cada uma expressa uma visão diferente da possibilidade de testar sem documentação. Na visão geral dos participantes da discussão, testar sem documentação não é impossível, mas essa abordagem também não foi defendida como a melhor prática. A comunidade do DFTestes parece ter uma tendência muito próxima a da Escola baseada em Contexto, entendendo que não devemos olhar as nossas dificuldades de documentação como muros intransponíveis, mas como empecilhos que devem ser vencidos.

Reflexões da Mesa Redonda

"À medida que os sistemas se tornam mais complexos, os riscos se tornam maiores, chegando ao ponto no qual ninguém conhece o que há no sistema. Isto é ruim não só para o teste, mas também para todas as fases anteriores do desenvolvimento.

Falando de casos de uso ou similares, um dos problemas comuns de quem não tem essa documentação é o de não saber o que testar. Recentemente fiz uma análise de cobertura de código em um programa de pedidos e concluí que ao incluir um pedido neste sistema com o máximo de opções que consegui informar, não cobri nem 20% do código. Se você não tem nada para consultar, como testar os outros 80% de forma eficiente? Agora imaginem se este programa estivesse 60% documentado com casos de uso e casos de teste. Já seria uma garantia bem maior. Outro problema é o de não saber o que alterar. Recentemente em um projeto foi esquecido de alterar um programa, e só esse programa gerou mais estresse que todas as outras alterações juntas porque estava dando problema lá no cliente. Sem documentação, não há rastreabilidade."

Na colocação do Ismael conclui-se que:

- Falta de documentação pode prejudicar a cobertura dos testes
- Falta de documentação implica em falta de rastreabilidade e aumento dos riscos nas manutenções do sistema

A visão do Marcelo Andrade complementa o primeiro cenário do Ismael:

"Claro que não dá para testar tirando conclusões sobre regras de negócio da própria cabeça do testador. Neste caso, se a informação está disponível (com o cliente ou com quem quer que seja) é ela que deve ser buscada."

Então, na falta de uma documentação, caso haja alguma outra fonte de conhecimento sobre os requisitos, ela viabiliza os testes e também aumenta a sua cobertura. Essa opinião também foi expressa pela Andrea Cruz quando ela fala que *"testar sem documentação é possível, porém podem existir em determinados sistemas requisitos implícitos importantes que podem não ser cobertos pelo teste."*

Existe sem dúvida um risco na falta de documentação. Para ser bem sucedido, o Analista de Teste precisa contar com alguma outra fonte de informação, ou a sua cobertura poderá ser seriamente prejudicada.

Segundo Daniel Goettenauer, *"o beneficio da documentação só é percebido atualmente em grandes projetos, onde os testes de regressão são constantes e existem muitas mudanças. dependendo do tamanho do projeto de teste a documentação poderia ser tratada de forma mais simples (documentos básicos) ou complexa (todos os documentos). Dessa forma não teríamos projetos desenvolvidos em 16 horas e testados em 72 horas por conta do volume de documentação."*

Ter algum nível de documentação é mesmo importante, entretanto o que define sua necessidade? Uma documentação necessária é aquela que é consultada e mantida, que serve de apoio para o time ou, não se encaixando em nenhum desses objetivos, ela se torna necessária apenas se for uma requisição do cliente. Jeffries, 2001 expõe em seu blog esse mesmo ponto quando fala que *"você pode precisar de um UML bem formatado para o seu projeto, ou você pode precisar imprimir o Javadoc quando distribuir o seu código para outros, ou você pode precisar documentar os requisitos para o gerenciamento ou como parte de um contrato. Se e quando você realmente precisar dessas documentações, você deve realmente tê-las."* Se elas não forem realmente necessárias apenas demandam tempo da equipe e nem sempre são mantidas adequadamente.

Um sem número de documentos que são criados no início do projeto e não sofrem mais atualizações é uma realidade em muitos projetos. De acordo com o Shmuel, *"uma documentação desatualizada é menos útil, mas ainda pode*

ser usada da mesma maneira. Mas essa ajuda não é um pré-requisito necessário, e podemos testar mesmo sem tê-la. Por meio de abstrações e inferências, é possível aprender sobre o programa de várias outras maneiras, durante os testes, durante conversas, durante as discussões sobre bugs etc."

Essa opinião está alinhada com uma pesquisa do IEEE, 2003 em que entrevistas com engenheiros de *software* revelaram as seguintes opiniões sobre documentação:

- Documento de arquitetura e outras documentações abstratas são freqüentemente válidos ou ao menos provêem um guia histórico que pode ser útil para manutenção.
- Documentações de todos os tipos freqüentemente estão desatualizadas
- Uma fração considerável da documentação não é confiável. "

O Felipe da Silva trouxe um ponto diferente sobre a necessidade e assinalou outro uso dado à documentação fora o projeto de testes que é o de contrato. É comum utilizarmos uma documentação e a aprovação do cliente como um contrato ou parte dele na definição de escopo e base para cronograma. Segundo Felipe, *"na maioria dos casos além de brigarmos para querer ter um sistema testável por qualquer um, em determinados processos de engenharia de software também brigamos porque queremos ter um documento como "defesa" contra a insatisfação do cliente. É aquele problema que se você não documenta, o cliente não assina, se ele não assina, não existe um registro que ele havia dito que era aquilo que ele queria, correndo o risco do cliente reclamar e ter melhorias no sistema sem pagar por elas nem ajustar cronograma e etc."*

Conclusão

As opiniões expostas na Mesa Redonda levaram a um entendimento de que a documentação pode não ser necessária, mas é importante. O tema é bastante abrangente e a cada participação na lista foi possível idealizar novos cenários para responder a mesma pergunta. E você? A que conclusão chegou após ler o que discutimos em mais uma edição da Mesa Redonda do DFTestes?

Referências Bibliográficas

Much Ado About Nothing: DocumentationRon Jeffries 2001

http://www.xprogramming.com/xpmag/FussAboutDocumentation

Schools of Software Testing Bret Pettichord, 2007
http://www.io.com/~wazmo/papers/four_schools.pdf

How Software Engineers Use Documentation: The State of the PracticeTimothy C. Lethbridge, Janice Singer, Andrew Forward, IEEE Computer Society 2003

Exploratory Testing Explained James Bach, 2003
http://www.satisfice.com/articles/et-article.pdf

Capítulo 5: Quando documentar não é preciso?

Edwagney Luz

Revisora: Keite Moraes

Introdução

Atualmente vivemos em um mundo onde os negócios possuem uma grande dependência pela TI (Tecnologia da Informação) e à medida que esse fator aumenta, também cresce a produção de *software* e complexidade dos sistemas.
Um exemplo é quando um risco operacional da TI se propaga para os negócios, deixando de ser um risco da TI, passando a ser um risco do negócio, ou seja, se uma falha ocorrer nos sistemas, consequentemente ocorrerá uma falha nos negócios. Desta forma, torna-se fundamental o desenvolvimento de software com mais qualidade e confiabilidade. Neste capítulo será abordado o porquê se devem ter profissionais qualificados na elaboração e execução de testes.

Mesa Redonda

Durante a discussão desse tema, diversos integrantes do grupo DFTestes colocaram suas opiniões e suas experiências, seja obtido de suas empresas ou em participações em projetos. Dessa discussão chegamos a alguns pontos interessantes, que trataremos no decorrer deste capítulo.

Para que serve a documentação?

No exposto por (Rezende, 2005) e descrito na introdução deste capítulo, chama a atenção o trecho onde diz que a documentação serve para orientar e treinar o cliente na operação do sistema. Esse é um enfoque interessante, visto que sem a documentação seria um tanto quando complicado passar essa orientação e treinamento ao cliente. Pensando em Teste de *Software*, para que serve então a documentação?
"Ajuda o entendimento do sistema, quando você cria a documentação de Teste de *Software*, você entende melhor o sistema e ainda abstrai as informações para o mundo do Teste de *Software*" (Fabrício Ferrari)
Vocês podem observar que, em outras palavras usamos a documentação de teste com o mesmo objetivo que levamos ao cliente. Entretanto, isso vai além. Ela não pode e não deve ser considerada apenas um guia para orientação e treinamento.

Ela deve ser tratada como sendo um referencial para o projeto de s*oftware*, como o compartilhamento de informações entre os diversos times que compõe um projeto. É a documentação que tem o poder de dar consistência e credibilidade ao projeto. (Ambler, 2002), expõe um raciocínio interessante. O *software* é o produto principal de um projeto, e afirma que ter esse*software* funcionando é o principal objetivo desse projeto e a produção de documentos ou artefatos gerenciais irrelevantes não é o foco. A equipe de desenvolvimento desenvolve *softwares* e não documentações.
Por outro lado ela também tem a responsabilidade de possibilitar a execução de um possível próximo trabalho. Ele chama de objetivo secundário da equipe de desenvolvimento. Ele afirma que para possibilitar que isso aconteça, a equipe não deve querer apenas desenvolver um *software* de qualidade, mas também documentação suficiente para que as pessoas que jogarão a próxima partida possam ser tão ou mais eficientes do que a equipe anterior, pois transferem habilidades para a equipe posterior, além de motivar o pessoal já existente a permanecer na equipe e desenvolver uma próxima versão.
Usando esse raciocínio, podemos perfeitamente transferi-lo para a equipe de teste, que em nada difere de uma equipe de desenvolvimento. Se uma equipe de desenvolvimento tem como principal responsabilidade desenvolver *software*, de forma análoga uma equipe de teste tem como principal atividade desenvolver testes, e como atividade secundária, não basta apenas desenvolver testes com qualidade e que encontrem o maior número de

defeitos, é necessário também documentar esses testes para que outra equipe possa, no futuro, garantir a mesma qualidade que foi garantida na primeira execução de teste no projeto.
Outro ponto levantado na discussão da mesa redonda é sobre a solicitação do cliente, onde tudo depende da solicitação dele. Se ele exige documentação de Teste de *Software*, a equipe tem o conhecimento sobre o sistema e o prazo para teste é curto, documentações são desnecessárias, e o teste a ser realizado seriam apenas usando técnicas de teste exploratório. O que é um assunto bem controverso, visto que a equipe de teste é a que deve ou deveria zelar pela qualidade do produto e este inclui o *software* e sua documentação. Assumir esse tipo de situação seria o extremo do extremo. Sugere-se fazer um estudo sobre a viabilidade de documentação, documentar por documentar, pode tornar-se perda de tempo, e consequentemente ocasionará perda de dinheiro.
Concluímos que documentar é preciso, porém com muito bom senso e que a mesma tenha uma finalidade, caso contrário estaríamos perdendo um precioso tempo que poderia ser utilizado para a melhoria da qualidade do produto. Mas quando é preciso documentar? Abaixo listamos alguns pontos levantados durante a discussão:

- Quando existir a necessidade de arquivar evidências;
- Quando existir a necessidade de arquivar e disseminar conhecimentos (gestão do conhecimento);
- Quando alguma cláusula contratual indicar que os níveis de documentação devem ser entregues;
- Quando existe uma alta rotatividade da equipe.

E quando documentar não é preciso?

- Quando o documento de fato não seja de nenhuma utilidade futura;
- Quando o documento não for mantido ou atualizado;
- Quando o documento não for compreendido pelas pessoas para quem ele foi escrito;
- Quando a equipe de desenvolvimento for gerar documentos difíceis de entender, que não atendam o propósito, ou que não sejam efetivamente utilizados;

- Quando a documentação gerada não é considerada importante, que não irá agregar valor e não for compartilhada com os *stakeholders* do projeto;
- Se a documentação criada não for gerar uma base histórica para auxiliar em pesquisas de projetos futuros.

Percebemos que nesse caso, todos os itens são uma consequência do item um, portanto, a primeira regra de utilidade é a primeira coisa que deve ser levantada no processo de decisão de se elaborar ou não um determinado documento.

Tornando a documentação de teste eficaz

Todos concordam que esse é um tremendo de um desafio, visto que antes é necessário convencer as equipes da necessidade de se documentar. Mas podemos simplificar essa tarefa, como algumas pessoas propuseram conforme descrito abaixo:

- Definir qual o objetivo do documento; a que se propõe; para qual finalidade ele deverá ser elaborado;
- Otimizar o documento eliminando tópicos irrelevantes ao projeto;
- Observando o grau de manutenibilidade do documento. Se o mesmo for difícil de manter, algo está errado e o mesmo deve ser revisto. Nesse caso volte ao item um e observe sua finalidade;
- Busca por ferramentas que agilize a produção do documento;
- Definição de local de fácil armazenamento e acesso por parte da equipe de teste e projeto;
- Verificar e garantir que a documentação esteja seguindo a padronização exigida;
- Identificar qual o público receptor da documentação para que a mesma seja elaborada com a linguagem desse público.

Como vimos no item anterior, devemos apenas gerar uma documentação

quando temos certeza de que será lida. Elaborar documentos apenas com o intuito de documentar detalhadamente determinada atividade, mas que de nada vai adiantar para o projeto ou projetos futuros, estaremos além da perda de tempo, consumindo recursos de armazenamento que poderão fazer falta no futuro e tempo desnecessário de profissionais para simplesmente organizar e manter as informações em repositório. Elaborar documentação deve antes passar pelo crivo de identificar para qual necessidade vamos armazenar tal informação.

Quais documentações devem ser geradas?

De acordo com o Guia de Implementação – Parte 10: Implementação do MR-MPS em organizações do tipo Fábrica de Teste (SOFTEX, 2009), as informações que devem ser armazenadas para qualquer projeto são: relatórios informais; estudos e análises; atas de reuniões; lições aprendidas; artefatos gerados; itens de ação; e indicadores. As informações podem estar armazenadas nos diversos meios de armazenamento como papeis impressos, fotografias, desenhos, eletrônicos e multimídia.
O guia também prega que as informações relevantes devem ser observadas primariamente, ou seja, devemos sim documentar, porém o que for mais relevante ao projeto. Vimos que essa informação é compartilhada por todas as fontes citadas nesse capítulo e com base nisso chegamos ao seguinte questionamento. Em teste, quais documentações então devem ser geradas para o armazenamento das informações?
De acordo com o discutido pelos integrantes da mesa redonda, várias documentações são utilizadas, dependendo da forma em que a empresa ou o projeto se porta em relação ao teste. Entretanto, colhendo dados durante a discussão, encontramos comumente entre todos os integrantes os seguintes documentos, que podemos dizer que podem seguramente serem considerados como sendo documentações que guardam informações relevantes ao teste de *software*. São eles:

- Plano de Teste – descreve o escopo, estratégia, ambientes, riscos, recursos, referentes ao projeto de teste;

- Especificação de Teste – descreve os cenários, casos, roteiros e massa de dados para teste;

- Relatórios Periódicos de Teste – descreve os resultados da execução do

teste contendo suas evidências e toda a gestão e histórico de defeitos;

- Relatório Final de Teste – descreve de forma sucinta todo o histórico da execução do teste contendo todos os casos de teste executados, percentual de casos com defeito, sua severidade e etc.

Esse conjunto de artefatos em geral é aplicado em projetos de teste que possui um nível de criticidade alto, pois necessitam de maior controle e armazenamento de informações. E nesse caso descrever de forma sucinta a estratégia, riscos, ambientes, recursos, relatórios periódicos e detalhados de execução, são informações extremamente relevantes, não só para o projeto atual, mas principalmente projetos futuros.
Para projetos com menor criticidade, somente os relatórios de testes são exigidos, principalmente para evidenciar os testes realizados. E na maioria dos casos pelos próprios desenvolvedores. Mas tudo isso faz parte de uma estratégia de melhoria dos processos, que devem visar sempre alcançar níveis mais elevados de maturidade.
Sugere-se que a documentação deve ser mais simples possível, mas que permita o registro das informações que a organização considera imprescindíveis. E nessa análise cada organização deve definir sua documentação básica, mas deve também explicitar aos seus colaboradores os objetivos a serem alcançados e as estratégias utilizadas, para que consiga da equipe o comprometimento e a motivação necessários.

Outras reflexões

No decorrer da discussão, surgiram algumas reflexões no sentido de melhoria do processo de teste, que não poderiam ficar de fora dessa coletânea de informações.

