Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

Vol. 9, No 2

2010

doi:10.5329/RESI.2010.0902

## Sumário

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente. Ela voltou a ser diagramada em uma única coluna para facilitar a leitura na tela do computador. Mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*, que gasta menos tinta.


(mapa de palavras com os termos mais frequentes nos artigos desta edição)

# REQUISITOS E ASPECTOS TÉCNICOS DESEJADOS EM FERRAMENTAS DE TESTES DE *SOFTWARE*: UM ESTUDO A PARTIR DO USO DO SQFD

## REQUIREMENTS AND WISHED FEATURES FOR SOFTWARE TESTING TOOLS: A STUDY BASED ON THE USE OF SQFD

(artigo submetido em março de 2010)

**Ismayle Sousa Santos**
Bolsista do CENAPAD-UFC
Universidade Federal do Piauí
ismayle07@yahoo.com.br

**Pedro Alcântara Santos Neto**
Núcleo de Processamento de Dados
Universidade Federal do Piauí
pasn@ufpi.edu.br

**Rodolfo S. Ferreira de Resende**
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
rodolfo@dcc.ufmg.br

**Clarindo Isaias P. da Silva e Padua**
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
clarindo@dcc.ufmg.br

***ABSTRACT***

This paper presents a set of requirements related to *software* testing tools. These requirements were elicited during several meetings with *software* development professionals. We adopted the *software* quality function deployment (SQFD) in order to prioritize the identified requirements and technical issues, generating relevant information regarding the testing tools. This information can be used to compare and select testing tools. It can also be used by testing tool developers to direct their development effort towards aspects considered important by the users of this kind of tools, the testers.

*Key-words:* QFD; SQFD; *software* testing tools; testing requirements.

**RESUMO**

Este artigo apresenta um conjunto de requisitos relacionados a ferramentas de apoio ao teste de *software*. Eles foram levantados a partir de várias reuniões realizadas com profissionais ligados à área de desenvolvimento de *software*. Foi utilizado o desdobramento da função de qualidade de *software* (SQFD) para priorizar os requisitos, inferir e correlacionar aos aspectos técnicos associados, gerando informações valiosas com relação às ferramentas de testes. Essas informações podem ser utilizadas para comparar ferramentas, assim como para direcionar esforços de desenvolvedores atuantes nessa área, permitindo focar naquilo que é mais importante para os seus usuários finais, os testadores.

Palavras-chave: QFD; SQFD; ferramentas de testes de *software;* requisitos de teste.

## 1 INTRODUÇÃO

Falhas de *software* geralmente correspondem a manifestações, para o usuário final, de um ou mais defeitos inseridos durante o processo de desenvolvimento. Embora existam várias ferramentas e técnicas para garantia de qualidade, muitos *software*s ainda são entregues aos usuários finais com uma grande quantidade de falhas. Essas falhas não são encontradas, em parte, por causa do uso de ferramentas de teste inadequadas, conforme apresentado em um relatório do NIST (2002). De acordo com esse relatório, os custos de falhas em *software*s desenvolvidos nos EUA, em 2002, são estimados de U$22,2 a U$59,5 bilhões.

Uma das principais técnicas para se desenvolver um *software* de qualidade é a utilização de testes. O Teste de *Software* é um elemento crítico da garantia de qualidade de *software* e representa a revisão final da especificação, projeto e geração de código (PRESSMAN, 2001). A atividade de teste geralmente demanda tempo, recursos e conhecimentos, necessidades essas que muitas vezes limitam sua execução por parte das organizações desenvolvedoras. Um meio de reduzir essas demandas se dá a partir do uso de métodos e ferramentas que auxiliem ou automatizem, de alguma forma, essa tarefa (PERRY, 2006).

Portanto, para redução dos gastos com a atividade de teste e construção de sistemas com mais qualidade, é preciso que os testadores tenham em mãos ferramentas de teste que atendam a algumas necessidades (PERRY, 2006). O primeiro passo para que isso aconteça é a identificação das necessidades dos profissionais que trabalham na área. Se isso fosse feito por meio do simples levantamento de requisitos, a indicação de quais têm maior prioridade e a definição de como eles podem ser atendidos só poderia ser feita por estimativas pessoais, que podem ser bastante imprecisas. Por conta disso, utiliza-se neste trabalho o método de desdobramento da função de qualidade de *software* (*Software Quality Function Deployment* - SQFD) (HAAG *et al.*, 1996). Com a aplicação desse método, é possível identificar e priorizar os requisitos relacionados às ferramentas de apoio aos testes de *software* de maneira sistematizada, ao mesmo tempo em que se pode realizar o desdobramento desses requisitos em características ou funções que deveriam existir nesse tipo de ferramenta.

Com a priorização dos requisitos e aspectos técnicos é possível obter a informação necessária para responder perguntas como: “Quais são as necessidades dos testadores?” e “Qual é a necessidade considerada de maior importância?” Além disso, os aspectos técnicos levantados podem ser usados como um meio mensurável para análise e comparação entre ferramentas de apoio ao teste de *software*.

Este trabalho teve como objetivo descobrir os aspectos técnicos mais importantes para ferramentas de apoio ao teste. Não foi encontrado na literatura nenhum trabalho nessa direção. O resultado obtido pode ser utilizado para avaliar e comparar ferramentas, além de indicar “o quê” os usuários de ferramentas de teste consideram mais importante, podendo

inclusive direcionar pesquisas na área, uma vez que os aspectos desejados podem ainda estar ausentes nas ferramentas atuais.

O restante deste artigo descreve a aplicação do SQFD com relação a ferramentas de apoio ao teste de *software*. Ele está organizado da seguinte forma: a Seção 2 discute alguns trabalhos relacionados; a Seção 3 descreve o QFD (*Quality Function Deployment)*, seus componentes chaves e o SQFD; a Seção 4 apresenta a metodologia adotada; a Seção 5 detalha a aplicação do SQFD para ferramentas de apoio ao teste; a Seção 6 conclui o artigo e aponta direções para trabalhos futuros.

## 2 TRABALHOS RELACIONADOS

Existem diversos trabalhos relacionados ao levantamento de requisitos para tecnologias associadas ao desenvolvimento de *software*. Hoffmann *et al.* (2004), por exemplo, identificaram um conjunto de requisitos para ferramentas de gestão de requisitos. De forma similar, Santos-Neto *et al.* (2007) identificaram requisitos para métodos de teste baseados em modelos. Esses trabalhos fizeram uma priorização dos requisitos com base na experiência dos autores, enquanto que no presente trabalho foi utilizada a percepção dos usuários finais, que nesse caso são testadores e profissionais relacionados ao desenvolvimento de *software*, para efetuar a priorização dos requisitos.

O QFD aplicado ao desenvolvimento de um produto utiliza o termo “necessidade” para definir algo que é desejado por um usuário do produto e utiliza o termo “aspecto técnico” para denotar uma característica que o produto deva conter, e que deve estar associada a uma ou mais necessidades identificadas (AKAO, 1990). Neste caso, o termo “necessidade” está diretamente associado ao conceito de “requisito”. Diversos trabalhos tratam do uso do QFD para o levantamento de necessidades e aspectos técnicos durante o desenvolvimento de *software*. Karlsson (1997), por exemplo, descreve e discute no seu trabalho o uso do QFD como um *framework* para gerenciamento de requisitos de *software*. Alves e Padua (2001) descrevem a aplicação do QFD na especificação de requisitos de usabilidade. Tan *et al.* (1998) discutem o uso de questionários para obter a voz do cliente, afirmando que eles são fáceis de aplicar e para entender as informações levantadas. Outro exemplo interessante da utilização do QFD para levantamento de requisitos foi apresentado por Ioannou *et al.* (2004), que exibem o uso do QFD para integrar a “voz do cliente” no processo de desenvolvimento de um sítio web, maximizando a satisfação dos usuários.

## 3 O MÉTODO QFD

O QFD foi criado no Japão por Yoji Akao e Shigeru Mizuno para o desenvolvimento de produtos manufaturados de alta qualidade (AKAO, 1990; MIZUNO e AKAO, 1994). Ele é um método para converter as necessidades dos clientes em características do produto (AKAO, 1996). Os objetivos principais do QFD são: capturar, de forma efetiva, as necessidades dos

usuários de um produto ou, simplesmente, capturar a voz do cliente, reduzir a perda de informações e fornecer mecanismos que permitam à equipe de desenvolvimento trabalhar eficientemente, com um foco maior no que é mais importante na visão dos clientes, para atender aos requisitos levantados.

