UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MOISÉS ARMANI RAMÍREZ

CROWD TESTING:
O PODER DA MULTIDÃO EM PROL DA QUALIDADE DE SOFTWARE

São Leopoldo
2012

Moisés Armani Ramírez

CROWD TESTING:
O PODER DA MULTIDÃO EM PROL DA QUALIDADE DE SOFTWARE

Monografia apresentada como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Profª Drª Margrit Krug

São Leopoldo
2012

*Aos meus pais**, César Augusto Jacobo Ramírez** e **Maria Armani Ramírez**,*
*e meus irmãos **Mateus Armani Ramírez** e **Marcos Armani Ramírez**.*

AGRADECIMENTOS

A minha família, pelo apoio incondicional durante todos esses anos de estudo e pelos valores ensinados;

A Sarah Pimentel, minha amiga, por ter contribuído na minha escolha do tema deste trabalho;

A Margrit Krug, minha orientadora, pelos valiosos conselhos e por ter apoiado as minhas ideias;

A Universidade do Vale do Rio dos Sinos (UNISINOS), minha universidade, pelo aprendizado e experiências proporcionados.

RESUMO

Controlar a qualidade de um software é uma atividade que exige tempo, esforço e conhecimento técnico sobre teste de software e sobre o sistema a ser testado. Não há garantia de que um software esteja 100% livre de defeitos e quanto mais se tentar aproximar desse percentual, maiores serão os custos do projeto. Diversas metodologias tentam reduzir os custos em testes sem impactar na qualidade do sistema. Contudo, existe um antigo e profundo impasse: esforço necessário de teste versus tempo disponível para testes. Este trabalho aborda o conceito de *crowd testing* e apresenta como uma multidão pode contribuir na qualidade de software a fim de amenizar o impasse entre esforço e tempo. Como não há trabalhos científicos que abordem o assunto, a proposta deste trabalho foi utilizar os conceitos de *crowdsourcing* e do processo de teste de software para, então, identificar como e onde o *crowd testing* pode ser utilizado no controle da qualidade de software. Trata-se de uma pesquisa exploratória baseada em bibliografias cientificas e documentais para a definição de *crowd testing*.

Palavras-Chave: *Crowd testing*. *Crowdsourcing*. Teste de software. Qualidade de software

ABSTRACT

Control the software quality is an activity that requires time, effort and technical knowledge on software testing and the system being tested. There is no guarantee that software is 100% defect-free and the more you try to approach this percentage, the higher the costs of the project. Several methods attempt to reduce testing costs without impacting the system quality. However, there is an ancient and profound dilemma: testing effort required versus time available for testing. This paper discusses the crowd testing concept and presents how a crowd can contribute to the software quality to ease the standoff between effort and time. Since there are no scientific studies that address the subject, the purpose of this study was to use the crowdsourcing concepts and the software testing process, then identify how and where the crowd testing can be used in software quality control. This is an exploratory research based on scientific bibliographies and documentation for the crowd testing definition.

Keywords: Crowd testing. Crowdsourcing. Software testing. Software quality.

LISTA DE FIGURAS

Figura 1 - Benefícios do investimento em teste de software 50
Figura 2 - Integração entre os processos de desenvolvimento e teste 51
Figura 3 - Conceito “V” de teste de software 54
Figura 4 - Conceito “V” de teste de software - QAI 54
Figura 5 - Onze passos do processo de teste de software 55
Figura 6 - Modelo 3P x 3E do ciclo de vida do processo de teste 56
Figura 7 - Ambiente de Teste 58
Figura 8 - Fluxograma de teste 58
Figura 9 - *Heuristic Test Strategy Model* 67
Figura 10 - Formas de recompensa no *crowd testing* 76
Figura 11 - Utilização do *crowd testing* em metodologias de desenvolvimento 80
Figura 12 - Atividades de teste na metodologia *Scrum* 82
Figura 13 - *Crowd testing* como uma atividade de teste 82
Figura 14 - Como introduzir o *crowd testing* na organização 83

LISTA DE QUADROS

Quadro 1 - Categorias principais de *crowdsourcing* 24
Quadro 2 - Eventos que deram origem ao *crowdsourcing* 25
Quadro 3 - Tipos de *crowdsourcing* na inteligência coletiva 34
Quadro 4 - 10 regras de *crowdsourcing* 37
Quadro 5 - Diferenças entre garantia da qualidade e controle da qualidade 41
Quadro 6 - Definições de verificação e validação 41
Quadro 7 - Exemplo de testes estáticos e dinâmicos 42
Quadro 8 - Níveis de teste 43
Quadro 9 - Técnicas de teste estrutural 43
Quadro 10 - Técnicas de teste funcional 44
Quadro 11 - Testes de caixa-branca 45
Quadro 12 - Testes de caixa-preta 46
Quadro 13 - Técnicas de testes e os níveis de teste 47
Quadro 14 - Participantes do processo de teste 47
Quadro 15 - Papéis e responsabilidades no teste de software 48
Quadro 16 - Modelo de teste estruturado 52
Quadro 17 - Distribuição do ambiente por atributos 59
Quadro 18 - Riscos no projeto 60
Quadro 19 - Riscos associados ao desenvolvimento de software 61
Quadro 20 - Riscos associados ao teste de software 62
Quadro 21 - Técnicas de teste baseadas na experiência 64
Quadro 22 - Características de um excelente testador exploratório 66
Quadro 23 - Elementos do *Heuristic Test Strategy Model* 68
Quadro 24 - Pesquisa para obtenção da importância dos fatores de qualidade 69
Quadro 25 - Principais áreas de utilização do *crowd testing* 75
Quadro 26 - Pesquisa do Forte Consulting Group sobre *crowd testing* 77
Quadro 27 - Razões para não substituir totalmente os testes tradicionais 79
Quadro 28 - Combinação do *crowd testing* com o teste tradicional 81
Quadro 29 - Quatro passos para implantar o *crowd testing* na organização 83

LISTA DE SIGLAS

CEO	*Chief executive officer*
CMMI	*Capability Maturity Model Integration*
CVDS	Ciclo de Vida de Desenvolvimento de Software
ISTQB	*International Software Testing Qualifications Board*
P&D	Pesquisa e Desenvolvimento
MIT	*Massachusetts Institute of Technology*
MPS.BR	Melhoria de Processo do Software Brasileiro
QA	*Quality Assurance*
QAI	*Quality Assurance Institute*
TI	Tecnologia da Informação
TMap	Test Management
UNISINOS	Universidade do Vale do Rio dos Sinos

SUMÁRIO

1 INTRODUÇÃO 18
1.1 Objetivos do Trabalho 19
1.1.1 Objetivo Geral 19
1.1.2 Objetivos Específicos 19
1.2 Metodologia 20
2 CROWDSOURCING 22
2.1 Conceito 22
2.2 Origem 24
2.2.1 Threadless 25
2.2.2 Converse Gallery 26
2.2.3 iStockphoto 27
2.2.4 Procter & Gamble 27
2.3 Participação dos amadores 28
2.4 A comunidade e o meio corporativo 31
2.5 A inteligência coletiva 33
2.6 Regras básicas de utilização 36
3 TESTE DE SOFTWARE 40
3.1 Definições 40
3.2 Objetivos do teste 49
3.3 Processo de teste 50
3.3.1 Conceito "V" de teste 53
3.3.2 Ciclo de vida do teste 56
3.4 Ambiente de teste 57
3.5 Riscos 60
3.6 Técnicas baseadas em experiência 64
3.6.1 Teste exploratório 65
3.7 Fatores de qualidade do software 69
3.8 Fatores que afetam o teste de software 70
3.8.1 Relacionamento das pessoas 70
3.8.2 Escopo de teste 71
3.8.3 Ciclo de vida do teste 71
3.8.4 Qualidade dos planos de testes 72
3.8.5 Restrições de testes 72
4 CROWD TESTING 74
4.1 Conceito 74
4.2 Utilização 78
4.3 Tendências 84
5 CONCLUSÃO 86
5.1 Dificuldades e limitações 87
5.2 Trabalhos futuros 88
6 REFERÊNCIAS 90

1 INTRODUÇÃO

Durante as décadas de 1970, 1980 e 1990, os testes eram efetuados pelos próprios desenvolvedores do software, cobrindo aquilo que hoje se chama de testes unitários e testes de integração (BASTOS et al., 2007). Desde então, a busca pela melhoria do processo de teste de software, visando sua eficácia e eficiência, tem sido constante. Modelos de maturidade do desenvolvimento de software e melhoria do processo de software como o CMMI (*Capability Maturity Model Integration*) e o MPS.br (Melhoria do Processo de Software Brasileiro) trouxeram expressivos avanços para a área de qualidade de software.

Atualmente os métodos ágeis, como o *Scrum*, possuem uma proposta bastante diferente quanto ao processo de teste de software tradicional. Métodos ágeis eliminam uma série de documentações e processos burocráticos visando a rapidez na execução de projetos, mas sem deixar de lado a sua qualidade.

De acordo com QAI (2006), a qualidade do software é definida pelo atendimento dos requisitos especificados no projeto e pelo atendimento das necessidades do cliente que solicitou o software. A área de qualidade de software é dividida em dois processos: um que garante a qualidade do software, responsável pelo o processo que produz o software, e outro que controla a qualidade do produto, responsável por assegurar que o software funcione conforme as especificações (BASTOS et al., 2007).

Controlar a qualidade de um software é um processo que, além de conhecimentos técnicos, exige também conhecimento sobre o sistema a ser testado. Deste modo, a equipe de teste deve compreender o negócio do cliente bem como as suas necessidades para que seja possível entregar um software com qualidade. Contudo, testar tudo é inviável. O teste exaustivo é praticamente impossível devido às restrições do projeto como tempo, esforço e orçamento (ISTQB, 2011). Por isso, de acordo com Pressman (2006, p. 293), “você acaba de testar quando o tempo acaba ou o dinheiro acaba”.

Todos e quaisquer projetos terão suas restrições e é responsabilidade dos gerentes e suas equipes identificá-las, analisá-las e elaborar estratégias que reduzam os riscos do projeto e não interfiram na qualidade do produto. O *crowd testing* é uma recente ferramenta que surgiu para contribuir com a qualidade do software, mais precisamente, no controle da qualidade. Ele tem a característica de poder utilizar mais testadores do que atualmente uma empresa dispõe para testar um software, tendo a mesma ou maior qualidade e em um prazo

mais curto. Por isso, o *crowd testing* pode ser utilizado no processo de teste de software. Contudo, de acordo com Narayanan (2011), "não se pode ainda substituir os testes tradicionais com o *crowd testing*".

O tema central deste trabalho é o estudo do conceito do *crowd testing* e da sua aderência no processo de teste de software tradicional. Desbravar um tema totalmente novo e explicar como o poder da multidão pode ser utilizado em prol da qualidade de software configura-se como a principal motivação desta pesquisa. Entretanto, há um fator motivacional peculiar: a ausência de referências bibliográficas sobre o *crowd testing* estimulou ainda mais a investigação sobre o tema.

Este estudo se diferencia dos demais trabalhos já existentes por aprofundar os conceitos do *crowd testing* de modo que seja possível compreender como e onde esta atividade pode ser inserida dentro do processo de teste de software tradicional. Esse tema é importante justamente porque auxilia os profissionais da área na utilização do *crowd testing* dentro dos atuais projetos de teste de software de suas empresas. A qual categoria da qualidade de software o *crowd testing* pertence, quais são os métodos e tipos de testes que ele utiliza, qual é o nível de teste em que está presente, quais são as técnicas de testes que podem ser utilizadas, quais são os papéis desempenhados e quais são os fatores que afetam o *crowd testing* são as principais perguntas respondidas neste estudo.

1.1 Objetivos do Trabalho

1.1.1 Objetivo Geral

O objetivo geral deste trabalho de pesquisa é descrever, baseando-se em conceitos, como e onde o *crowd testing* poderá ser inserido no processo de teste de software tradicional.

1.1.2 Objetivos Específicos

Com a finalidade de atingir o objetivo geral, os seguintes objetivos específicos foram estabelecidos:

- Analisar o funcionamento do *crowdsourcing*;
- Descrever o processo de teste de software tradicional;
- Investigar os conceitos do *crowd testing*;

- Aprofundar os conceitos do *crowd testing*, e;
- Identificar e analisar como o *crowd testing* pode ser utilizado no processo de teste de software tradicional;

1.2 Metodologia

Do ponto de vista da forma de abordagem do problema, esta é uma pesquisa qualitativa. Foi efetuada uma análise e interpretação de uma situação do mundo real, neste caso o *crowd testing*, para descrever como esta atividade funciona dentro do processo de teste de software.

Já do ponto de vista dos seus objetivos, esta é uma pesquisa exploratória baseada em bibliografias para melhor compreender o *crowd testing*, embasar o seu conceito e identificar como a multidão pode contribuir para a qualidade de software.

E, por fim, do ponto de vista dos procedimentos técnicos, esta é uma pesquisa bibliográfica e documental. Bibliográfica porque utilizou material publicado, de domínio científico, sobre o *crowdsourcing* e o processo de teste de software, e documental por não terem sido localizados materiais sobre o *crowd testing* que tenham recebido tratamento analítico.

2 CROWDSOURCING

A seguir são apresentadas as definições e formas de utilização do *crowdsourcing*. É descrito o seu conceito e origem, qual é a forma de contribuição dos amadores, a relação entre as comunidades *on-line* e as empresas, como utilizar a inteligência coletiva e, por fim, as principais regras a serem seguidas para obter sucesso na utilização do *crowdsourcing*.

2.1 Conceito

"Ao longo da história, as empresas se organizaram de acordo com linhas de autoridade estritamente hierárquicas. Todo mundo estava subordinado a alguém" (TAPSCOTT; WILLIANS, 2007, p. 9). Já nos últimos anos, pessoas de todo o mundo começaram a se reunir para executar tarefas que antes eram restritas aos empregados de uma determinada empresa, saindo, assim, do contexto corporativo e de suas limitações. Esse novo tipo de comportamento social não exige remuneração e é executado na forma de colaboração (coletiva ou individual), podendo ser utilizado por organizações, das mais diversas áreas, na criação, no desenvolvimento e na manutenção de produtos ou serviços (HOWE, 2009). A definição para essa nova forma de trabalho foi elaborada pelo jornalista Jeff Howe, que a publicou no seu artigo *The Rise of Crowdsourcing*, na revista *Wired*, em junho de 2006. O termo surgiu pela primeira vez quando Howe chamou de *crowdsourcing* a revolução dos amadores que passaram a criar conteúdo utilizando a internet como meio de produção (HOWE, 2009).

Baseando-se no poder da multidão, Howe (2006, p. 1) definiu assim o conceito de *crowdsourcing*:

> Simplesmente definido, *crowdsourcing* representa o ato de uma companhia ou instituição tomar uma função, uma vez realizada por funcionários, e externá-la a uma rede indefinida (e, geralmente, grande) de pessoas sob a forma de uma chamada pública. Isso pode assumir a forma de *peer-production*[1] (quando o trabalho é realizado colaborativamente), mas também é frequentemente realizado por um único indivíduo. O pré-requisito crucial é o uso do formato da chamada pública e da grande rede de trabalhadores potenciais.

[1] Novo modelo de inovação e criação de valor através de colaborações auto-organizadas que produzem novos bens e serviços dinâmicos que rivalizam com os das maiores e mais bem-financiadas empresas do mundo (TAPSCOTT; WILLIANS, 2007).

O *crowdsourcing* possui forte relação com a internet, mas a tecnologia não é o ponto principal do seu conceito. Segundo Howe (2009, p. 9): “Muito mais importante e interessante são os comportamentos humanos que a tecnologia engendra, especialmente o potencial que a internet tem de interligar a massa da humanidade em um organismo próspero e infinitamente poderoso”. Nesse contexto, a tecnologia incentiva a colaboração entre as pessoas que, ao tomarem a proporção de multidão, passaram a ser percebidas por empresas inteligentes que identificaram a oportunidade de utilizar esse poder em prol dos seus produtos e serviços (HOWE, 2009). As plataformas de *crowdsourcing* funcionam como intermediárias entre dois polos antes desconectados de pessoas, um que precisa de um trabalho concluído e outro que precisa trabalhar (GERBER, 2011).

As plataformas de *crowdsourcing* funcionam sob a premissa de que as pessoas possuem capacidades e talentos superiores aos que as atuais estruturas econômicas permitem utilizar. A importância da qualidade do trabalho e não quem você é (qual a sua identidade), configura-se como outra premissa a ser respeitada. Considera-se que no *crowdsourcing* as pessoas têm prazer em transmitir seus conhecimentos e em fazer o que de fato gostam e por isso a recompensa nada mais é do que a própria colaboração. O prazer é tão grande que em muitos projetos de *crowdsourcing* as pessoas colaboram sem esperar nenhum retorno financeiro (HOWE, 2009). Contudo, Gerber (2011, p. 1) alerta: “o estresse para implementar uma ideia com recursos limitados é um pesadelo para qualquer criador”.

O fato de muitas vezes não ser cobrado nenhum valor pela colaboração em algum trabalho não significa que o *crowdsourcing* deve ser considerado como um meio de obter mão de obra barata. A multidão sabe do poder que tem ao se reunir organizadamente para executar uma tarefa, dos direitos sobre o seu trabalho exercido e é capaz de identificar quando está sendo explorada. Por isso, o primeiro compromisso das pessoas é com a comunidade, o compromisso com a empresa que solicitou o seu trabalho fica em segundo plano. Para fazer parte de uma comunidade, a empresa deve apresentar uma mentalidade corporativa diferente da tradicional e investir em mudanças para se adequar ao método de trabalho do *crowdsourcing* (HOWE, 2009).

A forma de trabalho do *crowdsourcing* é simples: trabalhar junto, mas de modo descentralizado. O trabalho é divido em pequenas partes que serão distribuídas organizadamente entre os membros da comunidade (HOWE, 2009). Para identificar como

aproveitar melhor esta forma de trabalho, Howe (2009) identificou quatro categorias principais de *crowdsourcing* que são descritas no Quadro 1.

Quadro 1 - Categorias principais de *crowdsourcing*

Categoria	Forma de utilização
Inteligência coletiva ou sabedoria das multidões	Partindo do princípio que um grupo possui mais conhecimento do que um indivíduo isolado, esta categoria utiliza a sabedoria da multidão nos mercados de previsões e na resolução de problemas
Criatividade da multidão	Utilizando o alto poder de criatividade da multidão, empresas estão identificando que algumas atividades, antes restritas ao domínio de seus profissionais, podem ser executadas com mais qualidade por pessoas além dos limites corporativos
Voto da multidão	A opinião da multidão é levada em consideração para organizar grandes volumes de informação
Crowdfounding[2]	Utiliza a renda coletiva para financiar os mais necessitados

Fonte: Elaborado pelo autor

Por não ser um conceito cientifico, o *crowdsourcing* possui definições que podem ser melhores compreendidas utilizando casos e situações práticas. De acordo com Howe (2009), ele é um termo para identificar uma série de atividades. Sua capacidade de adaptação é o que o torna difundido e poderoso.

2.2 Origem

"O *crowdsourcing* começou no movimento em prol dos softwares de código aberto (ou livres)" (HOWE, 2009, p. 7), mas, de acordo com Tapscott e Willians (2007), esse movimento é apenas um exemplo de *peering*[3]. A comunidade Linux provou que as pessoas, quando organizadas, podem produzir um produto ou serviço tão bom ou até mesmo melhor do que as organizações privadas. A prova disso foi a capacidade da multidão de criar um sistema

[2] Também conhecido como *social bank*, é um atividade que aproveita a renda coletiva, permitindo que grandes grupos substituam bancos e outras instituições como fonte de recursos financeiros (geralmente atua no microcrédito). O *crowdfounding* permite que as pessoas financiem projetos em que acreditam (HOWE, 2009).

[3] O mesmo que *peer-production* (TAPSCOTT; WILLIANS, 2007).

operacional. Isto revela que "o trabalho geralmente pode ser organizado de modo mais eficiente no contexto da comunidade do que no contexto corporativo" (HOWE, 2009, p. 7). Contudo, o movimento de código aberto não foi o único a contribuir para o surgimento desta nova atividade. De acordo com Howe (2009), o Quadro 2 apresenta os quatros eventos que contribuíram para a origem do *crowdsourcing*.

