Integrando requisitos, gestão de defeitos e gestão de testes de software

Narcizo Thiago Maniçoba, Marcello Thiry, Anita Maria da Rocha Fernandes

Curso de Ciência da Computação - Campus Kobrasol - São José
Universidade do Vale do Itajaí – (UNIVALI) – São José – SC
{narcizo, thiry, anita.fernandes}@univali.br

Abstract. *Considering the management systems for requirements, tests and defects, it is clear, in the main free tools currently used, that there is no an integrated solution to document, in the same system, the tests performed and the defects found during the software development process . Besides the fact of having to use at least two different tools, these systems do not employ good usability practices, so companies tend not to document or document only part of the process. Given these facts, this work develops a single system that incorporates the essential features present in free tools and employs good usability practices.*

Resumo. *Considerando os sistemas para gestão de requisitos, testes e defeitos, percebe-se, nas principais ferramentas gratuitas usadas atualmente, que não há uma solução integrada para documentar, no mesmo sistema, os testes executados e os defeitos encontrados durante o processo de desenvolvimento de software. Além do fato de ter que usar no mínimo duas ferramentas diferentes, esses sistemas não empregam boas práticas de usabilidade, então as empresas tendem a não documentar ou documentar apenas uma parte do processo. Diante desses fatos, este trabalho desenvolveu um único sistema que contempla as funcionalidades essenciais presentes nas ferramentas gratuitas e emprega boas práticas de usabilidade.*

1. Introdução

Atualmente, o software está presente em quase todas as atividades dos seres humanos. Desta forma, problemas com seu funcionamento ou dificuldades em utilizá-lo corretamente podem acarretar em diferentes problemas e variados graus de criticidade. Cientes destes riscos e com o objetivo de agregar confiabilidade e qualidade aos softwares, autores como Myers, Badgett e Sandler (2011), e Pfleeger (2004), definem o processo de teste como sendo o ato de executar um software com o objetivo de encontrar defeitos e, além disso, verificar se o comportamento do software está de acordo com os requisitos levantados junto ao cliente fazendo o que deve fazer e não fazendo o que não deve fazer.

Existe uma relação muito próxima entre um caso de teste e um relato de defeito encontrado, mas na prática não é o que se vê na grande maioria das ferramentas gratuitas de apoio a teste de software. Enquanto algumas oferecem a gestão dos requisitos e testes, outras oferecem a gestão dos defeitos. Diante deste fato, os usuários acabam tendo que usar pelo menos duas ferramentas diferentes, com características diferentes como layout, operação e autenticação para documentar os casos de teste executados e os defeitos encontrados. Além disso, estas ferramentas não apresentam boas práticas de usabilidade. Nas pesquisas prévias para elaboração deste trabalho, foi encontrada a

ferramenta *RTH*[1] que oferece a gestão dos requisitos, testes e defeitos. Entretanto, por ser antiga e não sofrer atualizações há quase uma década, não recebeu novas funcionalidades e sua interface carece de requisitos de usabilidade.

Dentro deste contexto, foi desenvolvida uma ferramenta que integra a gestão de requisitos, testes e defeitos, oferecendo alta usabilidade para seus usuários. O foco está no armazenamento, associação e acompanhamento das informações de requisitos, planos de teste executados e defeitos encontrados. Espera-se que a gestão dos testes a serem executados e a gestão de defeitos encontrados integrados em um único sistema com usabilidade adequada possa ampliar a possibilidade das empresas documentarem o processo de teste de forma mais adequada.

Este artigo apresenta o desenvolvimento e avaliação da ferramenta Testicida, assim como os resultados e conclusões obtidos. O trabalho está organizado em 4 seções, sendo a seção 2 sobre a definição dos requisitos essenciais para que fosse desenvolvida a ferramenta Testicida; a seção 3 sobre a ferramenta e seus principais diferenciais em relação às existentes; a seção 4 sobre a percepção e feedback dos usuários após utilizarem a ferramenta; e a seção 5 a conclusão deste trabalho.

2. Requisitos para um ambiente integrado

Inicialmente, foram analisadas as ferramentas de gerenciamento de testes e de defeitos gratuitas mais utilizadas. Os critérios adotados para a seleção consideraram ferramentas com foco nos níveis de Teste de Sistema e Teste de Aceitação utilizando Técnica de Testes Caixa-preta. Também deveriam ser Web, gratuitas sem limitação de quantidade de usuários e projetos. As buscas na Internet foram feitas utilizando-se termos como gestão de defeitos e *bug trackers*. A Tabela 1 apresenta as ferramentas que atenderam aos requisitos estabelecidos.