> Capturar a "voz do cliente" significa identificar suas necessidades com relação a algum produto (AKAO, 1996). Essas necessidades, que são os requisitos do produto, podem então ser agrupadas em uma lista de "o quê" o produto precisa para satisfazer seus usuários. A primeira etapa do QFD corresponde à captura da voz do cliente sobre aquilo que ele deseja em um produto. Depois que todos os requisitos são obtidos, a lista pode ser refinada e ordenada. Além disso, o QFD provê um método sistemático para identificar a prioridade dos requisitos, a partir de uma atribuição de graus de importância por parte dos clientes (AKAO, 1996). Outro passo importante é o desdobramento dos requisitos em aspectos técnicos, por meio do qual é construída uma lista de "como" os requisitos dos clientes podem ser desenvolvidos no produto. Nesse momento, a ênfase muda de identificar o problema para resolver o problema (AKAO, 1996).

## 3.1 QFD NO DESENVOLVIMENTO DE SOFWTARE

A adaptação do QFD clássico, que fora criado inicialmente para melhorar a qualidade dos produtos manufaturados, para uso no desenvolvimento de *software*, é intitulada *desdobramento da função qualidade do software* (SQFD) (HAAG *et al.*, 1996). O modelo básico SQFD é uma adaptação da matriz de qualidade. Ele é baseado em um modelo de cinco fases. Os passos do modelo básico do SQFD, juntamente com uma exemplificação de como esse modelo é aplicado, são descritos a seguir. A Figura 1 ilustra a aplicação do SQFD em um exemplo hipotético, criado neste trabalho apenas para esclarecer a aplicação do modelo.

*Requisitos do cliente*: as necessidades do cliente são levantadas e registradas na tabela de requisitos do cliente. São utilizados, nesse passo, os métodos para levantamento de requisitos, tais como *brainstorm* ou reuniões para desenvolvimento em conjunto (*Joint Application Development* - JAD) (DENNIS *et al.*, 1999). Supondo um sistema simplificado de biblioteca *on-line*, por exemplo, que só permite que os usuários cadastrados renovem os livros emprestados, seria possível a identificação dos seguintes requisitos: (i) fácil de usar; (ii) seguro; (iii) rápido. Assim, após agrupar esses requisitos em níveis, para organizar a sua apresentação, um possível resultado dessa etapa é exibido na Figura 1a.

*Especificações técnicas do produto*: em cooperação com os usuários do produto, porém intensamente baseado na percepção da equipe de desenvolvimento, os requisitos são convertidos em especificações técnicas mensuráveis, que devem ser registradas na tabela de especificações técnicas do produto (HAAG *et al.*, 1996). É importante notar que alguns requisitos do cliente podem ser convertidos em múltiplas especificações técnicas do produto. Para o sistema de biblioteca hipotético comentado

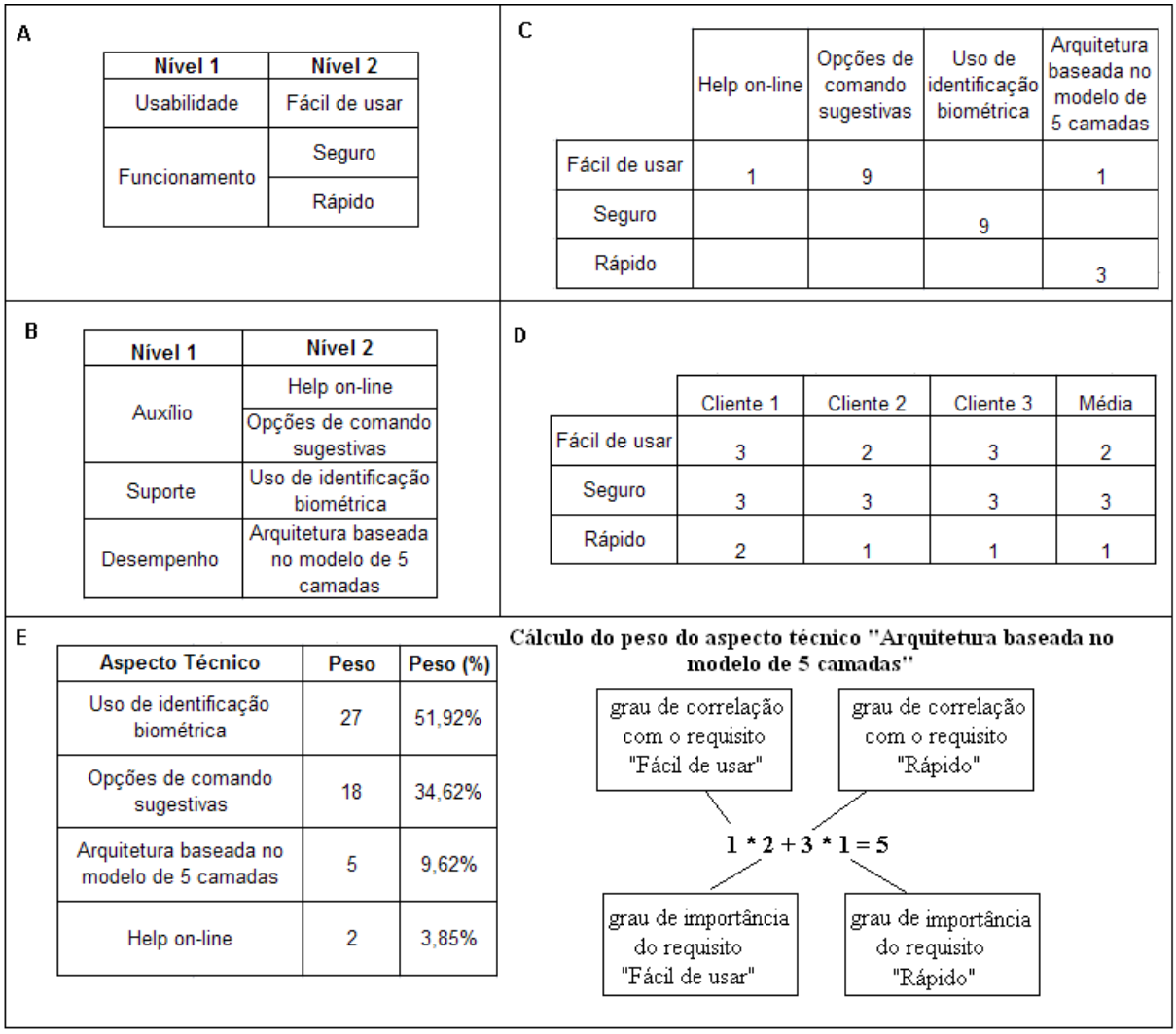
anteriormente, o requisito “fácil de usar”, por exemplo, poderia ser atendido por meio de um “*help on-line*” e de “opções de comando sugestivas”. Assim, um conjunto possível de aspectos técnicos, ilustrado na Figura 1b, relacionado aos requisitos já identificados poderia ser: (i) *help on-line*; (ii) opções de comando sugestivas; (iii) uso de identificação biométrica; (iv) arquitetura baseada no modelo de 5 camadas.

*Correlação*: com perguntas dirigidas ao cliente, deve ser criada a matriz de correlação, identificando os pesos dos relacionamentos entre os vários requisitos do cliente e as especificações técnicas do produto. Quando há muitos clientes envolvidos, é importante estabelecer um consenso quanto à intensidade dos relacionamentos (HAAG *et al.*, 1996). O grau de correlação pode ser “possível”, “fraco” ou “forte”, representado neste trabalho pelos números um (1), três (3) e nove (9), respectivamente (HAAG *et al.*, 1996). A Figura 1c ilustra uma matriz que poderia ser criada a partir da correlação entre os requisitos e aspectos técnicos para o exemplo utilizado nesta seção. Nela pode-se observar que o requisito “fácil de usar” não possui correlação com o aspecto técnico “uso de identificação biométrica”; possui correlação fraca com “*help on-line*” e “arquitetura baseada no modelo de 5 camadas”; e correlação forte com “opções de comando sugestivas”.

