

Héstia – Ferramenta de Apoio a Teste de Software com Base em Casos de Uso

Eduardo Pletsch Manini¹, Guilherme Silva Lacerda²

^{1,2}Centro Universitário Ritter dos Reis – 90.840-440 – Porto Alegre – RS – Brasil
edumanini@gmail.com, guilherme_lacerda@uniritter.edu.br

Abstract. *There is a growing awareness in the scenario of software development, the quality of their products. One way to control the quality of software is testing. This work deals with the use of a use case for a requirements of software specification, which will be used later on software test. To this will be shown the creation and the use of a managing test plans system, which will use the model of defined requirements.*

Resumo. *Observa-se uma crescente conscientização, no cenário atual de desenvolvimento de software, pela qualidade de seus produtos. Uma das maneiras de controlar a qualidade de software é por meio de testes. Este trabalho trata da utilização de um modelo de caso de uso para especificação de requisitos de software, os quais serão utilizados posteriormente em atividades de teste de software. Para isso, é apresentada a criação e utilização de um sistema para gerenciamento de planos de testes, em que será utilizado o modelo de requisitos definido.*

1. Introdução

O presente trabalho contempla o desenvolvimento de um sistema de informação para apoiar o processo de teste de software.

A principal motivação para elaboração desse trabalho foi a percepção do grande tempo gasto na construção de planos de teste, no processo de desenvolvimento de software da empresa onde trabalho. Além disso, muitas vezes, na criação de um plano de teste não são incluídos casos de teste que garantam que antigos requisitos continuem válidos, correndo-se o risco de perdas de funcionalidades a cada alteração efetuada em um software.

Como principal objetivo desse trabalho, se destaca o desenvolvimento de um sistema de gerenciamento de planos de teste. A partir de requisitos de softwares que serão testados, o sistema apoiará um analista de teste a criar um plano de teste de uma forma rápida e prática. O sistema gerenciará a execução desses testes por um testador. O sistema também servirá como uma base de dados que manterá um histórico dos testes efetuados nos sistemas. Também poderá ser usado como local para armazenamento de requisitos.

Na seção seguinte é apresentado um referencial teórico. Após, é exibido um comparativo de algumas ferramentas similares à ferramenta proposta. Em seguida há uma seção que traz a solução implementada. Após, são apresentados os resultados obtidos com a utilização da ferramenta e, por fim, são expostas as conclusões obtidas.

2. Referencial Teórico

Essa seção apresenta os principais conceitos utilizados nesse trabalho, embasados em obras de importantes autores.

2.1. Requisitos de Software

Requisitos de software são descrições de serviços que serão fornecidos pelo software e suas restrições operacionais. Eles refletem necessidades dos clientes para auxiliar a resolver algum problema [Sommerville 2007]. Os requisitos especificados servirão de base para todas as atividades da engenharia de software que se seguirão [Pressman 1995].

Os requisitos são classificados entre os seguintes tipos: requisitos funcionais, requisitos não-funcionais e requisitos de domínio. Requisitos funcionais descrevem o que o sistema deve fazer e funções detalhadamente, com suas entradas, saídas e exceções. Já os requisitos não-funcionais são requisitos que não estão diretamente relacionados a funções específicas do sistema, e sim a propriedades gerais do sistema, como confiabilidade, tempo de resposta, e espaço em disco. Requisitos de domínio são definidos como requisitos ligados diretamente ao domínio da aplicação e não em necessidades do usuário ou em função de propriedades do sistema [Sommerville 2007].

Validação dos requisitos é um subprocesso da engenharia de requisitos que se dedica a mostrar que os requisitos realmente definem o sistema que o usuário deseja, e, também ajuda a descobrir problemas com os requisitos. A facilidade de verificação deve ser validada; deve ser possível escrever um conjunto de testes para demonstrar que o sistema atende a cada requisito especificado. *“Se um teste for difícil ou impossível de ser projetado, isso significa geralmente que os requisitos serão difíceis de serem implementados e devem ser reconsiderados.”* [Sommerville 2007].

Para isso, os requisitos devem abranger os critérios de validação, que servirão para demonstrar o entendimento, viabilizar a implementação e servir de base para os testes. É fundamental que sejam dedicados tempo e atenção a eles [Pressman 1995].

Mudanças são um fato para grandes sistemas de software. À medida que um sistema cresce a complexidade de analisá-lo também cresce. Cada novo requisito adicionado ou alterado pode afetar aos demais requisitos. É preciso avaliar o impacto nos requisitos afetados pelas mudanças [Pressman 1995].

O gerenciamento de requisitos deve manter ligações entre os requisitos dependentes ou relacionados, stakeholders, módulos do sistema ou objetivos. Dessa forma será possível analisar o impacto de mudanças de requisito [Sommerville 2007].

2.2. Modelo de Casos de Uso

A idéia de Modelos de Casos de Uso foi introduzida em 1986, por Ivar Jacobson, um dos principais contribuintes da UML (*Unified Modelling Language*). A idéia de Jacobson foi amplamente aceita em virtude da simplicidade e utilidade.

Casos de uso são coleções de cenários relacionados de sucesso e fracasso, que descrevem atores usando um sistema como meio para atingir um objetivo [Larman 2007].

Deve-se observar que casos de uso não são diagramas. Eles são modelados basicamente redigindo textos. Entretanto, a UML define um diagrama para representar casos de uso.

Os casos de uso são escritos em diferentes tipos de formalidade (resumido, informal e completo), dependendo da necessidade, mas é no modo completo que todos os passos e variantes são descritos, com pré-condições e garantias de sucesso. O formato completo de duas colunas enfatiza a ocorrência de uma interação entre os atores e o sistema e é ilustrado no Quadro 1 [Larman 2007].

