Test-Module: uma ferramenta para gerenciamento de testes de software integrada ao *FireScrum*

Audrey B. Vasconcelos, Iuri Santos Souza, Ivonei F. da Silva, Keldjan Alves

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851, Cidade Universitária – 50.732-970 – Recife – PE – Brasil

{abv,iss2,ifs3,kao}@cin.ufpe.br

Abstract. *The increasing demand for software development with higher quality increases its competitiveness and turns critical the creation of a software testing process in parallel with the development's, including projects that adopt agile methodologies. In this context, the use of tools that support the management of testing activities is essential to project success. This paper presents an open source tool integrated to the FireScrum – a tool for projects management using Scrum - that allows the management of these activities.*

Resumo. *Com o aumento da competitividade e complexidade no âmbito do desenvolvimento de software, e a conseqüente exigência por produtos com qualidade assegurada, torna-se imprescindível a realização do processo de testes de software em paralelo ao processo de desenvolvimento, inclusive em projetos que adotam metodologias ágeis. Neste contexto, o uso de ferramentas de suporte à gestão das atividades de testes é essencial para o sucesso dos projetos. Este trabalho apresenta uma ferramenta de código aberto integrada ao FireScrum – ferramenta para gestão de projetos ágeis utilizando Scrum – a qual permite gerenciar as principais atividades de testes.*

1. Introdução

É notável a crescente necessidade do uso de produtos de software pelas organizações, sendo muitas vezes essencial à sua sobrevivência. O crescimento da indústria de desenvolvimento de software acarretou um aumento significativo na competitividade entre as organizações provedoras desse serviço. Os usuários de software, consumidores desse serviço, estão cada vez mais exigentes quanto aos atributos de qualidade do software, tais como: desempenho, confiabilidade e usabilidade.

Uma técnica de verificação e validação para certificar artefatos de software, como testes de software, é uma abordagem que contribui para o aumento da qualidade ao longo do processo de desenvolvimento de software [Sommerville 2007]. Todavia, testar software não é uma tarefa trivial, pois envolve pessoas, processos, ferramentas, aplicações, *releases* (conjunto de artefatos), grupos de *scripts* (roteiros de testes), dentre outros. Além disso, o uso de planilhas ou documentos de textos não é eficaz para uma gestão de testes produtiva e com baixo custo.

Na literatura da engenharia de software, observa-se que um projeto típico de desenvolvimento de software tem 30% a 50% do custo total do projeto referente aos

testes [Myers 1979], [Pressman 2006], [Sommerville 2007] e este número ainda pode ser mais alto para sistemas críticos [Harrold 2000].

Dessa forma, para que o custo da atividade de testes não seja um fator limitador para a sua aplicação, usar uma ferramenta adequada para auxiliar na execução dessas atividades otimiza a produtividade.

Neste contexto, ferramentas para gestão de testes em um âmbito de desenvolvimento ágil de software devem auxiliar os desenvolvedores nas tarefas de planejamento e execução dos testes [Beizer 1990]. A partir da execução dos ciclos de testes é possível extrair métricas sobre os casos de testes que obtiveram sucesso ou falha em sua execução. Outras características desejáveis para essas ferramentas de gestão de testes consistem no relacionamento entre casos de testes e requisitos, além do mapeamento e monitoramento dos defeitos detectados.

2. Uma Ferramenta para Gerenciamento de Testes

Esta seção apresenta a ferramenta *Test-Module*, desenvolvida com o objetivo de auxiliar a especificação e gestão das atividades de teste em projetos de desenvolvimento de software que adotam metodologias ágeis, em particular *Scrum*, integrada à ferramenta *FireScrum* [FireScrum 2010]. A *Test-Module* permite criar casos de testes, associá-los aos requisitos, organizá-los em bibliotecas, gerenciar planos para as execuções de testes em projeto de software e gerenciar ciclos de testes que serão executados, além de captar resultados das execuções, coletar métricas, registrar e monitorar defeitos.

2.1. Visão Geral da *Test-Module*

A *Test-Module* possui um fluxo lógico de execução. Primeiramente, é necessário criar a biblioteca de casos de teste (*Test Suite*), que consiste em um agrupamento de casos de testes relacionados. A Figura 1 apresenta a tela de cadastro de bibliotecas, na qual se observa como exemplo a biblioteca "Test Suite Cadastro Evento".

A Figura 2 apresenta a tela de cadastro de casos de teste (*Test Cases)*, na qual se observa o caso de teste "Test Case – Validar Campos Cadastro Eventos". Durante o cadastro de casos de testes, a ferramenta permite associá-los a requisitos do projeto *(Backlog Item)*, além de cadastrar o tipo de execução do teste (manual, automático, semi-automático), a prioridade, o cenário e os passos para execução do teste.