*Requisitos prioritários do cliente*: com base em pesquisas realizadas junto aos clientes, com atribuição de peso para cada necessidade identificada, pode-se gerar uma priorização das necessidades, construindo-se assim a tabela de requisitos prioritários do cliente. Uma forma de se fazer isso é solicitar para cada cliente que participou da identificação de necessidades a atribuição de um peso para o requisito. Pode-se utilizar o valor de um (1) a três (3), sendo três o mais importante. Esses valores foram utilizados para facilitar essa etapa, tendo sido definidos pelos autores do trabalho. Não existe consenso sobre o uso de valores para essa etapa, de modo que se adotou o que pareceu ser mais adequado para o contexto. A Figura 1d ilustra uma possível priorização dos requisitos levantados, supondo que apenas três clientes participaram das reuniões. Nela pode-se observar que o requisito “seguro” foi considerado mais importante e “rápido” o menos importante.

*Especificações técnicas prioritárias*: as especificações técnicas são priorizadas com base no somatório das multiplicações dos pesos dos requisitos pelos graus de intensidade da correlação entre os requisitos e as especificações técnicas do produto (HAAG *et al.*, 1996). Para detalhar melhor, o cálculo é feito da seguinte forma: (1) para cada aspecto técnico, verificar quais são os requisitos com os quais ele possui alguma correlação; (2) multiplicar o grau de importância do requisito pelo grau de correlação com o aspecto técnico em questão; (3) somar os valores obtidos, resultando no peso (prioridade) do aspecto técnico. A Figura 1e exibe a tabela de priorização dos aspectos técnicos, assim como o cálculo do peso do aspecto técnico “arquitetura baseada no modelo de 5 camadas”. Por fim, com base na tabela de priorização dos aspectos técnicos pode-se identificar quais os aspectos de maior importância na visão dos

usuários. No exemplo contido na Figura 1e, pode-se verificar que o aspecto técnico “uso de identificação biométrica” é o mais importante, pois possui maior peso.



A

| Nível 1 | Nível 2 |
|---|---|
| Usabilidade | Fácil de usar |
| Funcionamento | Seguro |
| | Rápido |

C

| | Help on-line | Opções de comando sugestivas | Uso de identificação biométrica | Arquitetura baseada no modelo de 5 camadas |
|---|---|---|---|---|
| Fácil de usar | 1 | 9 | | 1 |
| Seguro | | | 9 | |
| Rápido | | | | 3 |

B

| Nível 1 | Nível 2 |
|---|---|
| Auxílio | Help on-line |
| | Opções de comando sugestivas |
| Suporte | Uso de identificação biométrica |
| Desempenho | Arquitetura baseada no modelo de 5 camadas |

D

| | Cliente 1 | Cliente 2 | Cliente 3 | Média |
|---|---|---|---|---|
| Fácil de usar | 3 | 2 | 3 | 2 |
| Seguro | 3 | 3 | 3 | 3 |
| Rápido | 2 | 1 | 1 | 1 |

E

| Aspecto Técnico | Peso | Peso (%) |
|---|---|---|
| Uso de identificação biométrica | 27 | 51,92% |
| Opções de comando sugestivas | 18 | 34,62% |
| Arquitetura baseada no modelo de 5 camadas | 5 | 9,62% |
| Help on-line | 2 | 3,85% |



Figura 1: Exemplo hipotético para explicação dos passos do SQFD

Fonte: os autores

Uma questão interessante relacionada à criação da tabela com os aspectos técnicos priorizados é: como ela pode ser utilizada? A resposta é simples: ela é base para se comparar e determinar qual produto é o melhor dentre vários. Os aspectos técnicos priorizados indicam aquilo que o cliente acha mais importante no produto, uma vez que eles são obtidos a partir do desdobramento dos requisitos em elementos técnicos que devem constar nos produtos. Sua priorização é feita com base na importância dos requisitos e na forma como estes estão contemplados nos aspectos técnicos (HAAG *et al.*, 1996).

Com a construção da *tabela de desdobramento de características da qualidade*, exibida na Figura 1e, pode-se agora comparar dois sistemas

simples de biblioteca *on-line*. Supondo que existam dois sistemas apropriados para o caso, denominados aqui A e B, é possível utilizar a tabela de aspectos técnicos priorizados para definir qual é o melhor. Para fins didáticos, considera-se neste trabalho que a comparação entre esses dois sistemas hipotéticos A e B resulta nas seguintes conclusões, ilustradas na Figura 2: ambos possuem identificação biométrica; somente o sistema A possui opções de comando sugestivas e utiliza o modelo de cinco camadas; somente o sistema B possui um *help on-line.*

Assim, com base nessa análise, pode-se afirmar que o sistema A é melhor que o sistema B, pois a soma dos pesos dos aspectos técnicos a que o sistema A atende é maior que a respectiva soma para o sistema B.

| | Peso (%) | Sistema A | Sistema B |
|---|---|---|---|
| Uso de identificação biométrica | 51,92% | X | X |
| Opções de comando sugestivas | 34,62% | X | |
| Arquitetura baseada no modelo de 5 camadas | 9,62% | X | |
| Help on-line | 3,85% | | X |
| | | 96,15% | 55,77% |

Figura 2: Comparação hipotética entre dois sistemas fictícios de biblioteca *on-line*

Fonte: os autores

## 4 METODOLOGIA

Neste trabalho foi aplicado o SQFD para identificar e priorizar os requisitos e aspectos técnicos relacionados a ferramentas de apoio aos testes de *software*.

Para levantar os requisitos e aspectos técnicos, foram entrevistados profissionais de Belo Horizonte ligados ao desenvolvimento de sistemas de informação. Esses profissionais atuavam como arquitetos de *software*, testadores, gerentes de projeto, engenheiros de requisitos e de usabilidade, totalizando 17 pessoas. O critério utilizado para seleção dos participantes foi a conveniência (WOHLIN *et al.*, 2000). Foram convidados a participar do estudo todos aqueles que estavam acessíveis. Embora a equipe fosse bastante heterogênea, em termos das áreas da *engenharia de software* em que cada um atua, conforme apresentado na Figura 3a, todos, com exceção de um dos participantes, já tinham alguma experiência relacionada à área de testes (Figura 3b).

Conforme pode ser observado nas Figuras 3c e 3d, todos os profissionais que participaram desta pesquisa possuem nível de pós-graduação

e experiência profissional trabalhando com engenharia de *software*. Assim, consideram-se significativas as necessidades e os aspectos técnicos identificados, visto que os profissionais envolvidos possuem tanto conhecimento prático quanto teórico.