Quadro 1. Caso de Uso Processar Venda

Ator Principal: Caixa	
Interessados e Interesses: <ul style="list-style-type: none"> Caixa: deseja entrada rápida, precisa e sem erros, de pagamento, pois a falta de dinheiro na gaveta do caixa será deduzida do seu salário. Vendedor: deseja comissões sobre vendas atualizadas. Cliente: deseja comprar, receber um serviço rápido e com o mínimo esforço. Deseja um comprovante de compra, necessário no caso de devoluções de mercadorias. 	
Pré-Condições: <ul style="list-style-type: none"> O Caixa é identificado e autenticado. 	
Garantia de Sucesso (Pós-Condições): <ul style="list-style-type: none"> A Venda é salva. As Comissões são atualizadas. O Recibo é gerado. 	
Cenário de Sucesso Principal (ou Fluxo Básico):	
Ação do Ator	Responsabilidade do Sistema
1. O Cliente chega a um posto de pagamento com bens para comprar.	
2. O Caixa começa uma nova venda.	
3. O Caixa digita um identificador do item.	4. Registra cada linha de item de venda e exibe a descrição do item e o total parcial corrente.
O Caixa repete os passos 3 e 4 até que indique ter terminado.	5. O Sistema apresenta o total com impostos calculados.
6. O Caixa diz ao Cliente o total e solicita o pagamento.	
7. O Cliente paga.	8. Tratar pagamento.
	9. Registra a venda completada e envia as informações para o Sistema externo de Contabilidade (para contabilidade e comissões). Sistema emite recibo.
Extensões (ou Fluxos Alternativos):	
3a. Identificador Inválido:	
	1. O Sistema informa o erro e rejeita a entrada.
7a. Pagamento com dinheiro:	
1. O Caixa digita a quantia de dinheiro fornecida.	2. O Sistema apresenta o valor do troco e libera a gaveta de dinheiro.
3. O Caixa deposita o dinheiro fornecido e entrega o troco para o Cliente.	4. O Sistema registra pagamento em dinheiro.
Requisitos Especiais:	
<ul style="list-style-type: none"> Interface de Usuário por tela sensível ao toque em um monitor de painel plano grande. O texto deve ser visível a uma distância de 1 metro. 	

O cenário de sucesso principal (ou fluxo básico) descreve o caminho de sucesso que satisfaz os interessados. Os passos registram as interações entre os atores e o sistema, validações e mudanças de estado dos elementos.

As extensões (ou fluxos alternativos) descrevem instruções condicionais ou desvios do cenário de sucesso. Uma extensão é composta por uma condição e um tratamento. A condição deve ser algo que possa ser detectado pelo sistema ou ator. O tratamento é uma sequência de passos [Larman 2007].

Quando se usa modelos de casos de uso na fase de especificação de requisitos, deve-se focar nos objetivos e na essência dos requisitos e deixar de lado questões que poderão interferir na implementação e funcionamentos internos do software. Os casos de uso devem ser expressos em níveis de intenções e responsabilidades, independente de tecnologias, especialmente relacionadas à interface com o usuário. Deve-se especificar o que o sistema deve fazer, e não como deve ser feito [Larman 2007].

A documentação dos casos de uso são excelentes descrições de testes funcionais, pois já contém um levantamento de todas as condições que podem causar falhas e os respectivos tratamentos. Pelo menos um caso de teste é necessário para cada extensão [Cockburn 2005]. Os casos de uso também podem ser usados como base de casos de testes e em testes de regressão [Booch, Rumbaugh e Jacobson 2005].

2.3. Qualidade de Software

Qualidade de software dedica-se a assegurar que o software tem baixo número de defeitos e atinge os padrões estabelecidos.

O gerenciamento de qualidade deve utilizar métodos formalizados quando se trata de desenvolvimento de sistemas grandes e complexos. Um método formal ajuda a evitar erros em tarefas comuns e permitem um melhor entendimento e acompanhamento do processo. Ferramentas podem apoiar a utilização de padrões, simplificando o trabalho tedioso para algumas pessoas [Sommerville 2007].

A qualidade nunca é medida literalmente, o que é medido são manifestações que refletem na qualidade. Isso significa que o software deve atingir um determinado patamar referente a alguns atributos para ser considerado de alta qualidade [Pressman 1995]. Sommerville cita um exemplo muito pertinente a esse trabalho:

“Por exemplo, digamos que uma organização planeja introduzir uma nova ferramenta de teste de software. Antes de introduzir a ferramenta, você registra o número de defeitos de software descobertos em um dado intervalo de tempo; depois da introdução da ferramenta, você repete esse processo. Se mais defeitos forem descobertos no mesmo intervalo de tempo, depois da introdução da ferramenta, você pode concluir que ela fornece apoio útil ao processo de validação de software.” [Sommerville 2007].

2.4. Teste de Software

Existem diferentes definições para teste de software na literatura. Uma delas é de que teste de software é uma técnica dinâmica usada para certificar que o software está de acordo com suas especificações [Sommerville 2007]. Para Myers [1979, apud Kosciński e Soares 2007] teste de software é o processo de executar um programa com

o objetivo de encontrar erros. A definição da IEEE Standard 729 [1983, apud Koscianski e Soares 2007] é de que o teste de software é um processo que verifica divergências entre os resultados obtidos e esperados e se o sistema satisfaz os requisitos especificados.