Após criar um conjunto de casos de testes, cria-se o plano de testes para o projeto. A Figura 3 apresenta a tela de cadastro de planos de testes *(Test Plan)*, na qual se observa um plano de testes "Test Plan – Validação Cadastros Sistema de Eventos". Durante o cadastro do plano de testes, o usuário seleciona os casos de testes, além de inserir o objetivo, a estratégia, a agenda e outros dados importantes para um planejamento de testes.

Na última etapa desse fluxo de gerenciamento de testes, o usuário define o ciclo de execução dos testes. A Figura 4 apresenta a tela para a criação de ciclo de testes *(Test Cycle)* para o plano de testes selecionado.

O fluxo lógico apresentado acima segue a seqüencia de agrupamento, criação e planejamento de testes, podendo ser realizado em outra ordem, a critério do usuário.

Além desse, ainda há o fluxo de execução, coleta de resultados e coleta de métricas, também contemplados na ferramenta.

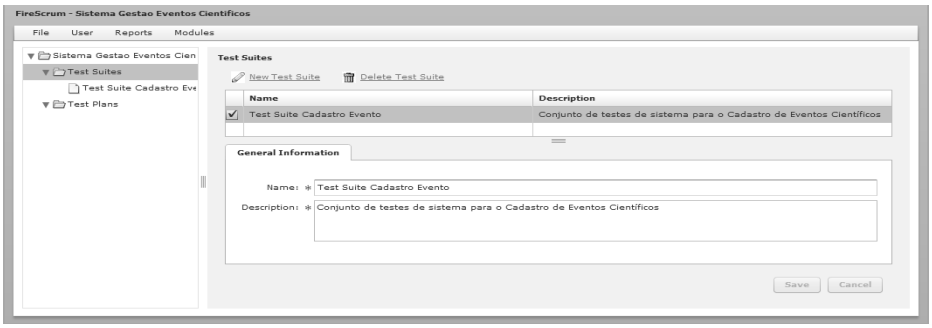

Figura 1. Cadastro de bibliotecas de casos de teste *(Test Suites)*

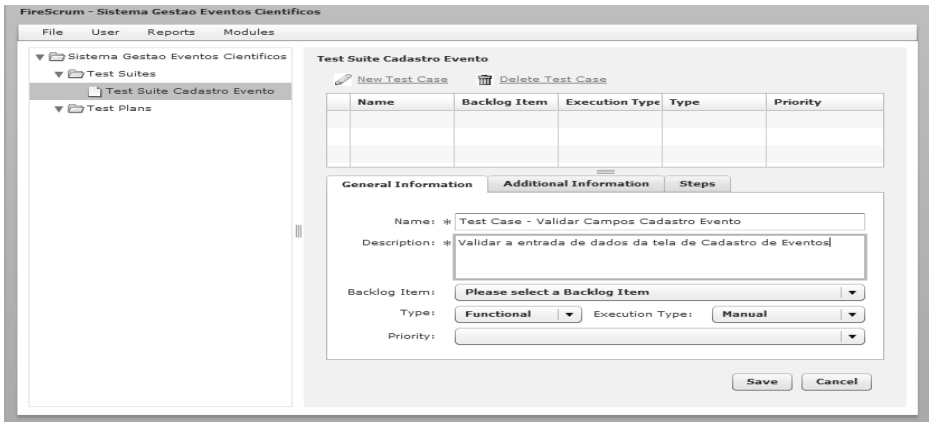

Figura 2. Cadastro de casos de teste *(Test Cases)*

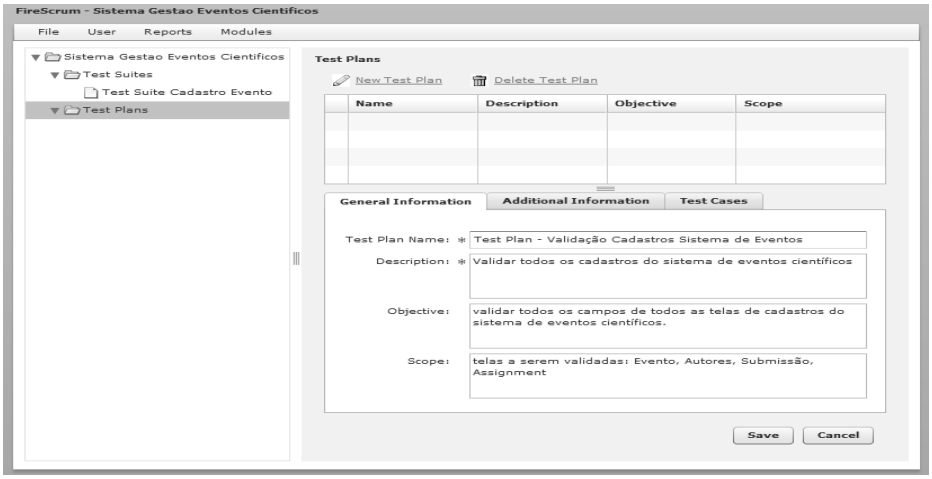

Figura 3. Cadastro de planos de teste *(Test Plans)*

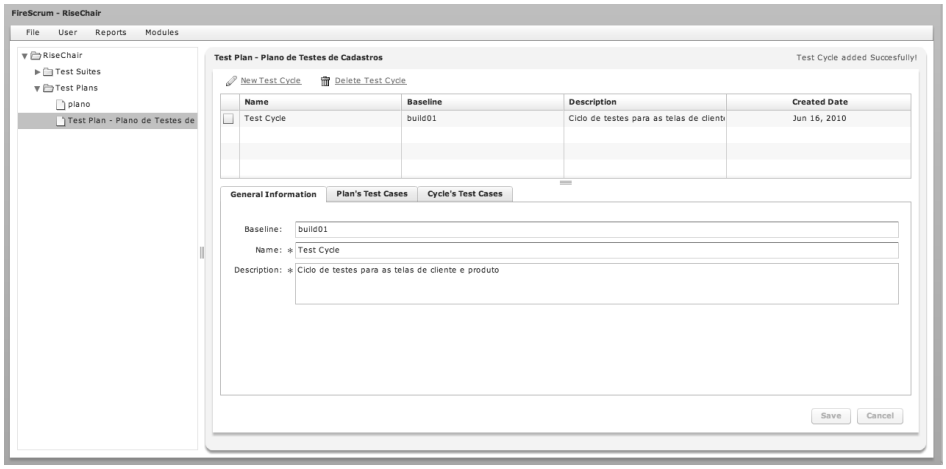

Figura 4. Cadastro de ciclos de execução *(Test Cycles)*

2.2. Arquitetura da *Test-Module*

A arquitetura geral da ferramenta *Test-Module* compreende o uso de Adoble Flex [Adobe Flex 2010] como camada de apresentação (*front-end*) e JAVA [Java 2010] com BlazeDS [BlazeDS 2010] como camada de processamento (*back-end*). A Figura 5 apresenta uma visão simplificada, porém completa, da arquitetura utilizada. A camada de visão foi desenvolvida através de uma interface RIA (*Rich Internet Applications*) [RIA 2010]. A camada de negócios empregou o uso de JAVA com o uso de padrões de projeto. O acesso aos dados é realizado através do *framework* Hibernate [Hibernate 2010], o qual fornece a característica de independência do banco de dados. A seguir, são explanados mais detalhes sobre os elementos presentes na arquitetura.