Uma dessas reflexões que chamou a atenção dos componentes da mesa foi relativa ao tratamento dos requisitos. Ao invés de criar casos de teste, basear os testes nos requisitos ou "*user stories*". Essa abordagem gerou controvérsia entre os integrantes no sentido de acreditarem não ser uma boa ideia a criação única de documento de requisitos direcionado para teste. Alguns acreditam que a criação dos casos de teste continua sendo necessário, mesmo que os requisitos sejam elaborados tendo a visão do teste. Melhorar a elaboração e gerência dos requisitos é uma tarefa a ser considerada sempre, mas não elaborar os casos de teste pode ser um risco para a qualidade do projeto, visto que o documento de requisitos e casos de teste possui diferentes significados.
Outra reflexão interessante é se, ao invés, de requisitos, documentarmos apenas os testes? Colocarmos as solicitações do cliente não como requisitos, mas como cenários de teste. Será que perderíamos algo?
De acordo com os praticantes do BDD (*Behaviour Driven Development*), costumam dizer que existe a possibilidade de utilizá-lo para detalhar os requisitos funcionais e nesse caso já estaríamos documentando e escrevendo um teste. É possível que não percamos nenhuma informação no caminho, entretanto corremos o risco de ter que detalhar muito cedo as informações de um projeto. Nesse momento a equipe ainda não está madura o suficiente para realizar esse detalhamento e em alguns casos nem mesmo o cliente tem esse nível de maturidade no projeto.
Em relação à abordagem TDD (*Test-Driven Development*), como ficaria a documentação de teste? Seria dispensável ou seria apenas em nível de Teste de Unidade? Possivelmente os documentos de teste continuariam sendo gerados, principalmente se no projeto existirem papéis específicos para a atividade de teste. Continuariam sendo desenvolvidos testes manuais e que em determinados projetos terão a necessidade da elaboração de documentações tradicionais em outras abordagens.

Conclusão

No decorrer da discussão dessa mesa redonda, pudemos perceber vários pontos de vista e várias abordagens diferentes quanto à elaboração e tratamento da documentação a ser elaborada em um projeto. Alguns optam pela elaboração vários tipos de documentos, outros optam por uma quantidade menor. Isso depende do tamanho e da criticidade projeto. Quando maior e mais crítico, maior nível de detalhamento é exigido. Entretanto o ponto em que todos concordam e isso vem ao encontro do que pregam as referências usadas como base teórica deste texto, foi que a documentação só é necessária quando existirem informações relevantes e que realmente sejam importantes de serem armazenadas. Caso contrário não faz muito sentido a elaboração de documentação. Analisar a informação que se deseja armazenar e o objetivo pelo qual se destina, é o primeiro passo e a primeira reflexão a ser feita antes de se iniciar a elaboração de qualquer documentação.

Referências Bibliográficas

Ambler, S. W. (2002). *Modelagem Ágil: Praticas Eficazes para a Programação eXtrema e o Processo Unificado.* Porto Alegre - RS: Bookman.

Rezende, D. A. (2005). Projeto de Documentação. In: D. A. Rezende, *Engenharia de Software e Sistemas de Informação* (p. 275). Rio de Janeiro: Brasport.

SOFTEX. (Outubro de 2009). MPS.BR - Melhoria de Processo do Software Brasileiro - Implementação do MR-MPS em organizações do tipo Fábrica de Teste.

Capítulo 6: Utilização de Processos Ágeis no Teste de Software (SCRUM, XP, TDD...)

Maíra M. Dutra e Patrícia Silva Corrêa

Revisora: Patrícia Silva Corrêa

Processos ágeis? Que isso mesmo?

Processo, ou metodologia de desenvolvimento de software pode ser enxergado como um conjunto de atividades que auxiliam a produção de software. Essas atividades culminam em um produto que reflete a forma como todo processo foi conduzido.

Muitas organizações desenvolvem software sem usar nenhum processo. Geralmente isso ocorre porque os processos tradicionais não são adequados as suas realidades. Em particular, as pequenas e médias organizações não possuem recursos suficientes para adotar o uso de metodologias "pesadas", e, por essa razão, normalmente não utilizam nenhum processo. O resultado dessa falta de sistematização na produção de software é a baixa qualidade do produto final, além de dificultar a entrega do software nos prazos e cursos predefinidos e inviabilizar a futura evolução do software.

E na busca por soluções para esses problemas, atualmente existe um interesse cada vez maior nos métodos ágeis de desenvolvimento de software. "O termo metodologias ágeis tornou-se popular em 2001, quando 17 especialistas em processo de desenvolvimento de software, representando os métodos (XP), Scrum, DSMD, Crystal dentre outros, estabeleceram princípios comuns compartilhados por todos esses métodos. O resultado foi a criação da Aliança Ágil e o estabelecimento do Manifesto Ágil."

Uma das características dos processos ágeis é que eles não são centrados nos artefatos (documentação). “Valorizando mais software em funcionamento, do que uma documentação abrangente”, acaba-se incentivando a produção de documentação apenas em momentos que se faz necessário, ou seja, deixa-se em aberto a quantidade e artefatos que serão produzidos, para que a pessoa decida, de acordo com a sua realidade. E orientam, que deve-se valorizar mais o software em funcionamento, do que a produção de uma documentação extensa.

Outra forte característica dos processosságeis é que são adequadas para situações, em que a mudança de requisitos é frequente. Para ser realmente considerada ágil, a metodologia deve aceitar a mudança, ao invés de tentar prever o futuro no começo do projeto. Além disso, reconhecer que as pessoas são os principais condutores e responsáveis pelo sucesso do projeto.

Mas e o Teste de Software, se encaixou nesses novos processos?

A discussão da sexta Mesa Redonda DFTestes foi sobre a “Utilização de Processos Ágeis no Teste de Software (SCRUM, XP, TDD…)”, discussão que gerou13 respostas, e teve oito participantes: Fabrício Ferrari, Renata Eliza, Marcelo Andrade. Juliana Kryszczun, Felipe Silva, Ronaldo Cruz, Jorge Diz e o Vitor Machel, e buscou responder as seguintes questões:

- Metodologias Ágeis e Teste de Software, tudo a ver ou não?
- Scrum e Teste de Software combinam?
- O que o XP tem a ver com o Teste de Software?
- Preciso seguir a risca a metodologia ágil ao implantá-la?

E também contou com as experiências dos participantes ao utilizarem processos ágeis.

Dito isso, refresquemos as ideias e vamos à leitura!

Metodologias Ágeis e Teste de Software, tudo a ver ou não?

O Marcelo Andrade transmitiu a seguinte ideia "não há como se seguir qualquer técnica dita ‘ágil’ se não se tiver uma forte disciplina em teste de software. Teste é uma das essências do desenvolvimento ágil."

Boa parte das metodologias ágeis surgiu da necessidade de fazer coisas diferentes, para que resultados diferentes fossem obtidos. Uma das diferenças percebida é que Teste de Software é essencial, pois ajuda a garantir a qualidade do produto.

O renascimento do interesse em teste de software é causado pela adoção de práticas de teste integradas ao desenvolvimento ágil, como formalismos (BDD, ATDD com planilhas, DSLs), novos mecanismos de isolamento de unidades para teste (ex: *test doubles*/ *mocks*) novas ferramentas (ex: JUnit) e principalmente uma nova forma de olhar para o teste, enxergando-o não apenas como uma forma de especificação verificável mecanicamente, mas como forma de comunicação e como instrumento de gestão, como nos mostrou o Jorge Diz, que também relembrou que a automação cresceu em importância e puxou avanços na tecnologia de testes como nunca antes na história da nossa área.

Como o termo ágil nos traz a idéia de velocidade, de rapidez, podemos nos apoiar na ideia do Ronaldo Cruz para enxergar a combinação entre Testes e Metodologias Ágeis: "uma vez que a elaboração de um produto tem de ser feita de forma mais rápida, os teste são ainda mais necessários, o risco de em meio a extrema velocidade se perder ou esquecer de alguma coisa é bem alto." Porém, como nos lembrou o Vitor Machel, existe a necessidade de que o Teste de Software se adapte aos valores da metodologia: "como qualquer processo de desenvolvimento de sistemas, mas não se pode ficar com o mesmo pensamento com metodologias tradicionais, tem que se enquadrar também."

Scrum e Teste de Software Combinam?

Existe uma ideia que o Scrum pode ser utilizado em qualquer contexto em que uma equipe precise trabalhar junta em prol de um objetivo comum. A partir essa ideia, a Renata Eliza nos guia "falando especificamente do Scrum, não há quem diga que ele pode ser utilizado em qualquer área da vida?! Por que não no Teste de Software?" E o Fabrício completou: "Scrum combina com qualquer tipo de projeto, desde para organizar uma faxina na casa com a família, até a construção de um foguete". Mas existe também que não concorde, o Vitor Machel, nos diz que: "Se pensar à risca, não porque teste de software por equipe de teste, não entra no conceito de Scrum". Examinemos as visões explicitadas pela mesa redonda.

Um projeto que utilize Scrum terá maior facilidade para responder as mudanças que vierem a ocorrer durante o decorrer do processo de Desenvolvimento. Buscando base nos princípios dos processos ágeis, torna-se possível enumerar uma série de motivos pra implementá-lo. O objetivo principal continuará sendo a garantia da qualidade dos sistemas.

O Scrum utiliza de meios e ferramentas bem claras para comunicação, como por exemplo, o quadro Kanban, que faz com que os objetivos e as tarefas estejam mais claras e para todos da equipe. Além disso, há um grande foco em reuniões de alinhamento, focadas em objetivos específicos, como por exemplo a *daily meeting*, onde o objetivo é alinhar como está o andamento do trabalho da equipe.

Essa dinâmica diferente, que acaba sendo mais simples e facilita a equipe ter foco no trabalho que é necessário ser feito, pode resultar num melhor gerenciamento dos testes e também num fluxo mais rápido das tarefas, em comparação, com metodologias tradicionais.

Com a implantação do Scrum é possível

- Verificar o andamento de cada projeto como um todo
- Permitir a detecção e remoção de riscos e impedimentos
- Oferecer uma estimativa mais apurada
- Controlar conflitos de interesses e necessidades
- Compartilhar o conhecimento entre a equipe

Vale lembrar também que um dos pontos mais importante do Scrum é trazer o cliente para perto do time.

Focando o Scrum na atividade de Teste, o Jorge Diniz nos trouxe outra importante percepção: " O conceito de pronto, aplicado às *user stories*, tem que incluir testes."

Como o Scrum é um framework de práticas de gestão, é necessário que também se adote em conjunto práticas técnicas que irão permitir que as entregas atendam sempre aos padrões de qualidade. As práticas de XP vêm a se encaixar bem nesse contexto.

Para não se prender somente nas práticas de gestão e chegar a um*anti-pattern*, é necessário foca-se também nas técnicas usadas para entregar software. É impossível melhorar o produto sem mudar as técnicas de requisitos, de desenvolvimento, de integração e de teste. Se não houver mudanças a equipe não conseguirá dar vazão as entregas. Os *post-its* na parede não possuem superpoderes, por isso nada de esperar melhorias sem mudanças.

Para concluir, refletimos sobre as colocações do Ronaldo Cruz:

"Todo modelo de desenvolvimento combina com teste. A questão é o quanto as pessoas e empresas estão dedicadas em fazer as coisas acontecerem. De nada vale dizer que usa Scrum, XP, RUP, abobrinha, chuchu ou cenoura, se a empresa e os envolvidos não estão realmente engajados em implementar o modelo e fazê-lo acontecer".

O que o XP tem a ver com o Teste de Software?

O XP - *Extreme Programming*, desenvolvido por Kent Back e Ward Cunningham, é conhecido como o mais popular dos métodos ágeis. É indicado

para equipes pequenas e médias, que necessitam desenvolver softwares em que os requisitos não estão totalmente especificados e que também se modificam rapidamente. Os valores principais são: comunicação, simplicidade, *feedback* e coragem.

Após essa breve visão, nos resta responder a questão: "O que o XP tem com o Teste de Software?"

Para o Jorge Diz, o "XP representou no início uma visão bastante peculiar em relação a teste de software".

De uma maneira bem sintetizada, pode-se dizer que o XP distribuiu o Teste de Software em dois: os Testes do Programador e os Testes do Cliente.

No Teste do Programador, o desenvolvedor se torna responsável por testar seu próprio código. Para cada *user story*, é escrito um teste que atenda a funcionalidade. Após a escrita do teste o desenvolvedor implementa o código. Dessa forma, o teste tem um teor altamente técnico e funciona como um apoio ao desenvolvimento do software.

A cada novo teste construído, é necessários que eles sejam integrados à bateria de testes existentes. A cada integração, os novos testes não podem causar falhar nos antigos e essa integração contínua precisa ser rápida para que o fluxo de trabalho de desenvolvedor não seja quebrado.

Já o Teste do Cliente, funciona como testes de aceitação. Os clientes validam cada *user story* e com essa validação, o requisito desta *user story* é declarada pronta.

Outra característica do teste cliente é o detalhamento dos requisitos funcionais do sistema. Uma vez que os clientes estão mais próximos do ambiente de operação do sistema, para eles é mais fácil levantar novas regras e requisitos sobre o software em desenvolvimento.

As atividades de inspeção também estão presentes no XP. Nesse processo, as práticas de inspeção se fundem dentro da prática de trabalho em pares ("pair programming"). Esta inspeção contínua, é mais eficaz que do que a inspeção tradicional, uma vez que as inspeções tradicionais são feitas após o fluxo de

desenvolvimento.

Mas nem tudo são flores em relação a inspeção. Essa prática exige uma mudança na cultura do programador, e ela não costuma ser fácil.

Outra forte tendência no XP é a automação de testes. Essa automação provocou uma série de inovações tecnológicas, para que o tempo de teste fosse suficiente para que o tempo do projeto não fosse quebrado, apoiando o ritmo do desenvolvedor.

Na cultura do XP, inicialmente não era dada importância aos testes manuais. Atualmente, essa visão mudou devido as contribuições da escola "context-driven testing" para testes exploratórios gerenciados.

Vale ressaltar que TDD (*Test Driven Development*) não é considerada uma prática de teste, e sim uma prática de apoio ao desenvolvimento, uma vez que ela é uma junção de duas práticas: 'test-first' (escrever o teste automatizado, antes do código) e 'refactoring' (limpeza do código, sem alterar seu comportamento funcional que já fora validado pelos testes).

Preciso seguir a risca a metodologia ágil ao implantá-la?

Essa é uma dúvida que pode surgir na cabeça dos responsáveis pelo desenvolvimento de um software. Surge um novo projeto e a oportunidade de

colocar em prática os conhecimentos sobre os processos ágeis. De que forma utilizar a metodologia ágil em seu projeto? Vejamos as ideias dos participantes da mesa para sanar essa dúvida.

Como o próprio nome diz, é um modelo, uma forma estruturada de executar o trabalho. Cada empresa tem sua particularidade, seja do projeto, dos funcionários, do cliente, em fim, da cultura. O que o Ronaldo Cruz nos recomenda é "você pega as práticas do modelo e as adapta ao seu ambiente."

Outro cuidado com "seguir a risca" é a forma como a metodologia é ensinada,

O Vitor Maciel aponta: "...ninguém ensina a risca metodologia ágil, é incrível o que eu leio por aí." E prossegui alertando sobre cargos: "Imagina criar empregos específicos de Scrum como um cara especifico para ser Scrum Master".

Já alguns (o Kent Beck do XP, por exemplo), dizem que sim, que é necessário seguir a risca a metodologia, por causa da sinergia entre as práticas. Como citou o Jorge Diz. Mas, o próprio Jorge, acredita que é necessário manter um jogo de cintura, sem descaracterizar as práticas que fazem a dinâmica funcionar. Sempre irão existir limitações e será preciso contorná-las.

No início do processo de implantação de uma metodologia ágil, o problema é a que os conceitos da metodologia ainda não foram totalmente incorporados. Só se aprende realmente fazendo, descobrindo, que é importante e o que não é; onde será preciso bater o pé e onde será possível ceder; o que funcionará bem num certo contexto e o que será preciso evitar. Por isso, uma das soluções para esse problema, é ter o acompanhamento de profissionais mais experientes durante o início de uma transição para a agilidade.

Outra prática comum, adotada pelas organizações que adotam uma nova metodologia, são as adaptações para evitar as práticas que mais mudam a forma de trabalho à qual estão acostumadas, por acomodação ou por medo da mudança. O tal "Scrum, but ...", como disse o Jorge Diz.

Muitas organizações saem com as máximas: "Estamos fazendo Scrum, mas dispensamos as reuniões diárias". "Estamos usando XP, mas não temos testes unitários". Em casos como esses, a chance de sucesso diminui drasticamente, e a metodologia em questão paga o pato e ainda aparece mal na foto.

Para concluir, O *by the book* está longe de ser a melhor maneira de implantar algo, principalmente na nossa área, como nos contou o Fabrício, e continuou dizendo: "Adaptação é a palavra chave, e adaptação não é feita uma única vez, temos que está sempre evoluindo e melhorando, e para isso a adaptação é essencial."

O Testador no Processo Ágil

"O testador é "o cara que aprova".