Quadro 2 - Eventos que deram origem ao *crowdsourcing*

Evento	Contribuição
Nascimento de uma classe amadora	Pessoas inspiradas e dispostas a colaborar com seus conhecimentos
Criação de um modo de produção	Baseou-se no modelo de produção do software de código aberto para criar produtos que não se restringem apenas a softwares
Proliferação da internet e de ferramentas baratas	Permitiram aos consumidores e amadores criar conteúdos e produtos de alta qualidade, que antigamente custavam caro e limitavam-se ao domínio de grandes empresas
Evolução das comunidades *on-line*	As comunidades mostraram que as pessoas podem organizar-se de modo eficiente e economicamente produtivo

Fonte: Elaborado pelo autor

Os eventos que deram origem ao *crowdsourcing* foram analisados por Howe (2009), que estudou a nova forma de trabalho de algumas empresas. Por não ser um assunto com características técnicas, examinar os trabalhos de *crowdsourcing* torna-se fundamental para melhor compreender o seu conceito. Os principais casos analisados por Howe são apresentados a seguir.

2.2.1 Threadless

A empresa Threadless surgiu após os *designers* Jack Nickell e Jacob DeHart terem se conhecido em um concurso *on-line* de *design* gráfico de camisetas. Não satisfeitos com a formatação desse concurso, os dois resolveram organizar o seu próprio concurso de *design* para, posteriormente, vender as melhores estampas de camisetas. O início do negócio se deu através da criação de um site e da implantação de um processo de trabalho simples: primeiramente as pessoas criariam estampas, depois elas seriam votadas pelos usuários da

comunidade e, por fim, os *designs* mais votados seriam estampados e os designers ganhariam camisetas gratuitamente (HOWE, 2009).

Como se percebe, o processo era participativo, onde as estampas das camisetas não seriam necessariamente criadas por profissionais da área de *design*. Qualquer pessoa com uma boa ideia e conhecimento básico em alguma ferramenta para edição de imagens poderia criar a sua estampa e enviá-la para votação. Como as pessoas gostavam de votar nas suas estampas preferidas, o processo participativo fez com que empresa rapidamente crescesse e ganhasse espaço na mídia. Naquela época, a Threadless recebia cerca de mil desenhos toda a semana e os submetia à votação da comunidade Threadless, que contava com mais de 600 mil usuários (HOWE, 2009).

A empresa não tem custos diretos na divulgação das suas camisetas porque a sua própria comunidade, que é formada por produtores e consumidores, se encarrega de fazer a propaganda dos produtos. Os produtores criam as estampas e as divulgam a fim de vencer os concursos. Já os consumidores ganham créditos (convertidos em dólares) ao trazerem amigos para a comunidade. Nota-se uma relação diferente das praticadas por grandes empresas do ramo, onde o cliente apenas pode consumir. Na Threadless você cria e consome (HOWE, 2009). De acordo com Brabham (2008), o *crowdsourcing* está revivendo a importância do consumidor no processo de *design*.

As pessoas não se associaram à Threadless por causa do dinheiro e das premiações, elas apenas queriam ter o seu trabalho reconhecido, não importava que os direitos autorais de suas criações ficassem com a empresa. E foi dessa forma que os donos da Threadless conseguiram atingir o seu principal objetivo: criar um site que pudesse atrair pessoas de interesses em comum e onde elas se sentissem bem (HOWE, 2009).

2.2.2 Converse Gallery

Outro emprego do *crowdsourcing* foi utilizado na campanha publicitária da empresa de calçados Converse, denominada Converse Gallery. Com o intuito de divulgar o tradicional modelo da marca, em abril de 2006 a agência publicitária da empresa criou uma iniciativa para receber vídeos de até 24 segundos. Os vídeos poderiam ser feitos utilizando uma câmera digital qualquer, mas desde que fosse possível perceber a paixão da pessoa pelo modelo do tênis. O sucesso foi tão grande que, além da Converse, o setor publicitário também ganhou

com esse novo modo de publicidade. Essa participação do público foi considerada um exemplo de conteúdo gerado pelo usuário (HOWE, 2009).

Conforme se constatou, os amadores começaram a contribuir de maneira expressiva e de uma forma nunca vista anteriormente. A mesma dinâmica de produção barata, excesso de talento e criatividade subutilizados e crescimento das comunidades *on-line* formadas por entusiastas com interesses comuns formam os alicerces desse tipo de trabalho (HOWE, 2009). Percebeu-se que a capacidade de criar conteúdo não se restringia mais apenas aos profissionais de uma determinada área, pois os amadores também poderiam fazer.

2.2.3 iStockphoto

Assim como a Threadless, a iStockphoto é outra empresa de *crowdsoursing*. No ano 2000, o *web designer* Bruce Livingstone decidiu criar um site para compartilhar imagens com seus amigos e colegas. O objetivo era criar uma alternativa aos tradicionais bancos de imagens que cobravam um alto valor por qualquer foto. Não demorou muito e logo fotógrafos amadores descobriram e aderiram à ideia. Em seguida, Livingstone começou a cobrar uma pequena taxa por cada imagem (uma parte para ele e outra para o fotógrafo) (HOWE, 2009).

O sucesso da iStockphoto, que não parava de aumentar, fez com que o lucro dos bancos de imagens fosse reduzido drasticamente, pois cada vez mais surgiam fotos de qualidade, tiradas por amadores, e que eram vendidas por um preço muito abaixo do que os bancos cobravam. Livingstone, então, havia criado uma comunidade de fotógrafos amadores dotados de muita criatividade e talento. Seis anos mais tarde esse sucesso todo seria vendido para a Getty Imagens por $50 milhões. A empresa manteve a filosofia da iStockphoto que, já de início, tinha um faturamento de $72 milhões (HOWE, 2009).

2.2.4 Procter & Gamble

A Procter & Gamble (P&G) não é uma empresa de *crowdsourcing*, mas tem utilizado os benefícios que esta atividade pode oferecer na inovação dos seus produtos. A P&G é uma empresa de bens de consumo, que produz alimentos, material de higiene e limpeza, produtos de beleza, e uma lista extensa de diversos produtos. Durante anos, a empresa adotou uma cultura corporativa sigilosa e estreita. Após passar por uma crise no ano 2000, a empresa resolveu reagir e substituiu o seu CEO visando promover uma mudança corporativa radical.

Contrataram A.G. Lafley que logo implantou um novo modelo de trabalho para aproximar as pessoas umas das outras e, da mesma forma, aproximar os funcionários de seus fornecedores e clientes (HOWE, 2009).

Quando assumiu, Lafley deparou-se com um cenário nada motivador: a criação de novos produtos estava praticamente estagnada. Apenas 15% dos novos produtos e inovações eram gerados fora da empresa e o seu objetivo era atingir 50% até 2007. No livro que escreveu sobre suas experiências na P&G, intitulado *O Jogo da Liderança*, Lafley (*apud* HOWE, 2009, p. 8) revela: "A P&G tem cerca de 8.500 pesquisadores e descobrimos mais 1,5 milhão de pesquisadores semelhantes com qualificação em áreas pertinentes. Por que não aproveitá-los?". Desta forma, o CEO da P&G não somente atingiu como aumentou a criação de novos produtos e inovações da empresa.

A revolução causada na estrutura organizacional da P&G só foi possível devido aos mecanismos baseados na internet que Lafley criou. Através de parcerias ele conseguiu estimular e utilizar o conhecimento coletivo de diversos cientistas anônimos (HOWE, 2009). Lafley externou alguns problemas que a P&G não conseguia resolver e identificou uma grande quantidade de cientistas dispostos a colaborar em pesquisas da empresa.

2.3 Participação dos amadores

Segundo Howe (2009, p. 19): "Cada vez mais habilidosos e capazes de uma colaboração organizada e sofisticada, os amadores estão concorrendo com êxito com profissionais de áreas que vão desde programação de computadores e jornalismo até ciências". O renascimento da atividade amadora no mundo inteiro tem impulsionado o *crowdsourcing* de maneira expressiva. Considerando que as pessoas são motivadas por colaborar espontaneamente em atividades que lhes dão prazer, durante a utilização do seu tempo livre, os amadores, com o seu comprometimento e determinação, tem muito a contribuir no contexto do *crowdsourcing* (HOWE, 2009). Segundo Gerber (2011, p. 1), "os amadores podem construir um portfólio e adquirir conhecimento que é tradicionalmente oferecido em escolas ou na indústria. Diferentemente do que ocorre na educação formal, há um retorno imediato sobre o valor de cada ideia".

Um exemplo de como o amador pode contribuir é Nick Monu, 22 anos e estudante de medicina na Brown University. Monu é um fotógrafo amador que passou a gostar dessa

atividade ainda durante a sua adolescência. Atualmente é um dos fotógrafos da iStockphoto e já recebeu $10 mil em um único mês com suas fotografias ditas amadoras (HOWE, 2009).

Quando um fotógrafo ganha $10 mil por mês em algo que considera um passatempo, a necessidade de redefinir o termo "amador" torna-se evidente. Se um químico sem nenhum treinamento prévio em biomedicina consegue um avanço-chave na busca da cura de alguma doença, será que é possível chamá-lo de amador? De acordo com Leadbeater e Miller (2004, p. 8) está nascendo uma nova designação para o termo amador: "os Pro-Am – amadores que trabalham seguindo os padrões profissionais". Para eles, o lazer não é consumido passivamente, mas sim ativa e participativamente.

A ornitologia , estudo dedicado das aves, é outro exemplo de como os amadores têm contribuído em atividades antes restritas a profissionais. Atualmente, através das facilidades de comunicação que a internet tem proporcionado, a observação de aves pode ser feita por qualquer pessoa que conheça as técnicas de observações que, agora, podem ser encontradas em blogs e web sites. Devido a esse conhecimento, observadores amadores são capazes de realizar observações e divulgá-las em suas próprias páginas na *web* ou em banco de dados de visão de aves. Todo esse esforço colaborativo gera dados que são utilizados por profissionais da área e por amadores que pretendem aperfeiçoar os seus conhecimentos (HOWE, 2009).

A observação de aves provou, mais uma vez, que o conhecimento deixou de ser restrito a um seleto grupo de profissionais. Gerber (2011, p. 1) destaca: "tarefas que envolvem criatividade não estão mais sob o domínio de profissionais especializados". Leadbeater e Miller (2004) acreditam que os Pro-Ams irão modificar o modelo de sociedade nas próximas duas décadas. Apesar disso, muitas empresas ainda julgam a inovação e a criatividade amadora como algo sem importância para os seus mercados (TAPSCOTT; WILLIANS, 2007).

O amador também pode ser utilizado para solucionar problemas mais complexos que profissionais não conseguiram resolver anteriormente. O InnoCentive é uma empresa de *crowdsourcing* que possui milhares de cientistas associados (dentre eles amadores e profissionais) dispostos a contribuir em projetos de clientes tais como Procter & Gamble (P&G), DuPont, SAP e Roche. Sempre que as equipes internas de P&D de alguma empresa não conseguem solucionar um problema, elas repassam para o InnoCentive que o disponibiliza em seu site. Assim, os cientistas podem acompanhar os projetos e manifestar interesse em participar e desenvolver uma solução. Muitos duvidaram do sucesso da empresa,

mas o InnoCentive conseguiu organizar uma frente de trabalho utilizando cientistas amadores espalhados pelo mundo inteiro. Essa forma de trabalho tem dado tão certo que, em média, os clientes do InnoCentive chegam a ganhar vinte vezes mais do que o valor pago ao solucionador. E as empresas não investem pouco: elas chegam a pagar entre $10 e $100 mil para cada solução e mais uma taxa para o InnoCentive (HOWE, 2009).

Além do surpreendente sucesso financeiro, um estudo do MIT sobre o InnoCentive revelou outro fato inesperado: a pesquisa descobriu que os solucionadores alcançavam mais êxito quando tinham menos experiência na disciplina relevante. Apesar de o resultado não parecer lógico, a P&G não teria recorrido ao InnoCentive se todos os seus problemas internos pudessem ser resolvidos por seus próprios cientistas. Howe (2009, p. 38) conclui sobre a revelação do estudo: "é um poder mágico: os não-treinados são também os não-contaminados", reforçando, assim, o potencial do trabalho de pessoas externas à empresa. Nesse contexto, Leadbeater e Miller (2004, p. 22) traçam a diferença entre profissionais das empresas e os Pro-Ams: "os profissionais estão mais propensos a entender a teoria por trás das boas práticas, enquanto Pro-Ams talvez tenham mais técnica e conhecimentos práticos".

Como último exemplo, outra corporação global recorreu ao *crowdsourcing*: a IBM. No final de 2005, a empresa resolveu reformar o sistema de patentes que durante anos lhe foi tão útil e valioso. Apenas para se ter uma ideia da dimensão, somente no ano 2007 a IBM conseguiu 3.125 patentes e gastava aproximadamente $6 bilhões por ano em pesquisa e manutenção de suas patentes. Mesmo tendo mais de 26 mil patentes somente nos EUA, a IBM queria reformar esse sistema devido ao alto custo de manutenção das patentes. Naquela época, o custo nas disputas judiciais chegava à quantia de $2 milhões e, além disso, o sistema de patentes estava falido. Era um sistema lento e confuso que implicava em desnecessários desgastes judiciais (HOWE, 2009).

David Kappos, o advogado que cuidava do portfólio de patentes da IBM, e Beth Noveck, professora de Direito da New York Law School, estavam insatisfeitos sobre o processo de obtenção de patentes norte-americana e decidiram fazer algo que pudesse melhorá-lo. Foi então que Noveck apresentou a Kappos o seu audacioso projeto: solicitar a patente para um wiki[4] e convidar o público em geral para ajudar a guiar o examinador de

[4] Software que permite aos usuários editarem o conteúdo das páginas na web (TAPSCOTT; WILLIANS, 2007).

patentes. O objetivo seria disponibilizar os pedidos de patentes *on-line* para que pessoas devidamente qualificadas pudessem analisá-los e comenta-los visando, então, tornar o processo mais ágil. Kappos estava trabalhando na ideia de usar a sua equipe interna para trabalhar em pares na revisão de patentes, mas Noveck queria utilizar o conhecimento de pessoas e comunidades *on-line* para efetuar esse trabalho (HOWE, 2009). Esse esforço de *crowdsourcing* baseou-se na ideia de trabalho do software de código aberto para criar um produto que não fosse um software. Howe (2009, p. 57) destacou a importância dessa iniciativa:

> O fato de a maior detentora de patentes do mundo, a IBM, e uma professora de direito de Yale estarem discutindo a terceirização da análise de patentes para a população era uma indicação do alcance que havia conseguido o modelo de produção definido pelo movimento de software de código aberto (HOWE, 2009, p. 57).

Embora parecesse arriscado, o plano de Kappos e de Noveck para tornar público o processo de análise de patentes estava em sintonia com os objetivos da IBM: contribuir cada vez mais em projetos de software de código aberto. A receita obtida nesses projetos é superior aos gastos no licenciamento de patentes que a IBM tinha. Além disso, as contribuições renderam inovações em seus produtos e uma boa imagem perante comunidades de software de código aberto (HOWE, 2009). Tapscott e Willians (2007, p. 39) afirmam: “Empresas inteligentes estão tratando a propriedade intelectual como um fundo mútuo – elas administram uma carteira de ativos de propriedade intelectual, sendo alguns protegidos, outros compartilhados”. Para Brabham (2008), a vantagem de não trabalhar sob as restrições das leis de propriedade intelectual é de produzir um produto melhor de forma coletiva e democrática.

2.4 A comunidade e o meio corporativo

Antes do surgimento da internet, o acesso à informação era limitado e nada fácil. Através da sua evolução a informação passou a ser distribuída e de fácil acesso. Segundo Tapscott e Willians (2007, p. 21): “O acesso crescente à tecnologia da informação coloca nas pontas dos dedos de todos as ferramentas necessárias para colaborar, criar valor e competir”. Contudo, a arquitetura da internet conspira contra sistemas fechados, pois permite que as pessoas aprendam, compartilhem e distribuam conhecimentos. Essas pessoas com interesses em comum passaram a se organizar em comunidades sob um modelo totalmente diferente dos

encontrados em uma empresa (HOWE, 2009). “A comunidade é a força organizacional básica por trás do *crowdsourcing*” (HOWE, 2009, p. 88). “Elas potencializam as motivações humanas básicas para transformar o trabalho que antigamente teria sido considerado não-remunetarivo em substancial econômico” (TAPSCOTT; WILLIANS, 2007, p. 121).

A partir do momento em que comunidades *on-line* passaram a produzir informações, elas começaram a rivalizar com a corporação pela supremacia, pois as comunidades *on-line* (a) conseguem reunir e organizar pessoas diferentes de um modo eficiente, (b) motivam a troca de ideias e ajuda mutua, (c) utilizam ferramentas tão boas ou até mesmo melhor que as utilizadas nas empresas e (d), na maior parte dos casos, os colaboradores abrem mão do direito autoral sobre as suas criações (HOWE, 2009). Todo esse conjunto favorece aos indivíduos inovadores produzirem com mais liberdade. Tapscott e Willians (2007, p. 91) destacam outro diferencial das comunidades:

> Enquanto tiverem os mecanismos para eliminar contribuições fracas, as comunidades grandes e auto-seletivas de pessoas em constante comunicação terão maior probabilidade de atribuir as tarefas certas às melhores pessoas do que uma única empresa cujo conjunto de recursos é muito menor.

“A cultura corporativa tradicional procura limitar o acesso de estranhos às informações internas, não o contrário” (HOWE, 2009, p. 135). Contudo, o aumento na terceirização indica que a empresa está deixando de lado esse comportamento. O *crowdsourcing* não se caracteriza como terceirização, mas, do mesmo modo, pode oferecer consideráveis vantagens às empresas para as quais as comunidades colaboram: eficiência na organização e execução do trabalho superior a da empresa. As comunidades *on-line* são autopoliciadas e não precisam de camadas gerenciais para controlar o desempenho e a produtividade dos colaboradores (HOWE, 2009). Por outro lado, de acordo com Gerber (2011), o *crowdsourcing* estimula a competição externa para os profissionais internos da empresa. Isso tende a trazer uma competição por melhores resultados. “Funcionários criativos vão continuar a ser contratados, mas projetos que não exigem acesso a propriedade intelectual vão ser distribuídos para plataformas de *crowdsourcing*” (GERBER, 2011, p. 1). Cabem as empresas investirem em tecnologia e na remodelação de sua estrutura com o objetivo de se tornarem mais abertas e integradas para, então, identificar como interagir com as comunidades *on-line* (TAPSCOTT; WILLIANS, 2007).

O fato das comunidades estarem repletas de pessoas talentosas, dispostas a colaborar na solução de problemas, pode contribuir para que o *crowdsourcing* funcione como um mecanismo de caça-talentos. Além do talento, muitas pessoas possuem os mesmos conhecimentos científicos que antes eram encontrados apenas no meio acadêmico e têm muito a contribuir para as organizações (HOWE, 2009). “Empresas de vanguarda têm aproveitado essa fonte de capital intelectual e mudado o aspecto de P&D” (HOWE, 2009, p. 131). De acordo com Tapscott e Willians (2007), empresas inteligentes têm estimulado ao invés de combater o crescimento de poderosas comunidades *on-line* de colaboradores talentosos.

O *crowdsoursing* pode oferecer à empresa uma força de trabalho dedicada que executará as tarefas com um custo muito baixo, mas não de graça. Criar e manter uma comunidade podem ser muito mais difícil do que se imagina. As pessoas de uma comunidade trabalham com um senso de propriedade sobre suas criações e, muitas vezes, a empresa também necessitará abrir mão dos direitos autorais ou até mesmo dividir os royalties. Além disso, a empresa não deve interferir no processo decisório da comunidade e deve estabelecer uma relação transparente, de parceria. Se as pessoas se sentirem exploradas ou enganadas, a imagem da empresa ficará prejudicada perante a comunidade e isso pode gerar altos custos para a empresa quando essa comunidade espalhar a má imagem da empresa para outras comunidades (HOWE, 2009).

Existe também a preocupação de que as comunidades virtuais utilizam demasiado tempo *on-line* que poderia ser utilizado com interações reais, feitas pessoalmente, como as comunidades reais (não virtuais) fazem. Contudo, com a evolução da internet as pessoas estão desenvolvendo novos hábitos e novas formas de comunidade (HOWE, 2009). Uma multidão pode ter bilhões de horas vagas por dia e cabem às empresas, empreendedores e qualquer pessoa com uma boa ideia identificar a melhor forma de utilizá-las. Por outro lado, segundo Brabham (2008, p. 84), existe a seguinte preocupação: “Na verdade, o *crowdsourcing* pode dar bastante força a uma reunião esperançosa do trabalhador e produto em uma economia pós-industrial de crescente alienação do trabalho”.

2.5 A inteligência coletiva

“O *crowdsourcing* requer a colaboração entre pessoas desconhecidas, dispersas no mundo todo, unidas apenas por interesses comuns” (HOWE, 2009, p. 106). Apesar dessa

diversidade, o *crowdsourcing* possui princípios igualitários: cada pessoa possui conhecimentos ou talentos que outro indivíduo poderá achar interessante e utilizá-lo. Essa diversidade e interesse em comum fazem parte da inteligência coletiva. De acordo com Howe (2009), o Quadro 3 apresenta as três formas de *crowdsourcing* na inteligência coletiva e os seus objetivos principais.