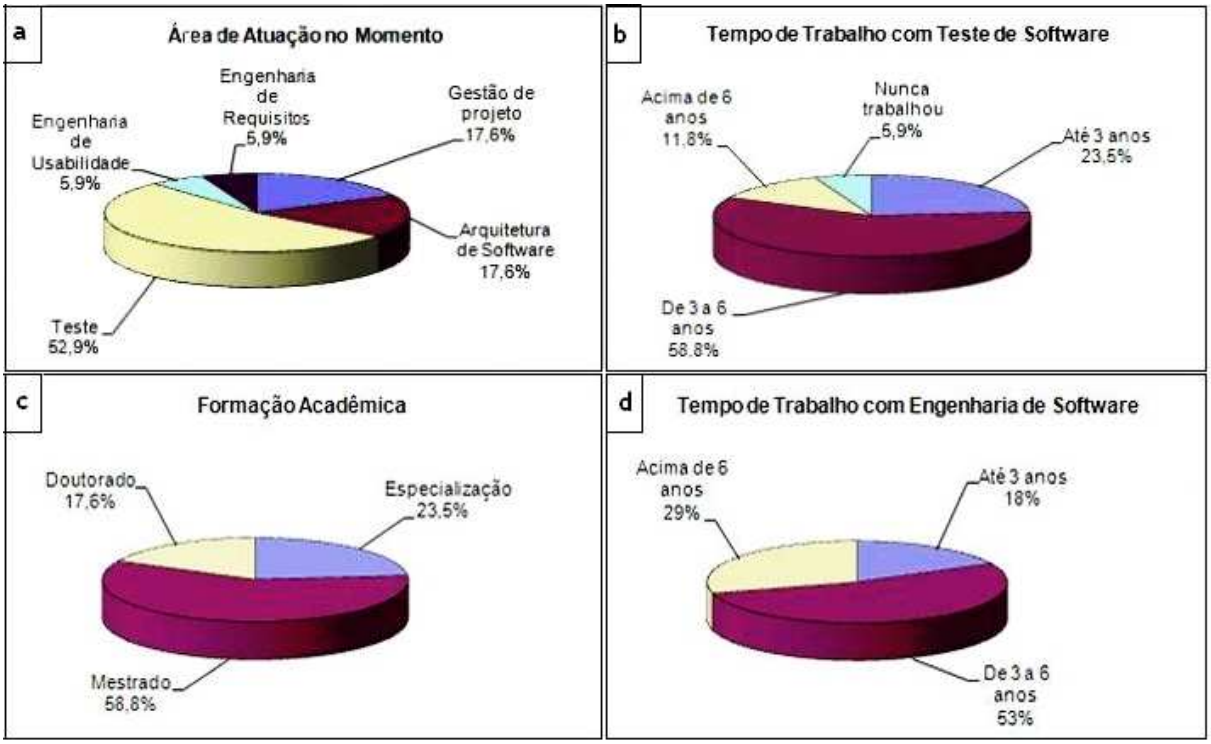



Figura 3: Perfil dos participantes do estudo.

Fonte: os autores

Durante a pesquisa seguiram-se os passos do SQFD apresentados na Seção 3. Para o levantamento dos requisitos, os participantes foram questionados com a seguinte pergunta: “O que você gostaria que uma ferramenta de apoio ao teste tivesse?“. As respostas foram registradas em planilhas eletrônicas e, a partir delas, foram feitas discussões envolvendo os participantes até que houvesse convergência de ideias. Isso gerou um conjunto de 63 requisitos que relatavam as necessidades levantadas pelos 17 participantes da pesquisa.

De forma análoga, foram identificados os aspectos técnicos. Os participantes foram questionados sobre “O que a ferramenta deveria ter para atender à necessidade identificada?”. Nos casos mais complexos, em que parte dos participantes não tinha certeza sobre os aspectos relacionados, eram feitos cenários dos requisitos com base no *who-what-when-where-why-how* (5W1H), que é uma técnica utilizada para identificar e relacionar conceitos, de forma a gerar um melhor entendimento sobre algo que se planeja fazer (SOWA e ZACHMAN, 1992).

A Figura 4 ilustra um trecho da planilha utilizada para aplicar o 5W1H. A primeira coluna da planilha contém os “atributos de qualidade”, que são

os requisitos que foram levantados. No exemplo em questão, tem-se apenas o requisito *documentação com fácil visualização*. Na coluna "necessidades" foram registradas as necessidades que levaram à identificação dos requisitos. Na coluna seguinte, definiu-se o cenário do requisito, respondendo as perguntas: *quem-onde-por quê-o quê-quando-como*. Por fim, a última coluna apresenta os "itens exigidos" para o atendimento do requisito ou que se relacionam a ele. Estes itens foram então sintetizados e deles extraíram-se os aspectos técnicos. Na Figura 4, por exemplo, dentre os aspectos que foram extraídos dos "itens exigidos", tem-se o *gerador de dados do plano de testes,* que surgiu a partir dos itens 2 e 3, e o *oráculo para gerar as saídas esperadas,* do item 4.

| **Atributo de qualidade** | **Necessidades** | **Cenários** | **Itens Exigidos** |
|---|---|---|---|
| | | **Quem, Onde, Porque, O que, Quando, Como (5W1H)** | |
| Documentação com fácil visualização ok | Documentação dos testes, especificando qual o objetivo do teste, quais dados de entrada, resultado esperado. | Onde: execução do fluxo de teste.<br>Porque: para poder visualizar que testes serão executados e com quais valores.<br>O que: documentação dos testes.<br>Quando: após a geração da especificação dos testes.<br>Como: gerar a documentação a partir dos testes já especificados. | 1. Gerador de relatório com formato de documentação de testes conforme especificado pelo IEEE.<br>2. Gerador de dados iniciais para o teste.<br>3. Gerador de entradas para os casos de teste.<br>4. Oráculo para gerar as saídas esperadas e determinar os dados iniciais requeridos para execução do teste. |

Figura 4: Trecho da tabela utilizada durante a pesquisa

Fonte: os autores

Para a identificação dos graus de correlação entre os requisitos e aspectos técnicos, foram utilizadas perguntas como: "Quais características uma ferramenta de teste precisa ter para atender o requisito X?" e "A característica A é mais ou menos importante que característica B para atender ao requisito X?", onde A e B são duas das características identificadas e X um dos requisitos levantados.

A priorização dos requisitos foi feita pela média dos valores atribuídos pelos diversos participantes a cada requisito. A priorização dos aspectos técnicos ocorreu sem a participação dos 17 profissionais envolvidos, uma vez que foi feita de forma automática, a partir dos cálculos descritos na Seção 3.

# 5 O SQFD APLICADO A FERRAMENTAS DE APOIO AO TESTE DE SOFTWARE

## 5.1 REQUISITOS DO CLIENTE

Como mencionado anteriormente, com base em reuniões feitas com os participantes do estudo, foi levantado um conjunto de requisitos relacionados às ferramentas de apoio ao teste de *software*. Esses requisitos refletem as necessidades dos usuários desse tipo. É importante ressaltar que as necessidades foram identificadas independentemente de serem ou não atendidas pelas ferramentas de teste de *software* disponíveis. O levantamento desses requisitos corresponde à primeira etapa de aplicação

do modelo básico do método SQFD, no qual os requisitos devem ser registrados em uma tabela de requisitos do cliente. Nas subseções a seguir apresentam-se os requisitos que foram identificados, já agrupados utilizando-se diagramas de afinidade (HAAG *et al.*, 1996). Nesse caso, o título da subseção corresponde ao nome do agrupamento utilizado e dentro da subseção descrevem-se as principais necessidades relacionadas.

### 5.1.1 Gestão e planejamento da atividade de teste

A atividade de teste necessita de planejamento visto que ela consome recursos e possui um prazo estabelecido para ser executada. Com base nisso, algumas necessidades relacionadas a esse item foram identificadas: (i) criação de *plano de teste*, com auxílio para a definição de tarefas para a equipe e extração de dados gerenciais de teste de projetos similares; (ii) auxílio na estimativa de prazos para as atividades de teste, baseada nas bases de dados históricos dos projetos já executados; (iii) estimativa da complexidade dos casos de uso, utilizando uma classificação que auxilie as atividades de planejamento; (iv) estimativa da estabilidade dos casos de uso, indicando quais casos de usos não deveriam ser testados, em virtude da sua instabilidade, minimizando a questão do retrabalho em partes do sistema ainda não estáveis; (v) integração com ferramentas de gerenciamento de projeto, permitindo a importação e exportação de dados para uso em tais ferramentas.

### 5.1.2 Geração automática de testes

A necessidade (i) geração automática de testes funcionais é, provavelmente, a maior necessidade de uma equipe de teste, pois essa geração automática conduz a uma redução do tempo gasto na atividade de testes. Contudo, com a crescente utilização de sistemas *Web*, os quais são acessados por uma grande quantidade de usuários com diferentes níveis de instrução e sendo suscetíveis à tentativa de invasão, outras necessidades também ganharam destaque: (ii) geração de testes de desempenho e estresse, visando à avaliação do funcionamento do sistema em condições anormais de funcionamento; (iii) geração de testes de segurança, avaliando a robustez do sistema com relação a certos tipos de vulnerabilidades; e (iv) apoio à execução de testes de usabilidade, facilitando a realização do teste de usabilidade, que visa à avaliação da adequação ao uso do sistema sob teste.

Outra necessidade associada à geração automática de testes é a de (v) geração de testes para falhas registradas em uma ferramenta de acompanhamento de falhas. Essa necessidade foi identificada porque mesmo havendo a geração automática de testes, uma falha pode ser registrada em uma ferramenta de acompanhamento de falhas, por exemplo, pelo próprio desenvolvedor, e não por uma ferramenta de automação. Nesse caso, é importante que o mecanismo de geração consiga gerar testes para essa falha, para garantir que ela não se manifeste na versão final do produto. Além disso, existe a necessidade de que os (vi) testes gerados sejam independentes e auto-contidos, evitando

assim que eles sejam dependentes de outros testes e permitindo que possam ser executados de forma paralela.