Nada é considerado "pronto" em um sprint, até que ele diga que está." - Gustavo Quezada

Durante a Mesa Redonda surgiu um novo questionamento. A Juliana Kryszczun levantou a seguinte dúvida:

"Qual é a maior diferença para o analista de teste entre processos ágeis e processos tradicionais de teste?"

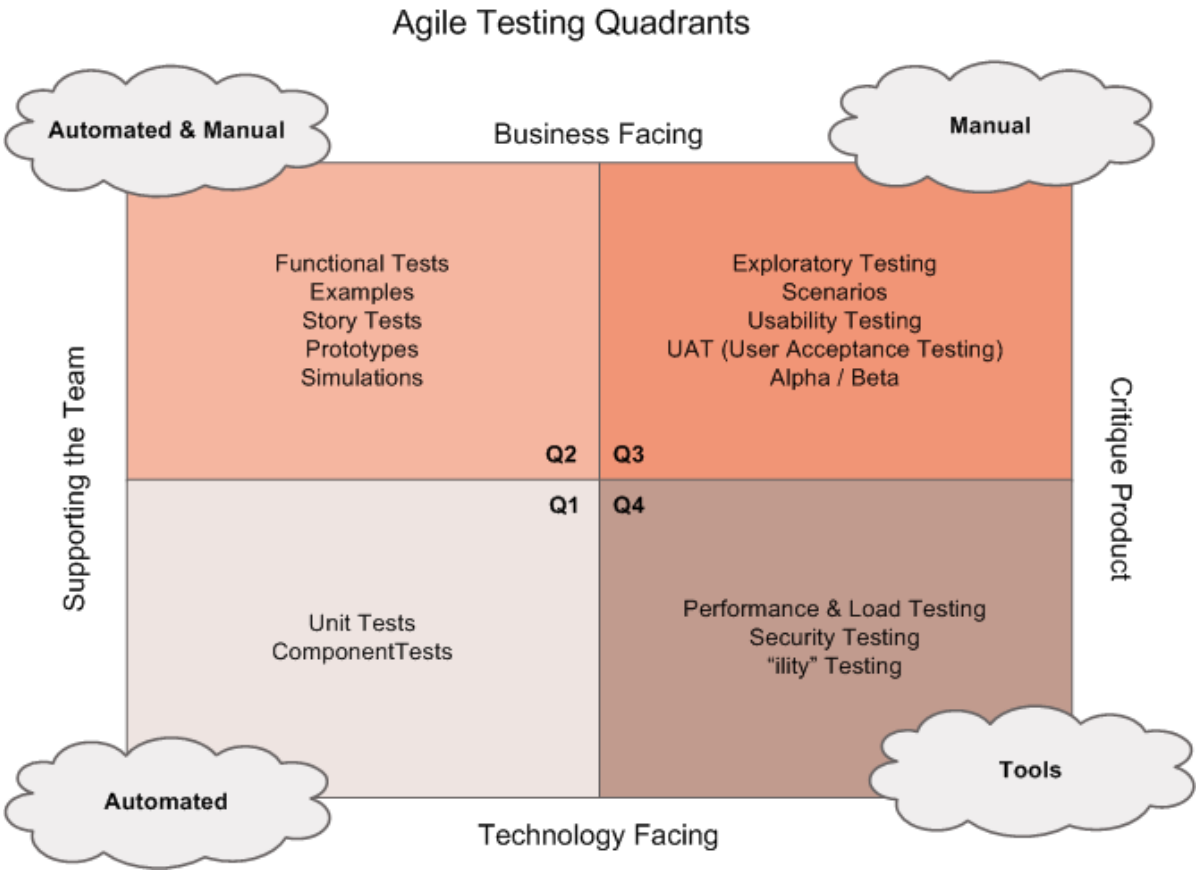
Para ajudar a sanar essa questão, o Marcelo Andrade citou o artigo do Professor José Papo, "O papel do analista de Testes dentro dos processos ágeis - Uma Introdução". Abaixo está uma compilação das principais colocações do artigo.

O professor inicia apresentando a ideia de que ainda existem muitas dúvidas no mercado, sobre qual é o papel do analista de teste dentro de uma equipe ágil. Pois a definição dos papeis pode levar a confusões e a crença, de que não existe lugar para o analista de teste em um time ágil.

Outro levantamento do professor é a diferença entre o processo tradicional e o processo ágil. Enquanto no processo tradicional, testes aparecem no fim do ciclo de desenvolvimento o analista de teste é mais reativo. Na prática a equipe de vive separada da equipe de desenvolvimento, e só se cruzam no momento da verdade da fase de teste do modelo cascata. Já nos processos

ágeis, o analista de teste deixa de ser reativo para se tornar um papel fundamental na interação com os desenvolvedores, analistas de negócios e clientes.

Para fundamentar essa mudança, o professor apoiou-se no quadrante de testes ágil, do livro de Lisa Crispin e Janet Gregory, Agile Testing.



Em seguida, citou os grandes Grupos de Teste:

Q1 - Testes que focam na arquitetura e suportam o time: São os testes unitários e de componentes. Estes são realizados e são de responsabilidade dos próprios desenvolvedores. O papel do analista de testes nesse quadrante é o de apoiar, suportar e expandir conhecimentos entre os desenvolvedores sempre que necessário. De preferência isso é feito fazendo "pairing" com o desenvolvedor no momento de elaborar os testes unitários automatizados.

Q2 - Testes que focam no negócio e suportam o time: São testes funcionais diferenciados, que idealmente utilizam a técnica de *Behaviour-Driven Development* e *Acceptance Test-Driven Development*. Isto é, são testes e cenários de exemplo realizados pelos testadores em conjunto com os clientes, usuários e analistas de negócio. Com base nesses exemplos e cenários os desenvolvedores terão melhores condições de desenvolver e entender os

requisitos. Além disso, utilizando-se ferramentas adequadas (como o Fitnesse ou o Concordion, por exemplo), uma parte desses testes será automatizada antes ou em paralelo com o desenvolvimento do cenário. Portanto, o foco desses testes não é encontrar o maior número de defeitos e sim ajudar clientes e desenvolvedores a se entender melhor.

Q3 - Testes que focam no negócio e criticam o produto: esses são o que chamo de testes "clássicos". Os testes de aceitação feitos na homologação do produto ou de suas partes, testes betas e testes exploratórios. São os testes feitos não com o objetivo de dizer que o software funciona, mas, pelo contrário, de encontrar defeitos. Essa categoria às vezes é negligenciada por alguns agilistas mais radicais. Mas a verdade é que bons analistas de testes possuem técnicas para encontrar defeitos que poucos desenvolvedores conhecem (até porque o papel do desenvolvedor é construir e o do testador, neste quadrante, é o de destruir!).

Q4 - Testes que focam na arquitetura e criticam o produto: São os testes de performance, de carga e de segurança. Esses são de responsabilidade dos analistas de testes e costumam ser feitos quando pedaços da aplicação já estão prontas e, especialmente, antes da entrada de um release em produção.

E concluiu com a seguinte recomendação: "Recomendo então a todos os analistas de testes: estudem bastante o processo ágil de testes, as novas técnicas de *Behaviour Driven Development* e as ferramentas automatizadas que as auxiliam. Assim você se tornará um recurso muito mais valioso para sua equipe e para o resultado final dos projetos."

Principais Benefícios dos Métodos Ágeis.

Existem várias razões pelas quais as empresas adotam os Métodos Ágeis irá agregar valor a seus negócios. Dentre as principais podemos citar:

- **Software correto** - O cliente está constantemente envolvido no projeto. Assim assegurando que o software que esta sendo desenvolvido é o que

lhe retornará o maior valor desejado, ou seja, atenderá todos os requisitos funcionais, de acordo com a sua prioridade. A todo instante o cliente irá dizer que o sistema estará desenvolvido de forma correta para a equipe de desenvolvimento sobre o incremento de software que foi entregue. Com esse alto grau de envolvimento do cliente em um projeto ágil, não é mais difícil desenvolver um software errado e que não traga o maior valor possível ao cliente.

- **Qualidade** - Métodos Ágeis sempre atribuem forte foco na qualidade do que esta sendo desenvolvido. Não se resumindo apenas nos testes de aceitação do
- cliente, mas sim na adoção de práticas de desenvolvimento que garantam alta qualidade técnica.
- **Prazos e custos** - O fato dos projetos ágeis fazerem com que o software seja desenvolvido em interações, e propondo que essas interações devam ser curtas, faz com que os atrasos sejam menores, e também proporciona um melhor ajuste da equipe quanto ao desenvolvimento do software, e do cliente quanto ao software que deseja. Na verdade a grande vantagem do métodos ágeis nesse ponto, é que eles incentivam a entrega constante e o mais cedo possível, o prazo em si também definido em projetos que não usam métodos ágeis. Caso o andamento do projeto não esteja seguindo o que foi planejado, os requisitos de menor valor ou baixa prioridade podem ser retirados do projeto. E caso o período definido de tempo do projeto tenha a necessidade de ser estendido, isso ocorre com o total consentimento do cliente e de todos envolvidos no projeto.
- **Alertas mais cedo** - Como um projeto ágil é essencialmente uma série de mini-projetos bem definidos, os problemas podem ser identificados e resolvidos com maior antecedência, sem causar maiores danos ao andamento do projeto.
- **Adaptação a mudança** - As mudanças são inerentes aos negócios. Um projeto ágil pode se adaptar as mudanças de um mercado, uma empresa, ou um cliente, efetivamente melhor do que os projetos tradicionais.

Experiências dos participantes ao utilizarem processos ágeis

Nessa parte, serão expostas as experiências compartilhadas pelos participantes da mesa, sobre a utilização de processos ágeis. Aproveitem as visões e absorvam o que julgarem necessário, nunca se sabe quando será a hora de

colocar a leitura em prática, e ideias são sempre bem-vindas. Como disse o Felipe da Silva: "Gosto sempre de utilizar exemplos que sendo analisados possa-se tirar uma lição extra".

Para o Ronaldo Cruz:

"Para mim foi muito boa. Tornou o trabalho mais fácil por ter uma interação maior com todos os envolvidos. Mas também exigiu uma mudança maior na postura. Qualquer membro da equipe, seja desenvolvedor, banco ou teste, tem de se tornar ágil. Sair da cadeira e correr atrás, seja para desenvolver a atividade de outro membro ou tirar dúvidas de teste ou negócio, não importa, cada um da equipe precisa deixar de ser acomodado e se tornar pró-ativo, buscar a solução e ajudar os demais a concluírem as suas atividades. Na equipe em que trabalho, me tornei analista de teste, negocio e dou suporte ao pessoal do desenvolvimento."

O Vitor compartilhou também a sua experiência dizendo:

"Totalmente satisfatórias, os documentos (não são todos) que só serviram para ir para o lixo, nem existe mais. O conceito que estamos usando é "faça somente documentos para usar e entregue o software funcionando". dificilmente estamos trabalhando(tanto teste quanto desenvolvimento) depois das 18 e sábados e domingo, o que eu mais gostei. Em resumo aqui, todos estão aprovando, cliente satisfeito é tudo, o resto é resto."

O Fabrício Ferrari apresentou também suas conclusões retiradas de suas experiências:

- A cultura das pessoas e da organização é o que mais influência a implantação de uma metodologia ágil;
- A maturidade das pessoas também é um fator bastante influenciador, principalmente, em relação ao tempo que a implantação vai levar e como

ela ocorrerá;
- A comunicação é essencial, tanto interna quanto externa;
- O PO é o cara (no bom sentido é claro), a sua participação é essencial para o sucesso do projeto;
- As pessoas são o diferencial de qualquer projeto. As pessoas certas com domínio da tecnologia, conhecimento do projeto e determinação são capazes de resultados incríveis, e a adoção de metodologias ágeis potencializa esses resultados;
- A metodologia utilizada pode ter dado certo em um projeto, mas isso não garante que ela também irá funcionar em outro;
- Busque ter uma pessoa da área de Teste de Software, no seu time Scrum;
- Antes de implantar uma nova metodologia, dê treinamentos a respeito dela para todos;
- Se algo deu errado, com certeza não é culpa da metodologia, e sim sua.

O Jorge Diz a partir de suas experiências identificou os problemas enfrentados pela disciplina de Teste de software no contexto dos processos ágeis. É comum as pessoas empacarem justamente por dificuldades na automação de testes no nível necessário para suportar a agilidade. Alguns pontos de atenção:

- Código legado difícil de testar
- Falta de disciplina de *test-first* por parte dos desenvolvedores
- Não domínio das técnicas de isolamento para testes unitários
- Investimento excessivo em testes frágeis (GUI), que pode diminuir a velocidade das atividades de teste.
- Indisponibilidade de ambiente próprio para teste, onde a aplicação possa ser testada sob controle total da equipe de desenvolvimento/ testes
- Testadores ainda perdidos numa nova forma de organizar as equipes (testador residente em vez de operário de uma fábrica de testes)
- Dificuldade para satisfazer o pronto-pronto
- Resistência à programação em pares

Conclusão

Voltando ao tema processos ágeis, nesse momento de fechar as ideias, vale relembrar da colocação do Marcelo Andrade, que alertou para o enfoque meramente mercadológico para a palavra ágil. De nada adianta uma empresa dizer que usa metodologia ágil se ela não possuir disciplina e seguir apenas o modismo que a palavra ágil nos traz: "Ontem usávamos metodologia própria,

hoje usamos ágil e continuamos seguindo o testa aí". O indicado, seria que empresas que querem entrar na moda, usassem um tempo para estudar e um modelo como mesclas de metodologias ágeis adaptadas para seu cotidiano. Nesse caso, essas empresas poderiam ser o destaque de um evento 'da moda' e ainda ganhar fãs e seguidores, como a globo.com que implantou um misto de metodologias ágeis para desenvolvimento de seus projetos.

E seguindo a ideia de adaptar e implantar, o principal objetivo ao se implantar uma nova metodologia, é sempre melhorar e melhorar, como lembrou o Felipe Silva e que ainda completou dizendo que pela melhoria vale a pena ir à guerra, tomando o cuidado de sempre lutar como um exército unido e não um contra o outro. Lembrando que será necessário mudar a cultura dos soldados, pois de nada adiantará usar, ágil, abobrinha, ou chuchu, se as pessoas da empresa, ou os nossos bravos soldados, não estiverem comprometidas e engajadas em implementar o modelo e fazê-lo acontecer. Não existe a bala de prata!

Inicie com o primeiro passo e siga passo a passo. Não de uma vez, como tentam fazer alguns, que pensam que um processo novo que cai de sem pára-quedas e irá funcionar. Traga o cliente para perto do time, busque ter uma pessoa da área de Teste de Software, no seu time. Já vimos que o teste tem muito, se não tudo a ver com essa guerra. Ele é a arma secreta para deixar seu cliente satisfeito. E principalmente, ao iniciar um novo processo ágil, lembre-se da máxima do Marcelo Andrade:

SE VOCÊ NO TESTA, VOCÊ NO É ÁGIL!

Referências Bibliográficas

CRISPIN, Lisa et al. **Agile Testing: A Practical Guide for Testers and Agile Teams.** Addison-Wesley Professional. 2009. 576P.

KOSCIANSKI, André. **Qualidade de Software**. São Paulo. Novatec. 2007 400p.

PAPO, José. O papel do analista de Testes dentro dos processos ágeis - Uma

Introdução. Disponível em < http://josepaulopapo.blogspot.com/2009/09/testes-agil-papel-analista.html> Acesso em 09/11/2010.

SOARES, Michel dos Santos**. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. Disponível em <http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02... > Acesso em 01/11/2010

Capítulo 7: Teste ágil, como implementar?

Lucas Gonçalves Nadalete

Revisora: Juliana Kryszczun

Introdução

O uso de metodologias e práticas de desenvolvimento ágil têm se tornado algo cada vez mais natural no cotidiano das empresas . Algumas dessas empresas, apesar de não admitirem sua adesão, implicitamente acabam aplicando algumas práticas comuns, com o objetivo de obter resultados mais cedo.

Essas práticas tendem a estimular e valorizar a equipe e a interação constante entre os seus membros, a colaboração constante com os clientes, o funcionamento do *software* em desenvolvimento e a capacidade de resposta a mudanças.

Poucas sabem, mas na realidade elas estão adotando os princípios ágeis definidos no Manifesto Ágil na data de 11 à 13/02/2001. No entanto, o que muitas não tem claramente definido, é como controlar a qualidade do que está sendo desenvolvido de forma ágil, ou seja, como praticar teste ágil.

Conforme opinião expressa pelos colaboradores e reforçada por meio da própria literatura, aplicar teste ágil exige quebra de paradigmas, mudança de cultura, dinamismo da equipe (pré e pró-atividade), conhecimento técnico, testes automatizados, e muitas outras características e fatores que viabilizam a aplicação de teste ágil, quando praticadas conjuntamente.