Quadro 3 - Tipos de *crowdsourcing* na inteligência coletiva

Tipo	Objetivo
Mercados de previsões ou mercados de informações	Usar o conhecimento de muitas pessoas para antecipar resultados futuros
Solução de problemas ou *crowdcasting*[5]	Disponibilizar um determinado problema para uma comunidade indefinida e diversificada de solucionadores potenciais
Idea Jam	Realizar *brainstorm on-line* por algumas semanas a fim de gerar novas ideias

Fonte: Elaborado pelo autor

A inteligência coletiva é formada por um grupo de pessoas interligadas por interesses em comum, mas quanto maior forem as características comuns, menor será a inteligência coletiva. Por outro lado, a inteligência coletiva se destaca na medida em que a diversidade dentro do grupo cresce e a capacidade das pessoas expressarem seus pontos de vista individuais seja respeitada. Ainda assim, outros fatores devem ser atendidos para que a diversidade supere a capacidade: (1) é necessário um problema real, (2) a multidão precisa ser qualificada para resolver o problema, (3) é necessário algum método que agregue e processe as contribuições individuais e (4) as pessoas escolhidas devem vir de um grupo suficientemente grande. Somente assim pode-se garantir uma variedade de abordagens e a capacidade de expressar a individualidade seja garantida (HOWE, 2009).

As chances de que alguém encontre a solução de um problema são maiores quando um grande número de pessoas aplicam métodos diversificados para tentar solucioná-lo. Da mesma forma, um grupo com pessoas escolhidas aleatoriamente tem um desempenho melhor do que um grupo formado somente com os melhores solucionadores de problemas (HOWE,

[5] Junção de *broadcasting* (transmissão) e *crowdsourcing* (HOWE, 2009).

2009). “Aproveitar a inteligência coletiva das pessoas envolve comerciar o que a multidão já sabe. Essas aplicações de *crowdsourcing* geralmente requerem pequenos investimentos de tempo e energia por parte dos colaboradores individuais” (HOWE, 2009, p. 154). Entretanto, de acordo com Brabham (2008), deve-se ter cuidado em assumir que as ideias que emergem da multidão em aplicações de *crowdsourcing* representam uma ascensão de ideias superiores através de um processo democrático. Estudos sobre a exclusão digital indicam que o usuário típico da web provavelmente seja branco, de classe média ou alta, fala inglês, com nível de educação superior e utilizam conexões de alta velocidade (BRABHAM, 2008). Deste modo, Brabham (2008) alerta que pode ser mais difícil encontrar diversidade entre uma multidão homogênea.

A essência do trabalho criativo do *crowdsourcing* está em identificar os talentos no meio da multidão. “A multidão deve ser diversificada e tida como versada na área em questão, mas os integrantes da multidão não precisam interagir entre si” (HOWE, 2009, p. 160). Howe (2009, p.160) complementa: “A interação leva à deliberação que, por sua vez, reduz a diversidade de ideias por meio da qual a inteligência coletiva frutifica”. Por isso, as individualidades devem ser preservadas, até mesmo para não contaminar a comunidade. Por outro lado, Tapscott e Willians (2007) defendem a integração entre as pessoas, porem as comunidades precisam de sistemas que permitam a interação entre os colaboradores e de líderes para gerenciar as interações. Para Gerber (2011), gerenciar a criatividade é o maior desafio do *crowdsourcing*, pois as pessoas precisam de autonomia para não ficar limitadas, mas também precisam de alguém que as digam o que fazer.

No *crowdsourcing* a qualidade do trabalho é mais importante do que saber quem é a pessoa que o executou. As melhores ideias são valorizadas e resultam em melhores status para o autor do trabalho. Naturamente, os demais integrantes do grupo tentam buscar melhores resultados para as soluções que já foram encontradas. Esse fato acaba por elevar a qualidade global do trabalho, bem como estimula o aperfeiçoamento das habilidades individuais (HOWE, 2009). “As pessoas gostam de aprender e de ensinar. A comunidade tem uma habilidade infalível de identificar seus integrantes mais talentosos e destacar o trabalho destes” (HOWE, 2009, p. 160). Deste modo, quando a empresa utilizar o *crowdsourcing* ela não terá o trabalho de identificar esses talentos, pois a própria comunidade o faz quando dispõe de inteligência coletiva.

2.6 Regras básicas de utilização

O *crowdsourcing* tem obtido aceitação positiva e tem sido visto com entusiasmo como uma possível força vital na política. Quanto mais cresce e ganha força, mais profissões ele ameaça. Seu conceito representa uma mudança radical para as empresas e certamente existem aqueles que o enxergam como algo negativo, pois acreditam que essas mudanças irão reduzir o quadro de funcionários e causar demissões (HOWE, 2009). Muitos colaboradores de *crowdsourcing* já tiraram trabalho de fotógrafos profissionais nos bancos de imagens tradicionais e carreiras já estão sendo arruinadas (BRABHAM, 2008). Por outro lado, o *crowdsourcing* acelera a globalização da mão de obra e a descentralização das empresas. Assim como a internet, ele não tem fronteiras e consegue tirar proveito das desigualdades econômicas entre países desenvolvidos e em desenvolvimento (HOWE, 2009).

Existem também preocupações de que o *crowdsourcing* impulsione uma mediocridade cultural ao ponto de multidões serem capazes de produzirem peças teatrais, músicas ou pinturas como as dos mais renomados artistas tais como Shakespeare, Beatles e Picasso. Howe (2009) afirma que isso é possível, embora seja pouco provável.

Preocupações e desconfianças semelhantes foram enfrentadas pelo o renomado autor de ficção científica, Theodore Sturgeon, ao longo de vinte anos na defesa da ficção científica. Muitas críticas foram feitas utilizando o que havia de pior nas produções dessa área e levaram o autor a criar a Lei de Sturgeon. Segundo essa lei, 90% de tudo o que é gerado não se pode aproveitar. Essa regra pode ser estendida para as mais diversas áreas e, ao utilizar o *crowdsourcing*, deve-se identificar os 10% que realmente tem utilidade (HOWE, 2009).

Outra importante regra é a regra do 1:10:89, sobre a participação de mídia. Segundo essa regra, para cada grupo de 100 pessoas de um determinado site, apenas uma criará algo útil, outras 10 votarão no que foi criado e as 89 restantes apenas consumirão o que foi criado. A regra 1:10:89 foi criada por Bradley Horowitz, o vice-presidente de estratégia de produtos da Yahoo na época, após observar como as pessoas interagiam com o Yahoo Groups, Flickr e a Wikpedia (HOWE, 2009).

As regras anteriores servem para identificar os principais contribuidores da comunidade, mas para garantir boas chances de sucesso e superar desafios, incertezas e desconfianças, sete passos devem ser considerados quando se pensa em utilizar o *crowdsourcing*: (1) envolver as pessoas desde o início, (2) gerenciar as pessoas, (3) utilizar

instruções claras, (4) saber trabalhar em comunidade *on-line*, (5) comunicação constante, (6) prover fóruns para discussões e (7) divisão correta do trabalho. Juntamente com esses passos, algumas regras devem ser respeitadas para manter os princípios básicos do *crowdsourcing*. De acordo com Howe (2009), o Quadro 4 apresenta as 10 regras de *crowdsourcing*.

Quadro 4 - 10 regras de *crowdsourcing*

Ordem	Regra	Definição
1	Escolha o modelo certo	Determine seu objetivo e escolha a categoria de *crowdsourcing* que melhor atende aos seus interesses
2	Escolha a multidão certa	Com objetivos traçados, divulgue-os nos meios mais adequados e não se esqueça de que a multidão precisa ser suficientemente grande para garantir a diversidade no grupo
3	Ofereça os incentivos certos	Crie uma comunidade vibrante e comprometida. Para isso, identifique as motivações de cada um e envolva as pessoas. Estabeleça uma forma de compensação (não necessariamente financeira). Lembre-se de que no *crowdsourcing* o reconhecimento do trabalho realizado é fundamental
4	Deixe as cartas de demissão na gaveta	Não utilize o *crowdsourcing* apenas como uma forma de reduzir custos na contratação de mão de obra barata
5	A burrice das multidões ou o princípio do ditador benevolente	Apesar de serem auto-organizadas, a multidão necessita de direção e alguém com quem possam se comunicar. Existe a necessidade de alguém para guiá-las e corrigi-las, quando necessário. Por mais que a comunidade seja auto-gerenciada, elas precisam de líderes
6	Mantenha a simplicidade e divida as tarefas	Procure dividir as tarefas em pequenas partes que podem ser produzidas de forma independente antes de serem integradas. Como muitas pessoas do *crowdsourcing* doam o seu tempo livre, tarefas muito grande podem ser de difícil execução. Lembre-se de que você terá uma multidão para executá-las. O fato de prezar-se pela diversificação dentro de uma comunidade implica na

		necessidade de haver clareza e simplicidade na divisão do trabalho
7	Lembre-se da Lei de Sturgeon	Essa lei diz que 90% de tudo que for produzido é lixo, ou seja, não possui utilidade alguma. Nem tudo o que for produzido pela multidão pode ser aproveitado ou apresenta a qualidade desejada
8	Lembre-se dos 10%, o antídoto contra a Lei de Sturgeon	Para identificar os 10% que são úteis, utilize a própria multidão para triar e selecionar os melhores trabalhos. As comunidades sabem muito bem identificar os talentos que elas possuem
9	A comunidade sempre está certa	Por mais que haja a necessidade de ter um líder, alguém que oriente e conduza os trabalhos, não tente controlar ou manipular a comunidade. Lembre-se que a multidão pode ter a proporção de bilhões de pessoas
10	Não pergunte o que a multidão pode fazer por você, mas, sim, o que você pode fazer pela multidão	As pessoas colaboram por prazer e por saberem que o seu trabalho será reconhecido e valorizado. Dentro do contexto de uma empresa tradicional, os funcionários trabalham para satisfazer as necessidades da empresa. Mude sua mentalidade e crie um ambiente agradável e participativo

Fonte: Elaborado pelo autor

"O *crowdsourcing* não é uma tarefa fácil. Ele consiste em um método novo e radical de trabalhar, e o fenômeno ainda está engatinhando" (HOWE, 2009, p. 246). Howe (2009) acredita que em curto prazo os problemas gerados pelas mudanças serão compensados pelos benefícios de longo prazo, com menos estruturas organizacionais verticais e um ambiente mais colaborativo que reconheça a qualidade do trabalho das pessoas. De acordo com Gerber (2011), esse modelo de trabalho somente terá sucesso se as pessoas estiverem motivadas para superar os desafios que o trabalho criativo oferece. Segundo Brabham (2008, p. 86), "pesquisas são necessárias para entender como os membros de uma multidão se sentem em relação ao seu papel como trabalhador para as empresas, e não examinar somente os casos de sucessos".

3 TESTE DE SOFTWARE

Sistemas de software estão cada vez mais presentes no nosso dia-a-dia e com certeza a maioria das pessoas já teve a experiência de um software não ter funcionado como o esperado. Softwares mal testados podem gerar diversos problemas, incluindo financeiros (ISTQB, 2011). “Durante as décadas de 1970, 1980 e 1990, os testes eram efetuados pelos próprios desenvolvedores do software, cobrindo aquilo que hoje chamamos de testes unitários e testes de integração” (BASTOS et al., 2007, p. 11). O nível de cobertura de teste permitia que diversos defeitos não fossem detectados ou eram encontrados já em produção. Com o passar dos anos novas tecnologias surgiram tornando os sistemas mais complexos e exigindo mais qualidade dos softwares. A solução foi aprimorar a atividade de teste criando um processo paralelo ao de desenvolvimento com técnicos especializados (BASTOS et al., 2007).

Atividades de teste de software foram introduzidas nos modelos de maturidade e melhoria de software, tais como o CMMI e a MPS.BR, respectivamente, e demandaram a criação de um processo de teste de software independente do processo de desenvolvimento para o cumprimento de novos padrões de qualidade. Para tal atividade, conhecimentos de técnicas específicas e profissionais especializados foram necessários para a execução do processo de teste de software (BASTOS et al., 2007).

“Com a ajuda do teste é possível medir a qualidade do software em termos de defeitos encontrados, por características e requisitos funcionais ou não funcionais do software” (ISTQB, 2011, p. 11). A aplicação correta de métodos e ferramentas, revisões técnicas, gerência e controle, levam à qualidade do software que é confirmada durante o teste (PRESSMAN, 2006).

3.1 Definições

Teste de software, segundo QAI (2006, p. 63): “É uma atividade de controle de qualidade”. De acordo com Myers (2004, p. 27), “teste é o processo de execução de um programa com a intenção de encontrar erros”.

Quanto ao termo qualidade, esse possui duas definições: a primeira, do ponto de vista do fabricante de software, é atender os requisitos. A segunda, do ponto de vista do cliente, é atender suas necessidades. O atendimento dessas necessidades representa a eficácia do produto e superá-las demonstra a eficiência de todo o processo de desenvolvimento e testes

(QAI, 2006). Segundo Pressman (2006, p. 349): “Qualidade de software é a satisfação dos requisitos funcionais e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas que são esperadas em todo o software desenvolvido profissionalmente”. A qualidade de software é dividida em duas categorias: a garantia da qualidade e o controle da qualidade. De acordo com QAI (2006), o Quadro 5 apresenta as diferenças entre esses dois métodos.

Quadro 5 - Diferenças entre garantia da qualidade e controle da qualidade

	Garantia da qualidade	**Controle da qualidade**
Método	De prevenção	De detecção
Preocupação	Com o processo	Com o produto ou serviço
Responsável	Equipe da qualidade	Time de testes
Propósito	Estabelecer e avaliar o processo que produz um produto ou serviço	Verificar se os atributos especificados estão (ou não) presentes no produto ou serviço
Principal objetivo	Identificar pontos fracos no processo e melhorá-los	Detectar defeitos no produto ou serviço

Fonte: Elaborado pelo autor

O profissional de garantia de qualidade não deve estar envolvido na atividade de controle de qualidade, a não ser para validá-lo (BASTOS et al., 2007). Deste modo, duas equipes ou times são necessários para garantir e controlar a qualidade do software. Ainda assim, pode-se destacar mais uma diferença entre esses dois métodos: enquanto a garantia da qualidade realiza atividades de verificação, o controle da qualidade executa atividades de validação (embora a verificação seja utilizada em revisões). De acordo com Pressman 2006; QAI 2006; Rios e Moreira 2006, o Quadro 6 apresenta as definições de verificação e validação.

Quadro 6 - Definições de verificação e validação

Autor	**Verificação**	**Validação**
Pressman	Garantir que o software implementa corretamente uma função específica	Garantir que o software construído corresponde aos requisitos do cliente
QAI	Garantir que o sistema (software, hardware, documentação e o funcionário)	Assegurar fisicamente que o sistema opera de acordo com o plano

	cumpre as normas e processos da organização, com base em métodos de revisão ou que não execute o código	executando as funções do sistema através de uma série de testes que podem ser observados e avaliados
Rios e Moreira	Realizar inspeções/revisões sobre os produtos gerados pelas diversas etapas do processo de teste	Avaliar se o sistema atende aos requisitos do projeto (usuário). Os testes unitários, de integração e de aceitação podem ser classificados como testes de validação

Fonte: Elaborado pelo autor

Em outras palavras, verificação responde a questão "Nós construímos corretamente o sistema?", enquanto validação visa responder "Nós construímos o sistema correto?" (QAI, 2006). "A primeira pergunta diz respeito ao que foi construído, e a segunda diz respeito ao entendimento do que era para ser construído" (BASTOS et al., 2007, p. 31).

Quando se fala em verificação e validação, é fundamental diferenciar duas técnicas de teste: estática e dinâmica. "Teste estático é realizado usando a documentação do software. O código não é executado durante o teste estático. Teste dinâmico requer que o código esteja em um estado executável para realizar os testes" (QAI, 2006, p. 117). A maior parte dos testes de verificação é do tipo estático, enquanto a maioria dos testes de validação são testes dinâmicos e, de acordo com QAI (2006), o Quadro 7 apresenta alguns exemplos de testes estáticos e dinâmicos.

Quadro 7 - Exemplo de testes estáticos e dinâmicos

Testes estáticos	Testes dinâmicos
Revisão de viabilidade	Teste unitário
Revisão de requisitos	Teste de integração
	Teste de sistema
	Teste de aceitação

Fonte: Elaborado pelo autor

Os exemplos de testes estáticos e dinâmicos ocorrem em uma determinada sequência que é representada por diferentes níveis de testes. O teste estático deve ser realizado antes do teste dinâmico (e pode se estender ao longo do processo de teste) para, ainda nas fases iniciais do projeto, detectar erros nas documentações antes de virarem defeitos no código (ISTQB,

2011). De acordo com QAI (2006), o Quadro 8 apresenta os níveis de teste presentes no processo de teste de software.

Quadro 8 - Níveis de teste

Nível	Objetivos
Teste de verificação	Realizar testes estáticos nas documentações
Teste unitário	Verificar se as funções do sistema funcionam adequadamente
Teste de integração	Verificar se as funções do sistema que envolve mais de uma aplicação ou banco de dados são executadas com precisão
Teste de sistema	Simular a operação do sistema e verificar se ele funciona corretamente
Teste de aceitação do usuário	Testar o sistema no mundo real. Isso significa que o usuário irá interagir com o sistema e verificar se ele funciona corretamente

Fonte: Elaborado pelo autor

As técnicas de teste são agrupadas em duas categorias dependendo se elas derivam da descrição de uma função do programa ou da descrição de uma estrutura interna do programa. Conjuntos de teste baseados em análise estrutural tendem a descobrir erros que ocorrem durante a codificação do programa, enquanto conjuntos de teste baseados em análise funcional tendem a descobrir erros que ocorrem na implementação de requisitos ou na modelagem (QAI, 2006).

"O teste estrutural garante suficiente testes da implementação de uma funcionalidade" (QAI, 2006, p. 123). Para isso, é necessário acessar a implementação, identificar a massa de teste que contemple toda a sua estrutura e testá-la. "A técnica não determina o funcionamento correto da aplicação, e sim da estrutura" (BASTOS et al., 2007, p. 143). De acordo com Bastos et al. (2007), o Quadro 9 apresenta as principais técnicas de teste estrutural.

Quadro 9 - Técnicas de teste estrutural

Técnica	Descrição	Exemplos
Estresse	Determinar o desempenho do sistema com volumes esperados	• Espaço suficiente alocado em disco • Comunicação de rede adequada
Execução	Determinar se o sistema funciona com proficiência	• Transações rodando em tempos adequados • Hardware e software otimizados
Contingência	Determinar de o sistema retorna a	• Indução de um erro

(recuperação)	um status operacional depois de uma falha	• Avaliação da adequação de um *backup* dos dados
Operação	Determinar se o sistema pode ser executado em status operacional normal	• Manual e treinamento dos operadores • Verificação dos procedimentos de gerenciamento de configuração
Conformidade (processo)	Determinar se o sistema foi desenvolvido em concordância com padrões e procedimentos	• Padrões seguidos • Documentação completa
Segurança	Determinar se o sistema está protegido segundo os padrões de segurança da empresa	• Acesso negado • Procedimentos locais

Fonte: Bastos et al. (2007) modificado pelo autor

Com relação ao teste funcional, esse garante que os requisitos sejam propriamente satisfeitos pela aplicação (QAI, 2006). A preocupação desse tipo de técnica não é como o processo ocorre, e sim quais são os seus resultados. Em outras palavras, pode-se dizer que o teste funcional é realizado para assegurar que as especificações e os requisitos do software foram atendidos (BASTOS et al., 2007). De acordo com Bastos et al. (2007), o Quadro 10 apresenta as principais técnicas de teste funcional.

Quadro 10 - Técnicas de teste funcional

Técnica	Descrição	Exemplos
Requisitos	Determinar se o sistema é executado conforme o especificado	• Validação do especificado versus implementado • Verificação das políticas e dos regulamentos
Regressão	Verificar se alguma coisa mudou em relação ao que já estava funcionando corretamente	• Mudança de segmentos funcionais do sistema • Mudança manual de procedimentos corretos
Erro no manuseio	Prevenir ou detectar erros para depois consertá-los	• Falha introduzida no teste • Erro reincidente

(usabilidade)		
Suporte manual	Preparar os dados para processamento e usar os dados fornecidos pelo sistema	• Informação de *input* (entrada) • Tomada de ações baseadas nas informações contidas nos relatórios
Integração	Assegurar que a interconexão entre as aplicações funciona corretamente	• Cenário de teste de integração
Controle	Analisar o aplicativo com um olhar negativo e assegurar que aquilo que poderia sair errado foi adequadamente protegido	• Seleção das transações e verificação da possibilidade de reconstrução do processamento de cada transação na base de teste
Paralelo	Comparar resultados do aplicativo atual com a versão anterior	• Comparação dos resultados obtidos com as versões nova e antiga para garantir a equivalência

Fonte: Bastos et al. (2007) modificado pelo autor

O teste estrutural pode ser chamado de teste de caixa-branca e o teste funcional de teste de caixa-preta (ISTQB, 2011). O teste de caixa-branca assume que a lógica interna do código é conhecida para a execução de testes (QAI, 2006). "Caminhos lógicos internos ao software e colaborações entre componentes são testados, definindo-se casos de testes que exercitam conjuntos específicos de condições e/ou ciclos" (PRESSMAN, 2006, p. 318). De acordo com QAI (2006), o Quadro 11 apresenta os principais testes de caixa-branca.