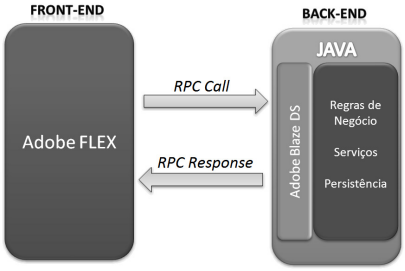

Figura 5. Arquitetura geral da *Test-Module* no *FireScrum*

2.2.1Camada de Apresentação

A camada de apresentação fornece ao usuário experiência de uso otimizado dos componentes e menus do sistema construídos com o Adobe Flex – um *framework* multi-plataforma para desenvolvimento de aplicações RIA. Assim, a aplicação cliente processa algumas informações do sistema no terminal cliente e o processamento complexo fica a cargo do servidor da aplicação no *Back-end*.

2.2.2Camada de Processamento

Esta camada utiliza diversas tecnologias para prover todas as funcionalidades desejadas. A linguagem JAVA foi utilizada como plataforma de desenvolvimento e

controle dos objetos necessários à *Test-Module*. O BlazeDS foi utilizado para conectar os dados do *back-end* com o *front-end* em Adobe Flex, e a persistência dos dados foi realizada através do *framework* Hibernate. Mais detalhes sobre a ferramenta *Test-Module* encontra-se na página do *FireScrum* [FireScrum 2010].

3. Trabalhos Relacionados

Geralmente as ferramentas comerciais para gestão de teste de software limitam-se a oferecer suporte para as principais atividades de teste de software, como criação e gestão de planos, casos e roteiros de testes, além do registro dos resultados e elaboração de relatórios [Beizer 1990]. Um exemplo é o software *HP TestDirector*, produzido e comercializado pela *Hewlett-Packard* [HP 2010], que adicionalmente tem suporte a gerenciamento de defeitos. Cobrindo todas as etapas de testes definidas pelo processo unificado da *Rational*, há o software fabricado e comercializado pela *IBM Rational*: o *Rational TestManager* [Rational 2010]. Oferecendo, além das principais atividades de teste, suporte ao desenvolvimento distribuído com gestão ágil de projetos de software, a *Borland* produziu e comercializa a *SilkCentral Test Manager* [Borland 2010].