Este capítulo apresenta uma explanação sobre "teste ágil e como implementá-lo" do ponto de vista das diversas perspectivas discutidas na Mesa Redonda DFTestes.

O que é Teste Ágil?'

Em se tratando de desenvolvimento ágil, alguns princípios definidos por meio do Manifesto Ágil são utilizados com o objetivo de nortear a linha de produção, como:

- Indivíduos e interações entre eles mais que processos e ferramentas;
- *Software* em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Capacidade de responder a mudanças mais que seguir um plano.

Apesar de se valorizar muito mais os itens da esquerda, os itens da direita não são desconsiderados, ao invés disso, os mesmos são aplicados de forma ponderada e conforme a necessidade do processo, sem que haja impacto nas entregas a serem realizadas, prazos estabelecidos e qualidade do produto final gerado.

Os mesmos princípios usados para direcionar o desenvolvimento ágil, devem ser considerados quando for aplicado teste ágil, ou seja, testar de forma ágil exige uma forte adaptação na rotina e dinâmica da equipe de teste, em relação ao processo de desenvolvimento adotado, com o objetivo de propiciar um processo relativamente simples e que possa ser executado com grande facilidade e agilidade, cobrindo o maior número de riscos, com um nível de qualidade que seja apreciada e valorizada pelo cliente ou usuário final.

Através da definição do processo ideal e simplificado, onde o teste ágil é suportado, um conjunto de práticas que proporcionem a diminuição do tempo entre o erro e a sua descoberta tendem a ser estabelecidos em conjunto com uma sistemática de trabalho que possibilite à área de teste de *software* ser mais pró-ativa do que reativa.

O que muda do Teste Tradicional para o Teste

Ágil?

Analisemos algumas das práticas do processo de teste tradicional aplicados na tentativa de gerenciar o "*chaos*", ou ao menos evitar culpados :

- A área de teste de *software* assumindo a postura de "Último Defensor da Qualidade";
- Restrições no gerenciamento de mudanças;
- Preparação detalhada e planejamento acima de tudo;
- Conjunto de documentação pesado para a terceirização dos esforços de teste;
- Critérios de entrada e saída rigorosos e com aprovações;
- Automatização de testes pesada e com foco nas regressões;
- Tentativas de execução do processo.

Ao se tratar de teste ágil, essas mesmas práticas não se adaptam à dinâmica almejada. Em um ambiente ágil de controle de qualidade deve-se considerar os prazos e as atividades de teste do começo ao fim da iteração. Ao contrário do teste tradicional, não se espera que uma única equipe se responsabilize pela qualidade final das entregas, mas sim que todas as equipes tenham sua colaboração no controle dessa qualidade, desde o levantamento das necessidades do cliente, até a implantação do produto final gerado.

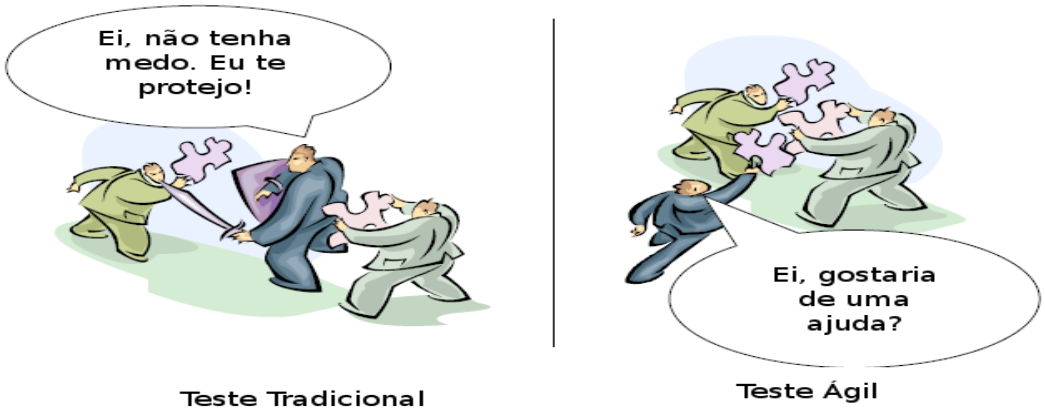


Figura 7.01 – Deslocamento de papéis - Adaptado de

Conforme pode ser observado na Figura 1, no teste tradicional, espera-se que

os defeitos sejam identificados no último nível, pela equipe de QA, enquanto que ao se aplicar teste ágil, isso é antecipado pela própria equipe de desenvolvimento por meio de práticas como desenvolvimento em pares, integração contínua, pequenas entregas, refatoração constante e padrões de codificação, de técnicas como TDD (Test Driven Development) e ATDD (Acceptance Test Driven Development), e através da automatização dos testes gerados.

Ao se trabalhar com testes ágeis, observa-se algumas mudanças de conceito em relação ao modelo tradicional, tais como:

- Mudanças são inevitáveis, e com base nisso toda equipe, incluindo os programadores, testadores e clientes, são responsáveis pelo resultado final;
- Todos da equipe devem estar acessíveis e se comunicando ativamente através do projeto;
- O teste de *software* se torna mais preventivo, ou seja, programadores testam mais cedo, com mais frequência e agressivamente. Neste ponto a prática de TDD pode e deve ser aplicada;
- Toda equipe solicita ativamente *feedback* das demais;
- Testadores e programadores tendem a ser mais pró-ativos (participação direta com o cliente) e técnicos (aplicação de práticas XP e técnicas como TDD e ATDD);
- Testadores precisam saber automatizar, com o intuito de manter o ciclo de entregas dos testes sempre no prazo estabelecido e com teste de regressão atualizado. Só não se automatiza, o que realmente não pode ser automatizado ou não vale a pena ser automatizado (*e.g.* Teste de Usabilidade);
- Os testes tonam-se uma rotina que nasce e morre junto à iteração planejada.

Como podemos implementar Teste Ágil no

dia-a-dia?

É possível trabalhar de forma ágil nas atividades de teste de*software* com qualquer tipo de metodologia de desenvolvimento, desde os mais tradicionais descendentes da família UP (*e.g.* RUP, EUP, OpenUP, AUP), até as mais ágeis como XP, Scrum, Cristal, Lean e outros. A questão é que nenhuma dessas metodologias garantirá "agilidade" no processo. Ser ágil não exige apenas adotar uma sistemática ágil, mas pensar e agir de forma ágil na plenitude do seu conhecimento.

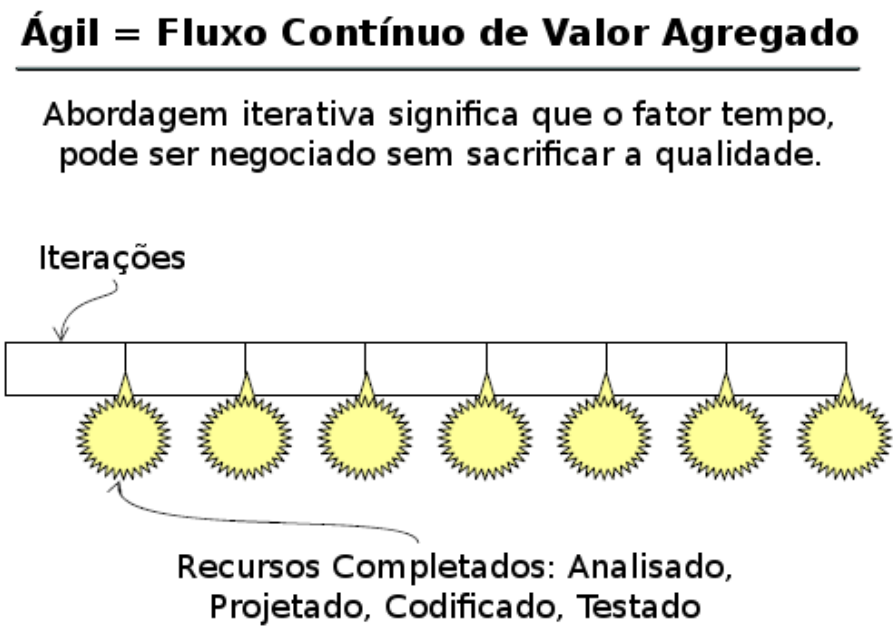


Figura 7.02 – Fluxo ágil de desenvolvimento

Em um projeto onde a sistemática de trabalho segue os princípios do Manifesto Ágil, o prazo de cada iteração tende a ser reduzido, visando agregar valor mais rápido ao produto final que é apresentado como subsídio para se obter o *feedback d*o cliente (Figura 2).

É neste tipo de iteração que as "atividades de teste" devem assumir características ágeis. A Figura 3 apresenta um processo simplificado de teste ágil que pode ser adotado a cada iteração realizada.

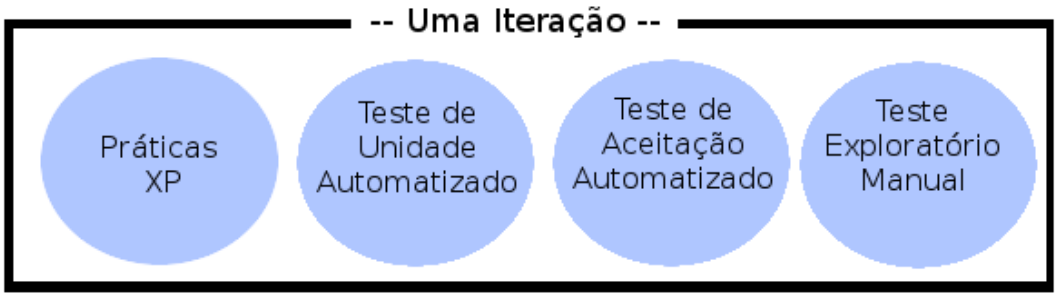


Figura 7.03 – Representação de uma iteração de teste ágil - Adaptado de

- **Práticas XP**: A aplicação de algumas das práticas de Programação Extrema possibilitam mitigar e minimizar os riscos de introduzir defeitos triviais na aplicação, por meio da aplicação de programação em par, *design* simplificado, testes, refatoração constante e a aplicação de padrões de codificação . Outras práticas podem ser adicionadas conforme necessidade.

- **Teste de Unidade Automatizado**: Feito pelos próprios programadores, geralmente com o auxílio de um *framework* xUnit, e frequentemente como resultado da prática de TDD. A bateria de testes de unidade gerada representa um conjunto de especificações executáveis que apoia o processo de desenvolvimento.

- **Teste de Aceitação Automatizado**: Representa o resultado da colaboração entre os programadores e os *stakeholders* de negócio, e são frequentemente implementados com o auxílio de um FIT (*Framework for Integrated Testing*). A bateria de testes de aceitação gerada representa um conjunto de requisitos executáveis.

- **Teste Exploratório Manual**: Além de ser necessário para complementar os testes automatizados, os testes exploratórios fornecem algum *feedback* adicional, cobrindo possíveis lacunas que passam despercebidas pelos testes automatizados de unidade e aceitação.

Para se aplicar teste ágil, não há receita de bolo a ser seguida. É preciso conhecer o processo ágil praticado muito bem, assim como a cultura da empresa e a maturidade dos seus colaboradores em relação ao mesmo, e em cima disso, trabalhar uma estratégia de implantação e execução das atividades de teste dentro do processo.

Algumas sugestões são apresentadas por profissionais e podem ser consideradas na implementação de teste ágil:

- Implantação de forma gradativa, ou seja, a realização de um projeto piloto pode ser adotada com o intuito de iniciar a adoção, adquirir experiência, aperfeiçoar, ajustar e comprovar progressivamente os ganhos;

- Análise antes de automatizar os testes, se realmente vale a pena ou não. A automatização na maioria das vezes proporciona diversos ganhos ao projeto, principalmente a médio e longo prazo, no entanto, a mesma pode custar mais caro que os ganhos proporcionados em determinadas situações;

- Ao se aplicar teste ágil, há uma inversão na pirâmide dos níveis de teste, ou seja, ao invés de termos muitos testes a nível de sistema e aceitação, haverão muitos testes a nível de unidade e integração, onde a responsabiliade primária tende a ser dos programadores;

- Atue junto ao cliente, com o objetivo de obter *feedback* constante sobre o que está sendo desenvolvido, em relação ao esperado;

- Criar teste de unidade e integração para sistemas legados (código desenvolvido sem teste), é algo impraticável e pode impactar no orçamento e prazo final planejado.

ATDD e TDD, como implementar?

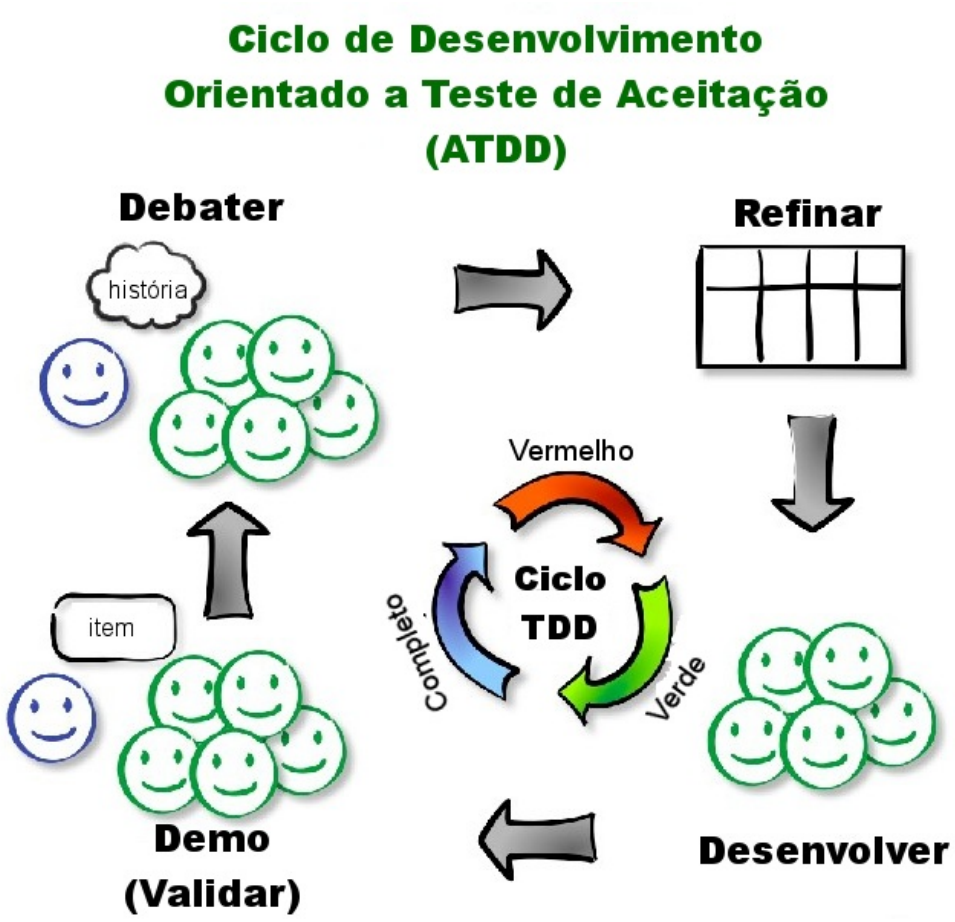
É muito comum ouvirmos a respeito de desenvolvimento orientado a teste (TDD) como uma "técnica para teste" onde os testes de unidade produzidos, guiam o desenvolvimento do código fonte da aplicação. Na verdade TDD não se trata de uma técnica, e sim de uma "prática de programação" que resulta em uma suíte de testes de unidade, onde tais testes são um efeito colateral e não o objetivo em si .Na aplicação da prática de TDD é guiada por três passos básicos, usados no desenvolvimento dos testes de unidade, e consequentemente do código fonte da aplicação:

- Desenvolve-se um teste de unidade que falhe, para demonstrar que a base de código existente ainda não contém tal recurso implementado;
- Uma vez que se tenha o teste de unidade que falhe, produz-se o código de produção para tonar o teste executável e fazê-lo passar;
- Com o teste passando, refatora-se o código com o objetivo de enxugar e eliminar duplicações para deixar o código fonte legível e melhorar o *design*.

Aplicar TDD nem sempre é uma tarefa fácil e muitas vezes é tida como complexa por pregar a criação dos testes antes mesmo do código da aplicação, além de representar expectativas quanto ao comportamento do *software*.Mas e o que a "prática" de ATDD tende a oferecer além da TDD?