Quadro 11 - Testes de caixa-branca

Técnica	Definição
Cobertura de sentença	Executa todas as sentenças pelo menos uma vez. No teste de sentenças derivam-se casos de teste para executar sentenças específicas, normalmente para se aumentar a cobertura
Cobertura de decisão	Executa cada decisão pelo menos uma vez
Cobertura de condição	Executa todas as possíveis saídas de cada condição pelo menos uma vez, ou seja, executa todos os resultados possíveis de uma decisão

Fonte: Elaborado pelo autor

Por outro lado, o teste de caixa-preta foca em testar a função de um programa conforme suas especificações, desconsiderando a estrutura interna do software. Ele determina se as combinações das entradas e as operações produzem os resultados esperados (QAI, 2006). “O teste de caixa-preta refere-se a testes que são conduzidos na interface do software” (PRESSMAN, 2006, p. 318). Segundo Pressman (2006, p. 327):

> O teste de caixa-preta tenta encontrar erros das seguintes categorias: (1) funções incorretas ou omitidas, (2) erros de interface, (3) erros de estrutura de dados ou acesso à base de dados externa, (4) erros de comportamento e (5) erros de iniciação e término.

Especialmente em projetos grandes e complexos, o número de combinações de entrada pode ser muito grande, sendo impossível cobrir os testes de todas as situações devido à restrição de tempo. De acordo com QAI (2006), o Quadro 12 apresenta os três principais testes de caixa-preta que gerenciam a quantidade de dados de entrada a serem testados.

Quadro 12 - Testes de caixa-preta

Técnica	Definição
Particionamento de equivalência	Uma classe de equivalência é um subconjunto de dados que é representativo de uma classe maior. Particionamento de equivalência é uma técnica para testar classes de equivalência, em vez de realizar testes exaustivos de cada valor da classe maior. Por exemplo, um programa que edita os limites de crédito dentro de um determinado intervalo ($10.000 - $15.000) teria três classes de equivalência: • Menor do que $ 10,000 (inválida) • Entre $10,000 e $15,000 (válida) • Maior do que $15,000 (inválida)
Análise do valor limite	Essa técnica consiste no desenvolvimento de dados e casos de teste que focam nos limites de entrada de uma determinada função. Utilizando o mesmo exemplo do limite de crédito, a análise de valor limite testará: • Limite mínimo: uma unidade abaixo e uma acima ($9,999 e $10,001) • Dentro do limite ($10,000 e $15,000) • Limite máximo: uma unidade abaixo e uma acima ($14,999 e

	$15,001)
Dedução de erros	Casos de testes são desenvolvidos baseados na intuição e experiência do testador. Testes intuitivos podem ser úteis para identificar testes específicos que não são facilmente identificados pelas técnicas formais

Fonte: Elaborado pelo autor

Diferentemente do teste de caixa-branca, que é executado no início do processo de teste, o teste de caixa-preta é executado em praticamente todos os níveis de teste. De acordo com QAI (2006), o Quadro 13 apresenta a relação dessas técnicas de teste com os níveis de teste.

Quadro 13 - Técnicas de testes e os níveis de teste

Níveis	Caixa-Branca	Caixa-Preta
Teste unitário	X	
Teste de integração	X	X
Teste de sistema		X
Teste de aceitação		X

Fonte: QAI (2006) modificado pelo autor

Além de conhecer as técnicas, para verificar se foi construído corretamente o sistema e validar se foi construído o sistema correto, o testador deve: (a) entender completamente o sistema, (b) entender completamente a tecnologia que está sendo implantada, (c) possuir criatividade, percepção e conhecimento do negócio e (d) compreender a metodologia de desenvolvimento utilizada (QAI, 2006). Contudo, o testador não é o único responsável no controle da qualidade do software. De acordo com Bastos et al. (2007, p. 37), o Quadro 14 apresenta os participantes do processo de teste de software :

Quadro 14 - Participantes do processo de teste

Participante	Responsabilidades
Líder do projeto de teste	Técnico responsável pela liderança de um projeto de teste específico, normalmente relacionado a um sistema de desenvolvimento, seja um projeto novo ou uma manutenção
Arquiteto de teste	É o técnico responsável pela organização da infraestrutura de teste: montagem do ambiente de teste, escolha das ferramentas

	de teste e capacitação da equipe para executar seu trabalho nesse ambiente de teste
Analista de teste	Técnico responsável pela modelagem e pela elaboração dos casos de teste e pelos scripts de teste
Testador	Técnico responsável pela execução dos casos de teste e scripts de teste

Fonte: Elaborado pelo autor

Para QAI (2006), a equipe de teste usualmente é composta por o gerente de teste (ou líder de teste) e os testadores. O Quadro 15 apresenta as principais responsabilidades desses papéis.

Quadro 15 - Papéis e responsabilidades no teste de software

Papel	Responsabilidade
Gerente de teste	• Planejar e estimar o teste • Elaborar a estratégia de teste • Revisar os artefatos de análise e de desenvolvimento • Gerenciar os esforços de teste • Assegurar que todos os critérios de entrada dos testes foram atendidos • Supervisionar os testes de aceitação
Testador	• Desenvolver os casos de testes • Preparar a massa de teste • Revisar os artefatos de análise e de desenvolvimento • Executar os testes • Utilizar ferramentas de teste automatizado para testes de regressão • Preparar a documentação de teste • Reportar e rastrear os defeitos

Fonte: Elaborado pelo autor

As atribuições de cada papel dependem do contexto do produto e projeto da organização. Por isso, uma pessoa pode realizar mais de um papel dentro de um determinado projeto e realizar um papel diferente em outro (ISTQB, 2011).

3.2 Objetivos do teste

O objetivo principal do processo de teste é encontrar o maior número possível de defeitos no software (BASTOS et al., 2007). Entretanto, QAI (2006) destaca que teste não é: (a) um estágio ou fase do projeto, (b) somente encontrar problemas no código, (c) um último exame no software e (d) *debugging*[6]. De acordo com ISTQB (2011), testes podem possuir objetivos diferentes: (1) encontrar defeitos, (2) ganhar confiança sobre o nível de qualidade, (3) prover informações para tomada de decisão e (4) prevenir defeitos. Pressman (2006, p. 316) complementa os objetivos do teste: "Ao mesmo tempo, os testes devem exibir um conjunto de características que atinge o objetivo de encontrar a maioria dos erros com um mínimo de esforço".

A eficiência em cumprir o objetivo do teste está na estratégia de testes adotada, que deve ser desenvolvida nos artefatos de testes tais como: plano de teste[7], casos de teste[8], scripts de teste[9] e dados de teste. Quanto ao objetivo de teste, esse será obtido a partir dos requisitos de negócio, reuniões com os *stakeholders*[10] e outros documentos do projeto (BASTOS et al., 2007).

Entender os riscos é outro objetivo do teste a ser considerado no projeto de teste. Segundo Bastos et al. (2007, p. 253): "O risco é uma perda em potencial para a organização e pode ser medido através da análise de riscos" e por isso devem fazer parte da estratégia de testes.

Por fim, a motivação dos testes se resume na seguinte afirmação: simplesmente testa-se software porque os desenvolvedores são incapazes de desenvolver um software livre de defeitos (QAI, 2006). Infelizmente, os desenvolvedores têm um interesse oculto em demonstrar que o software está livre de erros e que funciona de acordo com as especificações

---

[6] Processo de análise e correção lógica, sintática, e outros erros identificados durante os testes (QAI, 2006).

[7] Documento de planejamento do projeto (BASTOS et al., 2007). Descreve o escopo, abordagem, recursos e cronograma das atividades de teste (ISTQB, 2010).

[8] Uma descrição específica de um teste a ser executado (BASTOS et al., 2007). "Conjunto de valores de entrada, precondições de execução, resultados esperados e pós-condições de execução desenvolvidas para um determinado objetivo ou condição de teste" (ISTQB, 2010, p. 16).

[9] Uma especificação de procedimento de teste, especialmente em testes automatizados (ISTQB, 2010).

[10] Todos os participantes e envolvidos em um determinado projeto (QAI, 2006).

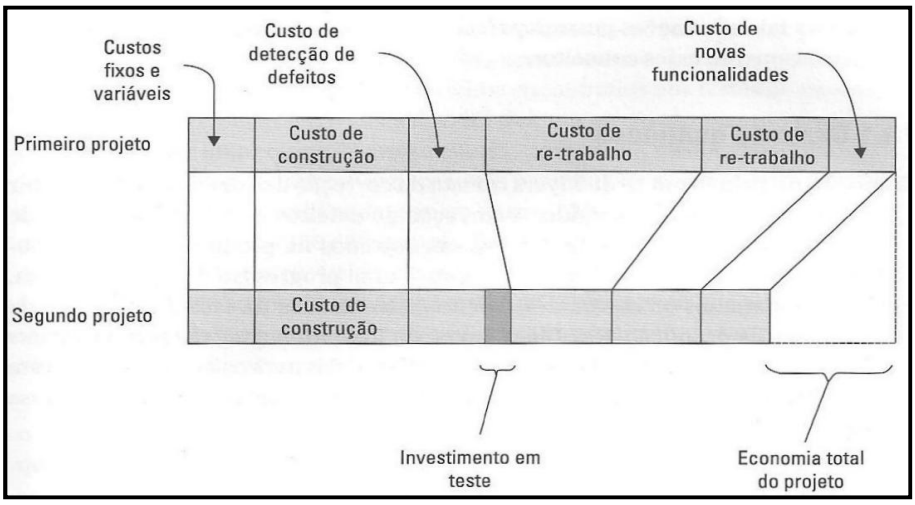
do cliente (PRESSMAN, 2006). Deste modo, justifica-se a necessidade dos testes serem preparados e executados por uma equipe especializada.

3.3 Processo de teste

Uma visão comum do processo de teste é de que ele consiste apenas da fase de execução, como executar o programa (ISTQB, 2011). Na verdade, existem atividades de teste antes e depois da fase de execução. Segundo Pressman (2006, p. 288): "O teste frequentemente responde por mais esforço de projeto que qualquer outra atividade da engenharia de software". Quando executado como sendo uma atividade dentro do processo de desenvolvimento, o teste não é capaz de garantir a qualidade do software, pois ele apenas consegue validar os requisitos do negócio. Analistas de sistemas e até mesmo os usuários não possuem o devido conhecimento técnico necessário para a execução de testes. Com a adoção de um processo de teste de software, defeitos podem ser detectados mais cedo, ao iniciar o projeto de teste paralelamente ao projeto de desenvolvimento (BASTOS et al., 2007).

Quanto mais cedo forem encontrados erros, menor é o custo de correção dos erros e maior é a probabilidade de corrigi-los corretamente (MYERS, 2004). Por isso é preciso investir em testes e não somente do desenvolvimento de software. De acordo com Bastos et al. (2007), a Figura 1 apresenta os benefícios do teste no custo do software, mostrando ser possível reduzir o custo total do software (desenvolvimento e manutenção) quando o teste for aplicado corretamente.

Figura 1 - Benefícios do investimento em teste de software

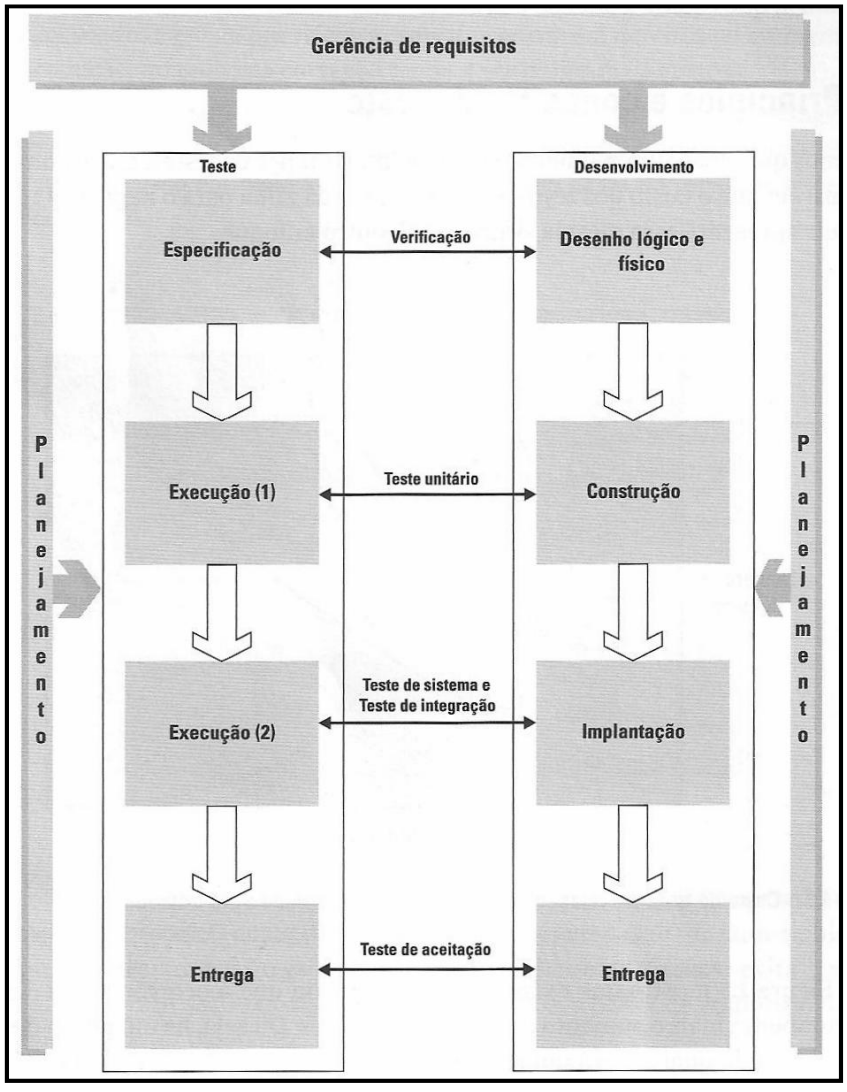


Fonte: Bastos et al. (2007)

Contudo, Bastos et al. (2007) destacam a importância de eliminar os seguintes pré-conceitos referente ao processo de teste: (a) o testador é um inimigo do desenvolvedor, (b) a equipe de testes pode ser montada com os desenvolvedores menos qualificados, pois qualquer um pode testar sistemas e (c) quando estiver tudo pronto, o software seguirá para o pessoal fazer o teste. Há de se eliminar estas barreiras para que o processo de teste possa ser implantado com mais facilidade e menos resistência por parte das equipes (principalmente de desenvolvimento).

Existem diversas formas de integrar os processos de desenvolvimento e de testes. A Figura 2 apresenta o modelo de integração proposto por Rios e Moreira (2006):

Figura 2 - Integração entre os processos de desenvolvimento e teste



Fonte: Rios e Moreira (2006)

Para melhor entender como funciona o processo de teste (o lado esquerdo na Figura 2), de acordo com Bastos et al. (2007), o Quadro 16 apresenta um modelo de teste estruturado destacando o ciclo de vida, atividades e documentos de testes.

Quadro 16 - Modelo de teste estruturado

Ciclo de vida de testes
1. Inicio 2. Planejamento 3. Especificação 4. Execução 5. Término
Atividades desenvolvidas no ciclo de vida de testes
1. Definição do escopo do projeto 2. Participação na revisão da documentação de desenvolvimento 3. Elaboração da lista de requisitos de teste usando os requisitos do sistema 4. Elaboração do planejamento dos testes (Plano de Teste e Estratégia de Teste) 5. Preparação do ambiente de teste 6. Elaboração dos casos de teste 7. Preparação da massa de teste 8. Execução dos testes (programas e *scripts*) 9. Registro dos defeitos encontrados 10. Documentação dos resultados dos testes 11. Revisão dos resultados dos testes 12. Manutenção da documentação de teste (bibliotecas de casos de testes etc.)
Documentos do processo de teste
1. Estratégia de teste 2. Análise de risco 3. Plano de Teste 4. Casos de Teste 5. *Script* de Teste 6. Registro de Defeitos

Fonte: Bastos et al. (2007) modificado pelo autor

O modelo de teste estruturado apresentado no Quadro 16 dever ser entendido como um exemplo, uma vez que os padrões de cada empresa podem ser diferentes (BASTOS et al., 2007).

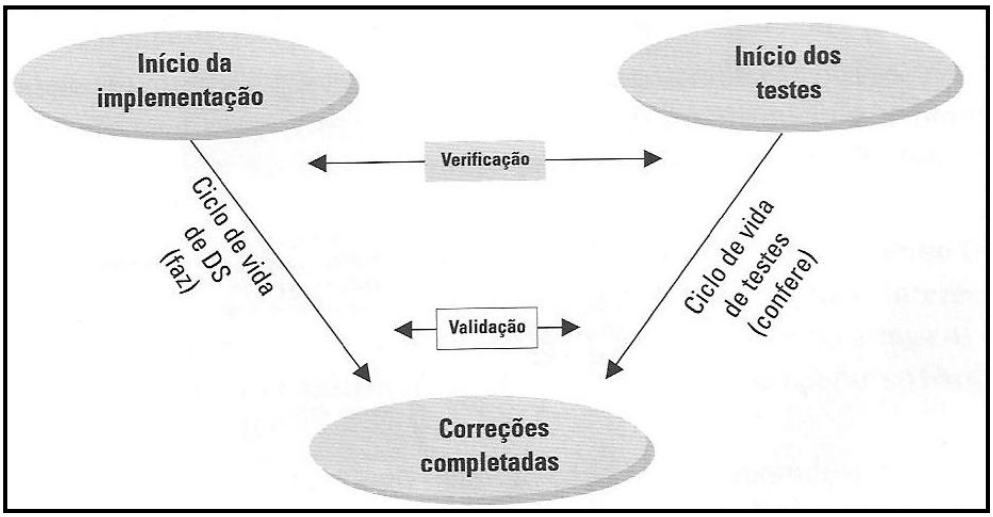
3.3.1 Conceito “V” de teste

O ciclo de vida de testes presume que sejam realizados testes ao longo de todo o processo de desenvolvimento (BASTOS et al., 2007). Segundo ISTQB (2011, p. 21): “Não existe teste isolado; a atividade de teste está intimamente relacionada com as atividades de desenvolvimento de software”. Deste modo, testes e desenvolvimento são interdependentes, pois cada saída do processo de desenvolvimento serve de entrada para o processo de teste. Em pontos pré-estabelecidos, os resultados do processo de desenvolvimento são inspecionados para determinar a exatidão das implementações (QAI, 2006).

Para garantir integridade no processo de teste, se deve utilizar uma equipe de testes independente, com o propósito de testar o software e não de desenvolvê-lo. Deste modo, a equipe de desenvolvimento utiliza uma metodologia para desenvolver o sistema e a equipe de testes utiliza outra específica para testá-lo (QAI, 2006). Segundo Bastos et al. (2007, p. 40): “A experiência tem demonstrado que não se alcançam bons resultados quando os testes são realizados pelos profissionais do próprio grupo de desenvolvimento de um projeto”. A experiência tem mostrado também que as pessoas não conseguem enxergar alguns dos seus próprios erros. Sendo assim, testes ineficazes são esperados quando o sistema é testado utilizando uma única metodologia para desenvolvimento e teste (QAI, 2006).

O conceito “V” de teste considera que o processo de desenvolvimento e teste é iniciado no mesmo momento, usando as mesmas informações. A equipe de desenvolvimento se encarrega de fazer o sistema enquanto a equipe de teste tem a tarefa de conferir as implementações e documentos. Durante todo o ciclo de vida de desenvolvimento de software (CVDS) a equipe de teste irá executar verificações e validações (BASTOS et al, 2007). Segundo Bastos et al. (2007, p. 41): “Se os grupos trabalharem juntos e de maneira integrada, o alto nível de riscos que caracteriza os projeto de desenvolvimento de software irá decrescer”. Na prática, o conceito “V” de teste pode ter diferentes níveis, conforme o projeto e o produto (ISTQB, 2011). De acordo com Bastos et al. (2007), a Figura 3 ilustra o conceito “V” de teste.