### 5.1.3 Apoio aos testes

A atividade de teste exige alguns mecanismos de apoio para, por exemplo, acompanhar os defeitos (*bugs*) encontrados e gerar os dados do teste. Foi utilizado o termo “povoador” para representar um elemento que gera um conjunto de dados no mecanismo de persistência de um sistema. Geralmente, antes da execução dos testes, é necessária a execução de um povoador no intuito de gerar os dados exigidos pelo teste a ser executado. Idealmente, as ferramentas de automação deveriam prover um gerador de objetos “inteligente” o suficiente para preencher com valores adequados os diversos atributos das entidades de negócio a serem testadas, de forma a não infringir as regras de negócio existentes. Além disso, existe a necessidade de suporte à gestão de configuração e à configuração do ambiente de teste, em que seja possível configurar tudo o que é necessário para o início da execução de um teste.

Resumindo, as principais necessidades relacionadas a esse item são: (i) apoio para a geração de dados para execução de um teste, de forma a gerar objetos automaticamente sem infringir as regras do negócio; (ii) geração de código de povoamento a partir de um banco de dados povoado, visto que foi identificada a necessidade de se utilizar uma instância de um banco como conjunto de dados iniciais para um teste; (iii) suporte à gestão de configuração, permitindo o controle de versão dos artefatos de teste mantidos pela ferramenta; (iv) configuração do ambiente de teste, realizando todas as tarefas necessárias para o início dos testes, como por exemplo, baixar código, compilar e povoar o mecanismo de persistência utilizado pela aplicação e (v) fornecer apoio para o acompanhamento de defeitos.

### 5.1.4 Execução automática

A execução automática dos testes é parte fundamental para qualquer ferramenta de automação. No entanto, a simples execução não é a única necessidade associada a este item. Além da (i) execução automática, são necessários (ii) a execução automática com possibilidade de distribuição, possibilitando assim uma redução no tempo para executar um conjunto de testes; (iii) agrupamento de testes em conjuntos, gerando baterias de testes; (iv) agendamento da execução, permitindo assim o escalonamento da execução de um agrupamento de testes e (v) interrupção e retomada da execução de testes.

### 5.1.5 Acompanhamento dos testes

Existem necessidades relacionadas ao acompanhamento dos testes, cujo objetivo é propiciar um maior controle sobre as atividades de teste em execução dentro de um projeto. Com isso, decisões mais precisas podem ser tomadas e potenciais riscos ao desenvolvimento podem ser mitigados mais cedo. As necessidades identificadas com relação a este

item foram: (i) identificação da produtividade dos testadores, para permitir um acompanhamento mais preciso do projeto e possibilitar assim melhores estimativas futuras e (ii) verificação da reincidência de falhas, procurando identificar problemas associados ao desenvolvimento.

### 5.1.6 Avaliação de resultados

A avaliação dos resultados dos testes é a atividade em que os resultados obtidos são comparados com as metas identificadas no plano de teste. Isso é feito para determinar se a atividade de teste pode ser considerada bem sucedida. A principal necessidade relativa a este item é a avaliação da cobertura dos testes levando em consideração diferentes critérios. A análise de cobertura deve ter diferentes opções de agrupamento, exibindo a cobertura alcançada no agrupamento e nas suas partes, como por exemplo, cobertura alcançada em uma classe e nos diversos métodos que compõem a classe. Idealmente, deveria haver um mecanismo que facilitasse a introdução de um novo critério na ferramenta, na forma de um *add-in* que pudesse ser desenvolvido por uma organização e adaptado para funcionar na ferramenta.

### 5.1.7 Documentação de testes

A documentação dos testes é uma necessidade relatada por grande parte dos profissionais da área de teste, pois é uma tarefa repetitiva e trabalhosa. Os principais requisitos levantados com relação a esse item foram: (i) geração da documentação dos testes, com entradas, saídas esperadas, dados iniciais para o teste e procedimentos de teste a serem executados; (ii) geração da documentação de execução dos testes, indicando resultado da execução e cobertura atingida; (iii) documentação gerencial sobre os testes, gerando relatórios estatísticos indicando quantidade, qualidade, cobertura, esforço gasto para criação, complexidade das partes de *software* envolvidas, percentagem de esforço no projeto e demais informações gerenciais; e (iv) fácil visualização e navegação dessa documentação, facilitando o seu uso pelas partes interessadas.

### 5.1.8 Rastreabilidade dos testes

A rastreabilidade é o grau de relacionamento que pode ser estabelecido entre dois ou mais produtos do processo de desenvolvimento. Esse é um aspecto muito importante para as atividades de gestão de um projeto, pois essa informação facilita o desenvolvimento e a manutenção de um produto por meio da fácil identificação de elementos relacionados a uma parte do sistema. As principais necessidades associadas a esse item foram: (i) identificação da rastreabilidade dos testes e dos seus artefatos relacionados (requisitos, desenho e código) e (ii) identificação de testes afetados por mudanças nos artefatos.

### 5.1.9 Apoio à implementação de testes

Uma ferramenta de automação de testes deve prover mecanismos para facilitar a criação de testes manuais. Isto é necessário mesmo exis-

tindo um mecanismo para geração automática de testes, uma vez que a equipe de teste pode achar que a cobertura alcançada pelos testes automáticos não seja a ideal ou porque pode ser muito cara, e até mesmo inviável, a automação de testes para uma parte do sistema. Isso implica na necessidade de (i) facilidade para criação de testes adicionais aos testes gerados automaticamente.

#### 5.1.10 Integração

O uso de uma ferramenta de teste específica pode criar uma dependência na organização desenvolvedora, tornando difícil uma mudança de ferramenta em virtude dessa dependência. Idealmente, deveria ser possível mudar a tecnologia dos sistemas e aproveitar todos os testes já desenvolvidos, minimizando os custos associados. Assim, foram identificadas algumas necessidades associadas a essa questão: (i) importação de testes criados em outras ferramentas; (ii) exportação dos testes criados na ferramenta para diferentes tecnologias; (iii) capacidade de trabalhar com sistemas *Web* e (iv) capacidade de trabalhar com outras tecnologias, como Java, C/C++ e Delphi.

#### 5.1.11 Aquisição e implantação

Embora ferramentas gratuitas sejam sempre bem recebidas pela comunidade de desenvolvimento de *software*, ferramentas pagas, mas que implementem corretamente as funcionalidades desejadas, ainda possuem um grande potencial de mercado. No entanto, além da necessidade relacionada ao “custo”, que versa que uma ferramenta deveria ser (i) gratuita ou de baixo custo, existem outras necessidades: (ii) possuir boa documentação e (iii) ser acompanhada de cursos de capacitação.

#### 5.1.12 Usabilidade e desempenho

Por mais que uma ferramenta possua um bom manual de usuário e cursos de capacitação, é importante que ela seja adequada ao uso, pois assim seus usuários terão maior produtividade. Destaca-se que nem sempre usabilidade está associada à facilidade de uso, principalmente neste caso, em que os usuários das ferramentas de teste são geralmente pessoas com conhecimentos avançados em informática e que não necessitam de tanto apoio gráfico como outros tipos de usuário.

Parte das ferramentas de teste existentes no mercado possui baixa usabilidade, sendo impróprias para seu público alvo. Essas ferramentas provavelmente devem ter sido desenvolvidas sem levar em consideração a característica de usabilidade do produto. Outras ferramentas, notadamente as mais recentes, consomem recursos excessivos das máquinas utilizadas, sendo inviáveis para adoção dentro de certas organizações. Com base nesses dados, duas necessidades foram identificadas: (i) boa usabilidade, favorecendo a produtividade dos usuários e (ii) baixo consumo de recursos (processador e memória).