ATDD visa capturar os critérios de aceitação para a funcionalidade em desenvolvimento. Normalmente isso é discutido, identificado e levantado quando a equipe de teste interage com os responsáveis pelo negócio, visando compreender sua real necessidade. Em complemento à prática de TDD, que visa garantir que as funcionalidades bases da aplicação sejam desenvolvidas em conformidade com a arquitetura e projeto, a prática de ATDD tende a prover *feedback* sobre o quão perto da conclusão da tarefa a equipe de desenvolvimento se encontra, demonstrando uma clara visão do progresso.

A Figura 4 apresenta claramente o ciclo de desenvolvimento orientado a teste de aceitação, e como a prática de TDD pode ser inserida visando aumentar a abrangência dos testes.



No artigo escrito por Elisabeth Hendrickson , fonte principal deste tópico, a autora apresenta um exemplo simples, prático e bem completo de como extrair o melhor de ambas as práticas de TDD e ATDD. As equipes que aplicam as práticas, principalmente ATDD, tendem a sentir os benefícios ainda na fase de discussão dos requisitos e estabelecimento dos critérios de aceitação, por meio de uma melhor e mais completa compreensão das necessidades do cliente.

Quando o ciclo proposto na Figura 4 é trabalhado do começo ao fim do projeto, os envolvidos tendem a concordar que o sistema tende a se tornar mais testável, manutenível e relativamente fácil. Com a suíte de testes gerada é possível efetuar os testes de regressão automaticamente, fornecendo um pronto *feedback* sobre as expectativas relacionadas às funcionalidades bases do sistema, e ao negócio como um todo.

Quais as dificuldades de se implementar Teste Ágil?

Nem todos os ambiente são propícios para a adoção e execução de desenvolvimento ágil. Barry Boehm e Richard Turner sugerem em seu trabalho que a análise de risco seja usada para escolher entre métodos adaptativos ("ágeis") e preditivos ("dirigidos pelo planejamento"), destacando o ambiente ideal para cada um destes:

- **Ambiente ideal para o desenvolvimento ágil**
  - Baixa criticidade;
  - Desenvolvedores sênior;
  - Mudanças frequente de requisitos;
  - Pequeno número de desenvolvedores;
  - Cultura que tem sucesso no caos.

- **Ambiente ideal para o desenvolvimento direcionado a planejamento**
  - Alta criticidade;
  - Desenvolvedores júnior;
  - Baixa mudança de requisitos;
  - Grande número de desenvolvedores;
  - Cultura que procura a ordem.

Muitas vezes as características acima apresentadas são consideradas ao se definir qual a metodologia a ser adotada em um projeto de desenvolvimento de *software* e, consequentemente na sistemática adotada para as atividades de teste. Dentre outros fatores que podem comprometer a adoção de uma abordagem mais ágil na dinâmica de testes dentro da empresa, cita-se:

- Cultura organizacional resistente;

- Requer muita mudança cultural;

- Pode levar a maiores dificuldades nas negociações contratuais;

- Equipe não multidisciplinar, ou seja, considerando o fato de que o time de teste é forçado a cada vez mais estar pequisando e se aperfeiçoando em relação à novas ferramentas, além de saber programar, pois a automação se torna um elemento essencial, espera-se que os colaboradores sejam multidisciplinar;

- Cliente não é ou prefere não ser ágil, assumindo uma metodologia direcionada a planejamento. Neste ponto, espera-se que ao optar por uma metodologia ágil, o cliente seja participativo e comprometido com o sucesso do sistema, passando para o time de desenvolvimento suas reais necessidades e opiniões do sistema;

- A área de teste participando de forma reativa, ou seja, contando sempre com todos os documentos à mão, o que nem sempre é uma realidade em um ambiente ágil.

Segundo José Correia, um profissional especialista da área de teste de *software*, o mesmo relata sua experiência prática em relação à resistência do mercado na adoção de metodologias ágeis:

"Nos projetos que acompanho, o uso de práticas ágeis é elegível apenas quando o número de horas totais do projeto é inferior a 1000 horas. Alguns clientes são mais conservadores e estabelecem um teto de 640 a 800 horas. Projetos acima são executados dentro das práticas do RUP, modelos CMMI/MPS-Br e os testes complementados com base no Syllabus/ISTQB, CBOK/QAI, normas ISO e IEEE."

Experiências como a citada acima, nos mostram que um ambiente ágil nem sempre é elegível para um determinado projeto, seja pelas especificações do projeto a ser executado, seja pela capacitação dos recursos envolvidos, seja pelo prazo estabelecido, ou até mesmo pela resistência organizacional ou do cliente final. O que nos cabe é analisar cada caso e avaliar qual a melhor metodologia de desenvolvimento e teste a ser aplicada, e não simplesmente aplicá-la por aplicar.

Quais as vantagens de se implementar Teste Ágil?

A aplicação de teste ágil proporciona diversas melhorias, não só a nível de qualidade do processo, como também na qualidade do produto. A equipe de desenvolvimento se apresenta mais motivada e segura em relação a qualquer mudança que seja efetuada na aplicação. Em um ambiente ágil, os *bugs* são identificados, relatados e corrigidos em um curto espaço de tempo, pelo fato do foco estar na prevenção e não na remediação dos mesmos. O desenvolvimento torna-se sustentável.Em um ambiente ágil, a tendência é que as entregas sejam feitas mais rápidas, em curtas iterações. A gestão de defeitos tende a ser, também, mais dinâmica, pois passa a ser executada diretamente pelo próprio desenvolvedor, sem o relato de inconformidades e intervenção do testador, como também pode ser fomalizada pela equipe de QA à equipe de desenvolvimento quando identificadas a nível de sistema ou aceitação.

A participação do cliente nas atividades de teste se torna mais efetiva, ou seja, faz com que o *software* tenha realmente aquilo que precisa ter, e faça aquilo que realmente precisa fazer. Em paralelo, a automatização de testes adquire muito mais importância, possibilitando a realização de ciclos enxutos e cada vez mais rápidos, além de garantir a validação regressiva das funcionalidades, e de forma econômica.Ao se aderir a teste ágil, o retrabalho e manutenção executados na aplicação são bem menores, pois as validações são realizadas em todos os níveis da aplicação (unidade, integração, sistema e por fim aceitação do cliente final). Com isso, o custo e tempo atrelados ao desenvolvimento também tendem a reduzir.

É possível observar que o fator "tempo" tem se tornado cada vez mais curto, levando muitos profissionais a casarem a ideia de **agilidade** com **qualidade**. Com base nisso, tenta-se estabelecer um equilíbrio unindo o "útil" (qualidade) ao "necessário" (agilidade), sem perder a real essência das atividades de controle de qualidade. Em outras palavras, "agilidade" não representa "ter pressa", da mesma forma que não simboliza evitar a "documentação", pelo contrário, ser "ágil" significa fazer direito, conciso e coeso, onde toda documentação que seja efetiva e útil, é mais que assimilada. É por esse motivo que a documentação de teste de *software* esta entre uma das melhores e mais efetivas.

Em um artigo publicado na InfoQ por Kay Johansen, a mesma questiona à

diferentes especialistas da área "Por quê você ama teste ágil?", obtendo as mais variadas respostas, as quais são compiladas em 10 razões chaves para isso:

- **Não há mais teste manual de scripts!** *Scripts* são executados automaticamente, disponibilizando mais tempo para o testador executar testes exploratórios.
- **Desenvolvedores realmente gostam de mim!**: Localizar problemas antes do final da iteração e enquanto o código está fresco na mente dos desenvolvedores, facilita o trabalho dos mesmos.
- **Agora eu posso verificar os recursos antes deles serem escritos!**(ambos Kay e Philip) – O testador pode evitar problemas ao iniciar o teste, antes que os recursos sejam definidos.
- **Os resultados do teste automatizado podem ser visto muitas vezes ao dia** – Fornecendo um *feedback* rápido após qualquer alteração.
- **A atmosfera é fortemente orientada a equipe**(John Overbaugh) – Cada membro da equipe se preocupa em terminar os testes e não somente o código (Lisa Crispin).
- **O testador pode ocasionalmente ajustar o defeito**(Lista Crispin) – Cada membro da equipe sente-se mais confortável já que o teste é automatizado.
- **Fornece a oportunidade para revisar constantemente as práticas de teste** (Adam Knight) – Ao invés de simplesmente repetir o que foi feito anteriormente, as práticas são constantemente revistas. No caso de Adam os testes que costumavam levar 5 dias para serem executados manualmente foram reduzidos agora para 30 minutos.
- **Eu gasto muito, muito menos tempo debugando**(Adrian Howard) – Eu tenho o *feedback* quase ao mesmo tempo em que cometi um erro, por isso, geralmente é trivial localizar e corrigir.
- **Há chance de realmente impactar na qualidade ao invés de somente documentá-la!** (Jonh Overbaugh) – Quando os defeitos são corrigidos imediatamente ao invés de colocar numa pilha de defeitos.
- **Sempre existe tempo para testar, porque o teste é feito primeiro** Josue

Barbosa dos Santos contou a história de trabalhar num escritório do governo no Brasil onde a prática era testar no final do projeto. O desenvolvimento estava sempre atrasado no cronograma do projeto, atingindo o prazo limite e sendo liberado para os usuários sem teste. Com a introdução das técnicas de TDD e ATDD pelo menos algum teste era executado enquanto o *software* era desenvolvido.

Conclusão

Os benefícios proporcionados pela aplicação de teste ágil tendem a ser extraordinários quando explorados da forma correta, com as práticas e técnicas corretas, e sob a análise correta dos recursos, infraestrutura, ambiente, *frameworks* e ferramentas a serem utilizados e aplicados no ciclo de vida de desenvolvimento do *software*.

A sistemática de teste aplicada tende a ser mais ousada e dinâmica, fornecendo *feedback* constante a equipe de desenvolvimento, que passa a se comprometer também com a qualidade do produto final, por meio da implementação de testes de unidade (TDD).

É comum ao se aplicar teste ágil, confundir o conceito de "agilidade" com "rapidez", quando na realidade está atrelado à "qualidade" e "integridade" dos artefatos finais entregues. Ao se aplicar uma abordagem ágil, as iterações tendem a ser menores, assim como as entregas tendem a ser efetuadas em um curto prazo de tempo, no entanto cabe a equipe de teste em conjunto com as demais áreas garantir a consistência e qualidade final do produto gerado, assim como a satisfação do cliente diante das suas expectativas. Isso pode ser feito de várias formas, inclusive através da adoção das práticas de TDD e ATDD.

Em cada projeto desenvolvido, pode-se observar um conjunto de variáveis que influenciam diretamente nas decisões tomadas antes, durante e após o projeto, muitas vezes a favor e outras vezes contra a adoção de abordagens ágeis de desenvolvimento. No entanto o que se pode observar, é que "teste ágil" é uma realidade viável que pode extrair o melhor da equipe e do processo aplicado, fornecendo retorno imediato e a curto prazo diante da sistemática de teste adotada.

Referências Bibliográficas

FOWLER, Martin; HIGHSMITH, Jim. The Agile Manifesto. SD Magazine. Agosto, 2001. Disponível em: <http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf>. Acessado em: 02/02/2010.

HENDRICKSON, Elisabeth. Agile QA/Testing. Quality Tree Software, Inc. Novembro, 2006. Disponível em: <http://testobsessed.com/wordpress/wp-content/uploads/2006/11/agiletesting-talk-nov2006.pdf>. Acessado em: 10/02/2010.

BECK, Kent. Test Driven Development: By Example. 1. Ed. Addisson-Wesley Professional. Novembro, 2002. 240 p.

Extreme Programming: A Gentle Introduction. Disponível em: <http://www.extremeprogramming.org/>. Acessado em: 11/02/2010.

BOEHM, Barry. Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA: Addison-Wesley, 2004. pp. 55-57.

InfoQ: Top 10 Motivos para Amar Teste Ágil. Disponível em: . Acessado em: 28/02/2010.

HENDRICKSON, Elisabeth. Driven Development with Tests: ATDD and TDD. Quality Tree Software, Inc. Agosto, 2008. Disponível em: <http://testobsessed.com/wordpress/wp-content/uploads/2008/12/atddexample.pdf>. Acessado em: 08/03/2010.

Capítulo 8: Quando Automatizar?

Fabrício Ferrari de Campos

Revisor: Marcelo Andrade

Introdução

A automação vem desempenhando um importante papel no progresso da sociedade, e isso tem ocorrido tanto para o bem quanto para o mal. Por exemplo: graças à automação foi possível a produção em série de carros, mas a mesma fez com que houvesse demissões em massa de funcionários em épocas de sazonalidade baixa no mercado automobilístico, já que o volume de produção é maior do que o de venda.

No entanto, neste capítulo não iremos abordar a automação no contexto automobilístico e sim no contexto de Teste de *Software*, uma das principais áreas da Engenharia de *Software* e onde a automação costuma ser um grande desafio, uma vez que pode contribuir para o progresso do projeto, como também, se usada de forma inadequada, ter o efeito reverso e causar vários problemas.

Automação no Teste de Software

A automação vem aos longos dos anos ganhando um papel importante na área de Teste de *Software*. E isso se deve a uma série de fatores, dentre os quais podemos destacar:

- Diminuição do uso de mão de obra;
- Diminuição dos custos;
- Aumento na velocidade do processo de Teste de *Software*;
- Maior sustentabilidade da garantia da qualidade, perante o "triângulo da gerência de projeto" (escopo, tempo e dinheiro).

Podemos utilizar a automação em qualquer fase do Teste de *Software*, como

especificação de casos de teste, geração de métricas de testes, execução de testes, montagem do ambiente, etc.

Porém, é importante salientar que a automação costuma ser um passo a ser dado, apenas após já termos um processo de Teste de *Software* bem estruturado e uma equipe preparada, pois ela muitas vezes exige um alto conhecimento técnico, principalmente para alguns tipos de testes específicos (que serão abordados mais a frente) e é um esforço que precisa ser apoiado por um processo maduro.

Em frente aos fatores citados anteriormente, tentaremos nas próximas páginas esclarecer melhor a automação no Teste de *Software*, pois tais fatores costumam ser mal interpretados, causando muitos problemas e o surgimento de falsas expectativas e impressões.

Quando automatizar?

Atualmente existem boas ferramentas (tanto livres quanto proprietárias) que podem nos auxiliar em diversas atividades do processo de Teste de *Software*, desde a revisão de documentos, até a execução de testes de desempenho. Muitas vezes, sente-se a necessidade de ferramentas para automação de testes, quando acabamos despendendo muito tempo, por exemplo: criando ou formatando casos de teste, gerando métricas de testes, etc.
Ao pensar na automação dos testes, não podemos esquecer da automação dos testes unitários e integrados, que geralmente, são feitos pelos desenvolvedores e que já costumam ser executados de forma automatizada. O problema é que a prática de criação de testes unitários e integrados não é muito comum, no âmbito do desenvolvimento de *software*, mesmo tendo resultados bem expressivos quando implantada.
Neste caso, a automação deve iniciar logo que a primeira linha de código for escrita, ou até mesmo (de preferência), antes mesmo da elaboração da primeira linha de código, como propõe a técnica TDD (*Test-driven development*).
Ao pensar em automatizar os testes de sistema, precisamos estudar a viabilidade da automação, quer dizer, se com a automação conseguiremos:

- Ganho de tempo;
- Redução de custos;
- Manter a qualidade.

A maior dificuldade está em conseguir medir a viabilidade da automação, pois

é necessário analisar vários fatores, dentre eles:

- A maturidade do time e do processo de Teste de *Software*;
- O grau de reutilização dos testes automatizados;
- O nível de capacitação das pessoas para operar o *software*, ou manter o *software*, nos cenários em que a ferramenta ou *script* foi desenvolvido pelo próprio time;
- O conhecimento acerca do comportamento que se espera do sistema a ser testado;
- O tempo disponível para automatização dos testes. Lembre-se que a automação é um investimento a médio e longo prazo;
- O grau de frequência de mudanças das funcionalidades a serem verificadas, pois em alguns casos, pode não ser viável automatizar um teste de uma funcionalidade que irá mudar amanhã (embora isso não seja uma verdade absoluta, pois há exceções);
- A testabilidade do sistema. Em alguns cenários é necessário realizar alterações no código para que seja possível a automação dos testes. Isto ocorre de forma mais frequente, em sistemas legados, nos quais não houve uma preocupação com a testabilidade do sistema e o desenvolvimento não utilizava técnicas que favorecem uma boa testabilidade, como por exemplo, o TDD;
- A garantia de que com a automação do teste, poderemos alcançar a mesma qualidade da execução manual do teste;
- Aumentar a cobertura dos testes, uma vez que com os testes automatizados, a equipe pode ganhar tempo para elaborar testes mais complexos e assim, tornar o processo de Teste de *Software* mais eficiente, em relação ao tempo e qualidade, já que será possível prover um *feedback* mais rápido aos desenvolvedores e garantir uma maior qualidade do sistema sob teste.