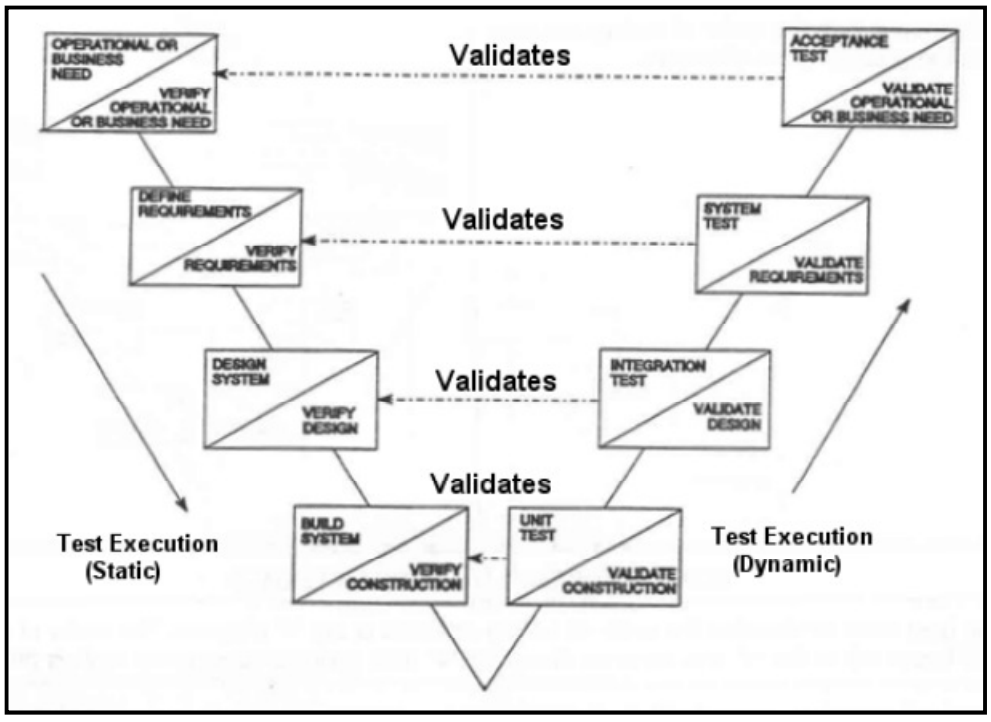
Figura 3 - Conceito "V" de teste de software



Fonte: Bastos et al. (2007)

Já QAI (2006), conforme Figura 4, apresenta o conceito "V" de teste de software mostrando como os níveis de teste interagem com os de desenvolvimento e análise.

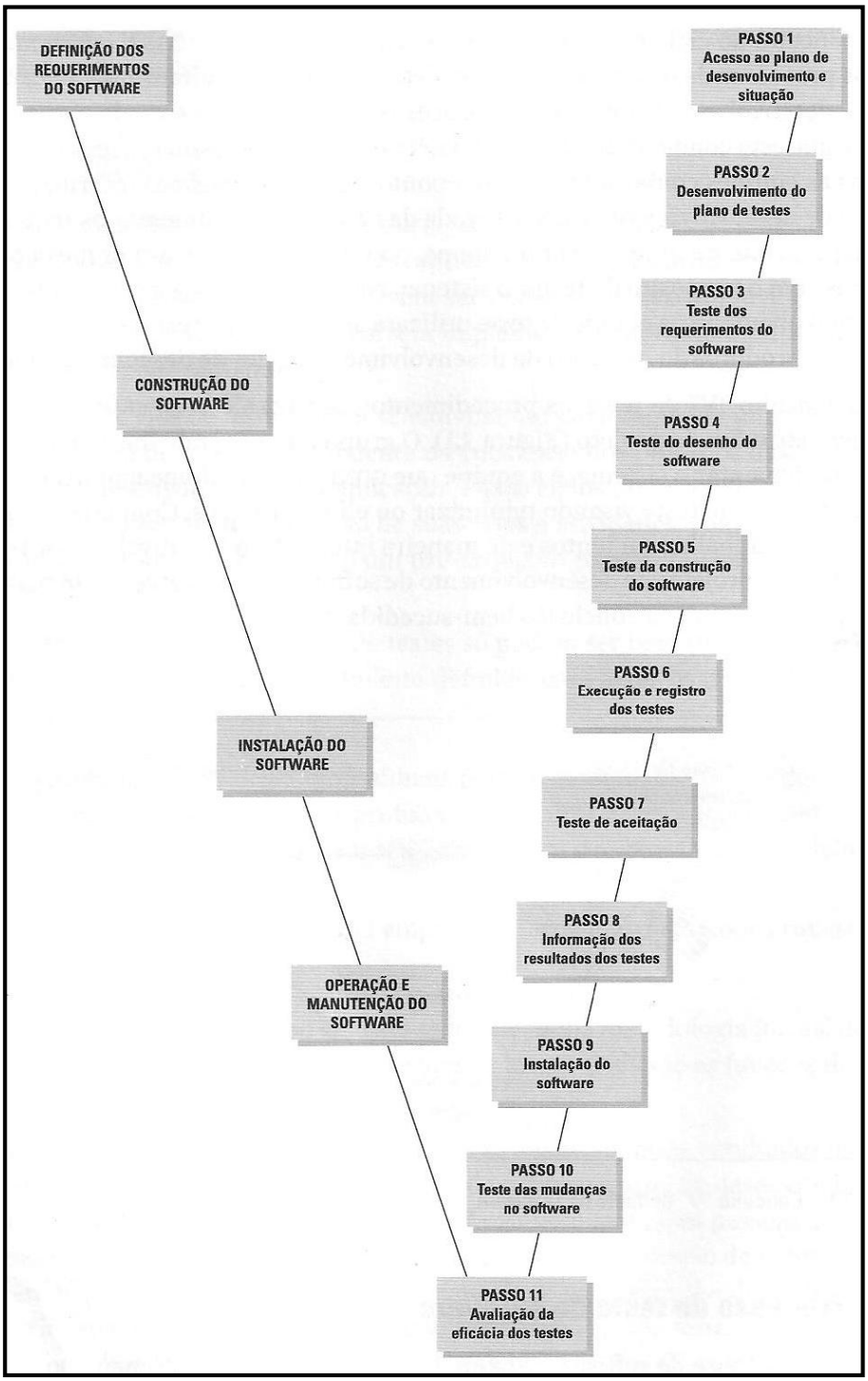
Figura 4 - Conceito "V" de teste de software - QAI



Fonte: QAI (2006)

O processo de teste de software baseia-se no conceito "V" de teste. De acordo com Bastos et al. (2007), a Figura 5 apresenta os onze passos do processo de teste de software.

Figura 5 - Onze passos do processo de teste de software



Fonte: Bastos et al. (2007)

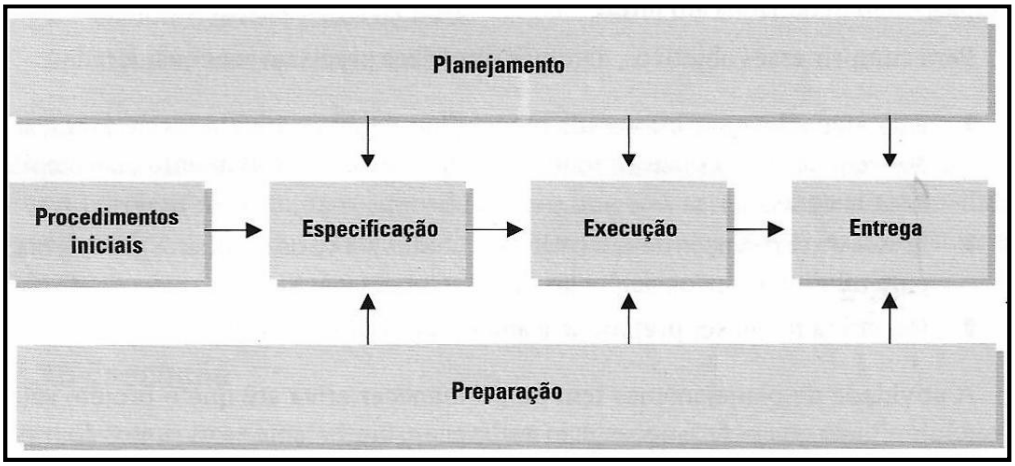
Os primeiros quatros passos do conceito "V" de teste usam verificação como o principal meio para avaliar a exatidão dos resultados entregues pelo desenvolvimento. Os três passos seguintes utilizam a validação para testar o sistema no modo executável. O último

passo representa os dois times de desenvolvimento e teste avaliando a eficácia dos testes (QAI, 2006).

3.3.2 Ciclo de vida do teste

O ciclo de vida do teste envolve testar continuamente o software. Isso significa testar o software desde o início do projeto. Baseado no conceito "V" de teste, o ciclo de vida do teste tem início no mesmo momento em que o processo de desenvolvimento inicia (QAI, 2006). De acordo com Rios e Moreira (2006), a Figura 6 apresenta um modelo do ciclo de vida do teste baseado na metodologia de *Test Management* (TMap). O modelo, que foi elaborado pelos autores, é chamado de Modelo 3P x 3E.

Figura 6 - Modelo 3P x 3E do ciclo de vida do processo de teste



Fonte: Rios e Moreira (2006)

A primeira etapa (Procedimentos Iniciais) sugere a análise dos requisitos do projeto através da técnica de verificação, onde revisões serão realizadas com o intuito de garantir que os requisitos estejam completos, precisos e sem ambiguidades. Além disso, um plano inicial deverá ser desenvolvido contendo as atividades que serão executadas ao longo do projeto. A segunda etapa (Especificação) refere-se ao desenvolvimento de casos de teste e roteiros de teste. Contudo, a elaboração desses documentos não se restringe a esta fase, pois devem ser desenvolvidos e revisados durante o andamento do projeto. A terceira etapa (Execução) diz respeito à execução dos testes conforme foram especificados na etapa anterior e ao registro dos resultados encontrados. A quarta e última etapa (Entrega) representa a finalização do

projeto de teste, onde serão entregues todos os artefatos de testes gerados durante o projeto bem como a apresentação final dos resultados obtidos (BASTOS et al., 2007).

As etapas Planejamento e Preparação são executadas paralelamente as outras etapas do ciclo de vida do teste porque são atividades de apoio. No planejamento, a estratégia de teste e o plano de teste são desenvolvidos para minimizar os riscos e orientar a execução das demais etapas. Essa atividade deve permanecer ativa até o final do projeto para que seja possível controlar possíveis desvios. Por outro lado, a etapa de preparação serve de apoio para as atividades operacionais do processo de teste. Basicamente, sua responsabilidade é preparar o ambiente de teste para que não haja obstáculos durante a execução dos testes. É fundamental que haja um processo de gerenciamento de configuração para controlar e armazenar todas as alterações feitas nos artefatos de teste (BASTOS et al., 2007).

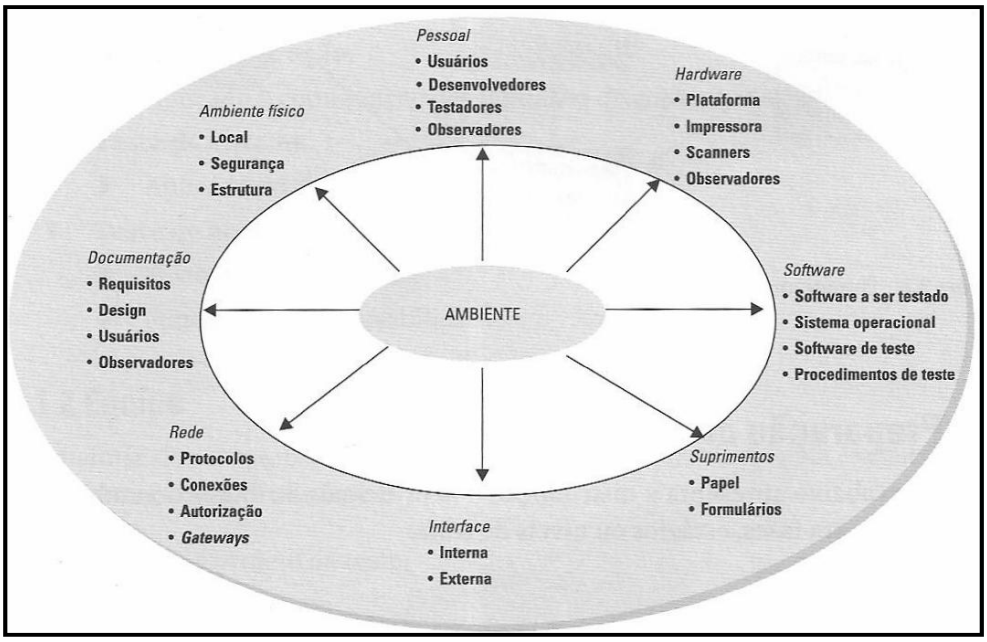
3.4 Ambiente de teste

Ambiente de teste é o "ambiente que contém hardware, instrumentação, simuladores, ferramentas de software e outros elementos de suporte necessários à realização de um teste" (ISTQB, 2010, p. 10). De acordo com Bastos et al. (2007) , a Figura 7 apresenta toda a estrutura que envolve o ambiente de teste. Ele é planejado antes de iniciar a elaboração dos cenários de teste e alguns outros elementos devem ser considerados na preparação do ambiente, tais como a criação da massa de dados de teste, o modelo de dados e a configuração dos softwares usados (BASTOS et al., 2007). Segundo Rios e Moreira (2006), ao definir o ambiente de teste, deve-se considerar: o sistema operacional, a arquitetura do sistema, a identificação dos componentes, o meio de acesso ao sistema, a linguagem de programação utilizada e a conectividade entre os ambientes.

O escopo do ambiente de teste é definido de acordo com o nível de teste ser executado. "Quanto maior o nível, mais o ambiente de teste deverá ser capaz de reproduzir as características do ambiente de produção" (BASTOS et al., 2007, p. 78). Considerando o ciclo de vida de teste, o ambiente deve ser planejado em dois momentos: na Estratégia de Teste e no Plano de Teste. Num primeiro momento identificam-se as necessidades básicas e durante o projeto informações detalhadas serão analisadas. Por outro lado, para ISTQB (2007) a preparação do ambiente de teste deve iniciar ainda na fase de modelagem e estender-se até alguns dias antes de iniciar a execução dos testes de sistema. De acordo com Bastos et al.

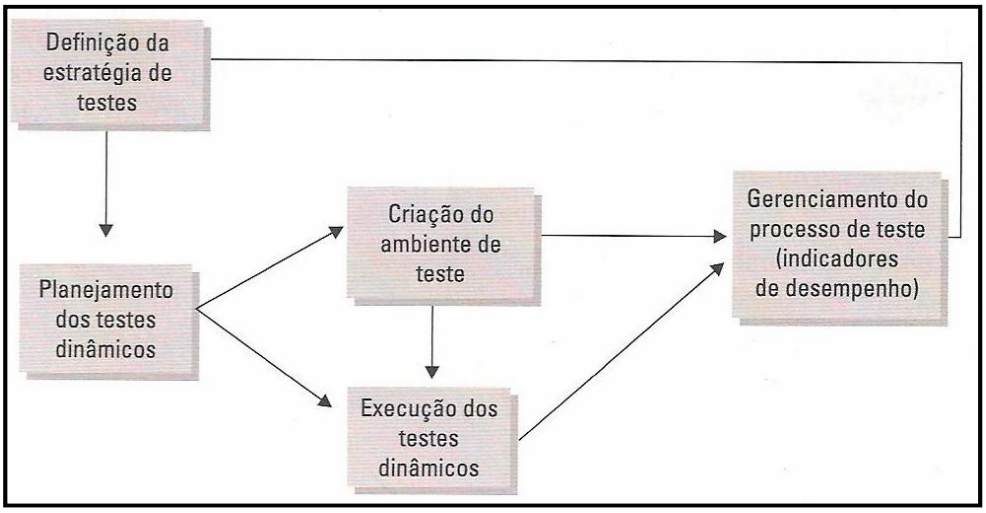
(2007) , a Figura 8 apresenta um exemplo de fluxograma para a preparação do ambiente de teste e o Quadro 17 mostra a distribuição dos principais atributos do ambiente a ser definido em fases, estágios ou níveis de teste.

Figura 7 - Ambiente de Teste



Fonte: Bastos et al. (2007)

Figura 8 - Fluxograma de teste



Fonte: Bastos et al. (2007)

Quadro 17 - Distribuição do ambiente por atributos

Atributos	Nível			
	Unitário	Integração	Sistema	Aceitação
Escopo	*Unidade individual*	*Unidades agrupadas*	*Sistema inteiro*	*Simulação da produção*
Equipe	*Desenvolvedores*	*Desenvolvedores e testadores*	*Testadores*	*Testadores e usuários*
Volume de dados	*Pequeno*	*Pequeno*	*Grande*	*Grande*
Origem dos dados	*Criação manual*	*Criação manual*	*Criação por processos*	*Dados reais*
Interfaces	*Não existem*	*Não existem*	*Simuladas*	*Reais/simuladas*
Ambiente	*Desenvolvimento*	*Desenvolvimento*	*Testes*	*Produção*

Fonte: Bastos et al. (2007)

Ao preparar um ambiente de teste isolado e o mais próximo possível do ambiente de produção, a equipe de teste poderá detectar erros reais que, provavelmente, não seriam descobertos durante a fase de desenvolvimento. Além de ser isolado, o ambiente deve ser restrito à equipe de teste e com dados representativos (BASTOS et al., 2007). Por isso que, de acordo com QAI (2006), os testes em cada nível de teste devem ser executados em ambientes separados a fim de garantir a integridade desses. Embora o volume de dados varie de acordo com o nível de teste, sugere-se que a base de teste tenha no máximo 30% do tamanho da base de produção. Outra vantagem é que um ambiente específico para testes permite que a equipe de desenvolvimento continue desenvolvendo sem afetar os testes que estão sendo executados no mesmo momento (BASTOS et al., 2007).

A gerência de configuração é importante no momento da preparação do ambiente para controlar e garantir que o que foi testado é o que será entregue ao cliente. “A garantia do ambiente de teste está diretamente relacionada à garantia de qualidade do produto” (BASTOS et al., 2007, p. 85). “As configurações do ambiente de teste deve também espelhar o ambiente de produção” (QAI, 2006, p. 244). A tarefa de controlar o ambiente é fundamental dentro do processo de teste de software e, de acordo com ISTQB (2007), dependendo da complexidade do sistema a ser testado, haverá um impacto expressivo no planejamento e custo para a preparação de um ambiente de testes.

3.5 Riscos

Risco, segundo ISTQB (2010, p. 50) é um "fator que pode resultar em consequências futuras negativas; normalmente expresso em termos de impacto e possibilidade". "Risco é usado para se decidir quando começar e onde deverá se testar com mais frequência; o teste é utilizado para reduzir o risco da ocorrência de um efeito indesejado, ou para reduzir seu impacto" (ISTQB, 2011, p. 54). Quando se analisa os riscos de um projeto, procura-se identificar os fatos cujas ocorrências poderão acarretar perdas para a empresa que, na maior parte, estão associadas à prejuízos financeiros para a organização (BASTOS et al., 2007).

O que faz um risco se tornar uma perda é a ameaça de sua ocorrência. A ameaça, por sua vez, é a capacidade de alguém explorar a vulnerabilidade de um sistema e pode ser reduzida quando é controlada. A vulnerabilidade pode ser uma falha de desenho, implementação ou operação que permite a concretização de uma ameaça e surge quando os meios de controles são insuficientes ou inadequados para evitar a ocorrência de um risco (BASTOS et al., 2007). De acordo com QAI (2006), o controle pode reduzir o efeito ou a frequência de ocorrência do risco.

Os riscos são divididos em dois grandes grupos: os riscos no projeto e os riscos do produto. Riscos no projeto são os aqueles que impedem o projeto de atender seus objetivos e, de acordo com ISTQB (2011), o Quadro 18 apresenta os possíveis riscos no projeto que as equipes de teste devem estar atentas.

Quadro 18 - Riscos no projeto

Fatore/Problema	Riscos

Fatores organizacionais	• Falta de conhecimento da equipe
	• Reciclagem e treinamento pessoal
	• Fatores políticos como
	  • Problemas com testadores comunicando suas necessidades com os resultados dos testes
	  • Dificuldade para passar informações (defeitos) encontradas nos testes e revisões, não permitindo o aprimoramento do desenvolvimento e da prática do teste
	• Atitudes impróprias em relação às expectativas do teste (ex.: a não

	valorização dos defeitos encontrados durante os testes)
Fatores Técnicos	• Problema na definição correta do requisito • Até que ponto os requisitos podem ser satisfeitos dadas restrições diversas • A qualidade da modelagem, do código e do teste
Problemas do fornecedor	• Falhas de terceiros • Problemas contratuais

Fonte: Elaborado pelo autor

Por outro lado, os riscos do produto representam potenciais falhas no software que comprometem a qualidade do produto final. De acordo com ISTQB (2011), são exemplos de riscos do produto: software instável, o dano potencial que o sistema ou hardware pode causar a uma pessoa ou empresa quando não funcionar corretamente, software com características pobres (funcionais e não funcionais), aplicação com integridade de dados e qualidade pobres e software que não cumpre com seus objetivos.