Tabela 1. Seleção das ferramentas de gestão de defeitos.

Nome	Site oficial
Bloodhound	bloodhound.apache.org
Bug-A-Boo	www.bug-a-boo.org
BugNET	www.bugnetproject.com
BugTracker.NET	www.ifdefined.com
Bugzilla	www.bugzilla.org
Eventum	www.launchpad.net/eventum
Flyspray	www.flyspray.org
Fossil	www.fossil-scm.org
JTrac	www.jtrac.info
Kwok	www.kwoksys.com
Mantis	www.mantisbt.org
oops-easytrack	www.oops-easytrack.sourceforge.net
phpBugTracker	www.github.com/a-v-k/phpBugTracker
Redmine	www.redmine.org
Request Tracker	www.bestpractical.com
Roundup	www.roundup.sourceforge.net
RTH	www.requirementsandtestinghub.wordpress.com
Scarab	www.memocomp.de/scarab/1.0.22
Trac	www.trac.edgewall.org
WebIssues	webissues.mimec.org
zenTrack	sourceforge.net/projects/zentrack

Com a seleção dessas ferramentas, a próxima etapa foi tentar encontrar pesquisas que apresentassem as três ferramentas mais utilizadas. O estudo apresentado pela revista

[1] RTH: www.requirementsandtestinghub.wordpress.com

Testing Experience (2010) apontou a ferramenta *Bugzilla* como a mais utilizada por usuários com 60%, em segundo *Mantis* com 30% e em terceiro *Trac* com 6%.

Com o intuito de confirmar os resultados apresentados por esta revista e verificar se houve mudança de popularidade nestes últimos 5 anos, optou-se por utilizar o *Google Trends* para realizar um comparativo de popularidade de busca por estas ferramentas. Partiu-se do pressuposto que as mais buscadas são as mais populares, logo as mais utilizadas. Comparando os resultados obtidos com a pesquisa realizada em (*Testing Experience*, 2010), chegou-se à mesma conclusão: *Bugzilla* e *Mantis* têm uma representação mais expressiva de popularidade, enquanto *Trac* e *Redmine* apareceram na sequência, mas com participação bem menor.

Mantis e *Bugzilla* foram consequentemente selecionadas devido aos resultados das duas pesquisas. Como as pontuações das ferramentas a partir da terceira posição não tiveram grande expressividade, buscou-se a próxima ferramenta que apresentasse mais recursos e boas práticas de usabilidade baseadas nos 10 princípios heurísticos de Nielsen (1993) e outras apresentadas por Krug (2006). Portanto, agregaria mais valor ao levantamento de requisitos para desenvolver a ferramenta proposta por este trabalho.

Na sequência iniciou-se a pesquisa pelas ferramentas de gestão de testes. Os critérios adotados foram os mesmos das ferramentas de gestão de defeitos. As buscas na Internet foram feitas utilizando-se termos como gestão de testes e *test management tool.* As ferramentas que atenderam aos requisitos são apresentadas na Tabela 2.

Tabela 2. Seleção das ferramentas de gestão de testes.

Nome	Site oficial
QATraq	sourceforge.net/projects/qatraq
Radi-testdir	radi-testdir.sourceforge.net/radi_home.html
RTH	requirementsandtestinghub.wordpress.com
Sitechco	www.sitechco.com
Test Case Web	tcw.sourceforge.net
Tesly	tesly.sourceforge.net
Testia Tarantula	www.tarantula.fi
Testitool	www.majordojo.com
TestLink	www.testlink.org
Testmaster	testmaster.sourceforge.net
Testopia	sourceforge.net/projects/testopia

A seleção das três ferramentas de gestão de testes mais adotadas também foi baseada inicialmente no estudo apresentado em (*Testing Experience*, 2010). A ferramenta *TestLink* foi indicada como a mais utilizada por usuários com 53%, em segundo *Testopia* com 17% e em terceiro *Redmine* com 12%. A ferramenta *RTH*, única encontrada que possui a gestão dos requisitos, testes e defeitos na mesma ferramenta foi a quarta colocada com 8%. Uma possível explicação para a pequena popularidade desta última é o fato de estar a quase dez anos sem receber atualizações.