Dentre as ferramentas de código aberto com licença GPL destaca-se a *QaTraq* [Traq Software 2010a], desenvolvido pela *Traq Software*, que atualmente também oferece versão comercial [Traq Software 2010b]. A *QaTraq* oferece suporte para as principais atividades de testes de software, além de modelos de relatórios para resultados de execuções de teste. Outra ferramenta popular de gestão das atividades de testes de software é a *TestLink* [TestLink 2010], que foi desenvolvida colaborativamente por uma comunidade composta por engenheiros de testes. Esta ferramenta oferece recursos adicionais, tais como priorização e atribuição de tarefas e suporte para integração com ferramentas de gerenciamento de defeitos.

A *FireScrum Test-Module* destaca-se, em relação a essas ferramentas, por centralizar em uma única aplicação as seguintes características: possui código aberto; está integrada a uma ferramenta de gestão ágil de processo de software; organiza os casos de testes em bibliotecas autônomas, facilitando o reuso destes; gestão dos ciclos execução; interface rica na camada de apresentação web; possui integração direta com ferramenta de gerenciamento de defeitos; e ainda possibilita o monitoramento dos passos que não alcançarem os resultados esperados durante a execução de um caso de teste.

4. Conclusões e Trabalhos Futuros

Neste trabalho foi apresentada uma ferramenta que auxilia os desenvolvedores de software, em contexto ágil, a fazer gestão de testes do projeto de software. A ferramenta *Test-Module* permite aos usuários definir planos e bibliotecas de testes, casos de testes, ciclos de execução e associá-los aos requisitos. Além disso, a ferramenta permite a integração com ferramenta de gerenciamento de defeitos rastreando a execução ao defeito identificado.

Para trabalhos futuros, a ferramenta deverá evoluir para atender a necessidade de relatórios mais elaborados, ampliar a coleta de métricas e permitir a integração com outras ferramentas de gerenciamento de defeitos consolidadas na comunidade, a exemplo do *Bugzilla* [Bugzilla 2010] *e* do *Mantis* [Mantis 2010]. Por ser um ferramenta

de código aberto, e originalmente idealizada dentro do projeto *FireScrum*, a *Test-Module* também terá uma evolução associada ao progresso das necessidades oriundas da comunidade de software livre que estiver interessada em gestão ágil de projetos e gestão de testes.

Referências

Adobe Flex. (2010). http://www.adobe.com/products/flex/, Maio.

Beizer, B. (1990). Software Testing Techniques, John Wiley & Sons, Inc., New York, NY, USA.

BlazeDS. (2010). http://opensource.adobe.com/wiki/display/blazeds/BlazeDS/, Maio.

Borland. (2010). "SilkCentral Test Manager", http://www.borland.com/us/products/silk/silkcentral_test/index.html, Maio.

Bugzilla. (2010). http://www.bugzilla.org/, Maio.

FireScrum. (2010). "FireScrum ...the open source Scrum Tool", http://www.firescrum.com/, Maio.

GPL. (2010). http://www.gnu.org/licenses/gpl.html, Maio.

Harrold, M. J. (2000). "Testing: A Roadmap." Em *International Conference on Software Engineering*.

Hibernate. (2010). http://www.hibernate.org/, Maio.

HP. (2010). "HP Quality Center", https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24_4000_100__, Maio.

Java. (2010). http://www.java.com/en/download/faq/whatis_java.xml, Maio.

Mantis. (2010). http://www.mantisbt.org/, Maio.

Myers, G. J. (1979). Art of Software Testing, John Wiley & Sons, Inc.

Pressman, R. S. (2006). Engenharia de Software, McGraw Hill, 6ª Edição.

Rational. (2010). "Rational TestManager", http://www-01.ibm.com/software/awdtools/test/manager/, Maio.

RIA. (2010). http://en.wikipedia.org/wiki/Rich_Internet_application, Maio.

Sommerville, I. (2007). Engenharia de Software, Addison Wesley, 8ª Edição.

TestLink. (2010). http://testlink.sourceforge.net/docs/testLink.php, Maio.

Traq Software. (2010a). "QaTraq - Open Source", http://sourceforge.net/projects/qatraq/, Maio.

Traq Software. (2010b). "QaTraq Software", http://www.testmanagement.com/, Maio.