Testes baseados em risco podem reduzir a probabilidade da ocorrência de um risco do produto quando abordado ainda no início do projeto. Identificar um risco serve para orientar as principais atividades do processo de teste (planejamento, especificação, preparação e execução do teste). Depois de identificados, os riscos são aproveitados para decidir quais técnicas de testes utilizar, avaliar o nível de detalhamento do teste, priorizar a execução dos testes dos requisitos mais críticos e para estabelecer outras ações que também possam reduzir os riscos (ISTQB, 2011). De acordo com QAI (2006), o risco deve ser usado como base para a alocação de recursos e priorização sobre o que deve ser testado primeiramente.

Contudo, existe outra forma de agrupar os riscos. Os riscos podem ser definidos como riscos associados ao desenvolvimento de software ou riscos associados ao teste de software. Os riscos associados ao desenvolvimento devem ser analisados de acordo com cada sistema e, de acordo com QAI (2006), o Quadro 19 apresenta alguns dos principais riscos associados ao desenvolvimento de software.

Quadro 19 - Riscos associados ao desenvolvimento de software

Risco	**Condições**
Uso impróprio da tecnologia	• Analistas e programadores de sistemas não qualificados para usar uma tecnologia

	• Primeiros usuários de um novo hardware ou software • Planejamento mínimo para a instalação de um novo sistema
Incapacidade de traduzir as necessidades do utilizador em requisitos técnicos	• Usuários sem habilidades técnicas • Pessoas técnicas sem suficiente entendimento de requisitos • Incapacidade do usuário para especificar requisitos
Incapacidade para controlar a tecnologia	• Muitos controles para a eficiência operacional • Inadequados procedimentos de recuperação e reinicialização • Inadequado controle sobre diferentes versões de programas
Entrada incorreta de dados	• Erros humanos de digitação • Falhas mecânicas de dispositivos de hardware • Incompreensão dos procedimentos de entrada de dados
Incapacidade de reagir rapidamente	• A estrutura dos arquivos de computador é inconsistente com as informações solicitadas • Os programas estão disponíveis para satisfazer o pedido desejado • O custo de processamento excede o valor da informação solicitada

Fonte: Elaborado pelo autor

Quanto aos riscos associados ao teste de software, esses devem ser identificados pelo gerente ou líder de teste, sejam eles simples ou não. Alguns tipos de riscos estão sempre presentes em qualquer projeto de teste, mesmo assim é preciso avaliá-los continuamente. De acordo com QAI (2006); Bastos et al. (2007), o Quadro 20 apresenta alguns dos riscos associados ao teste de software.

Quadro 20 - Riscos associados ao teste de software

Risco	**Condições**
Orçamento	• Orçamento insuficiente

Qualificação da equipe de teste	• Imaturidade da equipe • Insuficiente conhecimento sobre técnicas de teste
Suporte à equipe de teste	• Falta de apoio da alta gerência • Desenvolvedores não levam os testadores a sério
Ambiente de teste	• Frequência de atualização de versões dos aplicativos • Diferenças entre o ambiente de produção e o ambiente de teste • Lentidão na execução dos testes
Ferramentas	• Ferramentas específicas para a execução de certos tipos de testes, tais como carga e desempenho • Automatização dos testes de regressão
Metodologias	• Falta de metodologia • Discrepâncias entre o processo de teste e o processo de desenvolvimento • Controles inadequados
Cronogramas	• Prazos curtos • Mudanças no cronograma
Testware[11]	• Ausência de software para controle de versões • Dificuldade de rastrear os artefatos necessários para os testes
Novas tecnologias	• Falta de domínio sobre a tecnologia • Mudanças de tecnologia • Falta de treinamento adequado

Fonte: Elaborado pelo autor

A atividade de testar está diretamente relacionada ao risco e, na probabilidade da ocorrência de algum, as empresas devem investir em testes para que os riscos sejam minimizados. A equipe de teste deve cobrir os testes dos requisitos mais importantes do software, que concentram os maiores riscos para o negócio da empresa, com o intuito de minimizar a possibilidade de defeitos graves. Para isso, devem utilizar uma metodologia e

[11] Artefatos produzidos durante o processo de teste e necessários para planejar, projetar e executar testes. Exemplos: roteiros, procedimentos, arquivos, banco de dados, ambiente e tudo aquilo que possa ser utilizado nos testes (ISTQB, 2010).

ferramentas adequadas para cada projeto (BASTOS et al., 2007). De acordo com ISTQB (2007), testadores podem definir níveis de risco tanto quantitativa quanto qualitativamente, embora seja mais comum apurar os riscos qualitativamente.

Uma estratégia para minimizar os riscos do projeto de teste é, no momento de preparar o plano de teste, realizar a análise de riscos. A análise de risco estuda os riscos, categoriza-os e determina a probabilidade e impacto de cada risco. Percepções pessoais e opiniões sobre esses fatores devem ser considerados no momento da análise (ISTQB, 2007). Em uma visão mais aprofundada, segundo Bastos et al. (2007, p. 91): "A análise de riscos é o processo de avaliar riscos, ameaças, controles e vulnerabilidades". Para analisar os riscos durante o planejamento dos testes, quatro passos devem ser seguidos: (1) formar o time para analisar os riscos, (2) identificar os riscos, (3) estimar a magnitude do risco e (4) selecionar as prioridades de teste (QAI, 2006).

3.6 Técnicas baseadas em experiência

Encontrar erros é um talento que os testadores possuem mesmo não utilizando nenhuma metodologia (MYERS, 2004). Entretanto, existem algumas técnicas de teste que se baseiam em experiência para executar os testes. Competência e intuição são utilizadas, juntamente com experiência em aplicações e tecnologias, para encontrar defeitos no software. Embora sejam efetivos, os testes baseados em experiência não oferecem níveis de cobertura como outras técnicas de teste, uma vez que não têm critério de cobertura formal (ISTQB, 2007). As técnicas baseadas em experiência podem ou não serem empregadas com o uso de metodologia. De acordo com ISTQB (2007), o Quadro 21 apresenta detalhes sobre as principais técnicas de teste baseadas na experiência.

Quadro 21 - Técnicas de teste baseadas na experiência

Técnica	Descrição
Suposição de erro	O testador utiliza a experiência para supor erros que podem ser injetados no sistema e para estabelecer métodos que cubram os defeitos resultantes. É capaz de identificar falhas em potencial durante a análise de risco.
Baseada em lista de checagem	O testador experiente elabora uma lista de checagem com todos os pontos em que o software deve ser verificado. Além da experiência, padrões pré-estabelecidos são utilizados na elaboração da lista.

Exploratória	O testador planeja, executa e reporta os testes simultaneamente. Os objetivos de teste são ajustados dinamicamente durante a execução e não são elaboradas documentações formais. A criatividade também é explorada na preparação dos testes.
Ataque	O testador tenta provocar a ocorrência de falhas no sistema. Interações do software com seu ambiente operacional, principalmente onde há troca de dados, é o principal foco do teste de ataque.

Fonte: Elaborado pelo autor

3.6.1 Teste exploratório

O teste exploratório é um tipo de teste baseado na experiência e intuição do testador. De acordo com Bach (2003), o teste exploratório é qualquer teste em que o testador controla ativamente a modelagem dos testes, bem como sua execução, e usa as informações obtidas durante o teste para projetar novos e melhores testes. A intuição e conhecimento dos testadores são obtidos através da experiência em aplicações e tecnologias e podem ser úteis para identificar testes específicos que não são facilmente identificados pelas técnicas formais. Nesse caso, o teste exploratório pode complementar os testes mais formais. Entretanto, como esta técnica de teste depende da experiência do testador, a eficiência do teste pode variar muito (ISTQB, 2011). Segundo Bastos et al. (2007, p. 257): "O teste exploratório é indicado quando existe pouca documentação para orientar os testes ou quando o prazo é tão curto que não é possível preparar um teste mais formal".

Qualquer pessoa que testa executa teste exploratório. Porém, alguns confiam mais na exploração do que outros (TINKHAM; KANER, 2003). Essa técnica de teste consiste na aprendizagem, modelagem e execução de testes simultaneamente. Ao invés do testador se basear em especificações pré-estabelecidas que orientem os seus testes, ele age de modo mais proativo e busca executar o melhor teste possível no momento. Alguns elementos básicos tais como tempo, testador, produto, missão do teste e reportes compõem o teste exploratório. A missão do teste é elaborada pelo testador com base nas respostas de questionamentos feitos sobre o produto. Deste modo, os testes são modelados com base nas questões e executados para obter as respostas. Algumas vezes, o teste pode não responder completamente a uma questão e por isso deve ser ajustado e novas tentativas devem ser realizadas até que a missão

seja cumprida. Por isso é importante que os reportes sejam feitos dinamicamente para manter os *stakeholders* atualizados (BACH, 2003).

Testadores exploratórios aprendem continuamente sobre o sistema que estão testando e, durante esse aprendizado, eles também testam o software, reportam defeitos e projetam novos testes baseados na informação que obtiveram durante o aprendizado (TINKHAM; KANER, 2003). Deste modo, ao invés de se preocupar apenas em representar cada teste de forma escrita, a habilidade do testador é explorada ao máximo no teste exploratório. É através das habilidades que é possível distinguirem excelentes exploradores dos testadores amadores. De acordo com Bach (2003), o Quadro 22 apresenta as principais características de um excelente testador exploratório.

Quadro 22 - Características de um excelente testador exploratório

Atributo	Características
Modelagem de teste	Um teste pode ser elaborado por qualquer pessoa, mas um excelente testador exploratório é capaz de modelar testes que sistematicamente exploram o produto. Habilidade para analisar, avaliar riscos e utilizar ferramentas são algumas das propriedades desse testador na modelagem de testes eficientes
Observação cuidadosa	Excelentes testadores exploratórios observam mais cuidadosamente do que aqueles que apenas executam scripts de teste. Ao invés de ficarem analisando o comportamento de um script, eles precisam olhar atentamente para cada situação incomum ou misteriosa de um produto
Pensamento crítico	É a habilidade para revisar, analisar e explicar a lógica dos pensamentos. Um excelente testador exploratório é capaz de reportar coerentemente os status dos seus testes e as investigações de defeitos
Ideias diversas	Um excelente testador exploratório faz uso da heurística para formular diversas e boas ideias. A diversidade do time de teste e diferentes *backgrounds* também podem ser aproveitados para a geração de ideias
Riqueza de recursos	Ao contrário de testadores que ficam esperando recursos para teste, um excelente testador exploratório vai à procura de ferramentas, fontes de informação, dados de teste e fóruns para somar ao seu *testware*

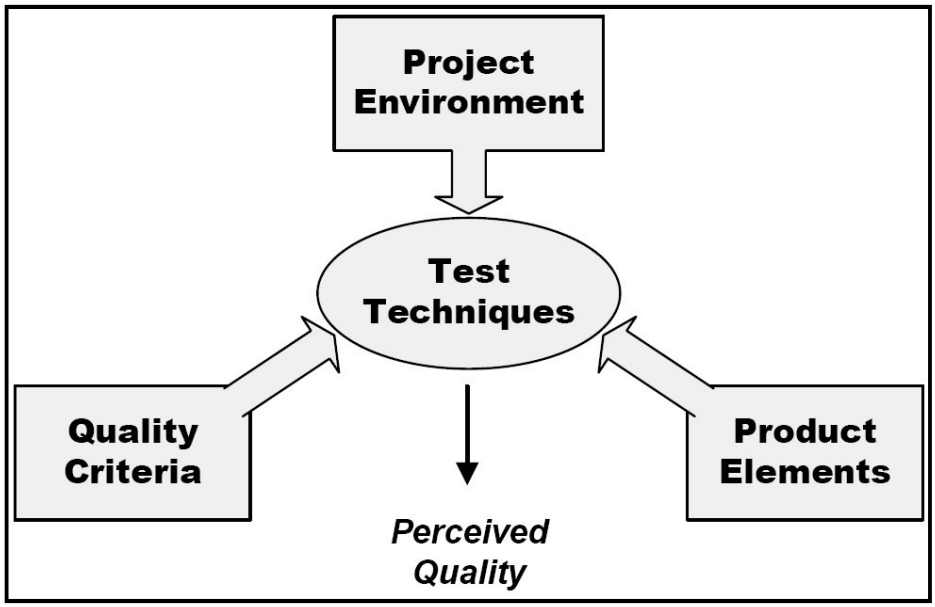
Fonte: Elaborado pelo autor

Tudo aquilo a partir da personalidade do testador até os seus modos de aprendizagem e suas experiências passadas tem um impacto em como ele percebe o risco, pensa que um

aplicativo pode falhar, projeta testes para encontrar essas falhas e em como ele pode cobrir os riscos identificados. A experiência do testador faz com que ele adquira todas essas competências. Entretanto, todas as decisões que um testador exploratório toma é feita sob condições incertas e insuficiente conhecimento sobre a aplicação. Isso faz com que todas as decisões tomadas tenham alguma probabilidade de estarem incorretas e também dificulta a escolha das próximas atividades ou dos próximos testes a serem realizados. Ao considerar que cada decisão tem um risco associado a ela, a heurística auxilia o testador exploratório a tomar as suas decisões de modo mais assertivo (TINKHAM; KANER, 2003).

Heurísticas são diretrizes que auxiliam o testador exploratório a pensar sobre os seus testes em diferentes ângulos, mas elas não são aplicáveis em todas as situações (TINKHAM; KANER, 2003). Um exemplo é o *Heuristic Test Strategy Model*, modelo heurístico criado por Bach (2006). O modelo é um conjunto de heurísticas para a rápida geração de diversas ideias. Em outras palavras, é um conjunto de padrões para a concepção de uma estratégia de teste que lembra o testador sobre o que ele deve pensar no momento de criar seus testes. De acordo com Bach (2006), a Figura 9 demonstra o *Heuristic Test Strategy Model* e o Quadro 23 apresenta descrição dos seus cinco elementos.

Figura 9 - ***Heuristic Test Strategy Model***



Fonte: (BACH, 2006)

Quadro 23 - Elementos do *Heuristic Test Strategy Model*

Elemento	Descrição
Project Environment	Inclui recursos, restrições e tudo aquilo que viabiliza os testes da aplicação. O testador deve certificar-se de que está utilizando todos os recursos disponíveis. Estes são os principais fatores que devem ser considerados no ambiente de projeto: clientes, desenvolvedores, time de testes, informações, equipamento e ferramentas, cronograma, itens de teste e os entregáveis
Product Elements	É tudo aquilo para ser testado. O testador deve garantir que realmente examinou todo o produto que ele deveria examinar. Estes são os principais aspectos de um produto que devem ser observados pelo testador: estrutura, funções, dados, plataforma, operações e tempo
Quality Criteria	São regras, valores e fatores que permitem o testador determinar se o produto tem problemas ou não. Os critérios de qualidade são alguns dos requisitos que definem como deve ser o produto, mas também podem oferecer riscos à ele. São divididos em dois grupos: Critérios de operacionais (capacidade, confiabilidade, usabilidade, segurança, escalabilidade, desempenho, instabilidade e compatibilidade) e critérios de desenvolvimento (suportabilidade, testabilidade, manutenibilidade, portabilidade e localização)
Test Techniques	São estratégias para criar testes. Todas as técnicas envolvem algum tipo de análise do ambiente do projeto, elementos do produto e critérios de qualidade. Existem nove técnicas gerais das quais outras técnicas são derivadas. São elas: teste funcional, teste de domínio, teste de estresse, teste de fluxo, teste de cenário, teste de afirmação, teste de usuário, teste de risco e teste automatizado
Perceived Quality	É o resultado do teste. É a informação que o testador obteve através da execução dos testes

Fonte: (BACH, 2006)

O teste exploratório é uma poderosa técnica porque a informação flui de trás para frente, ou seja, parte de uma execução para projetar ou remodelar os testes. Contudo, ele perde a sua força sempre que o processo de *feedback* for fraco, lento, muito longo ou até mesmo caro (BACH, 2003). Aliado a ele, o modelo heurístico de Bach (2006) auxilia os

testadores exploratórios, mesmo os mais experientes, a não esquecer detalhes essenciais sobre o produto, ambiente e critérios de qualidade.

3.7 Fatores de qualidade do software

O escopo de teste e os fatores de risco são os principais objetivos do teste e esses podem ser associados aos fatores de qualidade do software para melhor cobrir os testes. Segundo QAI (2006, p. 66): "Os fatores de qualidade do software são atributos do software que, se são requisitados e não estão presentes, põem risco para o sucesso do software e, portanto, constituem um risco de negócio". Definir e priorizar esses fatores auxilia na construção dos testes e melhora a qualidade do software quando aplicados desde a fase de desenvolvimento. Devido a isso, os fatores de qualidade devem ser identificados ainda na fase de requisitos (QAI, 2006).

Para identificar melhor os fatores de qualidade do software, de acordo com Bastos et al. (2007), o Quadro 24 apresenta um formulário que auxilia o entendimento desses fatores. Para indicar a importância de cada um, podem-se utilizar as seguintes opções: (MI) Muito Importante, (I) Importante, (AI) Alguma Importância e (NI) Nenhuma Importância.

Quadro 24 - Pesquisa para obtenção da importância dos fatores de qualidade

Respostas	Fatores de qualidade	Definições
	Correção	Extensão em que um programa satisfaz a suas especificações e atende aos objetivos dos usuários
	Confiabilidade	Extensão em que se pode esperar que um programa execute as funções programadas com a precisão requerida
	Eficiência	A quantidade de recursos computacionais e de código requerida por um programa para executar uma função
	Integridade	Extensão em que o acesso ao software ou aos dados por pessoa não autorizada pode ser controlado
	Usabilidade	Esforço requerido para aprender, operar, preparar as entradas e interpretar os resultados de um programa
	Manutenibilidade	Esforço requerido para localizar e corrigir um erro no programa. A facilidade de manutenção envolve também a

		documentação do software
	Testabilidade	Esforço requerido para testar um programa a fim de garantir que ele execute suas funções estabelecidas
	Flexibilidade	Esforço requerido para modificar um programa
	Reusabilidade	Extensão em que um programa pode ser usado em outra aplicação
	Interoperabilidade	Esforço requerido para juntar um sistema com outro
	Portabilidade	Facilidade do software de operar em vários ambientes

Fonte: Bastos et al. (2007) modificado pelo autor

3.8 Fatores que afetam o teste de software

Cada empresa possui liberdade para definir o melhor processo de teste de software que atenderá as suas necessidades. Contudo, vários fatores podem afetar os testes e acabam gerando riscos para o processo de teste. Os principais fatores são: relacionamento das pessoas, escopo do teste, má compreensão do ciclo de vida do teste, baixa qualidade dos planos de testes e restrições de testes (QAI, 2006).

3.8.1 Relacionamento das pessoas

Os testes podem ser afetados de acordo com o estado mental e emocional do testador e algumas atitudes que a empresa possa vir a permitir também contribuem para redução da qualidade dos testes. De acordo com QAI (2006), são elas: (a) testadores devem apoiar no desenvolvimento, (b) disponibilizar menos tempo para testes com o intuito de reduzir as chances de eles encontrarem defeitos, (c) deixar que os testadores encontrarem defeitos como uma forma de *debugging* e (d) testadores não necessitam de treinamento (somente os desenvolvedores).

Além disso, uma boa comunicação entre as pessoas é outro fator fundamental para o bom andamento dos testes. Por exemplo, quando um defeito é comunicado de forma construtiva, evitam-se as chances de desentendimentos entre a equipe de teste e a de desenvolvimento. Problemas de comunicação podem ocorrer, especialmente se os testadores forem vistos somente como mensageiros de más notícias ao informar defeitos. Outro exemplo é quando o testador, o analista ou o líder de teste consegue comunicar de forma clara e

objetiva o progresso dos testes, fazendo com que seja possível avaliar melhor os riscos do projeto (ISTQB, 2011).

Dependendo do ambiente, cada empresa poderá oferecer mais ou menos riscos ao processo de teste de software. Cabem às equipes de testes identificarem suas forças e fraquezas e melhorá-las. O processo de teste deve ser divulgado para todos os envolvidos no projeto de forma que as pessoas entendam como o teste poderá contribuir. Além disso, segundo QAI (2006, p. 91): "Um testador necessita entender o que testar, quem deve executar cada tipo de teste, quando o teste deve ser executado, como executar o teste e quando parar os testes".