Quais são as razões para automação dos testes?

No livro Agile Testing, da Lisa Crispin e da Janet Gregory, são citadas as seguintes razões:

- Teste manual é demorado;
- Reduzir a probabilidade de erros das tarefas de teste;
- Liberar tempo para fazer o trabalho da melhor forma;
- Prover uma rede de segurança (se eu fizer uma mudança no código eu terei os testes, que irão me avisar se eu quebrei algo);
- Prover feedback cedo e frequente;
- Os testes que direcionarão a codificação (utilizando técnicas como o TDD e BDD) podem fazer mais do que isso (ex.: se tornam testes de regressão);
- Os testes provem documentação, aliás, são uma excelente documentação;
- ROI, com o passar do tempo o esforço gasto na automação dos testes se paga.

Um dos grandes benefícios da automação é prover *feedback* cedo e frequente, e isso é possível, principalmente com a automatização dos testes unitários e integrados (que iremos ver com mais detalhes no próximo tópico), geralmente realizados pelos desenvolvedores. Ou seja, quando falamos de automação de testes, não estamos falando apenas da área de Teste de *Software*, mas sim em práticas que devem ser comum no desenvolvimento de *software* como um todo.

Essa é uma prática que pode ser utilizada em qualquer metodologia e não demanda de um grande investimento financeiro, pois boa parte das ferramentas que auxiliam esses níveis de testes, são *softwares* livres, como por exemplo: o JUnit para criação de testes unitários em Java.

Podemos perceber pelas razões citadas que a automação dos testes pode ser uma boa escolha, frente ao teste manual, porém como vimos anteriormente, precisamos analisar vários fatores, antes de partir para a automação.

Quais tipos de Testes devem ser

automatizados?

Quando tomamos a decisão de iniciar a automação na execução dos testes, é bom avaliar quais testes serão automatizados primeiro. De acordo com o Elias Nogueira, já há certos tipos de testes que sofrem uma tendência a automação, sendo eles:

- Testes de regressão;
- *Smoke Tests*;
- Tarefas repetitivas;
- Funcionalidades críticas do *software*;
- Testes com cálculos matemáticos.

Há outros testes que devido à grande dificuldade de sua execução de forma manual, a automação torna-se praticamente essencial:

- Teste de performance;
- Testes unitários e integração.

Patrick Wilson-Welsh fez uma boa metáfora usando a pirâmide de Mike Cohn, que nos ajuda a explicar, quais tipos de testes devem ser automatizados, tendo em vista o custo de se manter os testes. No topo, temos os testes de palha são aqueles que testam o sistema de forma caixa-preta, por meio da interface. No meio temos os testes de madeira que não são por meio da interface, mas tipicamente testam muito mais do sistema, do que os nossos testes de unidade. E na última camada temos os testes dos programadores: testes unitários que verificam partes do nosso sistema, de forma isolada.

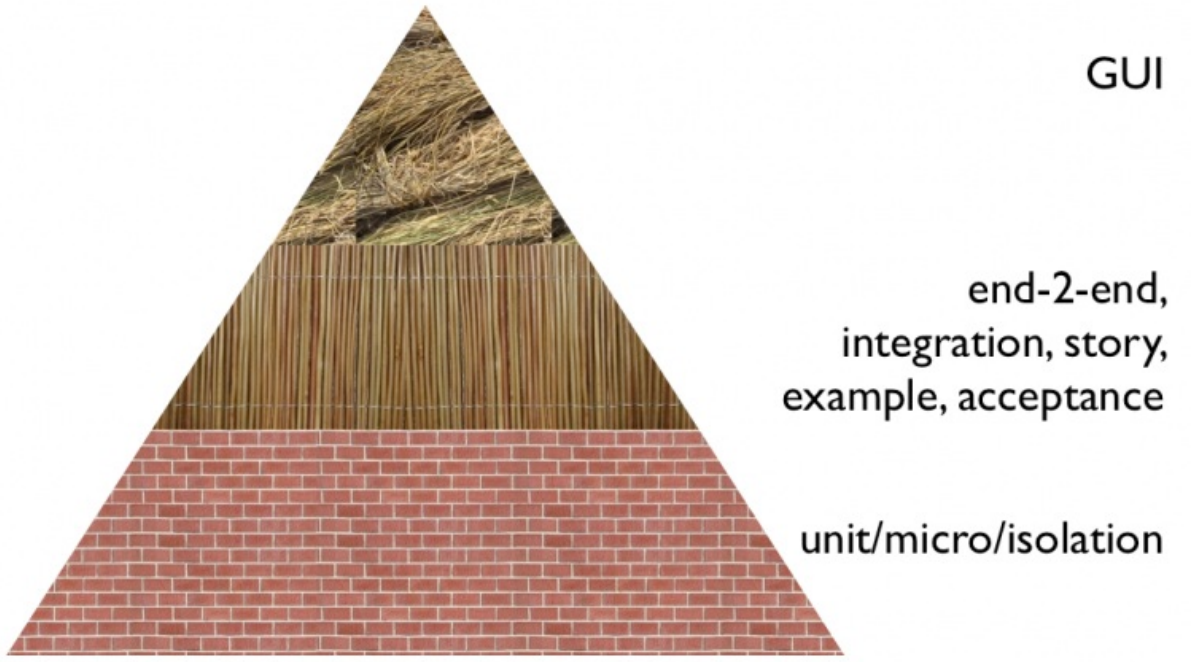


Figura 8.01 - Pirâmide da Automação (patrickwilsonwelsh.com, 11 de abr. 2010)

O interessante dessa pirâmide, é que ela costuma ser invertida em equipes tradicionais (não ágeis), pois geralmente a automação surgi como uma necessidade da equipe de Teste de *Software*, por conta dos testes de regressão.

Mas como podemos perceber com a metáfora do Patrick, precisamos apoiar a automação dos testes em uma base sólida, e essa base sólida, só pode ser construída com criação de testes unitários. E isso explica porque é importante pensar em testes sempre, testar não é uma atividade que deve ser realizada só no final, ou por apenas uma área.

Alinhamento das expectativas

Bret Pettichord diz em seu artigo "Seven Steps to Test Automation Success", que devemos tratar a automação dos testes da mesma forma que tratamos qualquer outro projeto. De acordo com o Bret, se iremos desenvolver um *software* para auxiliar a automação será necessário:

- Melhorar o processo de Teste de *Software*;
- Definir os requisitos;
- Realizar uma prova de conceito;
- O produto deve ser o melhor em testabilidade;
- Realizar a modelagem com foco na sustentabilidade;
- Elaborar o plano de implantação;
- Enfrentar os desafios do sucesso.

Já se iremos adquirir um *software* para auxiliar a automação serão necessárias outras ações, como por exemplo:

- Levantar as ferramentas existentes no mercado, tanto as comerciais quanto as gratuitas;

- Avaliar qual ferramenta atende melhor às suas necessidades. Muitas ferramentas possuem versões de demonstrações, com avaliação em períodos determinados (p. ex.: 30 dias);

- Capacitar as pessoas que irão utilizar o *software*. Infelizmente, ainda há pessoas que pensam que irão comprar uma ferramenta e as pessoas vão sair usando e tirando os melhores resultados, ledo engano, pois a ferramenta pode acabar ficando na "prateleira", caso as pessoas não sejam capacitadas;

- Envolver toda a equipe de teste, desde o levantamento das ferramentas existentes no mercado.

Mark Fewster e Dorothy Graham, lembram em seu livro "Software Test Automation - Effective use of test execution tools", que há uma expectativa de que os testes automatizados irão encontrar diversos novos defeitos. Porém, é mais provável que um teste encontre um defeito, na primeira vez que ele é executado, e geralmente, a primeira execução é manual. E se o teste já foi executado e passou, é muito menos provável que uma nova execução do mesmo teste encontre um defeito, ao menos que o teste execute um código que foi mudado ou poderia ser afetado por uma mudança realizada em uma parte diferente do *software*, ou ainda executando o teste em um ambiente diferente.

Portanto é importante saber que várias ferramentas de execução de testes são ferramentas de "repetição", ou seja, ferramentas de teste de regressão. Elas são usadas para executar testes que já foram executados. E isto é muito útil, mas o objetivo principal destes testes não é encontrar novos defeitos, e sim nos dá confiança de que o *software* está funcionando da mesma forma de antes.

Uma expectativa errônea, que às vezes ocorre quando iniciamos a automatização dos testes, é a de que teremos que ter 100% dos testes automatizados e isso faz que as pessoas busquem uma meta que muitas vezes não é coerente.

Dificilmente iremos utilizar apenas uma ferramenta para automatizar os testes. Por exemplo: se iremos testar uma aplicação *Web* podemos automatizar os testes funcionais com o Selenium, já os testes de performance podemos automatizar com o JMeter.

É importante ter em mente, que o projeto de automação não dará resultados do dia para a noite. Aliás, bem pelo contrário, pois como já dito, a automação é um investimento a médio e longo prazo.

Como vimos anteriormente, a automação é essencial para alguns tipos de testes, porém para outros tipos de teste, como por exemplo, testes de usabilidade, ela não é viável.

Além disso, a automação não deve ser medida pela quantidade de casos de teste que foram automatizados, mas sim por quais testes foram

automatizados e quanto tempo foi ganho com a automação. Em determinadas circunstâncias, pode até ser possível obter bons ganhos mesmo com apenas 10% dos testes automatizados. Por isso é importante saber o que será necessário automatizar e o que é mais prioritário. Automatizar apenas por automatizar costuma ser uma péssima ideia.

Automação no contexto ágil

A automação dos testes está entre as principais práticas ágeis, uma vez que as metodologias ágeis focam em entregas pequenas e curtas e que devem ter valor para o cliente.

E como garantir a qualidade de uma entrega que é feita em duas semanas, por exemplo?Por isso que a automação dos testes é uma prática imprescindível no desenvolvimento ágil, pois ela que irá ajudar o mesmo a se tornar sustentável, já que não é possível realizar ciclos curtos de entrega se tivermos que orçar um ciclo de teste manual.

É necessário poder executar os testes à vontade e de forma rápida (segundo o livro Agile Testing, 10 minutos), para obter *feedback* a respeito da qualidade do código. Você precisa escrever e executar os testes junto com o sistema sendo desenvolvido, para encurtar o ciclo de comunicação com o cliente e não perder tempo e esforço com mal-entendidos.

Os quadrantes de Brian Marick (figura 2) ilustra a ênfase da automação em *agile*. Mesmo para os testes do quadrante superior direito (crítica do produto / voltados ao negócio) é possível ser auxiliado por ferramentas de automação. Em outros quadrantes, não se descarta o teste manual. Na prática, apenas os testes de unidade (quadrante inferior esquerdo) são inviáveis sem automação e os testes do quarto quadrante são executados com o auxílio de ferramentas, muitos deles realizados de forma automatizada, como por exemplos os testes de desempenho.

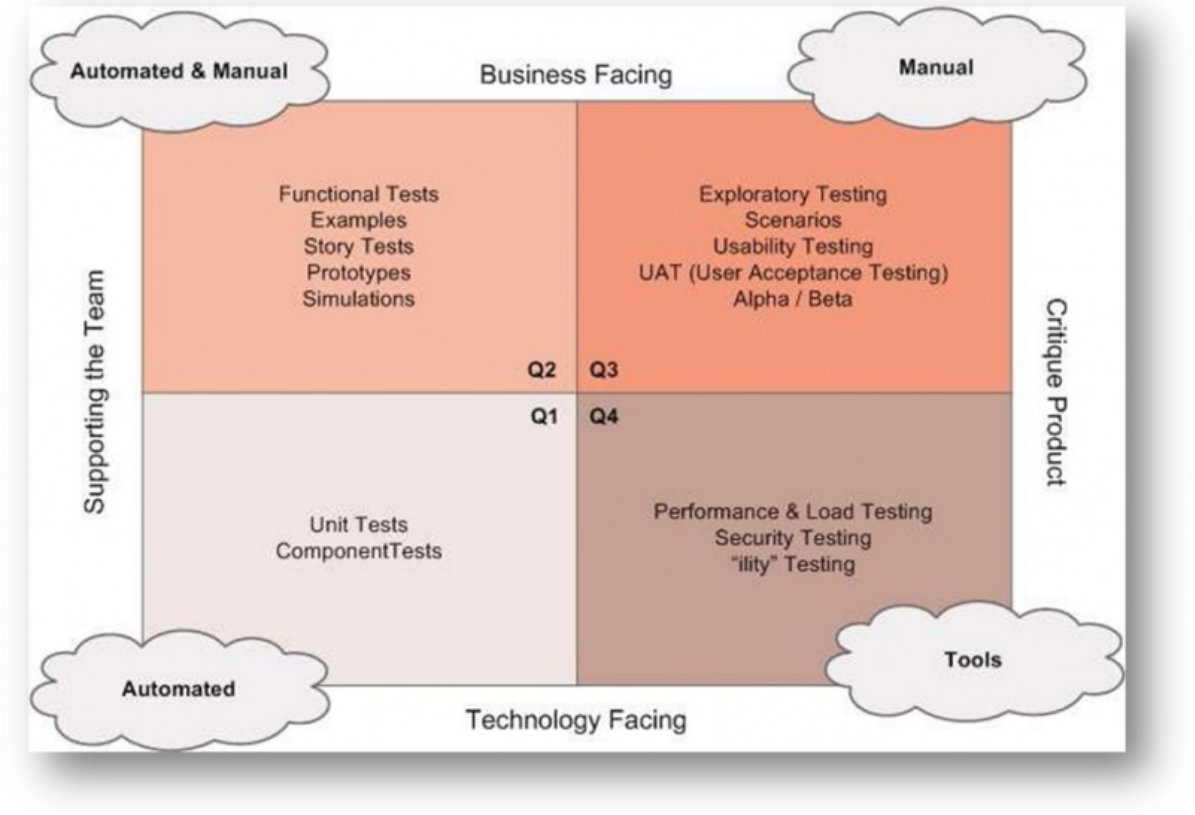


Figura 8.02 - Quadrante do Teste Ágil (Agile Vancouver, 7 de mar. 2010)

É bom lembrar que no modelo de Teste Ágil, testes são pensados não apenas como mecanismo de detecção de defeitos, mas que também é analisado seu papel de especificação do comportamento do sistema e suporte à equipe de desenvolvimento. Requisitos, testes e exemplos são aspectos diferentes das mesmas especificações.

Indo mais a fundo no papel da automação dos testes no contexto ágil, por meio do quadrante do Marick, podemos notar os seguintes pontos a respeito de cada quadrante:

- Q1: aqui entram os testes unitários e de integração, geralmente são feitos pelos desenvolvedores, mas os profissionais de Teste de *Software* podem participar da elaboração deles, por exemplo: programando em par com o desenvolvedor, quando o mesmo for criar os testes;

- Q2: os testes são realizados preferencialmente de forma automatizada, porém a automação pode não ser viável e então os testes podem ser realizados de forma manual. Neste quadrante os testes tem dois objetivos: validar o sistema com base no negócio e também dá suporte ao time de desenvolvimento, pois alguns testes, como os funcionais podem ser desenvolvidos antes mesmo da funcionalidade e assim poderão ajudar o desenvolvedor na modelagem do código, e futuramente, poderão entrar na suíte de testes de regressão do processo de integração contínua;

- Q3: os testes são executados de forma manual, devido aos objetivos dos testes e pessoas que estão executando-os;

- Q4: abrange testes mais específicos ligados a requisitos não-funcionais, e geralmente só podem ser realizados com o apoio de uma ferramenta, como é o caso do teste de desempenho. Além disso, em alguns casos se faz necessário a participação de um especialista, para poder realizar tais testes de forma efetiva, por exemplo, contratando um profissional da área de Segurança da Informação, para auxiliar nos testes de segurança.

Kent Beck, criador do XP e do TDD, tem uma postura bem radical, em relação aos testes automatizados:

"Qualquer funcionalidade que não possui testes automatizados simplesmente não existe."

A visão de Kent Beck pode até parecer radical demais, mas os testes tem um papel fundamental em um ambiente ágil, e os ciclos curtos exigem que eles sejam automatizados, sempre que possível.

Duas práticas fortemente difundidas com o surgimento de metodologias ágeis é o TDD e o ATDD. O primeiro que é sigla para *Test-Driven-Development* (Desenvolvimento Orientado a Testes), e como o próprio nome já sugere, a ideia é fazer que desenvolvedor desenvolva o sistema sendo guiado pelos testes, ou seja, os testes não são propriamente o objetivo dessa prática e sim o desenvolvimento do sistema de uma forma sustentável, que favorecer a manutenbilidade e testabilidade do mesmo.