### 5.1.13Acesso

Uma ferramenta de automação deve prover mecanismos que permitam a existência de diferentes usuários trabalhando em um mesmo projeto, com diferentes níveis de acesso. Esse acesso poderia ser remoto, facilitando assim o trabalho com a ferramenta. Essas características estão relacionadas às seguintes necessidades identificadas: (i) controle de permissões por função, configurável por usuário e grupo, dentro de um projeto, (ii) acesso remoto e (iii) funcionamento em diversas plataformas.

## 5.2 ASPECTOS TÉCNICOS RELACIONADOS

A Figura 5 apresenta os aspectos técnicos que foram levantados a partir das sugestões dos profissionais que participaram do estudo. Esses aspectos técnicos correspondem às funções que as ferramentas de teste deveriam conter para atender às necessidades identificadas. Para organizar as funções identificadas, essas especificações foram reunidas em 14 grupos: (1) geração de plano de teste; (2) geração de dados; (3) geração de testes funcionais; (4) avaliação de testes; (5) geração manual de testes; (6) rastreamento; (7) integração; (8) modelagem de teste; (9) geração de testes não-funcionais; (10) geração de relatórios; (11) suporte da ferramenta; (12) arquitetura da ferramenta; (13) auxílio da ferramenta; (14) execução de testes.

| Grupo | Aspecto Técnico |
|---|---|
| 1 | Gestão de dados do plano de teste |
| | Estimador de complexidade de caso de uso |
| | Estimador de prazo para execução de tarefa de teste |
| | Base de dados histórica de projetos |
| | Calculador de produtividade |
| | Registro de tarefas |
| | Alocador inteligente de tarefa |
| | Integração com ferramentas de gerenciamento de projetos |
| 2 | Gerador de objetos |
| | Gerador de povoador |
| | Gerador de valores |
| 3 | Gerador de entradas utilizando critérios |
| | Oráculo para gerar as saídas esperadas |
| | Interpretador de condições modeladas em uma linguagem formal |
| | Extrator de dados de modelos descrevendo o sistema |
| 4 | Avaliador de cobertura |
| | Analisador de qualidade de testes baseado em mutação |

| Grupo | Aspecto Técnico |
|---|---|
| 5 | Mecanismo de captura-reprodução |
| | Gerador de teste com apoio da especificação |
| | Uso de linguagens de alto nível |
| | Acesso as funcoes do SO |
| | Acesso ao mecanismo de persistência |
| 6 | Integração com ferramentas de gestão de requisitos |
| | Detector de alterações entre as visões |
| 7 | Integração com ferramentas de acompanhamento de bugs |
| | Integração com ferramentas de gestão de configuração |
| 8 | Representação dos testes usando um modelo independente de tecnologia |
| | Regras de transformação entre modelos |
| | Especificação de modelos para descrição do sistema |
| | Uso de uma linguagem para especificação de restrições no modelo |
| | Utilização de um padrão para descrever a arquitetura de software |

| Grupo | Aspecto Técnico |
|---|---|
| 9 | Verificador de arquitetura |
| | Gerador de testes de desempenho e estresse |
| | Integração com ferramentas para manipulação de requisições e respostas |
| | Integração com sistema de varredura de portas |
| | Gerador de testes de segurança |
| 10 | Gerador de relatório com formato definido pelo usuário |
| | Gerador de gráficos com fonte de dados e formato definido pelo usuário |
| | Acesso a qualquer informação de teste existente no modelo de testes |
| | Uso de hiperlinks e agrupadores nos relatórios |
| 11 | Cadastro de usuários |
| | Cadastro de grupos |
| | Cadastro de projeto |
| | Cadastro de equipe |
| | Mecanismo de busca para qualquer padrão informado pelo usuário |
| | Cadastro de perfis com possibilidade de definição de permissões |

| Grupo | Aspecto Técnico |
|---|---|
| 12 | Uso de software livres |
| | Uso de tecnologia Web |
| | Seguir um guia de estilo |
| | Utilizar terminologia adequada ao contexto |
| 13 | Help on-line |
| | Manual de usuário |
| | Sitio de apoio |
| | Curso de formação |
| 14 | Gerenciador de grids para execução distribuída |
| | Possibilidade de pausa e retomada da execução |
| | Agrupador e escalonador de testes |
| | Gerador de log de execução |
| | Simulador de interfaces de hardware e software |
| | Integração com uma linguagem de script para config. do ambiente de teste |
| | Comparador de arquivos ignorando padrões configuráveis |
| | Povoador de dados |
| | Gerenciador de transações |
| | Analisador de falhas |
| | Cadastramento automático de falhas |
| | Simulador de plataformas |
| | Simulador de banda de rede |

Figura 5: Aspectos técnicos identificados

Fonte: os autores

## 5.3 CORRELAÇÃO ENTRE OS REQUISITOS DO CLIENTE E AS ESPECIFICAÇÕES TÉCNICAS DO PRODUTO

Como já mencionado, primeiro foi feito o levantamento dos requisitos e aspectos técnicos com os profissionais participantes do estudo. Os requisitos foram agrupados em uma tabela, chamada de Tabela de Desdobramento da Qualidade Exigida e os aspectos técnicos em outra, na Tabela de Desdobramento de Características da Qualidade. Em seguida, aplicando a terceira etapa do SQFD, foi realizada, com a participação dos profissionais envolvidos no estudo, a correlação entre a Tabela de Desdobramento da Qualidade Exigida, que contém os requisitos levantados e a Tabela de Desdobramento de Características da Qualidade, criada com as funções identificadas. Essa correlação foi baseada em um consenso entre os participantes ao indicarem quais aspectos técnicos estavam relacionados a quais requisitos e qual era a intensidade desse relacionamento.

| | | | | Nível 1 | Gerador de plano de teste | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Funções do Produto / Requisitos | Nível 2 | Gestão de dados do plano de teste | Estimador de complexidade de caso de uso | Estimador de prazo para execução de tarefa de teste | Base de dados histórica de projetos | Calculador de produtividade | Registro de tarefas | Alocador inteligente de tarefa | Integração com ferramentas de gerenciamento de projetos |
| | Nível 1 | Nível 2 | | | | | | | | | | |
| 1 | Planejamento da atividade de teste | Apoio para a criação de um Plano de testes | | 1 | 9 | | 9 | 9 | 9 | 3 | 9 | 9 |
| | | Apoio para estimativa de prazos para as atividades de teste | | 2 | | 9 | 9 | 9 | 9 | 9 | 3 | |
| | | Apoio para identificação da complexidade das partes do sistema | | 3 | | 9 | | 3 | 3 | 3 | | |
| | | Integração com ferramentas de gerenciamento de projeto. | | 4 | 3 | | 3 | 3 | 3 | 3 | 3 | 9 |
| | de testes | Geração automática de testes funcionais | | 5 | | | | | | | | |
| | | Geração automática de testes de desempenho e estresse | | 6 | | | | | | | | |

Figura 6: Excerto da Casa de Qualidade construída

Fonte: os autores

O resultado dessa correlação permitiu a construção da Casa de Qualidade, em que é possível visualizar a correlação entre as necessidades e os aspectos técnicos identificados. Visto que foram levantados 43 requisitos e 67 aspectos técnicos, a matriz representando a Casa de Qualidade construída possui 43 linhas e 67 colunas. A Figura 6 apresenta um excerto da matriz construída. Quando existe alguma correlação entre um requisito e um aspecto técnico, foi utilizado o valor “9”, para representar uma correção forte, “3” para representar uma correlação fraca e “1” para indicar uma correlação possível.

Observando a Figura 6 pode-se constatar o grau de correlação entre os requisitos e os aspectos técnicos ilustrados. O requisito *apoio para criação de um plano de testes*, por exemplo, possui correlação forte com o aspecto técnico *gestão de dados do plano de testes*, correlação fraca com *registro de tarefas* e nenhuma correlação com o aspecto técnico *estimador de complexidade de caso de uso*.