A atividade de teste de software faz parte de um tema mais amplo, freqüentemente chamado de verificação e validação (V&V), que abrange muitas das atividades da garantia de qualidade de software. Verificação visa garantir que o software foi construído conforme especificado. Validação refere-se à garantia que os requisitos especificados refletem às exigências do cliente. Em conjunto, as atividades de verificação e validação de software visam estabelecer a confiança de que o sistema está de acordo com seu propósito [Pressman 1995]. Os testes sempre serão a principal técnica de verificação e validação de software, por isso necessitam de um bom planejamento [Sommerville 2007].

Como os testes de software consomem boa parte do esforço de um projeto, as ferramentas que apóiam essas atividades são muito valiosas e contribuem muito para melhorar a confiabilidade do software desenvolvido [Pressman 1995].

Quando a atividade de teste é realizada de maneira criteriosa e embasada tecnicamente, pode-se garantir, com certa confiança, que o software é bom o bastante para o uso operacional. Entretanto, os testes não servem para provar que o software está totalmente correto [Delamaro, Maldonado e Jino 2007]. Dijkstra cita: “*Os testes podem somente mostrar a presença de erros, não sua ausência.*” [1972, apud Sommerville 2007].

Uma das metas que devem ser usadas na geração de testes é demonstrar que o software atende aos requisitos; outra meta é de tentar descobrir falhas ou defeitos que gerem comportamento incorreto ou não desejável, como travamentos, cálculos incorretos, corrompimento de dados. Os testes devem ser projetados para expor possíveis defeitos, ou seja, tentar ‘quebrar’ o sistema [Sommerville 2007].

Nos testes funcionais, o sistema é tratado como uma caixa-preta, isto é, o comportamento pode ser avaliado somente pelo estudo das suas entradas e saídas, sem se preocupar com o funcionamento interno. Esses testes são derivados das especificações do sistema, em que os critérios de validação (definidos durante a análise de requisitos) devem ser testados. O testador concentra-se somente na funcionalidade, e não na implementação [Pressman 1995]. O principal objetivo dos testes de caixa-preta é garantir que todos os requisitos ou comportamentos da aplicação estejam corretos. Esses testes devem ser aplicados durante as últimas etapas do ciclo de desenvolvimento do software [Molinari 2003].

Grandes ameaças à qualidade do software são provenientes de mudanças, pois qualquer mudança efetuada em um software tem potencial para introduzir erros. [Pressman 1995]. Testes de regressão servem para verificar se mudanças ou correções feitas no sistema não introduziram novos defeitos. Essa é uma situação que ocorre com freqüência na engenharia de software. Testes de regressão também são necessários ao integrar um novo incremento ou módulo a um sistema.

Teoricamente, todos os testes devem ser executados novamente após uma alteração no sistema, mas na prática isso é muito dispendioso. Para contornar isso, a

elaboração dos testes de regressão pode envolver a rastreabilidade entre o requisito alterado e os requisitos relacionados, e, a partir desses requisitos gerar um subconjunto de casos de teste. [Sommerville 2007].

3. Estado da Arte

Nesta seção são apresentadas algumas ferramentas de gerenciamento de teste de software. São descritas as principais características de cada uma, e no final é traçado um comparativo entre as ferramentas apresentadas e o sistema proposto.

3.1. Testlink

Testlink é uma ferramenta de gerenciamento de casos de testes e de suas execuções. Trata-se de uma aplicação web desenvolvida em PHP (*Hypertext Preprocessor*) e distribuída sob licença Open Source.

A ferramenta permite a criação de projetos, cadastro de casos de teste e atribuições de casos de teste a testadores. Os casos de teste podem ser organizados em estruturas hierárquicas, de acordo com a preferência do usuário. A ferramenta permite também a geração de relatórios gerenciais e estatísticos sobre os testes executados e a integração com outras ferramentas de gerenciamento de *bugs* [Caetano 2007].

3.2. RSI/QA-Teste

RSI/QA-Teste é uma ferramenta desenvolvida e comercializada pela RSI Informática, empresa brasileira especializada em prover serviços nas áreas de tecnologias focados na qualidade de processos.

A ferramenta é usada na gestão e arquitetura de testes apoiando a análise, concepção, planejamento, execução, validação, controle e documentação de atividades executadas para testar sistemas. As principais características são o controle de planos, *scripts* e casos de testes, geração e controle de dados de massa de testes, integração com ferramentas de execução automática de testes, criação e controle de cronogramas, criação e exportação de dados para diagramas ou arquivos [Caetano 2007].

3.3. Conclusões e Comparativo

Apesar de as ferramentas Testlink e RSI/QA-Teste serem muito similares ao sistema proposto, nenhuma delas dispõe de um mecanismo que mapeie e gerencie automaticamente casos de uso como casos de teste. Esse é um diferencial que agrega um grande ganho de tempo na elaboração de planos de teste, pois não é preciso ‘escrever’ os casos de teste, o sistema Héstia automaticamente mapeará os casos de uso como casos de teste, abrangendo todas as condições de falha ou desvios.

Outros diferenciais do sistema proposto em relação às ferramentas apresentadas são as forma de alertas para seus usuários sobre suas tarefas pendentes. Primeiramente os alertas são enviados através de mensagens de e-mail. Acessando o sistema, logo na tela inicial o usuário também será alertado sobre suas tarefas, através de um quadro de tarefas pendentes, sem a necessidade de o usuário acessar um item de *menu* para visualizar todas suas tarefas.

O Quadro 2 apresenta um comparativo entre os sistemas apresentados e o proposto, em que foram analisadas as principais características de cada um.