O *Google Trends* novamente foi usado para tentar confirmar tais resultados obtidos pela revista e verificar se houve mudança de popularidade nestes últimos 5 anos. Apenas *TestLink* apresentou um pequeno volume de pesquisas suficiente para gerar um gráfico. Com isso, *TestLink* foi a primeira ferramenta selecionada para ser avaliada por este trabalho. Como as outras ferramentas não tiveram representação nas pesquisas do *Google Trends*, passou-se a considerar apenas os resultados apresentados pela revista. Esta apontou que a segunda ferramenta mais popular foi *Testopia*, um plugin que pode ser instalado ao *Bugzilla* para adicionar funcionalidades de gestão de testes. Mesmo sendo um plugin e não uma ferramenta independente, foi a segunda escolhida para ser avaliada por este trabalho, por ser um complemento do *Bugzilla* que é uma das ferramentas de gestão de defeitos mais utilizadas. A terceira ferramenta citada na

pesquisa foi *Redmine*, mas não foi escolhida para ser avaliada por ser uma ferramenta para gestão de projetos e não possuir funcionalidades específicas para gestão de testes. A próxima ferramenta citada foi *RTH*. Na sequência, *XQual Studio* possui uma versão gratuita, mas limitada a dez usuários, mil testes e poucos recursos. Como as ferramentas restantes não tiveram expressividade nas pesquisas, buscou-se a próxima ferramenta que apresentasse mais recursos e boas práticas de usabilidade baseadas nos 10 princípios heurísticos de Nielsen (1993) e outras apresentadas por Krug (2006). Portanto, a ferramenta *Testia Tarantula* foi escolhida para ser a terceira ferramenta a ser avaliada.

A Tabela 3 apresenta o comparativo dos requisitos identificados e avaliados nas três ferramentas de gestão de defeitos e nas três ferramentas de gestão de testes utilizando as 10 heurísticas de Nielsen. A sigla AT significa atende totalmente, AP atende parcialmente e NA não atende. A coluna "*Testicida*" representa a ferramenta desenvolvida com a indicação dos requisitos já implementados (AT), os que ainda estão em desenvolvimento (ED) e os que não serão implementados (NA).

Tabela 3. Avaliação dos requisitos identificados nas ferramentas selecionadas.

Requisitos	TestLink	Testia Tarântula	Testopia	Bugzilla	Mantis	BugNET	Testicida
Cadastrar usuário	AT	AT	NA	AT	AT	AT	AT
Definir a disponibilidade de um usuário	AT	NA	NA	AT	AT	AT	AT
Definir o perfil de acesso do usuário	AT	AP	NA	AT	AT	AT	AT
Armazenar operações realizadas por usuário.	AT	NA	NA	AT	AT	AT	AT
Exportar os usuários cadastrados para um arquivo formato .xml	AT	NA	NA	NA	NA	NA	ED
Cadastrar projeto de teste	AT	AT	NA	NA	NA	NA	AT
Definir os papéis dos usuários no projeto de teste	AT	AT	NA	NA	NA	NA	AT
Definir a disponibilidade de um projeto de teste	AT	NA	NA	NA	NA	NA	AT
Definir a visibilidade de um projeto de teste	AT	NA	NA	NA	NA	NA	AT
Clonar projeto de teste	AT	NA	NA	NA	NA	NA	AT
Cadastrar plano de teste	AT	AT	AT	NA	NA	NA	AT
Definir os papéis dos usuários no plano de teste	AT	NA	AT	NA	NA	NA	AT
Definir a disponibilidade de um plano de teste	AT	NA	AT	NA	NA	NA	AT
Definir a visibilidade de um plano de teste	AT	NA	AT	NA	NA	NA	AT
Clonar plano de teste	AT	NA	AT	NA	NA	NA	ED
Cadastrar build	AT	NA	AT	AT	NA	AT	AT
Definir a disponibilidade de um build	AT	NA	AT	AT	NA	AT	AT
Cadastrar marco	AT	NA	AT	AT	NA	AT	AT
Cadastrar suíte de teste	AT	AT	NA	NA	NA	NA	AT
Cadastrar caso de teste	AT	AT	AT	NA	NA	NA	AT
Atribuir suíte de teste a um plano de teste	AT	AT	NA	NA	NA	NA	AT
Atribuir caso de teste a um plano de teste	AT	AT	AT	NA	NA	NA	AT
Atribuir suíte e caso de teste a usuários para execução	AT	AT	AP	NA	NA	NA	AT
Permitir que o usuário insira o resultado do caso de teste como passou, falhou ou bloqueado.	AT	AT	AT	NA	NA	NA	AT
Associar, no caso de teste, um link para o relato de defeito	AT	AT	NA	NA	NA	NA	AT
Importar / exportar suítes de teste	AT	NA	NA	NA	NA	NA	ED
Importar / exportar casos de teste	AT	NA	AT	NA	NA	NA	ED