## 5.4 REQUISITOS PRIORITÁRIOS DOS CLIENTES

A Figura 7 exibe a lista das necessidades com o grau de importância médio atribuído pela equipe que participou deste trabalho. Foi utilizada uma escala que vai de um a três, sendo três o grau mais importante. Analisando-se a tabela de requisitos prioritários, pode-se identificar quais requisitos são considerados mais relevantes para ferramentas de apoio ao teste de *software*. Os requisitos que possuem o mesmo grau de importância são igualmente relevantes. Com base na Figura 7 pode-se constatar, por exemplo, que entre os requisitos considerados mais importantes estão a *execução automática* (Nr. 1) e a *identificação de testes afetados por mudanças em artefatos relacionados* (Nr. 15), pois ambos possuem grau três.

| Nr. | Requisito | Priorização |
|---|---|---|
| 1 | Execução automática | 3 |
| 2 | Acompanhamento de Bugs | 3 |
| 3 | Documentação gerencial dos testes | 3 |
| 4 | Facilidade para criação de testes adicionais aos testes gerados automaticamente | 3 |
| 5 | Possuir boa documentação | 3 |
| 6 | Agendamento da execução | 3 |
| 7 | Agrupamento de testes | 3 |
| 8 | Documentação de execução dos testes | 3 |
| 9 | Documentação dos casos de testes e procedimentos de teste | 3 |
| 10 | Execução de testes com possibilidade de distribuição | 3 |
| 11 | Geração automática de testes funcionais | 3 |
| 12 | Gestao de configuração de artefatos de teste | 3 |
| 13 | Apoio para estimativa de prazos para as atividades de teste | 3 |
| 14 | Facilidade para interrupção e retomada dos testes | 3 |
| 15 | Identificação de testes afetados por mudanças em artefatos relacionados | 3 |
| 16 | Apoio para a criação de um Plano de testes | 2 |
| 17 | Apoio para a geração de dados para execução de um teste | 2 |
| 18 | Avaliação da cobertura/qualidade dos testes | 2 |
| 19 | Fácil visualização da documentação | 2 |
| 20 | Geração automática de testes de desempenho e estresse | 2 |
| 21 | Importação de testes criados em outras ferramentas de teste | 2 |
| 22 | Rastreabilidade entre os teste e os artefatos relacionados (código, ERSw, Desenho). | 2 |

| Nr. | Requisito | Priorização |
|---|---|---|
| 23 | Suporte para geração de testes para sistemas tecnologias: Java, C++, Delphi. | 2 |
| 24 | Suporte para geração de testes para sistemas Web. | 2 |
| 25 | Apoio para identificação da complexidade das partes do sistema | 2 |
| 26 | Controle de permissões configurável por usuário ou por grupos | 2 |
| 27 | Funcionamento em múltiplas plataformas: window, linux. | 2 |
| 28 | Boa usabilidade, favorecendo a produtividade dos seus usuários | 2 |
| 29 | Geração de testes independentes e auto-contidos | 2 |
| 30 | Identificação da produtividade dos testadores | 2 |
| 31 | Integração com ferramentas de gerenciamento de projeto. | 2 |
| 32 | Configuração de ambiente para execução dos testes | 2 |
| 33 | Exportação dos testes criados na ferramenta para outras tecnologias | 2 |
| 34 | Geração automática de testes de segurança | 2 |
| 35 | Verificação da reincidencias de falhas | 2 |
| 36 | Possuir cursos de formação | 2 |
| 37 | Ser rápida, não consumindo muitos recursos | 2 |
| 38 | Geração automática de testes para bugs cadastrados por fora da ferramenta, através da descrição do bug | 2 |
| 39 | Geração de um povoador a partir de um banco de dados já povoado | 2 |
| 40 | Apoio para identificação da estabilidade de um caso de uso | 2 |
| 41 | Ser gratuita ou baixo custo | 2 |
| 42 | Acesso remoto | 1 |
| 43 | Apoio aos testes de usabilidade | 1 |

Figura 7: Priorização dos requisitos identificados

Fonte: os autores

## 5.5 ESPECIFICAÇÕES TÉCNICAS PRIORITÁRIAS

A partir dos valores utilizados na Casa de Qualidade desenvolvida e da priorização dos requisitos levantados, partiu-se para a última etapa do método básico do SQFD, na qual foram priorizados os aspectos técnicos identificados. Para isso, foi multiplicado, com respeito a cada aspecto técnico, o grau de importância do requisito pelo peso da correlação entre esse requisito e o aspecto técnico correlacionado. A soma desses produtos indica o peso de cada aspecto técnico e, com esses pesos transformados em porcentagem, foi montada a tabela, ilustrada na Figura 8, com a priorização dos aspectos técnicos levantados.

| Nr. | Função/característica do produto | Peso(%) |
|---|---|---|
| 1 | Representação dos testes usando um modelo independente de tecnologia | 3,83% |
| 2 | Regra de transformação entre modelos | 3,35% |
| 3 | Gerador de entradas utilizando critérios | 3,29% |
| 4 | Gerador de objetos | 3,29% |
| 5 | Interpretador de condições modeladas em uma linguagem formal | 3,29% |
| 6 | Gerenciador de transações | 3,17% |
| 7 | Especificação de modelos para descrição do sistema | 2,86% |
| 8 | Adoção de uma linguagem para especificação de restrições no modelo | 2,84% |
| 9 | Gerador de valores | 2,82% |
| 10 | Extrator de dados de modelos descrevendo o sistema | 2,64% |
| 11 | Acesso ao mecanismo de persistência | 2,52% |
| 12 | Acesso as funcoes do SO | 2,52% |
| 13 | Integração com ferramentas de gestão de requisitos | 2,52% |
| 14 | Oráculo para gerar as saídas esperadas | 2,47% |
| 15 | Povoador de dados | 2,11% |
| 16 | Cadastro de equipe | 2,02% |
| 17 | Cadastro de projeto | 2,02% |
| 18 | Base de dados histórica de projetos | 2,00% |
| 19 | Cadastro de grupos | 1,84% |
| 20 | Cadastro de usuários | 1,84% |
| 21 | Executor de teste com possibilidade de pausa e retomada da execução | 1,76% |
| 22 | Integração com ferramentas de acompanhamento de bugs | 1,76% |
| 23 | Integração com ferramentas de gestão de configuração | 1,76% |
| 24 | Gerador de log de execução de testes | 1,64% |
| 25 | Utilização de um padrão para uso de elementos descrevendo a arquitetura de | 1,64% |
| 26 | Acesso a qualquer informação de teste existente no modelo de testes | 1,58% |
| 27 | Gerador de relatório com formato definido pelo usuário | 1,58% |
| 28 | Avaliador de cobertura | 1,47% |
| 29 | Calculador de produtividade | 1,47% |
| 30 | Estimador de complexidade de caso de uso | 1,41% |
| 31 | Agrupador e escalonador de testes | 1,25% |
| 32 | Verificador de arquitetura | 1,25% |
| 33 | Comparador de arquivos ignorando padrões configuráveis | 1,23% |
| 34 | Gerador de testes de segurança | 1,23% |
| 35 | Registro de tarefas | 1,23% |
| 36 | Uso de linguagens de alto nível | 1,17% |
| 37 | Uso de tecnologia Web | 1,12% |
| 38 | Gerador de gráficos com fonte de dados e formato definido pelo usuário | 1,06% |
| 39 | Gerador de testes de desempenho e estresse | 1,06% |
| 40 | Simulador de banda de rede | 1,06% |
| 41 | Simulador de interfaces de hardware e software | 1,06% |
| 42 | Simulador de plataformas de hardware | 1,06% |
| 43 | Estimador de prazo para execução de tarefa de teste | 1,00% |
| 44 | Gestão de dados do plano de teste | 1,00% |
| 45 | Mecanismo de captura-reprodução | 0,92% |
| 46 | Definição de permissões por grupo, projeto, equipe | 0,88% |
| 47 | Gerador de povoador | 0,88% |
| 48 | Utilizar terminologia adequada ao contexto | 0,88% |
| 49 | Integração com ferramentas de gerenciamento de projetos | 0,82% |
| 50 | Cadastramento automático de falhas | 0,80% |
| 51 | Gerador de teste com apoio da especificação | 0,78% |
| 52 | Gerenciador de grids para execução distribuída de testes | 0,76% |
| 53 | Help on-line | 0,70% |
| 54 | Integração com ferramentas para manipulação de requisições e respostas em transações | 0,70% |
| 55 | Integração com sistema de varredura de portas | 0,70% |
| 56 | Manual de usuário | 0,70% |
| 57 | Sítio de apoio com exemplos de uso | 0,70% |
| 58 | Alocador inteligente de tarefa | 0,65% |
| 59 | Analisador de falhas | 0,57% |
| 60 | Curso de formação | 0,53% |
| 61 | Detector de alterações entre as visões, exibindo partes afetadas | 0,53% |
| 62 | Seguir um guia de estilo | 0,53% |
| 63 | Uso de software livres | 0,47% |
| 64 | Analisador de qualidade de testes baseado em mutação | 0,35% |
| 65 | Integração com uma linguagem de script para configuração do ambiente de teste | 0,35% |
| 66 | Mecanismo de busca para qualquer padrão informado pelo usuário | 0,35% |
| 67 | Uso de hiperlinks e agrupadores nos relatórios | 0,35% |