Característica	Testlink	RSI/QA-Teste	Héstia
Mapeamento automático de casos de uso para casos de teste			X
Controle e perfis de acesso	X	X	X
Geração de relatórios	X	X	X
Aviso de tarefas por e-mail	X		X
Quadro de tarefas pendentes			X
Ambiente web	X		X
Licença gratuita	X		X

Quadro 2. Comparativo entre as Ferramentas Apresentadas e o Sistema Proposto

4. Héstia – Ferramenta de Gerenciamento de Planos de Teste

Essa seção traz a solução implementada para a construção da ferramenta. Será apresentada uma descrição do sistema Héstia e também os principais diagramas modelados para o desenvolvimento do sistema.

O sistema foi desenvolvido com a utilização das tecnologias PHP e MySQL. A escolha dessas tecnologias se deu principalmente pela facilidade de utilização e obtenção de documentação e também por todas serem gratuitamente disponibilizadas para uso.

4.2. Descrição do Sistema

Para acessar o sistema Héstia é necessária a autenticação dos usuários por *login* e senha através de um navegador web.

Os administradores são responsáveis por cadastrar os usuários e sistemas a serem testados. Eles também definem os analistas de teste e testadores dos sistemas.

Os usuários analistas de teste cadastram os casos de uso dos sistemas a serem testados. Eles também criam os planos de teste, onde selecionaram os requisitos que devem ser testados e designam um usuário testador para executar o plano de teste.

Por sua vez, os usuários testadores executam os planos de teste com auxílio do sistema e registram os resultados obtidos. O diagrama de caso de uso e o diagrama de workflow, apresentados nas Figuras 1 e 2, respectivamente, ilustram todas essas atividades no sistema.

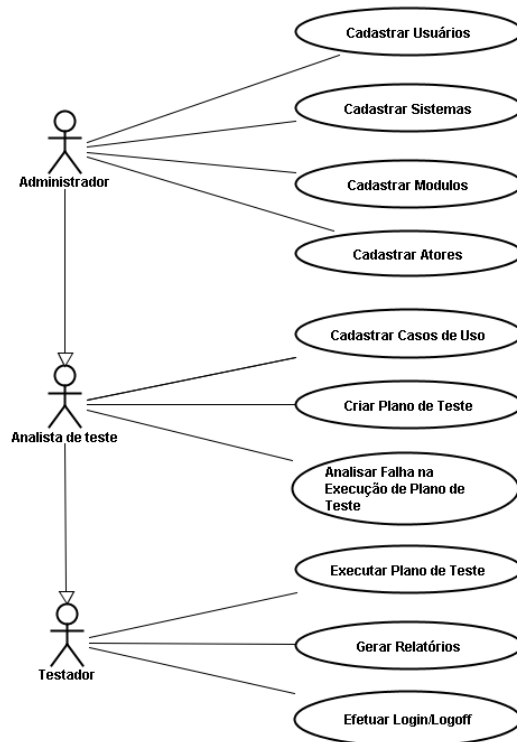


Figura 1. Diagrama de caso de uso

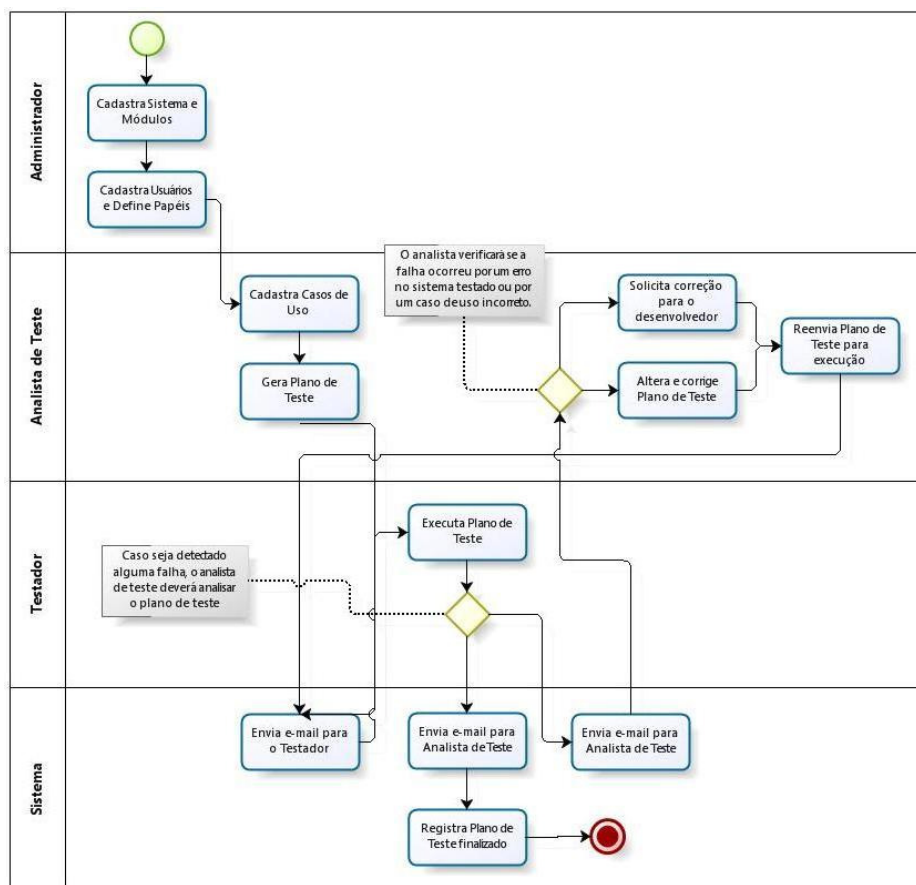


Figura 2. Diagrama de workflow

A tela de cadastro dos casos de uso é ilustrada na Figura 3. É preciso selecionar o sistema, o módulo, um ator do sistema que represente o ator principal do caso de uso e, opcionalmente, atores secundários. Os campos descrição, pré-condições, pós-condições e requisitos especiais são campos de preenchimento opcional. Há a possibilidade de anexar um arquivo ao caso de uso.