Mover / copiar casos de teste e suítes de teste	AT	AP	AP	NA	NA	NA	ED
Visualizar histórico de testes executados.	AT	AT	AT	NA	NA	NA	AT
Relacionar caso de teste com caso de teste	AT	NA	AP	NA	NA	NA	AT
Cadastrar requisitos	AT	AT	NA	NA	NA	NA	AT
Relacionar requisito com caso de teste	AT	AT	NA	NA	NA	NA	AT
Relacionar requisito com requisito	AT	NA	NA	NA	NA	NA	AT
Relatórios	AT	AT	AT	AT	AT	AT	AT
Integração com Bug Trackers	AP	AT	AP	NA	NA	NA	NA
Integração com Selenium	AP	NA	NA	NA	NA	NA	NA
Cadastrar projetos para relato de defeitos e sugestões	NA	NA	NA	AT	AT	AT	AT
Cadastrar subprojeto e associar a um projeto de defeitos existente.	NA	NA	NA	AT	AT	AT	AT
Definir projetos de defeitos ativo ou inativo	NA	NA	NA	AT	AT	AT	AT
Definir projetos de defeitos privado ou público	NA	NA	NA	AT	AT	AT	AT
Cadastrar relato de defeito	NA	NA	NA	AT	AT	AT	AT
Clonar um relato de defeito	NA	NA	NA	AT	AT	NA	AT
Monitorar relato de defeito	NA	NA	NA	AT	AT	AT	ED
Criar relação entre relatos de defeitos cadastrados	NA	NA	NA	NA	AT	AT	AT
Alternar formulário de relato de defeito entre básico e avançado	NA	NA	NA	AT	NA	NA	NA
Anexar arquivos a um relato de defeito	NA	NA	NA	AP	AP	AP	AT
Pesquisar relatos de defeitos cadastrados	NA	NA	NA	AT	AT	AT	AT
Adicionar comentários a um relato de defeito	NA	NA	NA	AT	AT	AT	AT
Notificações por e-mail	AP	NA	NA	AT	AT	AT	AT
Fluxo de trabalho para resolução de um caso de defeito (Workflow)	NA	NA	NA	AT	AT	AT	AT
Histórico de mudanças efetuadas no relato	NA	NA	NA	AP	AT	AT	AT
Suporte a autenticação através do protocolo LDAP	AT	NA	NA	AT	AT	AT	ED
Criação de campos personalizados	AT	NA	NA	AT	AT	AT	ED
Validações de preenchimento dos campos	AP	AP	AP	AP	AP	AP	AT
Interface responsiva	NA	NA	NA	NA	NA	NA	AT
Apresenta versão	AT	AP	AP	AT	AP	AT	AT
Suporte à multiplos idiomas	AT	NA	AT	AT	AT	NA	NA
Ajuda e Documentação	AT	AP	AP	AT	AP	AT	ED

A avaliação das ferramentas encontradas, em especial as apresentadas na Tabela 3, que foram avaliadas mais detalhadamente, reforçaram os dois problemas que motivaram a realização deste trabalho: não oferecem a integração de requisitos, gestão dos testes e gestão dos defeitos em uma mesma ferramenta; e a ausência de boas práticas de usabilidade é evidente. Percebeu-se também que o fato de usar duas ferramentas diferentes agrava o problema da usabilidade porque, por exemplo, os projetos, usuários, permissões precisam ser cadastrados nas duas ferramentas. As interfaces, operações, requisitos e regras de negócio das ferramentas são diferentes. Algumas ferramentas de gestão de testes oferecem integração com ferramentas de gestão de defeitos, mas talvez o termo integração esteja empregado de forma equivocada, de tão simples que é, não

elimina a necessidade das duplicidades de cadastros citados, o que eleva a probabilidade de inconsistência entre os dados.

