UM RELATO DE MELHORIA DO PROCESSO DE TESTE DE *SOFTWARE* APLICADO A UMA FÁBRICA DE *SOFTWARE*

Inácio Leite Gorayeb*
Sandro Ronaldo Bezerra Oliveira**

RESUMO

Este trabalho relata a experiência da mudança e evolução de um processo de software, além de descrever um arcabouço de ferramentas livres para apoiar o processo de teste de software, em uma fábrica de software. São fornecidas avaliações para a escolha de ferramentas de software livre de apoio ao processo, bem como problemas motivadores para as mudanças do processo inicial para o processo evoluído. O artigo mostra também resultados obtidos com o uso destes e de futuros trabalhos.

Palavras-chave: Teste de Software. Fábrica de Software. Software Livre. Automação de Teste. Processo de Teste.

ABSTRACT

This paper reports the experience of the change and evolution of a defined software process beyond to define a suite of free tools to support the Software Test process in a software house. Assessments for the choice of tools to support the process are discussed as well as the reasons that motivate the changes in initial process. It also shows the results obtained with the use of these and future works.

Keywords: *Software Testing. Software House. Free Software Tools. Test Automation. Test Process.*

1 INTRODUÇÃO

À medida que o emprego de sistemas de informação pela sociedade cresce ao ponto em que boa parte dos negócios depende cada vez mais de software e computadores, passa a ser de grande importância contar com um software de qualidade [1].

Com objetivo de se tornar mais competitivas, algumas empresas de software vêm implantando a gerência sistemática dos processos utilizados para o desenvolvi-

* Bacharel em Ciência da Computação pela UNAMA - Universidade da Amazônia. Mestrando do Programa de Pós-Graduação em Ciência da Computação – UFPA. igorayeb@gmail.com

** Doutor em Ciência da Computação pelo CIn/UFPE. Professor da Faculdade de Computação do ICEN/UFPA. srbo@ufpa.br

mento e a manutenção de software. Por meio da avaliação de seus produtos e da melhoria dos seus processos, essas empresas têm obtido a necessária melhoria da qualidade de seus produtos e, com isso, melhores resultados nos negócios.

Essas empresas que desenvolvem software compreendem a importância de se produzir sistemas com baixo nível de defeitos, bem como a importância de um processo maduro para atingir este objetivo. Mesmo assim, os sistemas atuais ainda contêm um alto índice de defeitos [1].

As etapas da engenharia de software de verificação e validação surgiram com a necessidade de garantir extrema confiabilidade de software nos sistemas, de maneira que um mínimo erro resultaria na falha da missão, resultando em perda de tempo e enorme recurso financeiro [8]. A validação de software não é uma atividade trivial. A atividade de teste, por exemplo, exige conhecimentos, habilidades, e infraestrutura específica. Um bom desenvolvedor ou projetista de software sem esta base dificilmente realizaria uma boa tarefa de testes.

Já o propósito do processo de verificação é confirmar que cada serviço e/ou produto de trabalho do processo ou do projeto atente apropriadamente os requisitos específicos [11]. Adicionalmente, Molinari [8] acrescenta ao conceito dois critérios fundamentais: o software deve executar todas as funções desejadas; e o software na sua execução, não deve passar por nenhum caminho que não tenha sido testado em alguma combinação com outras funções.

Este trabalho relata as mudanças ocorridas no processo de teste de uma fábrica de software no contexto do antes e depois da formação de uma equipe dedicada à execução desta disciplina, bem como, relata os critérios e motivações para escolha de ferramentas livres para o apoio a esse processo. Serão tratadas as motivações e necessidades dessas mudanças, bem como, mostradas as opções de ferramentas para apoio a determinados procedimentos e atividades do processo, além de justificada a escolha da ferramenta utilizada. Serão mostrados, também, resultados alcançados com as mudanças e a utilização de ferramentas.

O projeto desenvolvido pela fábrica consiste em um sistema de gestão de fomento. O sistema basicamente possui o escopo de gerenciamento de contratos, projetos, liberação de recursos, para projetos fomentados na Região Amazônica.