Caso de Uso - Alterar

Sistema Zeus
Módulo Alfa
Nome Cadastrar Pessoa
Ator Administrador
Atores Secundários

Descrição
 Processo de cadastro de pessoas no sistema Zeus.













Pré-Condições
 Usuário de perfil administrador autenticado no sistema.
 Usuário administrador de posse dos dados da pessoa que irá cadastrar.

Pós-Condições
 Pessoa cadastrada e ativa no sistema.




Requisitos Especiais




Arquivo
 Não há arquivo
☐ Alterar Arquivo



Cenário Principal

Ator	Sistema	
1. Entrar com o nome da pessoa.		  
	2. Valida nome da pessoa.	  
3. Ativa 'Salvar'.		  
	4. Armazena informações da pessoa.	  

Fluxos Alternativos

2.1 Nome já cadastrado no sistema.   

4.1 Dados obrigatórios não preenchidos.   

Ator	Sistema	
	1. Exibe aviso de campo obrigatório.	 


Retorna ao passo 1. 

Figura 3. Tela de Cadastro de Caso de Uso

No quadro do cenário principal é possível cadastrar os passos do caso de uso. Através dos quatro ícones a direita de cada passo é possível efetuar a edição, inclusão, criação de fluxo alternativo e remoção dos passos. Na edição e inclusão de um passo, será aberta uma janela *pop-up* para seleção do executor (ator ou sistema) e preenchimento da descrição do passo.

Para criar de um fluxo alternativo é necessário preencher a descrição da condição de desvio. Após um fluxo alternativo ser criado, ele será exibido no quadro de fluxos alternativos. Nesse quadro, os fluxos alternativos são identificados pelo número de sequência do passo do cenário principal relacionado e mais um número, ordenado sequencialmente, para possibilitar a identificação de fluxos alternativos de passos do

cenário principal que possuem mais de um fluxo alternativo. Os ícones com sinais de '+' e '-' possibilitam a exibição ou não dos passos de cada fluxo alternativo. É possível a edição, inserção e remoção dos passos dos fluxos alternativos da mesma maneira em que é feito nos passos do cenário principal. O último passo dos fluxos alternativos é um passo diferenciado dos demais, pois só é permitida a sua edição, onde é possível especificar se ao final do fluxo alternativo o cenário é encerrado ou se retorna a algum passo do cenário principal.

Após o cadastro dos casos de uso, os usuários analistas de teste podem iniciar a geração dos planos de teste. A geração de um plano de teste consiste na seleção do sistema que será testado, preenchimento do título, descrição e ambiente de teste do plano de teste, designação de um usuário testador para execução do plano de teste e seleção dos requisitos que devem ser testados. Há a possibilidade de anexar um arquivo ao plano de teste. A tela de geração de plano de teste é ilustrada pela Figura 4.

A interface 'Plano de Teste - Novo' apresenta os seguintes campos e elementos:

- Sistema:** Seleção por menu suspenso com 'Zeus' selecionado.
- Título:** Campo de texto com 'Solicitação de Mudança 310'.
- Descrição:** Campo de texto com 'Testar a mudança efetuada no sistema pela implementação da solicitação de mudança 310.'
- Ambiente de Teste:** Campo de texto com 'https://server004/desenvolvimento/em/zeus/'.
- Testador:** Seleção por menu suspenso com 'Eduardo Pletsch Manini' selecionado.
- Arquivo:** Campo de texto vazio e botão 'Enviar arquivo...'.
- Casos de Teste:** Tabela com as seguintes colunas: 'Módulo' e 'Caso de Teste'.

Módulo	Caso de Teste
Alfa	✗ Cadastrar Pessoa
	Cadastrar Pessoa - Dados obrigatórios não preenchidos.
	Cadastrar Pessoa - Nome já cadastrado no sistema.
Beta	✗ Emitir Boleto
	Emitir Boleto - Pessoa não possui débitos.
	✗ Efetuar Compra
	Efetuar Compra - Crédito não aprovado.
- Botão 'Adicionar Requisito' na base da tabela.
- Botões 'Salvar' e 'Retornar' na base da interface.

Figura 4. Tela de Geração de Plano de Teste

Os requisitos que podem ser selecionados durante a geração de um plano de teste são os casos de uso cadastrados. Cada caso de uso adicionado em um plano de teste se transformará automaticamente em, no mínimo, um caso de teste, que representará o cenário principal do caso de uso, mais um caso de teste para cada fluxo alternativo. Os nomes dos casos de testes gerados a partir de um fluxo alternativo são compostos pelo nome do caso de uso mais a condição de desvio do fluxo principal. Já os nomes dos casos de teste que representam o fluxo principal de um caso de uso são somente o nome do caso de uso.

Após a conclusão da geração do plano de teste o sistema Héstia enviará um e-mail para o testador designado informando sobre o plano de teste que deverá ser executado. Ao acessar o sistema, os usuários testadores visualizarão, na tela inicial, um quadro com suas tarefas pendentes, o qual conterá os planos de testes para os quais eles foram designados para execução. Ao selecionar um plano de teste para ser executado, uma tela de execução do plano de teste, semelhante à Figura 5, será exibida.