Já o ATDD do inglês *Acceptance Test Driven Development* (Desenvolvimento

Orientado a Testes de Aceitação) faz com que o desenvolvedor desenvolva o sistema focado no negócio, utilizando testes de aceitação que validem o comportamento esperado do sistema. Esses testes de aceitação são criados, geralmente, com a participação do cliente e do analista de teste. A geração de tais testes faz com que seja gerada uma "documentação viva" do sistema, uma vez que a sua execução costuma ser feita de forma automatizada, por meio de ferramentas como o Fitnesse, por exemplo.

Ambas as práticas são utilizadas antes do desenvolvimento do código, afinal o objetivo é que elas possam orientar e ajudar o desenvolvedor no processo de codificação. O grande ganho de tais práticas, é que utilizando elas o desenvolvedor tem um melhor entendimento sobre o sistema, sem ter escrito sequer uma linha de código e os testes que foram criados com o propósito de auxiliar a modelagem do sistema poderão ser utilizados como testes de regressão, garantindo assim a corretude do sistema e facilitando futuras mudanças, uma vez que não haverá o medo de quebrar algo, já que temos uma rede de segurança, os testes automatizados.

Conclusão

A automação está longe de ser a bala de prata para o Teste de*Software*, mas pode trazer benefícios significativos para o projeto, quando utilizada adequadamente dentro do mesmo. Precisamos executá-la de forma incremental, sempre analisando quais testes que se automatizados trarão maior benefício à equipe.

Automatizar os testes não é algo simples, e diversas premissas são necessárias para que possamos trazer todos os benefícios da automação, pois caso contrário, ela irá trazer mais problemas. É importante ter em mente que quando partimos para a automação, não podemos esquecer das pessoas e do processo, afinal como diz um velho ditado: “Tolos com ferramentas , continuam tolos”.

Quando trabalhamos com ciclos curtos de entrega, a automação de testes torna-se essencial para conseguirmos garantir a qualidade do software que está sendo desenvolvido. Essa necessidade fez surgir várias ferramentas voltadas para a automação dos testes, boa parte delas distribuídas de forma gratuita.

Por fim, a resposta para a pergunta “Quando automatizar?”, só pode ser obtida por cada um, pois os fatores que podem influenciar a decisão pela

automação podem variar muito de empresa para empresa. A intenção desse capítulo foi colocar em foco os desafios, características e fatores ligados a automação de teste, que aponta para ser uma das principais tendências na área de Teste de Software nessa nova década.

Referências Bibliográficas

BASSI, Dairton; CHEQUE Paulo. Testes Automatizados. Cursos de Verão 2007 - IME/USP. Disponível em: <http://ccsl.ime.usp.br/agilcoop/files/2-Testes.odp> Acessado em: 05/02/2010

PETTICHORD , Bret. Seven Steps to Test Automation Success. Disponível em: <http://www.io.com/~wazmo/papers/seven_steps.html> Acessado em: 05/02/2010

WELSH, Patrick. Flipping the Automated Testing Triangle: the Upshot. Disponível em: < http://patrickwilsonwelsh.com/wp-includes/class-read.php?fn=99fec55bf96d4a2bfad2f6718ab0a1e83bf11f08&sid=9&sb=wpnews> Acessado em: 11/04/2010

Gregory, Janet. Focus Your Testing Using the Agile Testing Matrix. Disponível em: <http://agilevancouver.ca/sites/agilevancouver/files/speakerslides/Vancouver-Quadrants.pdf> Acessado em 07/03/2010

CRISPIN, Lisa; GREGORY, Janet. Agile Testing: A Practical Guide for Testers and Agile Teams. 1 ed. Addison-Wesley Professional. 2009.

FEWSTER, Mark; GRAHAM, Dorothy. Software Test Automation - Effective use of test execution tools. Addison-Wesley Professional. 1999.

Capítulo 9: Estimativas de Testes (APT)

Fabrício Ferrari de Campos e Karine Birnfeld

Revisora: Patrícia Silva Corrêa

Introdução

A Análise de Pontos de Teste (APT) é uma técnica de medição para a área de Teste de Software, baseada na técnica de medição de Análise de Pontos de Função (APF).

Ela utiliza como base, o tamanho do sistema em pontos de função, oriundos da aplicação da APF, considerando todas as suas funcionalidades.

A APT aplica-se apenas aos testes de caixa preta, visto que para as demais técnicas de teste (caixa branca, por exemplo), as estimativas já estão incluídas na técnica de APF.

A técnica de APT pode ser utilizada quando existem horas de teste pré-determinadas, pois alguns riscos envolvidos podem ser identificados comparando o tempo de testes estimados com APT com o tempo de testes pré-determinados. Com a APT, é possível determinar ou calcular o real valor das funções do sistema, com o objetivo de utilizar o tempo da equipe de testes o mais eficientemente possível .

Porém, em torno das estimativas em geral e da APT em específico giram algumas dúvidas como, por exemplo, as dificuldades encontradas para implantar esta técnica de estimativa, as vantagens e custo-benefício bem como pré-condições e dados necessários para sua implantação.

Importância das Estimativas x Controle

Estimar é uma atividade que pode ser de grande valia no planejamento, gerenciamento e controle dos projetos. Porém, antes de estimar, é necessário saber por que será estimado, ou seja, qual a finalidade de cada estimativa e como ela será utilizada. Estimativas devem existir, principalmente, para melhorar o processo de planejamento e controle e não servir como uma forma de cobrança de profissionais, ou aplicado, apenas porque dizem que estimar é importante.

Segundo o Robson Agapito, atualmente as estimativas são extremamente importantes para fazer previsões mais certeiras, montar cronogramas, prever recursos e custos necessários. Caso não existam estimativas, será mais difícil acompanhar o andamento do projeto, não sendo possível fazer um acompanhamento efetivo a partir do cronograma. É realmente muito difícil planejar sem estimar, mas é bom ter consciência de que estimar não é adivinhar o futuro, ou seja, não é porque existem estimativas que o planejamento será exato, como disse o Fabrício Campos.

Planejar é diferente de controlar. Estimativas são muito importantes para planejar o esforço necessário para as atividades do projeto, que consequentemente, com um planejamento mais coerente, facilitará o controle e acompanhamento do projeto.

Não é impossível controlar algo sem medir, ou melhor, não é regra, que para controlar é necessário sempre medir. Como o próprio autor da frase (Tom DeMarco), disse recentemente: " a ideia de que controle seja talvez o mais importante aspecto de um projeto de software. Mas não é. Muitos projetos foram realizados quase sem controle e produziram produtos maravilhosos, como o Google Earth ou o Wikipedia." (tradução do José Papo).

Segundo o Fabrício Campos, medir é apenas uma das formas de controlar. Porém, não é uma tarefa tão simples, e nem sempre a mais importante para uma determinada realidade.

Análise de Ponto de Teste (APT)

O tamanho do sistema a ser testado, medido através da técnica de APF, é a base da APT. Além do tamanho do sistema, existem outros fatores que devem ser considerados para estimar a atividade de testes de software.

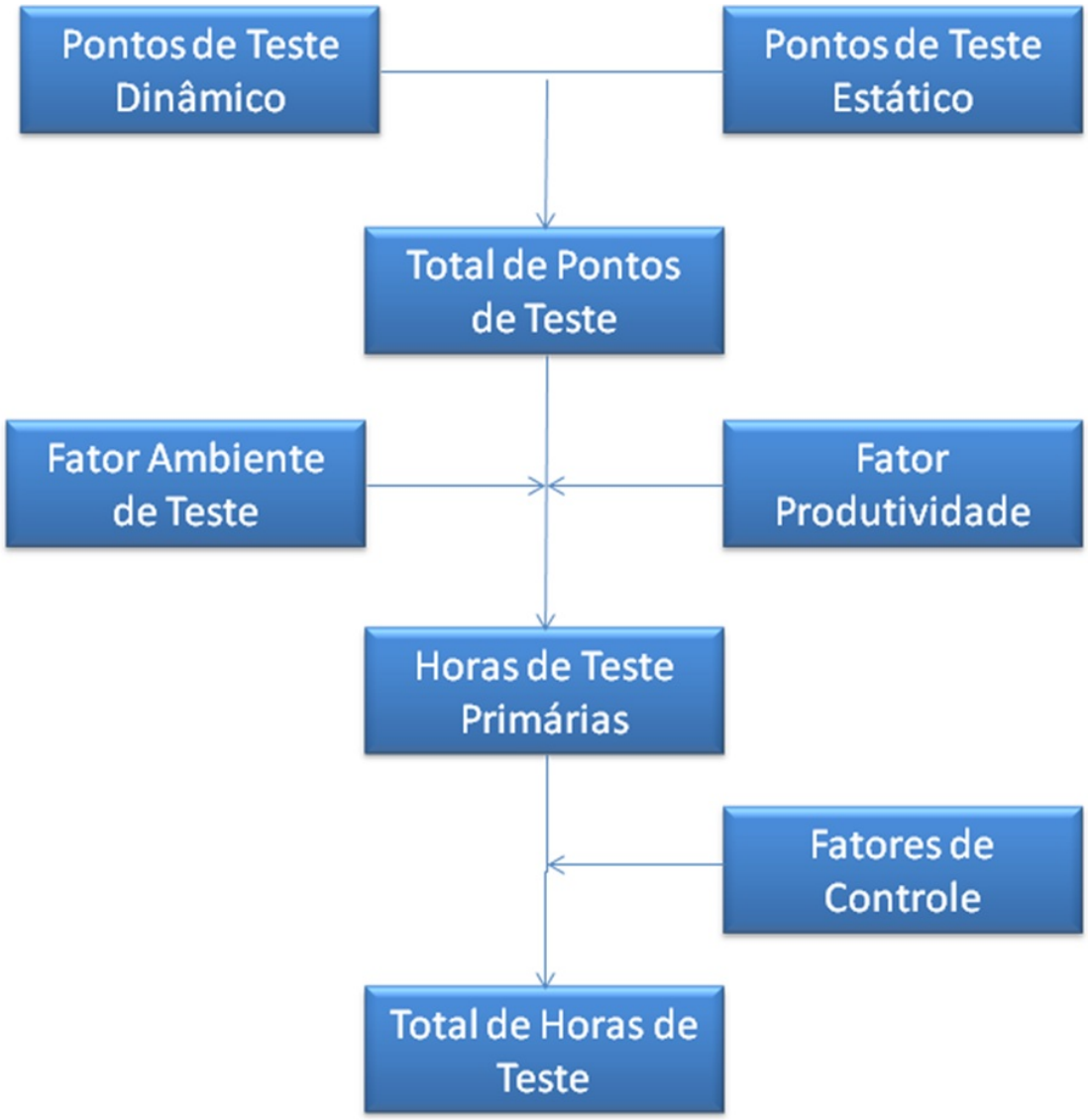
Na APT existem três elementos relevantes: o tamanho do sistema a ser testado, a estratégia de teste (seleção de componentes do sistema e características de qualidade a serem testadas e a cobertura dos testes) e o nível de produtividade. O primeiro e o segundo elemento determinam o volume do trabalho de teste empreendido (expresso em pontos de teste). Se o número de pontos de teste for multiplicado pela produtividade (o tempo total para realizar determinado volume de testes) é possível obter a estimativa de testes em horas .

A importância da utilização de uma técnica de medição específica para testes de software se deve ao fato da maioria das outras técnicas embutirem o esforço de testes no esforço de desenvolvimento, perdendo-se alguns detalhes que precisam ser considerados para estimar os testes de software de forma mais eficiente.

No livro Base de Conhecimento em Teste de Software, do Anderson Bastos, Emerson Rios, Ricardo Cristalli e Trayahú Moreira, são citados os seguintes fatores que afetam os testes:

- O grau de complexidade do processo de teste;
- O nível de qualidade que se pretende alcançar com os testes;
- O grau de envolvimento dos usuários com os testes;
- As interfaces que as funções testadas têm com os arquivos;
- A qualidade do sistema testado (o ciclo de reincidência de defeitos);
- O nível de cobertura esperado com os testes;
- A experiência e a produtividade da equipe de testes (medidos através de indicadores históricos);
- O grau de automação dos testes;
- A qualidade do ambiente de teste, até mesmo sua capacidade de simular o ambiente de produção;
- A qualidade da documentação do sistema e, em especial, dos requisitos;

A Figura 9.01 ilustra a visão geral do modelo de APT.



O número de pontos de teste necessários para os testes dinâmicos é calculado para cada função com base no número de pontos de função atribuídos à função, os fatores funções dependentes (complexidade, interfaces, uniformidade, importância do usuário e intensidade de uso) e a qualidade dos requisitos relacionados ou a estratégia de teste para as características de qualidade dinâmica. A soma destes pontos de teste atribuídos a cada função é o número de pontos de teste dinâmico.

O número de pontos de teste estático para medir as características de qualidade é calculado com base no número total de pontos de função para o sistema e a qualidade dos requisitos ou estratégia de teste para as características de qualidade estáticas. Isto resulta no número de pontos de teste estático.

O total de pontos de teste é a soma dos pontos de teste dinâmico e estático. O

número de horas de teste primárias pode ser calculado multiplicando o número total de pontos de teste pelos fatores ambiente de teste e produtividade. As horas de teste primárias representam o volume de trabalho envolvido nas atividades de teste primárias, como por exemplo, o tempo necessário para as fases de Preparação, Especificação, Execução e Conclusão.

Finalmente, o total de horas de teste é obtido adicionando subsídio para as atividades de teste secundárias (Planejamento e Controle) ao total de horas primárias. O tamanho deste subsídio, que representa o volume de trabalho envolvido nas atividades de gerenciamento, dependem do tamanho do time de teste e da disponibilidade de ferramentas de gerenciamento. O total de horas de teste é uma estimativa requerida para todas as atividades de teste, excluindo a elaboração do plano de teste.

Conforme mencionado no livro Base de Conhecimento em Teste de Software , a transformação do tamanho em esforço precisa de um trabalho adicional de calibragem, que demandará algum tempo até que as informações históricas estejam disponíveis.

Utilização da APT no Brasil

Pela discussão realizada no grupo DFTestes, a maior e mais ativa lista da comunidade de Teste de Software brasileira, percebeu-se que a técnica APT é pouco utilizada na realidade brasileira de Teste de Software.

Essa constatação pode levar a várias interpretações, mas é importante salientar que a realidade brasileira é diferente da Européia, onde a técnica é mais popular. Além do mais, a cultura brasileira também, o que faz com que os profissionais busquem outras soluções para suprir a necessidade de medição para a área de teste.

O Robson Agapito ainda ressaltou que para projetos fechados a utilização do APT é mais produtiva do que para projetos de manutenção de um software. Muitos dizem também que o APT é para projetos grandes, pois o PF inicial para se utilizar o APT é de 500 PF (o que equivale para muitas empresas em mais de 2000 horas no projeto) e não é qualquer projeto que tem este tamanho.

Outras formas de estimar

Frente ao pouco uso da APT no Brasil, procurou-se levantar durante a mesa redonda, como é feita a estimação na prática.

Percebeu-se que podemos estimar utilizando diversas técnicas e de formas diferentes, desde utilizando as reuniões do método Delphi até mesmo a utilização de técnicas de divisão de tempo em "caixas de tempo", como por exemplo utilizando a técnica de Pomodoro.

De acordo com Eduardo Gomes, o fato é que estimar não tem muito valor se não se constroem bases históricas. Só será possível comprovar a efetividade das estimativas quando tivermos bases históricas que contribuam para os ajustes necessários nos modelos e para a utilização com maior segurança dos resultados das estimativas. Antes disso, continua sendo chute. Mas talvez um chute "calibrado"; é um ponto de partida pelo qual temos que passar.

É importante ter em mente que não há uma fórmula mágica que irá te dizer como estimar, por isso é importante experimentar métodos e técnicas diferentes, e ir adaptando a realidade. Lembrando-se que para uma boa estimativa, é importante possuir experiência naquilo que será medido, seja por meio da própria experiência das pessoas, ou utilizando-se de bases históricas.

Conclusão

Estimar o tempo é uma atividade importante para um bom planejamento e gerenciamento do projeto de teste. A APT e uma técnica madura e focada na estimativa de tempo para a área de Teste de Software.

Todavia, é preciso analisar e avaliar a APT, perante a realidade do projeto no qual ela será aplicada, pois há pré-requisitos que caso não tenhamos, poderá ocasionar um mau uso de tal técnica e acabar resultando em mais problemas, do que soluções com a sua aplicação.

A realidade do mercado brasileiro de Teste de Software é diferente da de outros países, seja por questões econômicas ou culturais, portanto se faz necessário um esforço de adaptação, ao se optar por usar determinada técnica para estimar os esforços de testes.