Além desta seção introdutória, este relato apresenta seis outras seções. A seção 2 ambienta quanto ao cenário da fábrica de software, do projeto e da organização em que ocorre. A seção 3 explica o processo de teste inicial da fábrica de software. A seção 4 discute o processo de teste após as mudanças e atualmente executado na fábrica de software. Na seção 5 serão descritas as ferramentas de apoio ao processo de teste, justificando as escolhas. A seção 6 discute alguns resultados e melhorias obtidos com a adoção/utilização do arcabouço de ferramentas escolhidas, bem como as melhorias do processo evoluído. Por fim, a seção 7 trata das conclusões deste trabalho e aponta futuras adaptações no processo.

2 CONTEXTUALIZAÇÃO DO CENÁRIO DO PROJETO

A empresa foco deste relato foi fundada em 1974, e foi a primeira empresa genuinamente brasileira a produzir e comercializar tecnologia no segmento de informática. Nos anos 90, já fazendo parte de estrutura de um banco nacional, tornou-se uma integradora de solução.

Em 2004 essa empresa iniciou o desenvolvimento de um programa de excelência tecnológica com um cliente bancário situado na Região Norte, cujo objetivo era dotar os profissionais do cliente de um ferramental composto por novos processos, novas soluções tecnológicas e novos paradigmas de negócios, para que pudessem aprimorar os níveis de competitividade e qualidade de serviços e com isso viabilizar as diretrizes do planejamento estratégico da organização.

No ano de 2007, diante da necessidade do cliente em atualizar tecnologicamente (mudança de plataforma operacional) um sistema de fomento e em função da sua complexidade, o projeto ganha uma equipe dedicada para seu desenvolvimento. Com isso, a empresa cria seu projeto piloto de uma fábrica de software no Pará. A fábrica possui aproximadamente 60 (sessenta) recursos humanos desempenhando papéis bem definidos e seguindo um processo de software também definido.

Pela constante necessidade de comunicação com diversos stakeholders do cliente, foi acordado que a fábrica ficaria sediada no mesmo espaço físico do cliente. Acordou-se, ainda, que o sistema seria entregue por módulos funcionais, onde estes, após a entrega, ficariam por um período em homologação, para o cliente validação dos requisitos implementados.

Tratando do cenário do sistema, a empresa enfrentou diversas dificuldades quanto à aquisição e definição de recursos. O projeto não contemplava recursos de software e hardware, com isso, o projeto foi direcionado a usar os recursos que o cliente dispusesse a licença. Assim, diversas ferramentas de software livre foram usadas para apoiar diferentes disciplinas da engenharia. Quanto aos recursos humanos, o projeto até hoje possui dificuldade em compor a equipe, pela falta de mão de obra qualificada no mercado local para atender aos papéis definidos no processo de software. Para suprir essas necessidades, a empresa oferece de conhecimento, workshops e cursos internos.

Para caracterizar o tamanho do projeto em desenvolvimento, no início do projeto havia sido estimado 4560 UCPs (Pontos de Casos de Uso) a ser implementados, porém com o surgimento de novos requisitos este esforço está sendo revisado. Hoje o projeto possui aproximadamente 1000 UCPs desenvolvidos.

No contexto de SLAs (Service Level Agreement – Acordo de Nível de Serviço) do projeto, foi definido que a criticidade dos bugs (erro de funcionamento comum de um software) seria tratada como: crítico, importante e trivial. Para a homologação acontecer, o sistema: não poderia possuir nenhum bug crítico (bug que impede que o cliente realize sua tarefa) pendente; menos de 25% de bugs classificados como importantes (bug que dificulta ou atrapalha a realização da tarefa, porém não a evita), onde o fornecedor mesmo assim possui o compromisso de corrigi-los mesmo após o módulo ser homologado; e bugs triviais não seriam impeditivos para a homologação, devendo o fornecedor corrigir depois que módulo seja homologado.

3 CENÁRIO E PROCESSO INICIAL DE TESTE DE SOFTWARE

Nesta seção será mostrado como o processo geral e específico de teste de software eram conduzidos inicialmente na fábrica de software relatada na seção anterior.

A figura 1 mostra de forma geral as etapas do processo de desenvolvimento de software da fabrica com ênfase na disciplina de teste. Para melhor entendimento do fluxo de atividades do processo foram utilizadas notações de UML – *Unified Modeling Language*, para descrever como ocorre o processo de teste de software atualmente. Uma descrição do fluxo será discutido logo abaixo.