Plano de Teste - Execução

Código	77
Sistema	Zeus
Título	Solicitação de Mudança 310
Data de Criação	30/11/2009 23:18
Analista de Teste	Eduardo Administrador
Testador	Eduardo Pletsch Manini
Descrição	Testar a mudança efetuada no sistema pela implementação da solicitação de mudança 310.
Ambiente de Teste	https://server004/desenvolvimento/em/zeus/
Arquivo	Não há arquivo

Casos de Teste	
Módulo	Casos de Teste
Alfa	✓ Cadastrar Pessoa
	✓ Cadastrar Pessoa - Dados obrigatórios não preenchidos.
	Cadastrar Pessoa - Nome já cadastrado no sistema.
	Emitir Boleto
	Emitir Boleto - Pessoa não possui débitos.
Beta	Efetuar Compra
	Efetuar Compra - Crédito não aprovado.

[Retornar](#)

Figura 5. Tela de Execução de Plano de Teste

A tela de execução de plano de teste conterá as informações sobre o plano de teste e também uma espécie de *checklist* com os casos de teste. Nesse *checklist*, os casos de teste que já foram executados terão um sinal verde ao lado esquerdo. Desta maneira, os testadores poderão interromper a execução de um plano de teste para executar alguma outra tarefa ou outro plano de teste de maior prioridade e, após, prosseguirem a execução do plano de teste do ponto em que pararam.

O testador deverá clicar no ícone azul, ao lado direito de cada caso de teste, para iniciar a execução de um caso de teste. Ao clicar uma janela *pop-up* será aberta, semelhante à Figura 6, onde serão apresentadas as informações do caso de teste e também um quadro contendo as ações que devem ser realizadas para o caso de teste ser executado. As ações dos casos de teste gerados a partir somente do fluxo principal de um caso de uso serão exatamente idênticas aos passos do cenário principal do caso de uso. Já, para os casos de teste gerados a partir de um fluxo alternativo, as ações serão exatamente as mesmas que representam os passos do cenário principal até o passo que contém a condição de desvio do cenário principal para o fluxo alternativo; nesse ponto será exibida a condição de desvio e, a partir daí, as ações que se seguirão serão exatamente os passos do fluxo alternativo. Após, caso o último passo do fluxo alternativo seja do tipo que encerre o caso de uso, o caso de teste se finalizará. Caso o último passo do fluxo alternativo seja do tipo que retorne a um passo específico do cenário principal, todos os passos do cenário principal, a partir do passo em que se deve retornar serão exibidos como ações do caso de teste e após, o caso de teste se finalizará.

Ao concluir a execução de todos os casos de teste, o plano de teste será atualizado para o estado finalizado, e o analista de teste criador do plano de teste será informado via e-mail.

Plano de Teste - Execução de Caso de Teste

Código	<input type="text" value="78"/>
Sistema	<input type="text" value="Zeus"/>
Plano de Teste	<input type="text" value="Solicitação de Mudança 310"/>
Caso de Teste	<input type="text" value="Cadastrar Pessoa - Nome já cadastrado no sistema."/>
Ator	<input type="text" value="Administrador"/>
Atores Secundários	<input type="text"/>
Descrição	<input type="text" value="Processo de cadastro de pessoas no sistema Zeus."/>
Pré-Condições	<input type="text" value="Usuário de perfil administrador autenticado no sistema.Usuário administrador de posse dos dados da pessoa que irá cadastrar."/>
Pós-Condições	<input type="text" value="Pessoa cadastrada e ativa no sistema."/>
Requisitos Especiais	<input type="text"/>
Arquivo	<input type="text" value="Não há arquivo"/>

Ações

Ator	Sistema
1. Entrar com o nome da pessoa.	
	2. Valida nome da pessoa.
Nome já cadastrado no sistema.	
	3. Exibe aviso de nome já cadastrado no sistema.
4. Entra com outro nome para a pessoa.	
	5. Valida nome da pessoa.
6. Ativa 'Salvar'.	
	7. Armazena informações da pessoa.

Resultado

Observação

Arquivo Enviar arquivo...

Salvar
Fechar

Figura 6. Tela de Execução de Caso de Teste

Caso, durante a execução de algum caso de teste, seja detectado alguma falha ou comportamento anormal, o resultado da execução do caso de teste deverá ser classificado como falha e o testador deverá descrever a falha no campo observação. Também será possível anexar um arquivo com maiores informações sobre a falha. Ao salvar, a execução do plano de teste será interrompida e um e-mail será enviado automaticamente ao analista de teste criador do plano de teste informando a falha. Assim, o analista de teste deverá acessar o sistema Héstitia e visualizará no quadro de tarefas pendentes o plano de teste que resultou em falha. Será exibido o caso de teste específico em que a falha ocorreu e todas as informações fornecidas pelo testador. O analista de teste deverá analisar a falha e classificar o plano de teste em um dos seguintes estados: ‘Aguardando analista de teste’ ou ‘Aguardando desenvolvedor’, de acordo com a análise feita. Quando a devida providência ter sido tomada, o analista de teste deverá encaminhar o plano de teste novamente para execução.

Os usuários do sistema Héstitia também poderão gerar relatórios dos dados armazenados resultantes das execuções dos planos de testes. Será possível identificar

quais os módulos que apresentam mais falhas, a média de erros detectados ao longo de um período de tempo, as justificativas de falha, entre outras informações.

Nas Figuras 7 e 8 é apresentado um exemplo de relatório. Na Figura 7 o usuário seleciona os campos que deseja que sejam exibidos e configura os filtros do relatório de Planos de Teste. Ao selecionar ‘Gerar’, o relatório de Planos de Teste é apresentado, conforme a Figura 8. Será possível ordenar os campos por um determinado campo apenas clicando no título de cada coluna.