3. A Ferramenta Testicida

Diante dos problemas apresentados nas seções anteriores, a modelagem e desenvolvimento da ferramenta Testicida recebeu uma atenção especial em relação ao banco de dados e a interface de usuário com o intuito de oferecer uma ferramenta realmente integrada e com boa usabilidade.

Começando pelo banco de dados, o módulo principal é o projeto. É a base para os módulos requisitos, testes e defeitos. Desta forma, o módulo projeto concentra e compartilha as mesmas informações pertencentes a ele como, por exemplo, versões de software, categorias, papéis dos usuários com os outros três módulos. Isso garante a consistência nas informações e poupa tempo. Outra vantagem desta ferramenta está na geração de relatórios mais claros e completos, pois poderá relacionar as informações de todos os módulos. Além disso, poderá oferecer relatórios extras como, por exemplo, defeitos sem casos de teste relacionados. Relatórios como esse poderão auxiliar a equipe a tomada de decisão quanto a necessidade de cadastrar um novo caso de teste para este defeito encontrado com a intenção de detectá-lo em um próximo plano de testes a ser executado. Outros exemplos de relatórios podem ser casos de teste que mais geram defeitos e os que menos geram ou nada geram. Isso pode apontar casos de teste que precisam ser revisados e melhorados, caso seja necessário.

A preocupação com a modelagem do banco de dados pode ser notada na interface de usuário. A ferramenta oferece flexibilidade na criação de um projeto ao permitir a escolha dos módulos que serão habilitados. Claro que a proposta da ferramenta de unir requisitos, testes e defeitos é de oferecer uma solução integrada e com boa usabilidade e com isso ampliar a possibilidade das empresas documentarem de forma mais adequada o processo de teste, porém essa configuração personalizada atenderá também aos usuários que necessitam somente do módulo requisitos ou testes ou defeitos. Podem existir situações temporárias tal qual uma empresa que já utiliza um software de gestão de defeitos e pretende fazer a transição para a Testicida em um período de tempo planejado. Neste cenário hipotético, a ferramenta Testicida poderá servir inicialmente para a gestão dos casos de teste e ter seu módulo defeitos habilitado posteriormente até que os colaboradores estejam treinados, familiarizados e aptos a utilizarem apenas a ferramenta Testicida.

Com relação à usabilidade, autores como Nielsen e Loranger (2007), Benyon (2011) a definem como sendo um atributo de qualidade relacionado à facilidade de uso e à rapidez com que os usuários podem aprender a usar alguma coisa, a eficiência deles ao usá-la, a facilidade de memorização, seu grau de propensão a erros, quanto se sentem seguros ao utilizar e o quanto gostam de utilizá-la. Para Krug (2006) a primeira lei de usabilidade é: “Não me faça pensar”. Dentro do que é humanamente possível, o usuário deve olhar para uma página Web autoexplicativa e óbvia. Deve ser capaz de entender o seu propósito e como usá-la sem gastar qualquer esforço em pensar nisso. O autor ainda acredita que a maioria dos usuários de páginas Web não leem seu conteúdo e sim as exploram visualmente. Portanto, o ideal é criar uma hierarquia visual clara em cada página tirando vantagem das convenções já construídas ao longo da história da Web.

O aspecto visual da interface de usuário da ferramenta foi implementada baseando-se nesses e outros conceitos e técnicas como, por exemplo, as 10 heurísticas de Nielsen. Como resultado obtido, a interface foi implementada apenas com as informações e elementos essenciais, organizados de maneira a facilitar o entendimento visual do usuário. Na barra superior, mais à esquerda, está presente a seleção do projeto.

À sua direita encontram-se as opções relacionadas ao projeto selecionado. O primeiro é a Visão Geral, na qual o usuário poderá visualizar uma página indicando sua relação com tarefas atribuídas a si, ou que esteja envolvido, em cada um dos módulos habilitados. Ao lado da opção Visão Geral, na ordem, estão os módulos Requisitos, Testes e Defeitos. A sequência segue a lógica de um ciclo de testes de software: os requisitos são necessários para criar os testes que provavelmente encontrarão os defeitos. Por último, o menu Relatório dispõe de opções de relatório que relacionam e apresentam as informações geradas ao longo do processo. Acima deste menu, caso o usuário tenha permissões de administrador, será apresentado o menu Administração contendo as opções para gerenciar Usuários, Projetos e Perfis de Acesso. Ao lado, o menu Notificações apresentará notificações importantes como mudanças no estado ou anotações adicionadas em um defeito que esteja envolvido. Ao lado direito, a identificação do usuário autenticado no sistema. A interface visual foi desenvolvida com recursos *Web* atuais, o que possibilita se adaptar automaticamente à resoluções diferentes de tela, incluindo *tablets* e até mesmo celulares com maior resolução. A Figura 1 apresenta capturas de tela da interface da ferramenta Testicida.