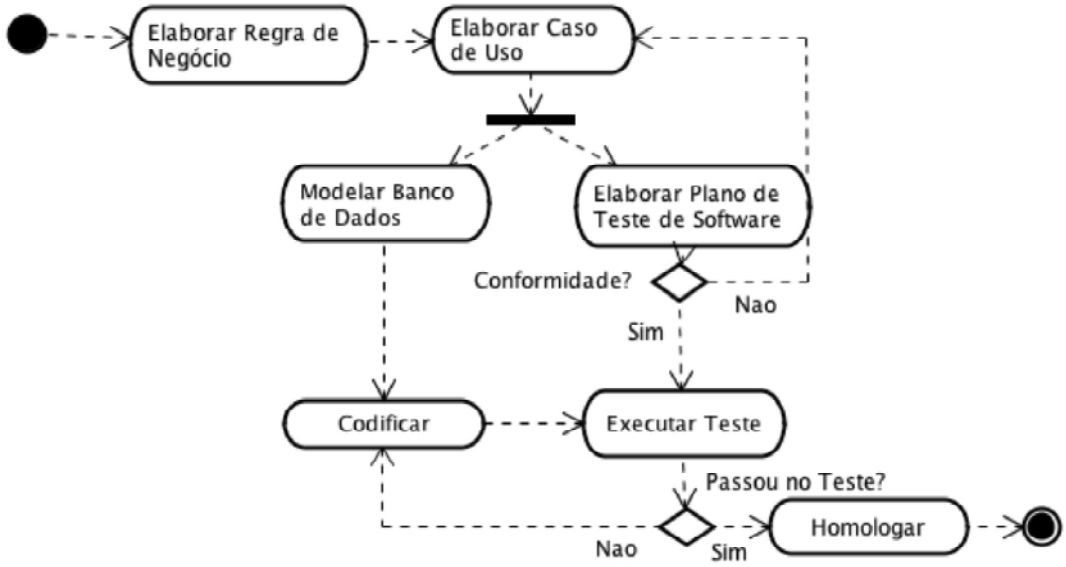

Figura 1 – Fluxograma do processo geral incial.

Inicialmente as Regras de Negócio (RN) são entendidas com o cliente e é elaborado um documento de regra de negócio. Na segunda etapa, tendo como insumo a documentação de regras de negócio, uma documentação de Casos de Uso (UC) é elaborada, onde a documentação aproxima da linguagem utilizada na regra de negócio do nível de linguagem adotada pela programação (implementação).

Na etapa de modelagem da base de dados (Modelagem BD) é feito, então, o modelo de banco de dados que dará suporte aos engenheiros de software para a implementação do caso de uso. Paralelamente a etapa de modelagem de banco de dados, e também baseado na documentação de caso de uso, o projeto de teste (contempla informações necessárias sobre o planejamento dos testes no módulo funcional em desenvolvimento e seus casos de testes para execução) é elaborado dando subsídio para a execução dos testes.

Após a liberação do caso de uso implementado, é então iniciada a execução dos testes. Inicialmente, os testes, pela falta de uma equipe dedicada à realização desta disciplina, eram realizados pelos próprios analistas de requisitos. Após o término dos

testes e realização das devidas correções o U.C. implementado, é então disponibilizado para homologação.

Alguns problemas foram detectados com este cenário. O primeiro é que pelo fato do teste ser elaborado pelo próprio analista de requisito, a execução do teste acabava tornando-se viciada/tendenciosa, ou seja, um teste sem outra perspectiva de visão crítica no contexto dos requisitos. O segundo problema diz respeito ao fato de que como o analista deveria elaborar os testes de software e posteriormente executá-los, este cenário acarretava no atraso da execução das suas atividades principais, a análise de requisitos. Outro problema, e não menos importante, percebido é que pela falta de uma equipe especializada, o teste não contemplava outros contextos, prejudicando a completude e correção dos próprios testes.

Outra questão percebida, é que durante a correção de defeitos não existia um gerenciamento destes através de uma ferramenta automatizada, ou seja, o controle (quem, quando, prioridade, tipo etc.) dos defeitos relatados pelo cliente e pela própria equipe de desenvolvimento acabava por se perder. Além disso, o processo em questão também não contemplava a prática de reteste dos defeitos encontrados, deixando com que estes defeitos não fossem retestados após sua correção, ou seja, sem se ter a certeza de que o mesmo realmente foi corrigido.

Na fase de homologação o UC implementado é oferecido em um ambiente próprio para a homologação (aceite) do cliente. A homologação significa para o projeto relatado no cenário, que o módulo funcional está de acordo com o especificado e satisfaz às necessidade do cliente e ainda que está pronto para ser promovido à produção.