3.8.2 Escopo de teste

O escopo de teste define o que será e o que não coberto pelo teste (BASTOS et al., 2007). Dependendo do nível de cobertura de teste desejado, a extensão do escopo de teste pode se tornar um problema devido ao prazo de entrega do projeto. Por outro lado, um escopo muito limitado poderá impedir as devidas verificações e validações do software. Além disso, o cliente pode solicitar alterações e melhorias durante a execução do projeto, fazendo com que o escopo do teste tenha que ser alterado e exigindo possível retestes (QAI, 2006).

3.8.3 Ciclo de vida do teste

Implantar um processo de teste de software como sendo uma fase do processo de desenvolvimento aumentará os custos do projeto quando defeitos forem encontrados. Isso porque sem as revisões iniciais nos documentos do projeto, a possibilidade de encontrar defeitos antes de eles serem implementados no código é muito maior (QAI, 2006). De acordo com a regra 10 de Myers (MYERS 2004), quanto mais tarde o defeito for encontrado, maior será o custo de correção. Mesmo tendo recursos financeiros para cobrir esses custos, muitas vezes não há tempo suficiente para a correção, testes e retestes.

Estudos demonstraram que a maior parte dos defeitos ocorre nas fases iniciais do projeto. Praticamente dois terços de todos os defeitos são encontrados na fase de modelagem. Isso demonstra que, quando iniciado juntamente com o processo de desenvolvimento, o processo de teste pode eliminar praticamente dois terços dos defeitos antes mesmo deles

serem desenvolvidos (QAI, 2006). Por isso, verificações e validações devem ser feitas em cada fase do projeto, do início ao fim.

3.8.4 Qualidade dos planos de testes

O plano de teste define como os testes serão executados e é elaborado a partir dos requisitos do sistema. Porém, quando não houver nenhuma documentação formal sobre o projeto, informações importantes podem não ser consideradas no momento da preparação do plano de teste. Uma vez que requisitos e ricos são suscetíveis à mudanças, deve-se haver controle para que o plano não fique desatualizado (BASTOS et al., 2007). Além disso, um mau entendimento sobre os requisitos permitirá que defeitos não sejam encontrados e requisitos não sejam validados. Problemas também ocorrem quando os objetivos de teste de cada fase do projeto não são definidos (QAI, 2006).

3.8.5 Restrições de testes

Tudo aquilo que restringir a execução dos testadores pode ser considerado como restrições de testes. Alguns exemplos: prazos e orçamentos limitados, baixa qualidade dos requisitos de negócio, mudança de tecnologia e limitações das habilidades dos testadores (QAI, 2006).

4 CROWD TESTING

O mundo tem estado em constantes transformações. Fronteiras estão desaparecendo, cada vez mais organizações tem uma economia 24/7 e a internet está em todos os lugares. Essas mudanças podem influenciar nossas vidas e também o nosso jeito de testar software. Atualmente, os testes baseiam-se em riscos, na cobertura de requisitos e/ou orçamento, entre outras tantas técnicas e métodos. Hoje em dia, até atividades de teste podem ser transferidas para a multidão (ROODENRIJS; PRINS, 2009). ”Usar *crowdsourcing* para testar é o novo fenômeno que vai ser um substituto para uma parte de nossas atividades de teste” (ROODENRIJS; PRINS, 2009, p. 8). Isso é chamado de *crowd testing* e a seguir é apresentado o seu conceito, formas de utilização e tendências sobre esta recente atividade de teste de software.

4.1 Conceito

O *crowd testing* é derivado do *crowdsourcing* e, segundo Roodenrijs e Prins (2009, p. 9) é definido da seguinte forma: “É o uso de pessoas (des)conhecidas, a multidão, para testar software”. Com uma definição semelhante, Narayanan (2011 b) define o *crowd testing* como sendo um meio de assegurar a qualidade do software através de testes realizados por uma multidão. Já em uma visão menos simplista, segundo Forte Consulting Group (2012, p. 2): “*Crowd testing* é uma comunidade de testadores que usam a sua própria infraestrutura e recursos, proporcionando flexibilidade, economia de custos e tempo de resposta rápido”. A ideia geral é focar na experiência do usuário para a execução de testes de software em situações do mundo real.

Utilizar a experiência dos usuários tem como objetivo fazer uso dos seus *feedbacks* no ciclo de desenvolvimento de software para reduzir os custos referentes aos problemas funcionais e de usabilidade. A premissa básica do *crowd testing* é ter habilidade de usar um grande número de pessoas para que seja possível atingir os objetivos da estratégia de negócio em curto prazo. Isso é possível porque o *crowd testing* dá valor a opinião da multidão, produz resultados mais universais e de acordo com a vida real (FORTE CONSULTING GROUP, 2012). Essa multidão consiste de pessoas de todas as partes do mundo com diferentes *backgrounds*, culturas, conhecimentos e criatividade. Todos podem participar, desde os iniciantes até os mais experientes testadores, usuários ou pessoas simplesmente interessadas

em testar. Mesmo assim, é necessário saber o que a multidão pode ou não pode fazer, o porquê eles querem testar, quais são as suas habilidades, conhecimentos, etc. (ROODENRIJS; PRINS, 2009).

O *crowd testing* oferece acesso a ambientes tecnicamente diversos e arquiteturalmente exclusivos, garantindo a validação de uma aplicação particular independente da sua infraestrutura. Para que a estratégia do *crowd testing* seja eficaz, é necessária uma análise detalhada dos cenários de teste para garantir que todos os cenários possíveis sejam cobertos (FORTE CONSULTING GROUP, 2012). Isso o torna apto a ser utilizado em diversas áreas. De acordo com Narayanan (2011 a), o Quadro 25 apresenta as três principais áreas onde a multidão pode atuar nos testes.

Quadro 25 - Principais áreas de utilização do *crowd testing*

Área	Utilização
Aplicações web	• Testar a interface do usuário • Testar cenários específicos
Dispositivos móveis	• Espalhar geograficamente os testes dos usuários finais • Testar diferentes combinações de dispositivos disponíveis • Organizações que adotam as melhores práticas usam uma combinação de *crowd testing*, emuladores e reais dispositivos como parte da estratégia de teste
Jogos	• Garantir que todos os cenários possíveis sejam cobertos • Gerar interesse por um produto (jogo) através do marketing social

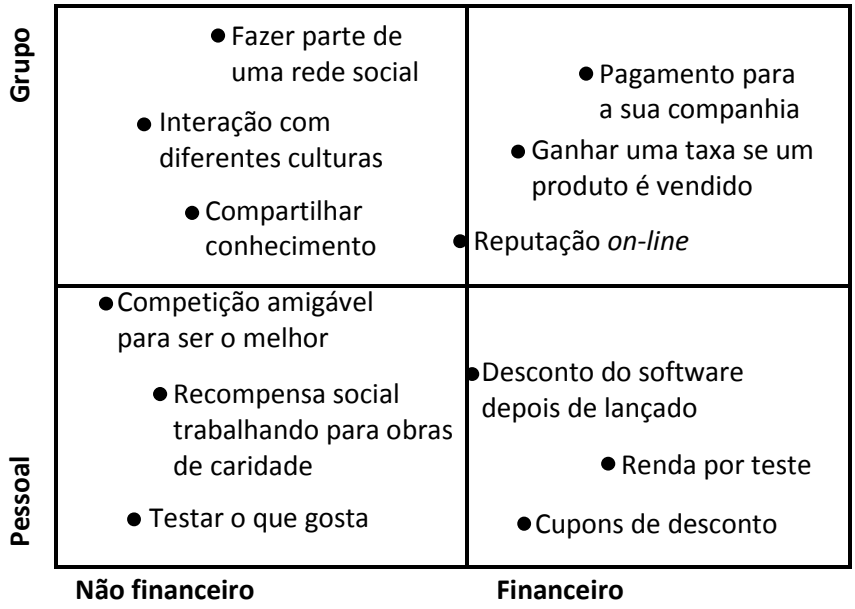
Fonte: Elaborado pelo autor

No tempo em que o *crowd testing* não era conhecido, o custo dos testes impedia o acesso ao controle da qualidade de software para muitas empresas (BUNERJEE, 2012). Hoje isso já é possível porque o modelo de pagamento no *crowd testing* é diferente do atual modelo praticado no teste tradicional. O dinheiro, no *crowd testing*, não é a única forma de recompensa, pois a reputação também é um fator motivacional. Recompensa social, contatos internacionais ou adquirir conhecimento são importantes incentivos para participar do *crowd testing*. Quanto à recompensa financeira, os modelos tradicionais para teste de software nem sempre são aplicáveis. Uma vez que o pagamento por defeito encontrado tem funcionado bem nos projetos de *crowd testing* e cupons de desconto também tem sido outra opção viável para

pagamento (ROODENRIJS; PRIS, 2009). A compensação do trabalho também pode ser uma combinação de um prêmio e um preço variável por defeito encontrado ou até mesmo um preço fixo para um número de horas ou projeto acordado (NARAYANAN, 2011 b).

Contudo, pode ser mais eficaz para os clientes do *crowd testing* se os testes forem oferecidos como um pacote de serviços. Deste modo, o preço e as formas de compensação devem ser praticados de acordo com a estratégia de teste de cada empresa (NARAYANAN, 2011 a, 2011 b). De acordo com Roodenrijs e Prins (2009), a Figura 10 apresenta um resumo das formas de pagamento no *crowd testing*, tanto financeiras ou não, quanto pessoais ou para o grupo.

Figura 10 - Formas de recompensa no *crowd testing*



Fonte: Roodenrijs e Prins (2009) modificado pelo autor

Em março de 2012 o Forte Consulting Group divulgou os resultados de uma pesquisa realizada com profissionais de TI na área de testes, garantia da qualidade, gerentes e analistas. Essa pesquisa revelou que 47% dos participantes tinham conhecimento íntimo do que é e como o *crowd testing* funciona. O estudo contou com a participação de novatos na área de testes e garantia da qualidade até os mais experientes: 30% possuíam de um até cinco anos de experiência, 45% de seis até dez anos e 25% possuíam mais de dez anos de experiência (FORTE CONSULTING GROUP, 2012). Além desse fato, o estudo revelou outras importantes informações a respeito do *crowd testing*. De acordo com o Forte Consulting Group (2012), o Quadro 26 apresenta algumas das conclusões obtidas através da pesquisa.

Quadro 26 - Pesquisa do Forte Consulting Group sobre *crowd testing*

Conclusão	Observações
O *crowd testing* está rapidamente se tornando uma opção viável de teste em ambientes corporativos	Mesmo com um ceticismo e carência de conhecimento em torno do *crowd testing*, a sua adoção tem indicado um grande interesse sobre o assunto. O seu uso pode resultar em web sites ou aplicações que são rapidamente implantados em produção, com custos reduzidos e capacitados para funcionar em diversos ambientes
O trabalho para mover as atividades *off-shore* de QA e testes continua sendo de alta prioridade	A maioria dos participantes concordam que o modelo *off-shore* de QA e testes (onde transfere-se a execução dessas atividades em outro país) é sólido e rentável, independente da inclusão de uma estratégia de *crowd testing*
Segurança, privacidade e vantagem competitiva da empresa vão proibir a adoção do *crowd testing* nas indústrias	Um grande percentual dos participantes identificou uma barreira para a adoção generalizada do *crowd testing* por ele representar uma mudança significativa no modo de comunicar-se e relacionar-se. Diversos fatores suportam essa relutância tais como potenciais riscos de segurança, o comprometimento da imagem corporativa e controles de qualidade
Os principais benefícios do *crowd testing* criam razões imperiosas para explorar essa estratégia	Os principais benefícios do *crowd testing* representam razões viáveis para investir nele. Baixo custo, rápido retorno, teste sob demanda e testes em condições do mundo real são alguns dos contribuidores para explorar a sua estratégia
O *crowd testing* ainda está a ser adotado e acreditado por tomadores de decisão	Nesse estudo, somente 16% dos participantes com cargo gerencial concordam que o *crowd testing* veio para ficar. Essa informação é crucial ao determinar os fatores da tomada de decisão para adotar ou não o *crowd testing* na empresa

Fonte: Elaborado pelo autor

A pesquisa também constatou que, dentre os principais benefícios do *crowd testing*, 61% dos participantes acreditam que a habilidade para testar em condições do mundo real é a característica mais importante. Rápido retorno (44%), testes sob demanda (39%) e baixo custo (39%) completam a lista dos principais benefícios. Enquanto 56% dos entrevistados concordam que segurança, privacidade e vantagem competitiva da empresa vão proibir a

adoção do *crowd testing* nas indústrias, somente 8% discordam e 36% não concordam nem discordam dessa declaração. Outra grande diferença identificada foi que 39% concordam que o *crowd testing* pode ser muito eficaz no ambiente corporativo, 14% discordam e a maioria, 47%, não possuem opinião definida. Por fim, 38% acreditam que o *crowd testing* é principalmente um movimento para o B2C (*Business to Consumer*), software de código aberto e aplicações comerciais, mas não para aplicações de negócio e corporativa (FORTE CONSULTING GROUP, 2012).

Além dos benefícios do *crowd testing* apontados pela pesquisa do Forte Consulting Group (2012), Roodenrijs e Prins (2009) destacam outras características que também beneficiam as empresas que adotam o *crowd testing*. São elas: (a) não há necessidade de contratos de longo prazo com testadores, (b) testes em diversas plataformas (incluindo dispositivos móveis), (c) possibilidade de adicionar ou remover o número de testadores de acordo com as necessidades e (d) aproximar-se dos usuários ou potenciais usuários do software.

4.2 Utilização

O uso do *crowd testing* é determinado por diversos fatores, tais como: o tipo da multidão, as condições de teste e as atividades que a multidão irá executar. Deste modo, quem utilizá-lo é responsável por decidir o uso da multidão de acordo com as suas necessidades. O tipo de multidão e o controle dela devem ser determinados ao preparar um projeto de *crowd testing*, pois não se podem utilizar qualquer pessoa da multidão e alguns sistemas exigem maior controle do que outros (ROODENRIJS; PRINS, 2009).

Mesmo considerando esses diversos fatores, os testes tradicionais não podem ser substituídos pelo *crowd testing* (FORTE CONSULTING GROUP 2012; NARAYANAN 2011 b). De acordo com Banerjee (2012), o Quadro 27 apresenta as razões para não substituir totalmente os testes tradicionais. Acrescentando mais um motivo, Roodenrijs e Prins (2009) estacam que a diferença entre os dois está no foco do teste. Enquanto o *crowd testing* foca na aceitação do software, o teste tradicional está focado em confirmar se o software está pronto para ser usado.

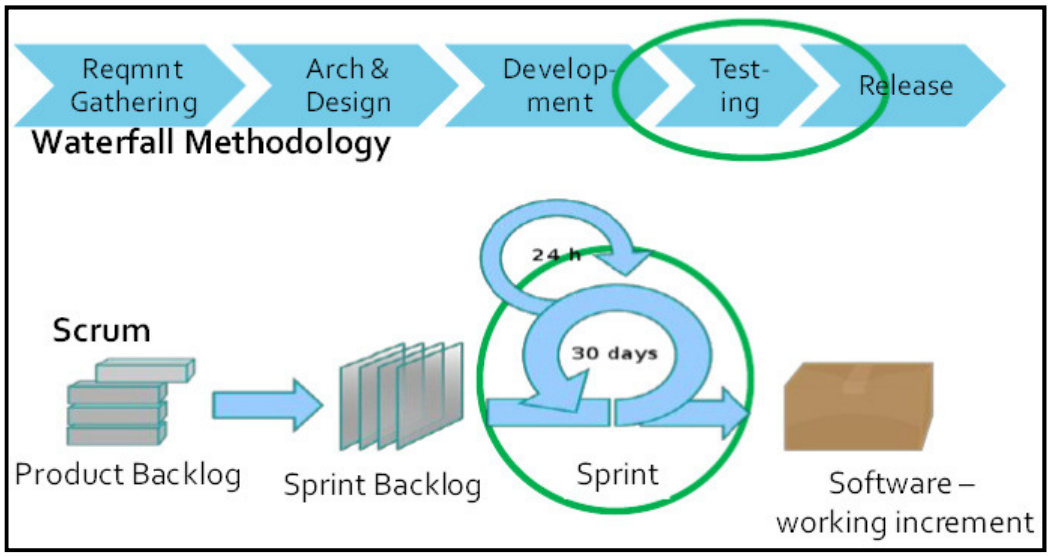
Quadro 27 - Razões para não substituir totalmente os testes tradicionais

Razão	Motivo
Cenários críticos talvez não sejam testados	O *crowd testing* foca nos casos de uso que são fáceis de identificar, baseado na disponibilidade de documentação limitada. Cenários críticos ou funcionalmente relevantes talvez não sejam devidamente cobertos
Cenários específicos não podem ser testados	Muitas vezes, durante os testes, cenários específicos são replicados para criar um ambiente adverso no servidor web ou aplicação. Isso não é possível por meio da multidão
Tornar os aplicativos disponíveis para a multidão pode ser um grande desafio	Muitas vezes os próprios aplicativos não são compatíveis com o acesso compartilhado
Uma aplicação requer certo nível básico de instabilidade antes de ser disponibilizada para a multidão	Caso a aplicação não seja estável, constantes paradas e ociosidade vão fazer com que a multidão perca o interesse de testar
Porque a maioria dos defeitos vai ser comum e reportado por diversas pessoas	Conduzir a triagem dos defeitos reportados vai ser a maior atividade, consumindo significante tempo e esforço
A maturidade da prática de teste nunca vai evoluir com o tempo	Ninguém está realmente selecionando os membros da multidão. Cada iteração pode incluir um conjunto totalmente desarticulado de pessoas, e as variações vão existir nos resultados e na qualidade dos testes.

Fonte:Elaborado pelo autor

Diferenças a parte, o *crowd testing* pode ser aproveitado, independentemente da metodologia de desenvolvimento, seja ela ágil ou cascata. De acordo com Narayanan (2011 a), a Figura 11 apresenta a fase ou estágio em que o *crowd testing* é usado na metodologia cascata (*Waterfall Methodology*) e ágil (*Scrum*, neste caso).

Figura 11 - Utilização do *crowd testing* em metodologias de desenvolvimento



Fonte: Narayanan (2011 a)

A força do *crowd testing* concentra-se na capacidade de aumentar as metodologias tradicionais de teste nas principais situações estratégicas e não deve ser considerada uma estratégia isolada de teste (FORTE CONSULTING GROUP, 2012). Embora seja importante, utilizar apenas o *crowd testing* como única atividade de encontrar defeitos torna o processo de garantia da qualidade imprevisível. Além disso, Narayanan (2011 a) destaca os seguintes riscos ao adotar somente o *crowd testing* como única atividade do processo de teste: (a) testes ineficazes em aplicações instáveis, (b) exposição de funcionalidades estratégicas ou competitivas para o público, (c) falta de domínio e melhor conhecimento da aplicação por parte dos testadores, (d) não cobre todos os tipos de testes e (e) dificuldade de detectar defeitos não funcionais. Por isso, o ciclo de vida de testes deve existir a fim de encontrar defeitos nas fases iniciais do projeto, caso contrário pode se criar um ciclo vicioso de encontrar cada vez mais defeitos nas fases finais do processo de desenvolvimento de software (NARAYANAN, 2011 b).

Para evitar encontrar defeitos somente no final do projeto, as organizações precisam adotar uma estratégia que combine a abordagem do teste tradicional com o *crowd testing* para obter o máximo retorno sobre o investimento possível (NARAYANAN, 2011 b). Atualmente, organizações maduras têm encontrado a correta combinação do teste tradicional com o *crowd testing*. Elas usam o teste tradicional para garantir que a aplicação está funcionalmente estável e de acordo com os requisitos e, em seguida, utilizam a multidão para reduzir o tempo de colocação do produto no mercado, custos e dar flexibilidade aos testes. Além disso, algumas

empresas também disponibilizam os seus scripts de automação e casos de teste para garantir que o *crowd testing* tenha certo nível de maturidade de teste (BANERJEE, 2012).

Enquanto grandes companhias, tais como Microsoft e Google, possuem um grande número de seguidores para usar os seus softwares e também executar testes beta[12], o *crowd testing* pode auxiliar as empresas que não possuem esse tipo de seguidores e oferecer uma multidão para testar os seus softwares (ROODENRIJS; PRINS, 2009). Contudo, de acordo com Narayanan (2011 b), o *crowd testing* pode ser mais poderoso e atuar como um elemento chave no ciclo de vida de teste. De acordo com o autor, o Quadro 28 apresenta as quatro abordagens que combina o *crowd testing* com o teste tradicional.