Uma versão de demonstração[2] está disponível e pode ser acessada com usuário *demo* e senha *demo12.*

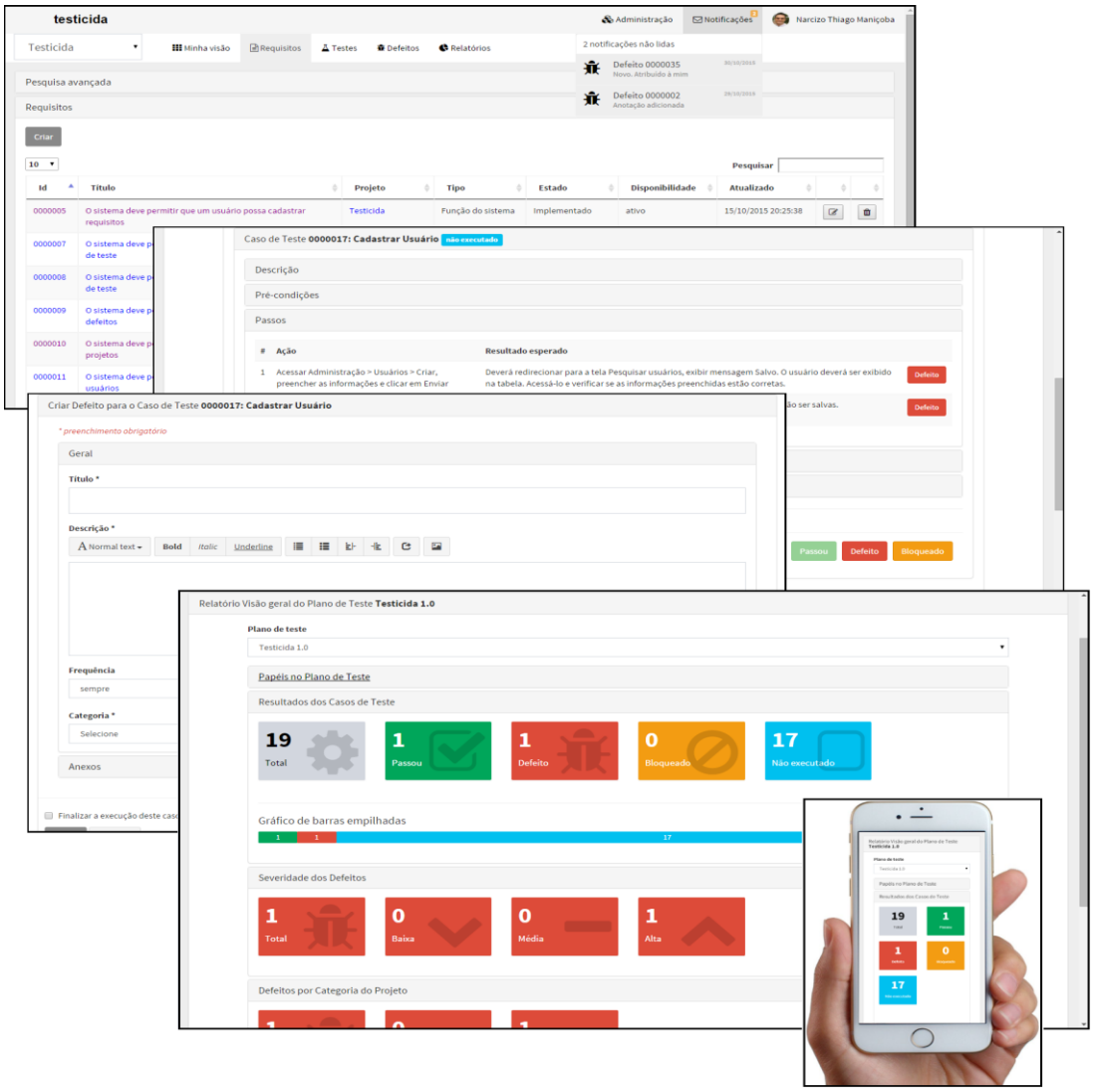

Figura 1. Capturas de tela da interface da ferramenta Testicida.