Figura 8: Funções do produto priorizadas por seus pesos em porcentagem

Fonte: os autores

Utilizando como exemplo o caso do aspecto técnico *alocador inteligente de tarefa* (Nr. 58) é possível entender melhor o cálculo dos pesos. Todas as correlações desse aspecto estão ilustradas na Figura 6. Ele possui correlação forte com o requisito *apoio para a criação de um plano de testes*, que possui grau de importância 2 e correlação fraca com os requisitos *apoio para estimativas de prazos para as atividades de teste*, o qual possui grau 3 e *integração com ferramentas de gerenciamento de projetos,* que possui grau 2. Assim, para o cálculo do peso da função denominada *alocador inteligente de tarefa* devem ser feitas as seguintes operações: 9x2+3x3+3x2 = 33. A porcentagem é obtida dividindo-se o valor obtido (33) pela soma de todos os valores de todas as funções priorizadas (5111). Nesse caso, a porcentagem relativa do item é 33/5111=0,65%.

Com os requisitos e aspectos técnicos priorizados, é possível avaliar as ferramentas de testes, bem como efetuar comparações entre ferramentas utilizando essas informações. A partir desses dados, pode-se criar um mecanismo para avaliação de ferramentas de testes funcionais e de desempenho (VELOSO *et al.*, 2010). É importante destacar que, com base nas avaliações dessas ferramentas, confirmou-se que as ferramentas de apoio ao teste ainda deixam muito a desejar, visto que muitos aspectos

técnicos encontrados neste trabalho foram considerados ausentes nas avaliações realizadas.

## 6 CONCLUSÕES

Neste trabalho foram levantados os principais requisitos relacionados às ferramentas de apoio às atividades de teste, identificadas a partir de uma série de reuniões envolvendo profissionais relacionados ao desenvolvimento de *software*. Foi realizada uma priorização dos requisitos identificados, a partir do grau de importância médio identificado pelos participantes da pesquisa realizada, conforme indicado pelo uso do SQFD. A partir desses requisitos também foram identificados aspectos técnicos relacionados, além de ter sido realizada a correlação entre os aspectos técnicos e os requisitos, permitindo assim a identificação das funções que devem ser priorizadas no desenvolvimento de uma ferramenta de apoio ao teste de *software*. Essa informação é valiosa para os pesquisadores saberem quais são os aspectos de maior relevância para os clientes e usuários de ferramentas de teste, permitindo assim a seleção de tópicos de investigação considerados mais interessantes. Essa é uma importante contribuição deste trabalho.

Outra contribuição importante está diretamente ligada à Casa da Qualidade obtida, uma vez que ela possui informações para definir estratégias relacionadas ao desenvolvimento de ferramentas de apoio aos testes. A partir do uso da Casa de Qualidade é possível realizar uma avaliação competitiva entre ferramentas de teste. Essa avaliação corresponde a uma pesquisa de mercado quantitativa, que busca identificar como os usuários de ferramentas de teste percebem o desempenho de uma ferramenta, em comparação com outras ferramentas concorrentes. A partir desse conhecimento e da avaliação do cliente de uma ferramenta de teste, é possível criar uma referência de características versus satisfação dos usuários. Isso pode ser extremamente útil para se avaliar ferramentas e para se guiar a construção de novas ferramentas, a partir da projeção da qualidade da ferramenta a ser desenvolvida. Essa é uma contribuição estratégica para qualquer empresa desenvolvedora de ferramentas de apoio ao teste.

A identificação dos requisitos e aspectos técnicos, conforme mencionado, possibilita uma avaliação competitiva das ferramentas de apoio ao teste existentes. A partir dos resultados desta pesquisa foi possível construir um mecanismo para avaliação de ferramentas de teste. O mecanismo construído foi aplicado na avaliação de duas categorias de ferramentas: ferramentas para o teste funcional e ferramentas para o teste de desempenho. As avaliações realizadas deixaram claro que existem muitos aspectos para melhorar nas ferramentas, além de permitir uma comparação simples, baseada em dados científicos, de opções comerciais existentes para uso prático.

Uma extensão para este trabalho é uma ampliação do conjunto de participantes, visando assim a obter mais informações sobre os requisitos e com isso encontrar mais aspectos técnicos associados, elevando ainda mais a relevância do resultado já obtido.

## REFERÊNCIAS


AKAO, Y. *Quality function deployment*: integrating customer requirements into product design. Cambridge, MA: Productivity Press, 1990.

AKAO, Y. *Introdução ao desdobramento da qualidade*. Belo Horizonte: Fundação Cristiano Ottoni, 1996.

ALVES, N. R.; PADUA, C. I. P. S. Especificação de requisitos de usabilidade utilizando-se o método desdobramento da função qualidade. *Proceedings of the First Ibero-American Symposium on Software Engineering and Knowledge Engineering*, JIISIC'01, Buenos Aires, Argentina, 2001.

DENNIS, A. R.; HAYES, G. S.; DANIELS, R. M. Jr. Business process modeling with group support systems. *Journal of Management Information Systems*, p. 115-142, Spring, 1999.

HAAG, S.; RAJA, M.; SCHKADE, L. Quality function deployment usage in *software* development. *Communications of the ACM*, v. 39, p. 41-49, Jan. 1996. doi:10.1145/234173.234178

HOFFMANN, M.; KUHN, N.; WEBER, M.; BITTNER M. Requirements for requirements management tool. *Proceedings of the IEEE International Requirements Engineering Conference* (RE'04), p. 301-308, Kyoto, Japão, 2004.

IEEE. *Guide to the software engineering body of knowledge*. IEEE Computer Society, 2004.

IOANNOU, G.; PRAMATARIS, K.; PRASTACOS, G. Quality function deployment approach to web site development: applications for electronic retailing. *Les Cahiers du Management Technologique*, v. 13, n. 3, 2004.

KARLSSON, J. Managing *software* requirements using quality function deployment. *Software Quality Journal*, v. 6, p. 311-326, 1997. doi:10.1023/A:1018580522999

MIZUNO, S; AKAO, Y. *QFD: the customer-driven approach to quality planning and development*. Tokyo: Asian Productivity Organization, 1994.

NIST. *The economic impacts of inadequate infrastructure for software testing*. National Institute of Standards and Technology, 2002. Disponível em: http://www.nist.gov/director/planning/upload/report02-3.pdf. Acesso em: 27/10/2010.

PERRY, W. *Effective methods for software testing*, 3. ed, Wiley Publishing Inc., 2006.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*, 5. ed., McGraw-Hill Higher Education, 2001.

SANTOS-NETO, P.; RESENDE, R.; PÁDUA, C. Requirements for information systems model-based testing. *Proceedings of the 2007 ACM Symposium on Applied Computing*, Seoul, Korea, 2007, p. 1409-1415. doi:10.1145/1244002.1244306

SOWA, J. F.; ZACHMAN, J. A. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, v. 31, n. 3, p. 590-616, 1992. doi:10.1147/sj.313.0590

TAN, K.; XIE, M.; CHIA, E. Quality function deployment and its use in designing information technology systems. *International Journal of Quality and Reliability Management*, v. 15, n. 6, p. 634-671, 1998. doi:10.1108/02656719810196234

VELOSO, J.; SANTOS NETO, P.; SANTOS, I. S.; BRITTO, R. S. *Avaliação de ferramentas de apoio ao teste de sistemas de informação*. Anais do VI Simpósio Brasileiro de Sistemas de Informação (SBSI), Marabá, PA, 2010.

WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, B.; REGNELL, B.; WESSLEN A. *Experimentation in software engineering*: an introduction. Kluwer Academic Publishers, 2000.