Quadro 28 - Combinação do *crowd testing* com o teste tradicional

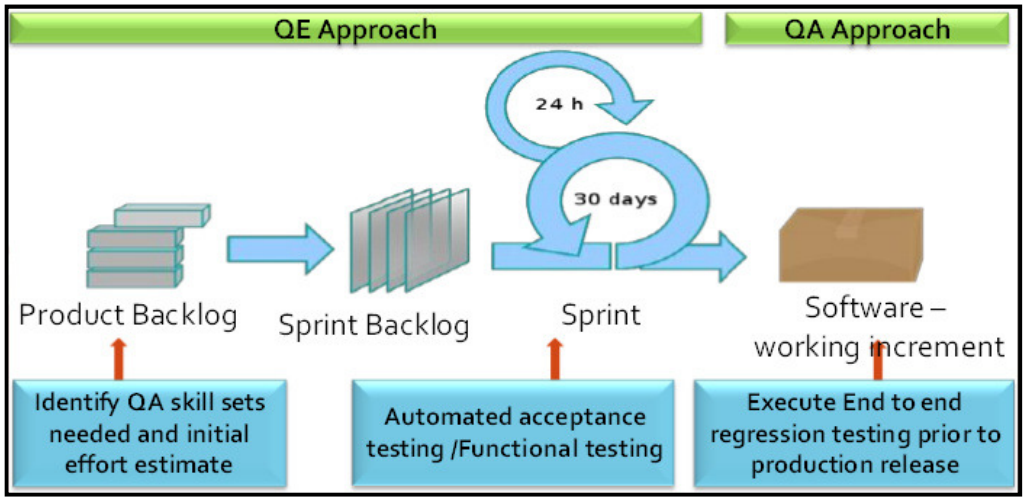
Abordagem	Definição
Extensão dos testes	O *crowd testing* pode ser usado como uma extensão complementar aos testes antes de liberar a versão de produção. É eficaz na captura de qualquer defeito de configuração ou na interface do usuário
Cenários de testes especializados	Usuários finais podem ser utilizados para múltiplos e específicos testes, onde um grupo efetua um tipo de teste ou testa um grupo de componentes específicos e o outro grupo executa outros tipos de testes ou testa outros componentes. Testes de desempenho também podem ser executados (manual ou automatizados)
Teste baseado em risco e prioridade	Usar a multidão para testar aplicações de criticidade alta ou com alta taxa de defeitos. Isso resulta em priorizar a utilização do orçamento para testes em áreas onde ele irá ter maior impacto no negócio da empresa
Integrado a estratégia de teste	Considerando o *crowd testing* como serviço, os testes podem ser combinados com outras ferramentas com o objetivo de atender a estratégia global de teste, focando em testes o mais cedo possível no ciclo de vida de teste

Fonte: Elaborado pelo autor

[12] É um teste operacional realizado por usuários/consumidores fora das dependências da empresa, sem envolvimento dos desenvolvedores, a fim de determinar se o sistema satisfaz, ou não, as necessidades de usuário (ISTQB, 2010).

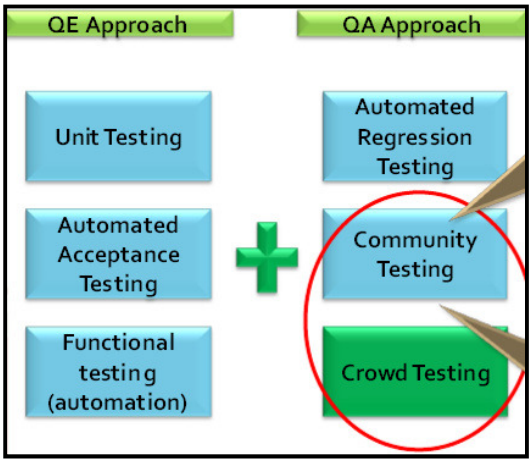
Para demonstrar como o *crowd testing* pode ser integrado no processo de teste, de acordo com Narayanan (2011 a), a Figura 12 apresenta algumas das atividades de teste utilizadas na metodologia ágil *Scrum* e a Figura 13 apresenta o *crowd testing* como sendo uma atividade de teste. Para Narayanan (2011 a), além do *crowd testing* os testes podem ser executados sob a forma do *community testing* que, segundo o autor, representa o estado mais elevado de maturidade do *crowd testing*. Isso significa uma combinação de testes conduzidos através de domínio especializado e ferramentas desenvolvidas pelas próprias comunidades. Organizações que adotam as melhores práticas esforçam-se por uma combinação ideal de *community testing* e *crowd testing*. Entretanto, o grau de teste necessário pode variar de acordo com a estratégia e maturidade dos testes da organização (NARAYANAN, 2011a).

Figura 12 - Atividades de teste na metodologia *Scrum*



Fonte: Narayanan (2011 a)

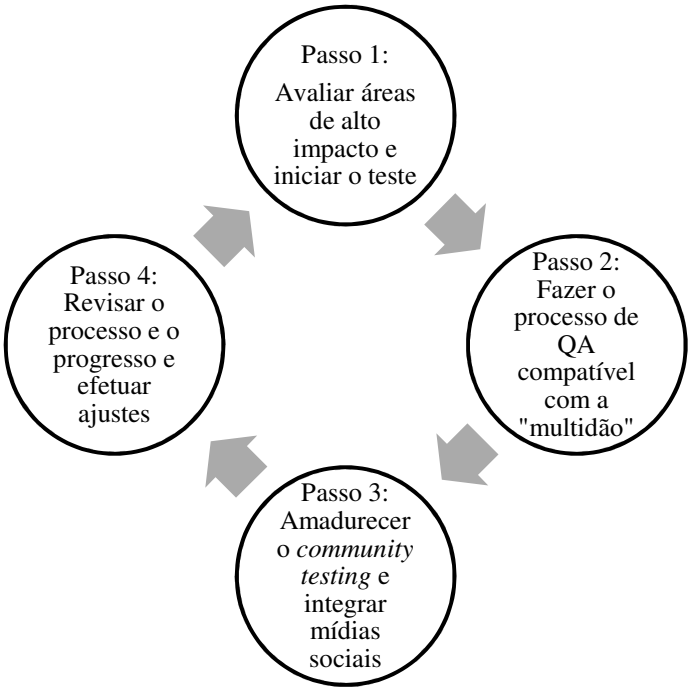
Figura 13 - *Crowd testing* como uma atividade de teste



Fonte: Narayanan (2011 a) modificado pelo autor

Enxergar o *crowd testing* como uma atividade de teste é fundamental para superar o desafio de integrá-lo no processo de teste da empresa. De acordo com Narayanan (2011 a), a Figura 14 apresenta orientações iniciais de como introduzir o *crowd testing* na empresa e o Quadro 29 detalha cada um dos quatro passos básicos. Além de entender essa integração, a aplicação também deve ser claramente definida e entendida por todos antes de implantar o *crowd testing* para garantir um produto eficaz (FORTE CONSULTING GROUP, 2012).

Figura 14 - Como introduzir o *crowd testing* na organização



Fonte: Narayanan (2011 a) modificado pelo autor

Quadro 29 - Quatro passos para implantar o *crowd testing* na organização

Passo	Atividades
1 - Avaliar áreas de alto impacto e iniciar o teste	• Identificar as áreas mais críticas e as prioridades • Iniciar a execução do *crowd testing* para benefícios imediatos. Reduza os esforços de outras atividades • Certificar-se das questões de segurança
2 - Fazer o processo de QA compatível com a multidão	• Identificar as opções de mudança de QA • Automatizar testes de regressão • Implantar a gestão do conhecimento para facilitar a mudança para o *community testing*

3 - Amadurecer o *community testing* e integrar mídias sociais	• Integrar-se com mídias sociais. Foque em grupos e comunidades *on-line* • Determinar a maturidade e disponibilidade interna e de fornecedores de serviços de teste • Influenciar as mídias sociais e os atuais times promovendo a mudança de "*crowd*" para "*community*"
4 – Revisar continuamente o processo e o progresso; e efetuar os ajustes necessários	• Revisar as áreas que estão sendo testadas, processos e integrações com mídias sociais • Efetuar as devidas alterações no processo • Garantir um processo de melhoria continua

Fonte: Narayanan (2011 a) modificado pelo autor

Além dos passos descritos por Narayanan (2011 a), algumas condições devem ser satisfeitas antes de transferir a aplicação para a multidão. De acordo com Roodenrijs e Prins (2009), antes de utilizar o *crowd testing* se deve (a) pensar sobre os testes do sistema que precisam ser executados em uma fase anterior, (b) definir os objetivos de testes e (c) verificar a disponibilidade do ambiente de teste. Por outro lado, o tipo de aplicação também deve ser analisado. De acordo com Bunerjee (2012), a aplicação ideal para o *crowd testing* é aquela em que (1) a internet é o mecanismo de conexão, (2) o servidor está publicamente hospedado para aceitar requisições globais e tem capacidade suficiente para suportar simultâneas cargas e (3) o cliente tem requisitos mínimos e a aplicação suporta a maioria das configurações.

4.3 Tendências

O *crowd testing* está crescendo rapidamente como uma nova tendência na indústria de desenvolvimento de software e logo será incorporado pela equipe de desenvolvimento das organizações (FORTE CONSULTING GROUP, 2012). Fornecedores de serviços estão lutando cada vez mais para expandir a oferta de serviços especializados, tais como testes de desempenho, e o *crowd testing* logo irá se tornar sinônimo de *community testing*. Com um mercado cada vez mais concorrido, os prestadores de serviços globais estão se diferenciando pela prestação de serviços de testes isolados ou através de parcerias com fornecedores estratégicos. Significativos investimentos, além de aprimorar o domínio do conhecimento e

aumentar os recursos de testes, serão capazes de preencher a lacuna entre o *crowd testing* e o *community testing* (NARAYANAN, 2011 b).

Apesar de estar sendo visto como uma opção viável de teste, o *crowd testing* é um serviço de teste complementar e as organizações devem utilizá-lo como parte da estratégia de teste. Esse tipo de abordagem pode rapidamente e de forma rentável eliminar defeitos, se devidamente integrado no ciclo de vida de desenvolvimento de software. Sólidas diretrizes de teste e muitos e experientes recursos de teste permitem ao *crowd testing* testar projetos com competência nos padrões de qualidade e com conhecimento sobre as necessidades de negócio do cliente (NARAYANAN, 2011 b). Para realmente ser considerado uma estratégia de teste, o *crowd testing* requer mix diversificado de usuários e arquitetura colaborativa. Uma estratégia de teste combinado vai melhor atender a crescente necessidade pela qualidade dos testes, enquanto permanecer dentro dos parâmetros de custo (FORTE CONSULTING GROUP, 2012).

Quanto às mídias sociais, no *crowdsoursing* elas já estão funcionando como propulsoras dessa atividade e Narayanan (2011 b) prevê uma maior integração entre o *crowd testing* e as mídias sociais. Grupos de interesses especiais, incluindo aqueles no *Facebook* e *Twitter* que são focados em negócios, vão ajudar a construir conhecimento e entregar produtos mais eficazes. De acordo com Narayanan (2011 a), a integração com as mídias sociais faz dele um grande aliado do marketing, onde grupos sociais enxergam o teste como uma oportunidade de divulgar seus produtos.

Apesar de todos os seus benefícios, Narayanan (2011 a) afirma que os testes executados sob o conceito puro do *crowd testing* não irão durar muito tempo, pois fornecedores de serviços de testes tendem a incorporar o *crowd testing* nos seus atuais portfólios e o *community testing* tende a ser um substituto. Já para o Forte Consulting Group (2012), apesar de o *crowd testing* ser uma ideia atraente, com a sua promessa de teste *On-demand* (sob demanda), baixo custo, e testes em condições do mundo real; questões como (a) será que esse método vai substituir as atuais estratégias de teste e QA das organizações, (b) como ele vai integrar eficazmente com as metodologias tradicionais e (c) será que o *crowd testing* veio para ficar; são questões que estão em aberto e ainda a serem estudadas. Para Roodenrijs e Prins (2009), apesar do *crowd testing* já ser uma realidade, ele ainda é pequeno e não está disponível para todas as empresas. Ele vai mudar na medida em que as pessoas trabalhem cada vez mais unidas e isso vai adicionar outra visão para os testes de software.

5 CONCLUSÃO

O processo de teste de software tradicional tem evoluído nos últimos anos e apresenta conceitos eficazes e eficientes quando aplicados corretamente. Contudo, ele pode se tornar um processo muito burocrático e oneroso se não for adaptado de acordo com a realidade de cada empresa. Isso não é uma tarefa fácil e barata, pois exige investimento de tempo e dinheiro para montar uma equipe de teste, estabelecer processos e documentações, definir ferramentas, realizar treinamentos e mudar a mentalidade corporativa para aceitar o teste de software como um processo e não apenas como mais uma atividade do processo de desenvolvimento de software.

Sua importância não se concentra no seu principal objetivo, que é encontrar defeitos, mas sim na sua capacidade de mostrar a qualidade do software testado. Nesse sentido, a utilização das técnicas de teste, bem como o entendimento de cada nível de teste, é fundamental para garantir a maior cobertura possível dos testes necessários para a aplicação. Entretanto, clientes cada vez mais exigentes, prazos de entrega curtos e constantes mudanças nos requisitos fazem com que o teste de software tradicional seja repensando dentro da organização e novas alternativas sejam buscadas.

O *crowd testing* é uma nova alternativa para o teste de software, mas deve ser visto como uma atividade de teste e não como um processo a parte, uma vez que ele não possui capacidade suficiente para substituir o processo de teste de software tradicional. Seu grande diferencial é o fato de dispor uma multidão diversificada de pessoas (uma multidão) para testar software. Sua principal aplicação se dá em sistemas web e dispositivos móveis, pois esses permitem ser acessados de qualquer parte do mundo e utilizando diversos tipos de configurações de ambientes (sistema operacional, *browser*, etc.). Contudo, por ser um conceito relativamente novo, seu conhecimento ainda não é compreendido em sua totalidade e a ausência de material científico sobre o *crowd testing* o mantém ainda pouco explorado.

Muitas pessoas sabem que o *crowd testing* surgiu do *crowdsourcing*, mas não conhecem o funcionamento do *crowdsourcing*. Esse é um conceito que obrigatoriamente deveria ser abordado com o *crowd testing*, mas que na prática não vem sendo implementado. Ele mostra como os profissionais amadores podem contribuir na execução de tarefas, o potencial das comunidades *on-line* e como elas podem interagir com as organizações (e vice-

versa) e como a inteligência coletiva deve ser construída para garantir um ambiente diversificado, criativo e altamente produtivo.

Alinhar o *crowdsourcing* com o processo de teste tradicional permite embasar conceitualmente o *crowd testing*, fazendo com que deixe de ser apenas um conceito vago e podendo preencher lacunas de conhecimento existentes. Da mesma forma, é possível diferenciá-lo do processo de teste de software tradicional e, ao mesmo tempo: (a) identificar qual é a categoria da qualidade de software que pertence, (b) quais são os métodos e tipos de testes que utiliza, (c) qual é o nível de teste em que está presente, (d) quais são as técnicas de testes que podem ser utilizadas, (e) quais são os papéis desempenhados e (f) quais são os fatores que afetam o *crowd testing*.

Preencher essas lacunas de conhecimento e responder à essas questões é o objetivo da próxima etapa deste trabalho. Deste modo, será possível identificar e analisar como o *crowd testing* pode ser utilizado no processo de teste de software tradicional como sendo mais uma atividade de teste.

5.1 Dificuldades e limitações

Durante a elaboração desta pesquisa, as seguintes dificuldades e limitações foram encontradas:

- Foram localizadas poucas referências bibliográficas sobre *crowdsourcing* para fundamentá-lo;
- O tema *crowdsourcing* não é um tema científico e exigiu uma leitura analítica mais apurada para descrevê-lo;
- A maior parte do conteúdo encontrado sobre *crowd testing* não estava em formato científico dificultando, assim, conceituá-lo;
- As atuais referências bibliografias referentes ao teste de software tradicional não abordam o *crowd testing* como uma possível atividade de teste, o que contribui ainda mais para a lacuna existente entre eles;
- Tanto *crowdsourcing* quanto *crowd testing* são assuntos muito recentes e é possível que as atuais literaturas não os descrevam detalhadamente.

5.2 Trabalhos futuros

Para dar continuidade a esta pesquisa, sugere-se que os seguintes trabalhos sejam realizados:

- Realizar o estudo de caso em uma empresa de *crowd testing*, visando comparar a teoria com a prática, identificar as lacunas existentes entre elas e propor soluções;
- Realizar o estudo de caso em uma empresa de software que utiliza somente o *crowd testing* como atividade de teste, com o intuito de mensurar e analisar a qualidade de software obtida;
- Escrever artigos e desenvolver treinamentos de curta duração para compartilhar o conhecimento obtido e divulgar esta nova atividade de teste, o *crowd testing*.

6 REFERÊNCIAS

BACH, J. **Exploratory Testing Explained**. 2003. Disponível em <http://www.satisfice.com/articles/et-article.pdf>. Acesso em: 10 jun. 2012.

BACH, J. **Heuristic Test Strategy Model**. 2006. Disponível em <http://www.satisfice.com/tools/satisfice-tsm-4p.pdf>. Acesso em: 10 jun. 2012.

BANERJEE, K. **Crowd Testing – Applicability and Benefits**. 2012. Disponível em <http://ers.hclblogs.com/2012/02/crowd-testing-%E2%80%93-applicability-and-benefits/>. Acesso em: 25 mai. 2012.

BASTOS, A. et al. **Base de Conhecimento em Teste de Software**. Rio de Janeiro: Martins, 2007.

BRABHAM, D. **Crowdsourcing as a Model for Problem Solving: An Introduction and Cases**. 2008. Disponível em <http://www.clickadvisor.com/downloads/Brabham_Crowdsourcing_Problem_Solving.pdf>. Acesso em 15 mar. 2012.

GERBER, E. **Para onde vai o crowdsourcing?** 2011. Disponível em <http://info.abril.com.br/noticias/mercado/para-onde-vai-o-crowdsourcing-31082011-45.shl>. Acesso em: 01 mar. 2012.

FORT CONSULRING GROUP. **Efficacy and Applicability of Crowd Testing**. 2012. Disponível em <http://4tegroup.com/crowd-testing-white-paper>. Acesso em: 14 mai. 2012.

HOWE, J. **Crowdsourcing: A Definition**. 2006. Disponível em <http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html> Acesso em 07 mar. 2012.

HOWE, J. **O poder das multidões: por que a força da coletividade está remodelando o futuro dos negócios**. 2ª Ed. Rio de Janeiro: Elsevier, 2009.

ISTQB. **Certified Tester Advanced Level Syllabus**. 2007. Disponível em <http://www.bstqb.org.br>. Acesso em: 05 jun. 2012

ISTQB. **Certified Tester Foundation Level Syllabus**. 2011. Disponível em <http://www.bstqb.org.br>. Acesso em: 05 jun. 2012

ISTQB. **Glossário Padrão de Termos Utilizados em Teste de Software**. 2010. Disponível em <http://www.bstqb.org.br>. Acesso em: 16 jun. 2012

LEADBEATER, C.; MILLER, P. **The Pro-Am Revolution: How enthusiasts are changing our economy and society**. 2004. Disponível em <http://www.demos.co.uk/publications/proameconomy>. Acesso em: 19 mar. 2012.

MYERS, G. J. **The Art of Software Testing**. 2ª ed. John Wiley, 2004.

NARAYANAN, M. **Crowd sourced testing – An emerging business model**. 2011 a. Disponível em <http://www.slideshare.net/manoj7698/star-west-2011-manoj-narayanan-presentation-10>. Acesso em: 14 mai. 2012.

NARAYANAN, M. **The Power of 'Crowd' Testing**. 2011 b. Disponível em <http://www.cognizant.com/InsightsWhitepapers/The-Power-of-Crowd-Testing.pdf >. Acesso em: 05 mar. 2012.

PRESSMAN, R. S. Engenharia de software. 6ª ed. Rio de Janeiro: McGraw-Hill, 2006.

QAI. **Guide to the CSTE Common Body Of Knowledge**. 2006.

RIOS, E.; MOREIRA, T. R. **Teste de Software**. Rio de Janeiro: Alta Books, 2006.

ROODENRIJS, E.; PRINS, A. **Join the crowd: Worldwide testing**. Revista Test Focus, África do Sul, v. 10, n. 4, p. 8-11, 2009. Disponível em <http://www.tmap.net/en/news/publication-about-crowdtesting-test-focus-magazine>. Acesso em: 19 mai. 2012.

TAPSCOTT, D.; WILLIAMS, A. D. **Wikinomics: como a colaboração em massa pode mudar o seu negócio**. Rio de Janeiro: Nova Fronteira, 2007.

TINKHAM, A.; KANER, C. **Exploring Exploratory Testing**. 2003. Disponível em <http://www.testingeducation.org/a/explore.pdf>. Acesso em: 13 jun. 2012.