[2] Demonstração do Testicida: http://188.93.231.189/~nsoftwar/testicida/view/login.php

4. Avaliação

A ferramenta Testicida foi avaliada com base nas 10 heurísticas de Nielsen, da mesma forma que as ferramentas selecionadas. Em seguida, a ferramenta foi avaliada por usuários com o objetivo de obter feedback. O objetivo foi verificar principalmente os requisitos de usabilidade e suas funcionalidades. A avaliação ocorreu no contexto de uma empresa que atua na área de telecomunicações. Foram selecionados 10 colaboradores que atuam em diferentes atividades do ciclo de vida de desenvolvimento, conforme apresentado na Tabela 4.

Tabela 4. Colaboradores selecionados para avaliar a ferramenta Testicida.

Quantidade de colaboradores	**Papel/Cargo**	**Experiência (anos)**
3	Marketing de Produto (MP)	2 (média)
2	Analista e Engenheiro de desenvolvimento (AE)	4 (média)
4	Técnico e Analista de confiabilidade (TA)	3 (média)
1	Gestão de Projetos (GP)	6

A maior parte destes colaboradores já utilizam a ferramenta *Mantis* há pelo menos três anos para gerenciar os defeitos. Um deles, TA, também utiliza *TestLink,* e outro, TA, *Redmine*, mas já trabalhou em outra empresa com *TestLink* e *Bugzilla*. Portanto, a avaliação também foi feita com base nos requisitos oferecidos por estas ferramentas. Algumas delas apresentadas na Tabela 3.

A avaliação iniciou com os colaboradores de MP e com os AE. Como responsáveis por propor, discutir e implementar os requisitos de um produto, estes colaboradores cadastraram três novos projetos, com seus respectivos requisitos: central telefônica *PABX,* roteador *Wireless* e Terminal Inteligente para *PABX*. Na sequência, os colaboradores TA (responsáveis pelos testes) cadastraram os casos de teste a partir dos requisitos cadastrados, criaram planos de teste e executaram os casos. Cada caso foi marcado como "Passou" ou "Defeito". Para aqueles marcados como "Defeito" os usuários cadastraram os relatos de defeito e atribuíram aos responsáveis para correção. Os colaboradores AE acessaram os defeitos atribuídos a eles, alteraram o estado para "Admitido", incluíram anotações, alteraram estado para "Resolvido" e depois atribuíram ao relator do defeito para validar a correção e fechar o relato de defeito. O colaborador de GP envolveu-se na decisão de etapas como criação dos planos de teste e acompanhamento através de relatórios.

A percepção destes usuários foi avaliada a partir de um questionário previamente definido. O objetivo foi verificar suas expectativas em relação a dois aspectos: conclusão da tarefa desejada e nível de dificuldade encontrado para realiza-la.

Em relação ao primeiro aspecto, todos os participantes conseguiram concluir a tarefa desejada. As considerações sobre o segundo aspecto são apresentadas na Tabela 5.

Tabela 5. Considerações dos participantes em relação ao aspecto facilidade de uso.

Item	**Consideração**	**Entrevistados**
1	(+) A inclusão de requisitos foi rápido, sendo simples, objetivo e muito bem organizado.	1 MP, 1 TA
2	(+) A disposição dos campos chamou a atenção pela facilidade que o sistema poderá agregar no dia-a-dia.	1 MP, 1 TA
3	(+) Bem intuitivo, e boa organização na sequência das tarefas.	2 TA
4	(+) A interface mais moderna, limpa e de melhor usabilidade facilita para que o aprendizado de seu uso seja mais rápido, colaborando para que os profissionais usufruam rapidamente de suas *features*.	1 TA
5	(+) Interligando o caso de teste com o defeito ficou muito fácil e ágil de realizar e verificar inconsistências no teste.	1 TA