A figura 2 apresenta as seguintes etapas do processo de teste: Elaboração do Projeto de Teste (PTS) e execução de teste. Na etapa de elaboração, o projeto de teste de software era preenchido pelo analista de requisitos, após o término da elaboração do requisito e contemplava, no mesmo documento, os resultados da execução dos testes. Caso algum defeito fosse encontrado, o responsável pelo teste encaminhava a inconsistência ao desenvolvedor responsável que então executava a correção. Posteriormente, o módulo era incluído para fase de homologação.

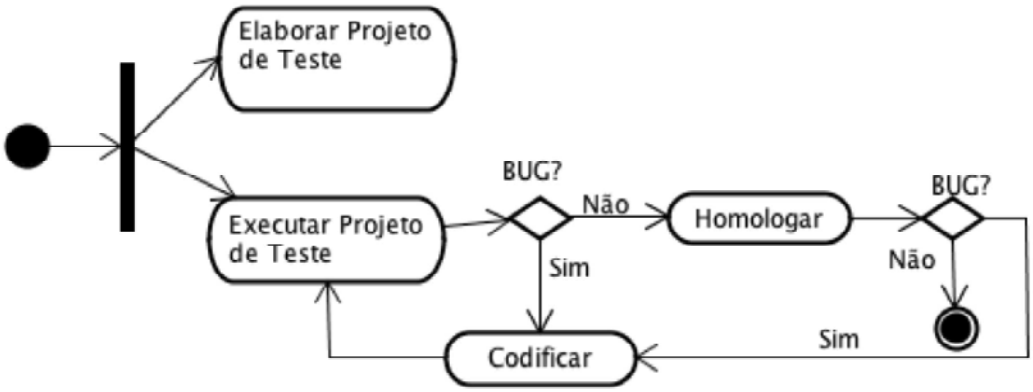

Figura 2 – Processo de teste de software inicial.

Como se percebe, na figura 2, as fases de elaboração e execução de teste eram feitas em paralelo, o que indica que não havia planejamento prévio para o teste do sistema. Além disso, o processo de teste contemplava apenas testes funcionais; os testes executados não tinham a preocupação de testar o software em aspectos também não funcionais, além de conformidade, performance, estresse e regressivo, que eram necessários para a homologação e bom funcionamento do módulo funcional em produção.

Outros problemas foram apontados em função do processo acima não contemplar outras práticas de teste de software, não possuir um processo mais completo e ainda não possuir uma equipe de teste dedicada e qualificada. O atraso na descoberta de defeitos acabava ocasionando atraso nas entregas dos módulos funcionais, alto custo no sistema e perda de qualidade do produto recarregado.

A partir dos problemas apresentados, com a necessidade de agilidade na homologação do sistema e com o aumento na qualidade do produto, algumas medidas foram tomadas para prover melhorias neste processo, desde a disponibilização de uma equipe dedicada até a adoção de melhores práticas. Estas melhorias serão discutidas na seção a seguir.

4 CENÁRIO E PROCESSO DE TESTE DE SOFTWARE PÓS-MELHORIAS

Nesta seção será discutido como se encontra institucionalizado o processo atualmente na fábrica de software, após alguns aportes de melhorias realizados para a execução da disciplina de teste de software, como a contratação de uma equipe dedicada e especializada. Para melhor entendimento do processo de teste de software, foi definido o fluxo de atividades visualizado na figura 3.

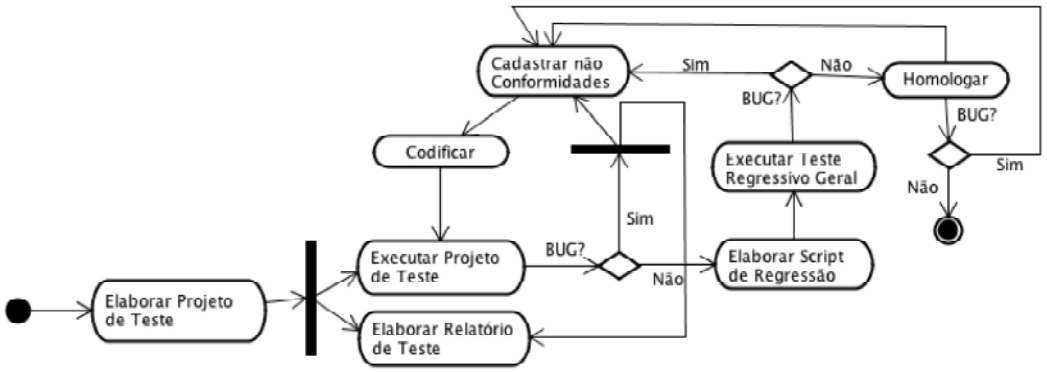

Figura 3 – Processo de teste de software pós-melhorias.