Relatório - Planos de Teste

Campos

☒ Código
 ☒ Data de Criação
 ☒ Estado

☒ Sistema
 ☐ Data de Finalização
 ☐ Ambiente

☒ Título
 ☒ Analista de Teste
 ☐ Arquivo

☐ Descrição
 ☒ Testador

Filtros

Sistema

Estado
☒ Aguardando execução
 ☒ Em execução
 ☒ Falha na execução

☒ Aguardando analista de teste
 ☒ Aguardando desenvolvedor
 ☐ Finalizado

☐ Cancelado

Data de Criação

Maior ou igual que
 / /

Menor ou igual que
 / /

Data de Finalização

/ /

/ /

Analista de Teste

Testador

Figura 7. Tela de Seleção de Campos e Configuração dos Filtros do Relatório de Planos de Teste

Relatório - Planos de Teste

Filtros

Sistema

Estado

Data de Criação

Maior ou igual que 01/11/2009

Menor ou igual que 01/12/2009

Figura 8. Relatório de Planos de Teste

Além dos dados disponibilizados para os usuários através da geração de relatórios, estará também disponível, em todas as partes do sistema onde é apresentado o

código dos planos de teste, um *link* que abrirá uma janela com os dados do plano de teste e um histórico de execuções do plano de teste, conforme ilustra a Figura 9.

Histórico			
Testador: Jorge Silva Antunes		Data de Término: 01/12/2009 13:49	
Data de Início: 01/12/2009 13:33		Estado: Finalizado	Resultado: Falha
Caso de Uso Falha: Cadastrar Usuário			
Observação do Testador: Os caracteres especiais utilizados no cadastro do usuário não foram armazenados corretamente.			
Observação do Analista de Teste: Falha detectada como bug do sistema. Falha comunicado ao desenvolvedor para realizar a correção.			
Testador: Paulo Farias Frota		Data de Término: 02/12/2009 14:46	
Data de Início: 02/12/2009 11:02		Estado: Finalizado	Resultado: Falha
Caso de Uso Falha: Cadastrar Usuário - E-mail inválido.			
Observação do Testador: O cadastro do e-mail 'eduardomanini.com.br' foi aceito.			
Observação do Analista de Teste: Comunicado ao desenvolvedor responsável para atualizar a regra de validação de e-mail para não aceitar e-mails sem '@'.			
Testador: Paulo Farias Frota		Data de Término: 04/12/2009 10:08	
Data de Início: 02/12/2009 18:37		Estado: Finalizado	Resultado: Sucesso

Figura 9. Histórico de Execuções de um Plano de Teste

4.1. Modelagem

Todo o sistema foi modelado e desenvolvido utilizando orientação a objetos com a linguagem de programação PHP 5. O diagrama de classes das classes de negócio é esquematizado na Figura 10.

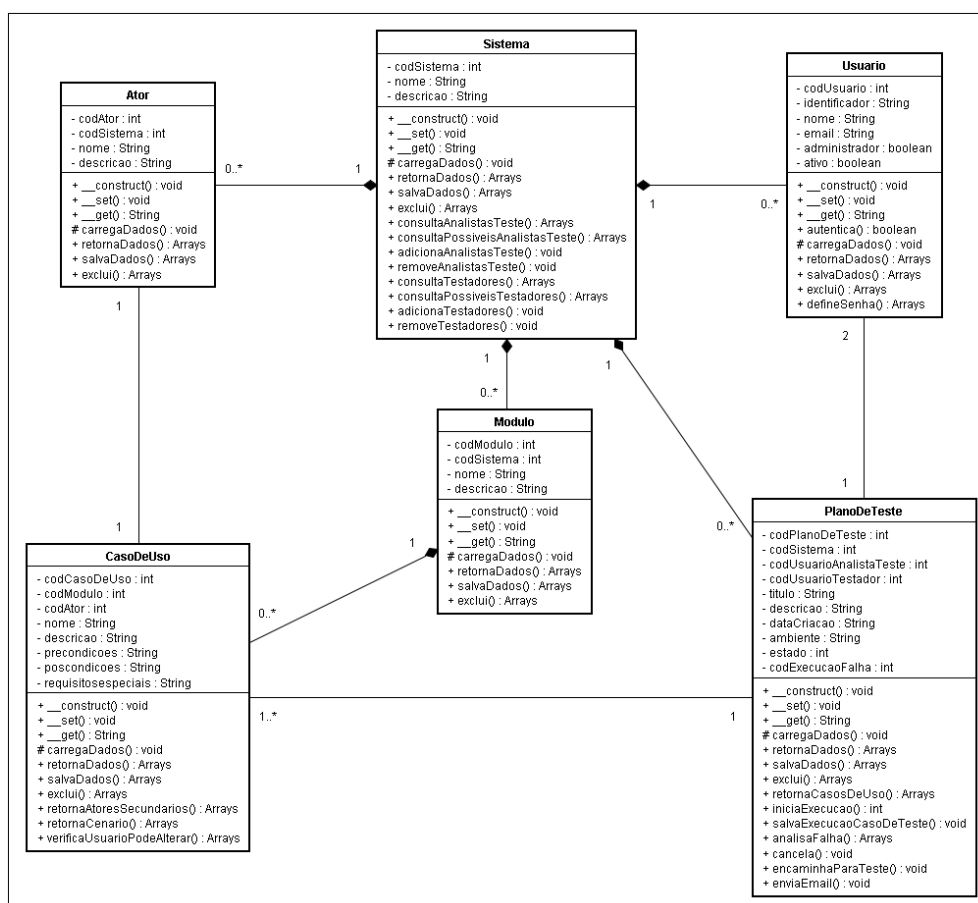


Figura 10. Diagrama de Classes

desse trabalho, que era o ganho de tempo e agilidade na elaboração dos planos de teste, foi atingido.