6	(+) O grande ponto positivo é unir 3 etapas importantes de um projeto em um lugar. Tudo fica fácil de ser encontrado e de ser relacionado para fazer algum tipo de consulta e a possibilidade de realizar rastreabilidade bidirecional (requisitos > testes > bugs, bugs > testes > requisitos).	2 TA
7	(+) O cadastro de defeitos pela ferramenta durante a execução dos testes é excelente, pois dá mais agilidade.	1 TA
8	(-) O ponto negativo é que todas as pessoas envolvidas nestas 3 etapas tem que fazer o seu papel no sistema, pois todas as informações são interligadas em uma cadeia. Exemplo: se alguém não escrever o requisito já fica complicado para a área de testes escrever os casos de testes.	1 TA
9	(-) Senti falta apenas de um mini *help*, para apresentar informações adicionais.	1 MP

Sobre o item 8, embora o usuário tenha classificado como negativo, pode ser considerado como positivo do ponto de vista do processo de teste. A proposta da ferramenta é evidenciar a falta de requisito, teste e/ou defeitos e fazer com que as pessoas se cobrem para fazê-los. Inicialmente, a ferramenta não irá impedir que sejam cadastrados casos de teste sem requisitos relacionados, pois a ferramenta pretende oferecer flexibilidade para atender o nível de maturidade atual de cada organização que queira adota-la.

Quanto ao item 9, vale ressaltar que este recurso já está no *backlog* da ferramenta, mas ainda não estava disponível no momento da avaliação.

5. Conclusões

As considerações feitas pelos usuários, logo após o primeiro contato com a ferramenta Testicida, dão a entender que o objetivo inicial de desenvolver uma ferramenta que integrasse requisitos, gestão de testes e gestão de defeitos, e empregando boas práticas de usabilidade, realmente pode incentivar os colaboradores a aumentar a efetividade e qualidade da documentação destas três importantes etapas presentes nos ciclos de desenvolvimento de software. Portanto, a ferramenta pode oferecer uma contribuição à comunidade de testes de software.

No entanto, apesar de os requisitos essenciais estarem funcionais, a ferramenta encontra-se atualmente em desenvolvimento e precisa de mais testes e correções para que, então, possa ser disponibilizada a primeira versão estável aos usuários.

Tem-se consciência, também, de que a ferramenta precisa de outros requisitos. Alguns já estão sendo especificados e desenvolvidos como, por exemplo, o módulo Requisitos permitir a gestão de requisitos e regras de negócio através da rastreabilidade. Em caso de alteração de um requisito, possa alertar quais casos de teste associados provavelmente precisarão ser revisados. E, também, controlar o versionamento dos requisitos.

Pretende-se, ainda, pesquisar e implementar outros requisitos, como a integração com ferramentas de automatização de testes, controle de versão e bases de serviço de diretório (LDAP e AD); suporte a outros idiomas como Inglês e Espanhol; permitir a criação de campos personalizados.

Até o momento da entrega deste artigo, a ferramenta contempla 7 das 10 heurísticas de Nielsen. A intenção é atender plenamente as 10 e melhorar ainda mais a usabilidade.

Por fim, mais testes e avaliações com usuários, preferencialmente de outras empresas, serão realizados com o intuito de obter mais feedbacks, para que dessa forma a ferramenta possa ser aprimorada.

6. Referências

IEEE Standard 610-1990: IEEE Standard Glossary of Software Engineering Terminology, IEEE Press.

KRUG, Steve. **Don't Make Me Think! A Common Sense Approach to Web Usability.** 2. ed. Berkeley, CA, 2006.

LEWIS, Willian E.; VEERAPILLAI, Gunasekaran. **Software Testing and Continuous Quality Improvement.** 2. ed. Boca Raton: Auerbach Publications, 2004.

MYERS, Glenford J; BADGETT, Tom; SANDLER, Corey. **The Art of Software Testing.** Editora John Wiley & Sons, New Jersey, USA, 2011.

NIELSEN, Jakob. **Usability Engineering.** Editora Academic Press, Boston, 1993.

NIELSEN, Jakob; LORANGER, Hoa. **Usabilidade na Web – Projetando Websites com qualidade**. Rio de Janeiro: Elsevier, 2007.

NIELSEN, Jakob. **Usability 101: Introduction to Usability** 2012. Disponível em: < http://www.nngroup.com/articles/usability-101-introduction-to-usability/>. Acesso em: 01 mar. 2015.

PFLEEGER, S. L. **Engenharia de Software: teoria e prática**. 2. ed. São Paulo: Prentice Hall, 2004.

RILEY, Tim.; GOUCHER, Adam. **Beautiful Testing: Leading Professionals Reveal How They Improve Software (Theory in Practice).** Sebastopol, CA: O'Reilly, 2010.