A primeira atividade descrita no fluxo é a elaboração do projeto de teste de software, onde são elaborados os casos de testes e os cenários de teste contemplados em um único documento (Projeto de Teste - PTS). Cada módulo funcional possui seu projeto de teste. A contemplação em um único documento dar-se pelo fato de tornar o processo menos burocrático, já que a demanda de desenvolvimento de casos de uso é alta.

A segunda atividade é formada por duas tarefas: a execução do projeto de teste e a Elaboração do Relatório de Teste (RTS). A execução do projeto de teste consiste em aplicar os cenários e os casos de teste de software projetados para o módulo funcional desenvolvido. Já a elaboração do RTS significa a criação de um relatório de teste onde para cada passo constante no cenário do PTS é anotado um resultado no documento RTS, podendo este resultado assumir três aspectos distintos: OK, bug, não executado; e como discutido na seção 2, caso o resultado seja "bug", três diferentes níveis de criticidade (trivial, importante e crítico) são associados.

No relatório de teste, o passo do cenário em análise recebe como resultado: OK, quando o passo é executado sem problemas e o sistema responde como o esperado (descrito na especificação de casos de uso) sem desvios de comportamento; bug, é atribuído para o passo que apresenta algum comportamento inesperado; e não executado, é atribuído quando por algum motivo o passo não pode ser executado, isso ocorre, por exemplo, quando algum bug torna-se impeditivo para a execução deste passo ou ainda por algum motivo inesperado que impeça a sua execução.

O documento de relatório de teste possui, ainda, um papel fundamental no processo de teste e mesmo no processo geral da fábrica gerando métricas para avaliações da melhoria contínua do processo. Um exemplo de métricas é bugs/UCs, ou seja, porcentagem de bugs encontrados na fase de testes/homologação contraposto a quantidade de UCPs especificados no módulo funcional para desenvolvimento, permitindo uma análise da eficiência dos testes dada a complexidade da UCP, e da quantidade de bugs que a equipe de teste está deixando de descobrir para serem descobertos apenas na fase de homologação.

Quando é finalizada a fase de execução do projeto de teste e de elaboração do relatório de teste, caso haja bugs no módulo funcional em análise, o BUG é registrado na ferramenta de software para a gestão de bugs e atribuída para o responsável pela correção (recurso humano alocado no projeto para o desenvolvimento do caso de uso). Após a correção, o caso de teste é retornado à equipe de testes responsável pela descoberta do bug para reteste (o ciclo de correção se repete até que o bug seja corrigido). Caso o bug tenha sido corrigido, o registro realizado no Mantis é "fechado". Este procedimento se repete até que o último bug seja corrigido.

Na fase de Homologação o U.C. implementado é oferecido em um ambiente próprio para a homologação (aceite) do cliente. A homologação significa para o projeto relatado no cenário, que o módulo funcional está de acordo com o especificado e satisfaz às necessidade do cliente e, ainda, que está pronto para ser promovido à produção.

Em vez de bug, se a solicitação cadastrada na ferramenta for uma mudança, então é seguido o procedimento de gestão de mudança. O procedimento de gestão de mudança consiste em convocar uma reunião com o grupo responsável pela gestão de mudanças, alocado para o projeto, para definir impacto, esforço, tempo, custo e tomar a decisão se a mudança será realizada ou não. Esse procedimento é executado tanto internamente quanto em homologação. Importante mencionar que estas etapas foram definidas na forma de um procedimento que se encontra disponível no processo de desenvolvimento de software.

Quando não houver mais defeitos é então feito um script da automação de teste regressivo seguindo o projeto de teste, para que futuramente possa ser realizado teste no módulo funcional. Após a criação dos scripts o teste total de regressão do sistema é executado. Ao término, caso não haja mais nenhum bug, o módulo funcional é então publicado para homologação junto ao cliente. Antes de o módulo funcional ser promovido à homologação, o mesmo entra em um processo de GC (Gestão de Configuração) para que possa ser assegurada em que versão poderá ser recuperada, caso necessário, gerando baselines, versões e seu release notes.