No entanto, a especificação e cadastro dos requisitos em forma de caso de uso consumiram um tempo maior que o esperado. Verificou-se que isso se deu por dois principais fatores: 1) o sistema Héstita não havia sido especificado no formato de casos de uso no início desse trabalho, 2) a especificação e cadastro dos requisitos em formato de caso de uso não seguiram, de forma gradual, o desenvolvimento do sistema, isto é, todos os requisitos foram especificados no modelo de caso de uso e cadastrados no sistema ao final da implementação do mesmo, em uma única etapa. Sendo assim, sugere-se para utilizações futuras, a especificação e cadastro dos requisitos dos sistemas a serem testados de forma gradual, acompanhando o desenvolvimento do sistema. Apesar dessa sugestão não reduzir o tempo gasto em um projeto, ela visa evitar um trabalho repetitivo e muitas vezes cansativo em uma mesma etapa.

Quanto à questão da execução dos planos de teste, o sistema também atingiu seus objetivos. Verificou-se que todas as condições especificadas foram aplicadas nos casos de testes gerados pelo sistema. Dado ao tamanho do sistema, pode-se dizer que uma quantidade razoável de erros foi detectada na execução dos testes. Esses erros foram corrigidos e os testes executados novamente, até que todas as execuções resultassem em sucesso. Isso comprova que o sistema visa a garantia de preservação dos requisitos e conseqüentemente uma preservação da qualidade dos sistemas testados.

É importante ressaltar que os casos de testes gerados pelo sistema Héstita são apenas uma espécie de guia passo-a-passo para o testador seguir, a aplicação dos mesmos no sistema que se está testando é de inteira responsabilidade do testador. O sistema Héstita armazenará os registros que comprovarão que o testador confirmou a execução de cada caso de teste.

6. Considerações Finais

Ao longo do desenvolvimento desse trabalho, se pode concluir que o processo de teste é uma importante etapa no desenvolvimento de software, pois é vital para se atingir com certa garantia a qualidade desejada. A utilização de ferramentas tendem a aumentar a qualidade do processo, tornando-o mais rigoroso e com menor risco de falhas.

A ferramenta proposta visa auxiliar o processo de testes funcionais de software, empregando um método simples e abrangente, baseado em casos de uso. A utilização do modelo de casos de uso torna a utilização da ferramenta muito ampla, pois esse é um modelo muito aceito e utilizado. Também a torna compreensível para um grande número de usuários, pois o modelo de caso de uso é simples e fácil.

Os objetivos desse trabalho foram atingidos. Um modelo para especificação de requisitos foi definido para poder ser usado nos testes e uma ferramenta foi construída para auxiliar nesse processo. A geração de planos de teste se deu uma forma rápida e prática, conforme estudo de caso realizado e descrito na seção anterior. O sistema Héstita ainda fornece um histórico com os registros dos testes efetuados nos sistemas e também pode ser usado para armazenar e manter os cadastros dos requisitos dos sistemas.

Cabe salientar que o sistema Héstia supre uma parte do que as atividades de testes devem contemplar. Outros testes, como teste caixa-branca, de desempenho, de carga e de unidade devem fazer parte do processo de desenvolvimento de software.

Uma dificuldade enfrentada e que não foi prevista no planejamento desse trabalho foi o grande tempo gasto para a especificação e cadastro dos requisitos no formato de casos de uso, no estudo de caso realizado.

Apesar disso, é sempre importante lembrar que todos os requisitos devem ser especificados de forma completa e consistente, não só para serem usados nos testes com sucesso, mas também em todas as demais etapas do processo de desenvolvimento de software. Pode-se concluir que talvez essa seja uma das maiores dificuldades enfrentadas no processo de desenvolvimento de software e também a origem da maioria dos problemas. A criação e atualização constante dos requisitos dispensam uma razoável quantidade de tempo, e hoje a maioria das equipes de desenvolvimento de software trabalha em seus projetos com pressões constantes sobre os prazos de entrega. Embora tudo isso, se o tempo ideal for dedicado para as fases de especificação de requisitos e testes, o produto desenvolvido já terá agregado um grau maior de qualidade e estará menos sujeito a falhas.

Como trabalhos futuros destacam-se os seguintes itens:

- Integração do sistema desenvolvido com outras ferramentas gratuitas de apoio ao processo de teste de software, tais como gerenciadores de *bugs* (Bugzilla, Mantis), automação de testes (Selenium) e geradores de dados para teste;
- Normalização do sistema Héstia para atender as principais normas e padrões de desenvolvimento de software, como o CMMI, MPS.BR e IEEE 829;
- Customizações diversas no sistema, detectadas como melhorias ao longo do desenvolvimento desse trabalho.

Referencias

- Booch, G., Rumbaugh, J. e Jacobson, I. (2005), UML: Guia do usuário, Editora Campus, 2ª edição.
- Caetano, C. (2007), Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas.
- Cockburn, A. (2005), Escrevendo Casos de Uso Eficazes, Bookman.
- Delamaro, M., Maldonado, J. e Jino, M. (2007), Introdução ao Teste de Software, Elsevier.
- Koscianski, A. e Soares M. (2007), Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software, Novatec Editora, 2ª edição.
- Larman, C. (2007), Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientado a objetos e ao desenvolvimento iterativo, Bookman, 3ª edição.
- Molinari, L. (2003), Teste de Software: Produzindo Sistemas Melhores e Mais Confiáveis, Editora Érica, 3º edição.

Pressman, R. (1995), Engenharia de Software. Makron Books.

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