Técnicas acopladas fortemente a pessoas, como por exemplo oplanning poker e pomodoro, podem ser revelar um bom início para a atividade de estimar, principalmente, devido ao fato que estimar envolve pessoas e muitas

vezes estimamos um trabalho criativo, que possui variáveis muito inconstantes e subjetivas.

Por fim, é importante buscar arquivar uma base histórica desde o primeiro dia do projeto, pois ela costuma ser uma boa fonte para a atividade de estimar e ao longo do projeto a estimativa acaba sendo mais efetiva, pois uma maior experiência provoca domínio sobre estimativas de determinada realidade

Referências Bibliográficas

Kusters R., A Cowderoy, F. Heemstra and E. van Veenendaal (eds).**Testpointanalysis: a method for test estimation**. Disponível em: <http://www.erikvanveenendaal.nl/UK/files/Testpointanalysis a method for test estimation.pdf> Acessado em: 04/07/2010

Bastos, A., et al. (2007) Base de Conhecimento em Teste de Software. São Paulo, Martins Fontes, 2ª Edição.

Capítulo 10: MPT.BR – Melhoria do Processo de Teste Brasileiro

Ueslei Aquino da Silva

Revisora: Carla Regina Florentino Sampaio

Introdução

Na década de 60, os *softwares* eram desenvolvidos com baixo nível tecnológico e ficavam a cargo dos analistas e programadores. O foco nesse momento era apenas **demonstrar** que o software estava funcional. À medida que a tecnologia evoluía, *softwares* mais sofisticados eram desenvolvidos, impactando diretamente na realização dos testes que, desta vez, se focalizavam: (1) na **detecção** dos defeitos; (2) nos erros e deficiências existentes no software; (3) na definição das capacidades e limitações e (4) no fornecimento de informações sobre a qualidade dos componentes, sistemas e outros produtos. Mas, o grande problema neste momento estava em quem realizava tais processos, pois eram realizados pelos próprios usuários e testadores sem experiência. Apesar de desde a década de 70 já existirem trabalhos mostrando que o teste precisava ser re-estruturado, (Em 1979 Glenford Myers publicou aquele que atualmente ainda é considerado um dos melhores livros de Teste de Software existentes no mercado, sob o título de "The Art of Software Testing" -publicado por John Wiley and Sons Inc. em edição revisada em 2004) foi a partir da década de 90 que o Teste de *Software*, enfim, recebe um novo olhar. Os holofotes agora estão sob a perspectiva da **prevenção**. Assim como comprovado por Myers, quanto mais cedo encontrar e corrigir um defeito, mas barato se torna para empresa. Desta forma, inspeções são realizadas nos artefatos do *software*, possibilitando a detecção e redução de defeitos logo nas fases iniciais do desenvolvimento.

Com a sofisticação dos *softwares*, processos de maturidade no desenvolvimento de *software* foram criados para se garantir cada vez mais a qualidade do produto. Em consequência, Processos de Teste surgiram para atribuir uma melhoria contínua aos serviços de teste. Atualmente vários são os processos destinados a Teste de *Software*. Citando alguns deles temos :

- Testability Support Model (TSM);
- Testing Maturity Model (TMM);
- Test Process Improvment (TPI), entre outros.

Alguns desses modelos tiveram origem a partir de um modelo de processo de *software*, como por exemplo o TMM, cuja base original é o CMM.

Desenvolvimento

Antes mesmo de entrar no cerne da discussão da 5ª mesa-redonda do DFTestes, cujo tema foi MPT.BR (Melhoria do Processo de Teste Brasileiro) e apresentar as opiniões dos participantes da mesa, vou de forma breve explanar o que vem a ser o MPT.BR

Tendo como fundamento toda história em torno de "Teste de*Software*" e a realidade do mercado brasileiro de desenvolvimento de *software*, a ALATS em parceria com a RioSoft dão início ao desenvolvimento do modelo chamado de **MPT.BR (Melhoria do Processo de Teste Brasileiro)** modelo de maturidade de processo de teste compatível com o MPS.BR e o CMMI. Desta forma, empresas que implementam MPS e CMMI poderão, com pequeno esforço, aumentar e comprovar o nível de maturidade da área de Teste de *Software*.

Conforme apresentado na Guia 1 do MPT:

"O MPT é voltado para a melhoria das áreas de Teste de Software de empresas de qualquer porte. O modelo é leve e possível de ser adotado por áreas de Teste de Software para apurar o seu nível de maturidade, sem com isso onerar os seus processos anteriormente implementados."

"O objetivo principal será garantir que áreas de Teste de Software de tamanho reduzido possam ser avaliadas e estimuladas a alcançarem níveis maiores de maturidade, sem que para isso tenham que incorrer em altos custos operacionais."

O Guia de implementação do MPT.BR está subdividido em níveis de maturidade, a exemplo do MPS.BR e CMMI. O nível um (1) contempla apenas a área de Gerência de Projetos. O objetivo é atender áreas de testes de todos os tamanhos, mesmo aquelas com número reduzido de profissionais.

O MPT.BR é apresentado com uma estrutura de cinco (5) níveis de maturidade, sendo o nível 1 o mais baixo e o nível 5 o mais alto. Na tabela 1, veremos como estão subdivididas as áreas de processo do MPT.BR.

Áreas de processo do MPT.BR :

Nível 1

- Gerência de Projetos de Teste - GPT

Nivel 2

- Gerência de Requisitos de Teste - GRT

Nivel 3

- Aquisição – AQU (opcional)
- Gerência de Configuração – GCO
- Garantia da Qualidade - GQA
- Medição - MED

Nivel 4

- Gerência de Recursos Humanos - GRH
- Gerência de Reutilização - GRU (opcional)
- Gerência de Riscos - GRI

Nivel 5

- Verificação - VER
- Validação – VAL

De forma a garantir a aderência a uma das áreas de processo do modelo, a

organização deverá implementar as práticas específicas e as práticas genéricas, que se aplicam a todas as áreas de processo, correspondentes ao nível de maturidade visado. A avaliação de que a unidade de teste alcançou um determinado nível será feita por meio da comprovação objetiva dos resultados alcançados e do exame das evidências (diretas, indiretas e afirmações) de que a empresa implantou cada uma das práticas específicas e genéricas para aquela área de processo e grau de maturidade visado.

Desta forma, temos a seguinte organização:

- Área de processo
  - Práticas específicas
  - Objetivos genéricos
    - Práticas genéricas

Iniciando o Bate-Papo sobre MPT

A chegada do MPT ao mercado brasileiro traz novas oportunidades de qualificação profissional para os interessados pela área de processos e também para as empresas que desejam melhorar sua área de teste baseada em níveis de maturidade.

Com esta tão nova realidade, e em processo de gestação, no mercado de Testes de *Software* brasileiro, algumas questões, quase que inevitavelmente, são levantadas:

- Ser certificado MPT vale à pena?
- Por que os "selos" de qualidade e melhoria, são tão criticados hoje em dia, principalmente pelos agilistas?
- Qual a posição do mercado em relação às empresas certificadas?
- Para uma empresa cujo *core business* não é o Teste de *Software*, é interessante a certificação?

Com o objetivo de ajudar os profissionais de teste a pensar sobre as questões colocadas, a equipe do DFTestes se reúne em uma mesa redonda e partilha opiniões e conhecimentos sobre o assunto como segue:

Ser certificado MPT vale à pena?

Os profissionais poderão se certificar como implementadores e/ou avaliadores do MPT e a meu ver, é uma ótima oportunidade para os profissionais que gostam da área de processo e/ou querem investir na área obtendo um diferencial de mercado.

Por que os "selos" de qualidade e melhoria são tão criticados hoje em dia, principalmente pelos agilistas?

Algumas experiências ouvidas sobre o MPS.BR, mostram que empresas buscaram o selo apenas para serem melhor colocadas em licitações. De início, a empresa aceita o investimento e passa por toda burocracia para implementação do processo mas, depois que o avaliador vira as costas engavetam toda documentação e volta à vida normal. Implantar processo, seguir práticas é um tanto quanto burocrático, se na organização não tiver o apoio da alta gerência para que tudo seja seguido, mesmo sem a presença do implementador e após a certificação, nada funciona. Uma solução adotada pelo MPS.BR para resolver essa situação foi colocar prazo de validade nas certificações, ou seja, a cada 3 anos a empresa passa pelo processo de reavaliação, se não se recertificar, sai da lista dos certificados em determinado nível.

Para o Shmuel Gershon, os agilistas não inventaram as críticas aos selos de qualidade. As reclamações são antigas, é só olhar para as críticas ao ISO9000 e ao Six Sigma.

Uma das limitações de todos os processos uniformizados é a uniformidade do processo. Como ele é estático e uniforme, quem tem que se adaptar é a empresa e os empregados. Empresas não são uniformes, são compostas por interações não uniformes entre pessoas não uniformes.

Empresas são diferentes e programam *softwares* diferentes. O processo para tratar de qualidade é o mesmo em um produto *web-based e embedded*? Um controlador de equipamento médico e um administrador de conteúdo? Uma empresa com 700 empregados e uma com 7?

O Shmuel é certíssimo em seus comentários, mas quanto ao MPT.BR temos uma nova perspectiva. Como explanado pelo objetivo do MPT, ele está sendo preparado para áreas de teste de todos os tamanhos, inclusive para "áreas de teste de tamanho reduzido".

Para o Fabrício Ferrari, o foco do MPT implica na fé de que existem contextos

diferentes e o tamanho é um dos parâmetros – uma boa novidade na área de estandardização de processos, que tende a ver tudo com 'tamanho único'.

Por outro lado:

- Tamanho da empresa é um parâmetro fraco sobre seu contexto;
- Comumente empresas com tamanho reduzido sentem menos a falta de processos do que grandes;
- Comentário implica que tamanho reduzido resulta em níveis menores de maturidade.

Sobre a limitação citada pelo Shmuel, acredita-se que é uma verdade, e o pior é como alguns profissionais encaram esses selos. Infelizmente, muitos ganham os seus uniformes (adorei essa analogia) e percebem que ficam bonitinhos com ele, daí então, a sua maior preocupação é deixar o uniforme bem limpinho.

Mas… e o cliente? ahh… ele terá uma primeira impressão muito boa da empresa uniformizada, e como dizem a primeira impressão é a que fica (ou não). No entanto após um tempo, o cliente vai perceber que as aparências enganam.

O mais importante para qualquer fornecedor é atender bem o cliente, isso é tão "lógico". Selos, certificações profissionais, entre outros são *plus*, primeiro você tem que comer o feijão com arroz, pra depois comer a sobremesa!

O Edwagney afirma ter o mesmo entendimento do Shamuel, além de compartilhar o material que está em (https://itqualityview.wordpress.com/), Edwagney complementa dizendo que o ponto crucial de alguns processos de qualidade, e processos em geral, não dar certo em algumas organizações é que elas não lançam mão da sua cultura, da sua política, não usam o que tem de melhor. Não adaptam um determinado padrão aos padrões já existentes na empresa.

Como experiência profissional adquiridas nas empresas por onde passou, Edwagney relata que a questão era: "Vamos implantar isso de qualquer forma." Conhece o ditado: "*Top-guela-down*"? É mais ou menos por aí…

Usar padrões e processos faz parte da inovação e da melhoria da qualidade, porém, arrisco dizer que 80% do sucesso de um projeto nesse sentido, se

ganha usando o lado bom do conhecimento e cultura da empresa.

Para Shmuel Gershon em um primeiro momento, pareceria uma questão de oferta e demanda. É igual ISO9001… se a empresa vai perder mercado por falta de certificação, então certamente vale à pena. Se a empresa consegue manter diferenciação comercial sem o certificado, então não vale à pena. Certo? O único problema na situação descrita é que no caso de uma certificação sobre testes, quem cria a demanda são as próprias empresas certificadas. Assim como nas certificações de indivíduos, existe tanta confusão ao redor do que testes e testadores fazem que é fácil apresentar uma certificação e dizer "Aqui oh, não se preocupe, nós somos certificados, por isso somos melhores que a concorrência". Ao criar a demanda, criamos um círculo vicioso onde quem não se certifica fica pra trás. No fim das contas, se uma empresa acredita que deve passar pelo processo de certificação, tudo bem, mas com cuidado e delicadeza.

Qual a posição do mercado em relação às empresas certificadas?

"Tanto o mercado nacional quanto mercado internacional estão cada vez mais exigentes por qualidade e produtividade, e menos tolerantes a variações de prazo e custo."

Uma alternativa adotada para se exigir um nível aceitável de qualidade foi a comprovação de que a empresa está desenvolvendo *software* com foco em qualidade. Esta comprovação vem com os selos de certificação tais como: CMMI, MPS.Br, entre outros.

Deve-se manter sempre em mente que é a melhoria do processo que realmente traz o aumento da qualidade e da produtividade da empresa. A certificação é um importante diferencial de *marketing*, mas não deve ser um fim em si mesma.

A melhor atitude é planejar a capacitação do processo e da equipe aos poucos, de acordo com a possibilidade financeira da empresa. É recomendada uma consultoria específica para auxiliar nesse planejamento, identificando e priorizando cada área do processo e cada treinamento necessário para a equipe de projeto.

Para uma empresa cujo core business não é o Teste de Software, é interessante a

certificação?

A melhor resposta depende de cada empresa, gerente, líder e equipe. A equipe deve se conhecer o suficiente para escolher.

Como citado anteriormente, temos diversos modelos de processo de testes. A maioria deles do exterior. Implantar um processo aqui no Brasil seria trabalhoso, além, de muito caro, pois não há instituições implementadoras/avaliadoras assim como profissionais qualificados para tal.

Para o Shmuel Gershon, depende: Uma possibilidade: Sim! – pelo menos vai ter um processo mais estruturado. Outra: Não! Onde existia incompetência caótica, agora existirá uma incompetência organizada e rigorosa. O problema desta última, é que agora ninguém mais percebe que tem que melhorar. Ou ainda: Depende! Pode ser que o dono da empresa se sinta inseguro e a certificação vai deixá-lo mais cordato para guiar a empresa. Ou pode ser que a equipe seja madura o suficiente para não deixar a certificação ou o rigor do processo atrapalhar a operação. Pode ser que o gerente de testes não queira a certificação de jeito nenhum, e certificar-se vai criar rixas desnecessárias. Pode ser que a equipe seja tão madura e com ótima performance que inserir as mudanças da certificação vai atrapalhar. Tudo pode ser.

Com o MPT, toda esta realidade muda, pois é um modelo gestado internamente possuindo, espalhados pelo Brasil, profissionais implementadores qualificados e instituições avaliadoras, além de ser muito mais barato.

"Se uma fábrica de software possui internamente profissionais que testam o software, acredito que sim. Trabalhei em uma empresa que mantém no mercado um grande ERP. A empresa possui uma fábrica de desenvolvimento de aproximadamente 70 desenvolvedores (em sua matriz) e um departamento de teste com 8 testadores. Para o mercado seria interessante mostrar para os clientes que a qualidade do software desenvolvido é assegurada por processos estabelecidos, maduros, entre outros." (Ueslei Silva)

Vários benefícios já são conhecidos quando uma fábrica de *software* implanta um processo de desenvolvimento. Com a implantação de processo de teste,

não vem a ser diferente e concatenando ambos, essa lista só tende a aumentar. Como exemplos, podemos citar:

- Aumento da qualidade do software produzido;
- Aumento da satisfação do cliente;
- Aumento da produtividade da empresa;
- Redução dos custos de desenvolvimento;
- Diminuição da rotatividade de pessoal;
- Aceleração da curva de aprendizado dos profissionais envolvidos no projeto;
- Aumento da moral da equipe;
- Entre outros.

Conclusão

Concluímos que processo é ótimo para a empresa como um todo, traz vários benefícios, mas submete a empresa a um investimento inicial caro. O processo de implementação é burocrático e necessita do apoio da alta gerência e colaboração de todos os envolvidos no projeto, caso contrário a implementação tornar-se-á um processo lento e doloroso. No fim, poucas mudanças poderão ser vistas. Uma vez sendo apoiado por todos os interessados, tudo torna-se mais fácil e rápido e assim a empresa sofrerá menos em seu processo de implementação. Contar com o apoio de uma consultoria especializada é uma prática comum, pois, garante de alguma forma a efetividade do processo.

Quanto ao fato de certificar-se, uma empresa deverá estudar vários pontos para uma avaliação de Custo X Benefício, de forma a comprovar a viabilidade do investimento a ser feito para a implementação do processo.

Referências Bibliográficas

DIAS, André. Por que investir em melhoria de processos? Pronus. Disponível em: <http://www.pronus.eng.br/artigos_tutoriais/processo_desenvolvimento/melhoria_processo.php?pagNum=0>. Acessado em: 14/03/2010.