O processo de homologação consiste em o sistema ficar disponível para o cliente testar, validar e verificar, a fim de sinalizar que o sistema encontra-se realmente dentro do que foi especificado na fase de requisitos, ou ainda, registrar possíveis inconsistências ou bugs. Para isso, foi criada uma nova instância da ferramenta de gestão de bugs e mudanças, apenas para o fim de homologação, ou seja, o cliente registra suas considerações de mudanças ou bugs encontrados, e a equipe de teste é responsável por dar prosseguimento aos registros feitos pelos clientes. Com essa nova instância, conseguimos manter em ambientes separados as demandas dos clientes das solicitações realizadas pela própria equipe interna de testes alocada para o projeto. Estas demandas classificam-se tanto como possíveis correções de bugs quanto de solicitação de mudanças. Isso ajuda na extração de métricas e melhorias contínuas para garantir a satisfação do cliente quanto ao desenvolvimento do projeto.

Durante a definição do processo de teste atual, procurou-se realizar sua evolução em conformidade com os Resultados Esperados do Processo[1] (Resultado esperado em cada constantes nos processos de Verificação e Validação do modelo MPS.BR[2], presente no nível de maturidade[3] D (Largamente Definido). Esta avaliação de compatibilidade, no que tange à análise da conformidade, assemelha-se a uma avaliação de aderência dos processos do CMMI[4] (Capability Mature Model Integration) na sua visão contínua (visão que classifica em ordem os níveis de maturidade), apesar de se reconhecer que no modelo MPS.BR a implementação dos resultados esperados não ocorre seguindo a visão contínua. A tabela 1 permite uma análise deste atendimento, apresentando quais das recomendações propostas no Guia de Implementação do modelo MPS.BR [7] para os resultados esperados dos processos de verificação e validação podem ser reconhecidas como implementadas pelo processo atual do processo, bem como as recomendações ainda não implementadas.

[1] Resultados esperados do processo estabelecem os resultados a ser obtidos com a efetiva implementação do processo[1].

[2] MPS.BR é um programa para Melhoria do Processo de Software Brasileiro[1].

[3] Nível de maturidade estabelecem patamares de evolução de processos na organização[1].

[4] CMMI é um modelo de referência que contém práticas (genéricas ou específicas), necessárias a maturidade de processos de software.

Tabela 1 – Atendimento do processo às boas práticas do processo de VER e VAL do MPS.BR.

Atividade do Processo Atual	Resultado Esperado do MPS.BR	Boas práticas do MPS.BR Implementadas	Boas Práticas do MPS.BR não Implementadas
Elaborar projeto de teste	VER1/VAL1	• Analisar produtos de trabalhos que serão produzidos; • Selecionar os produtos de trabalhos que serão verificados.	• Análise deriscos dos produtos de trabalhos para alcance aos objetivos.
	VER2/VAL2	• Definição de uma estratégia de verificação descrevendo procedimentos, infraestrutura e responsabilidades; • Definição de ferramentas de apoio; • Inspeção de artefatos feita por um membro da equipe que não seja o revisor.	• Revisão por pares dos produtos produzidos; • Reuniões do método walkthrough.
Elaborar projeto de teste	VER3/VAL3	• Definição decritérios e procedimentos que serão utilizados para verificação de cada produto de trabalho e na preparação de cada ambiente.	• Relatórios técnicos apresentando um conjunto de critérios apresentado: critérios para avaliação do produto associados as características de qualidade.
Executar projeto de teste/ Scripts regressivos	VER4/VAL4	• Realização dos testes.	• Revisão por pares.
Elaborar Relatório de Teste Elaborar Script regressivo	VER5/VAL5	• Registro dos defeitos identificados; • Utilização de classificação de defeitos (crítico, importante etrivial); • Classificação de origem dos defeitos.	• O documento derelatório de teste e o registro de não conformidade em ferramenta de *bugtracking*.
Cadastrar não conformidades	VER6/VAL6	• Disponibilização dos resultados através de relatório de teste colocado sob gerência de configuração e registro em ferramentas de*bugtraking*; • Análise de resultados apenas do relatório de testes para verificar o atendimento ao SLAou não.	• Análise de dados obtidos ao término de cada atividade de verificação.
Homologar	VAL7	• Nada se aplica.	• Realizar testes do sistema validado em ambiente de produção ou cópia do mesmo.

5 FERRAMENTAL DE APOIO AO PROCESSO DE TESTE DE SOFTWARE

Para o suporte e apoio ao processo atual executado, foram estudadas e avaliadas diversos tipos de ferramentas de software livre e proprietário para o atendimento automatizado/sistematizado das atividades constantes no fluxo da disciplina de teste. Em quase toda a totalidade as ferramentas escolhidas foram de cunho livre por três motivos: o primeiro retrata o fato da instituição incentivar e colaborar com desenvolvimento e uso de ferramentas livres; o segundo diz respeito à excelência apresentada por essas ferramentas em critérios de qualidade e atendimento às necessidades das atividades e dos processo; e o terceiro trata das questões de custo, as ferramentas são gratuitas e livres para ser utilizadas e adequadas conforme a necessidade apresentada. Nesta seção será dada ênfase para as ferramentas que participam ou apoiam o processo de teste.

Para os testes regressivos automatizados foram analisadas duas ferramentas o WebLoad e o Selenium. O WebLoad é uma ferramenta gratuita, porém não open source, que realiza testes regressivos to tipo on click, ou seja o sistema grava a navegação do analista de teste e transforma em script. Porém o WebLoad funciona unicamente na plataforma Windows não dando possibilidade da utilização e outros sistemas operacionais. A escolha da ferramenta para essa função foi o Selenium pelo fato da facilidade de multiplataformas e ainda outras que serão descritas no decorrer da seção.

O Selenium [9] é um arcabouço de ferramentas para apoio ao teste automatizado de sistemas web independente de plataforma. O Selenium é dividido em quatro outras ferramentas: a primeira é o Selenium IDE que consiste basicamente um add-on para o navegador Mozilla Firefox, possui a funcionalidade de gravar clicks, frases e outras ações em um script a fim de ser reproduzido para realizar testes automatizados. A figura 4 mostra a interface de gravação de scripts do Selenium IDE.

A segunda ferramenta é chamada de Selenium CORE, que consiste de uma aplicação web desenvolvida em Javascript que possui a funcionalidade de prover um "servidor" de scripts de teste. O Selenium CORE fornece uma interface de controle para automação de testes onde podem ser ativados os scripts independente de navegador, apresentando uma ordem de execução, relatório e histórico de sucesso e falha, como mostra a figura 5.

Figura 4 – Tela de captura de ações do Selenium IDE..

A terceira ferramenta é chamada de Selenium RC (Remote Control), que consiste de uma ferramenta de automação teste que permite ao usuário escrever testes automatizados em diferentes tipos de linguagens, além disso, tem a capacidade de trabalhar com múltiplos navegadores executando diferentes scripts. A figura 5 mostra como funciona a arquitetura do Selenium RC.

Figura 5 – Tela de execução de Scripts do Selenium RC.

A quarta e última ferramenta da suite Selenium é o Selenium Grid que consiste em um hub, que é conectado a vários Selenium RC e consegue manipular todos ao mesmo tempo de maneira a gerar concorrência proporcionando realizar testes mais complexos. A suíte de ferramentas de testes automatizados Selenium, contribui para o processo da fábrica de software automatizando testes regressivos de funcionalidade a fim de assegurar que novas funcionalidades ou alterações não tenham inseridas falhas em partes do produto já testadas.

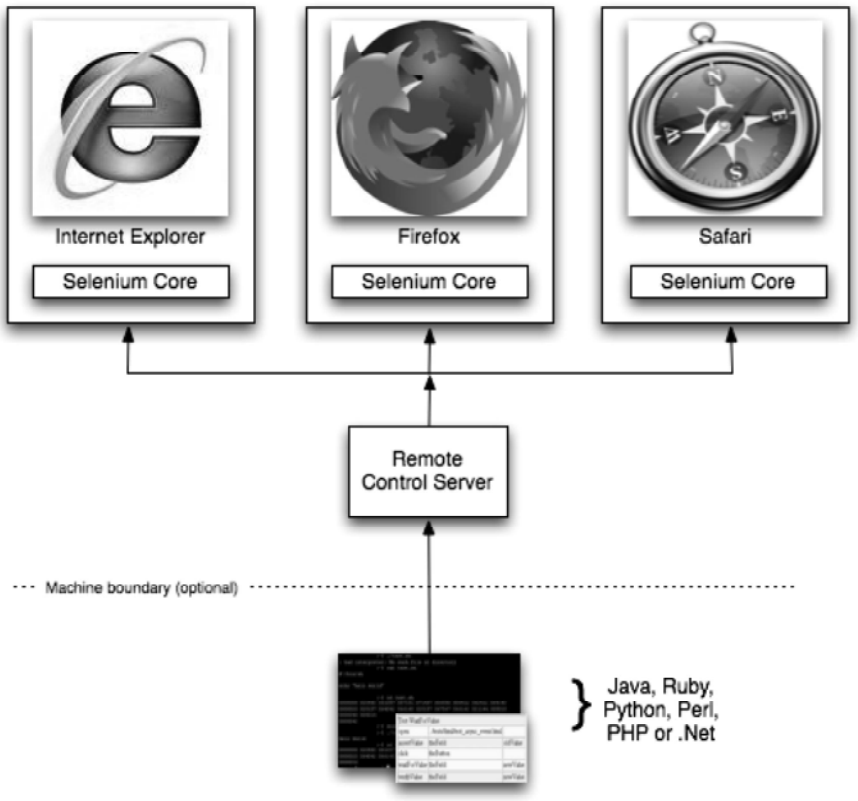

Figura 6 – Arquitetura de funcionamento do Selenium RC.

Para o teste de performance, foram testadas duas ferramentas, o WebLoad Console e o Apache JMeter. A ferramenta escolhida foi o JMeter, uma vez que é multiplataforma e ainda o WebLoad, não sendo disponibilizado em sua totalidade gratuito.

O JMeter consiste em uma ferramenta open source desenvolvida em Java e possui funcionalidades para teste de carga e performance em sistemas web. Além de uma análise estatística precisa, como mostra a figura 7. Outro fator importante para a escolha do JMeter é possuir um suporte de teste para sistemas JAVA que utilizam o framework Java Faces Bean, que para maior segurança da aplicação em cada seção, os objetos recebem pid (números identificadores) diferentes e com essa característica do framework, a maioria das ferramentas de performance acabam falhando por ser perder na hora de executar seus scripts como em alguns casos o WebLoad.

Na atividade do processo de elaboração do projeto de teste, são utilizadas duas ferramentas de apoio. Para o controle de versão de documentação e código fonte é utilizado o CVS (Control Version Server) [3], um sistema de controle de versão que permite que se trabalhe com diversas versões de arquivos organizados em uma estrutura de diretórios, mantendo e guardando históricos de quem e quando manipulou os arquivos. O CVS contribui para a primeira fase processo de teste, desde o momento em que o analista de teste acessa o repositório para consultar os requisitos, até o momento em que o projeto de teste é colocado também no repositório. Quando o analista de teste começa sua atividade de entendimento dos requisitos especificados, os mesmos

são acessados através de uma TAG, que contém mapeado os documentos participantes do produto e suas versões "congeladas", independentemente destes documentos estar em evolução ou não.

Figura 7 – Apresentação de resultados do teste de Stress no JMeter.

Para apoio à fase de elaboração de projeto de teste e elaboração de relatório de teste nas quais são produzidos documentos de texto e planilhas foram levadas em consideração duas suítes de escritório: o BrOffice e o Microsoft Office. Optou-se por utilizar o BrOffice que consiste em uma suíte de ferramentas de escritório [2]. O BrOffice é uma suite de ferramentas derivada do projeto OpenOffice, sendo adaptada através de pacotes para a língua portuguesa praticada no Brasil. Além de atender às necessidades das atividades citadas acima e o BrOffice ser gratuito, a suíte é compatível com a suíte utilizada pelo cliente, o Microsoft Office.

Para as atividades de execução do projeto de teste e elaboração do relatório de teste de software, além da suíte de escritório citada acima, estas fases contam com o apoio de outras ferramentas que dependem do tipo de sistema em desenvolvimento, no caso do cenário da fábrica de software retratado neste trabalho, o sistema é em plataforma web, onde são utilizados diferentes navegadores (Firefox, Internet Explorer, Google Chrome etc.) nas versões estabelecidas em contrato e sistemas operacionais distintos também estabelecidos em contrato.

Na atividade de cadastrar não conformidades e elaboração de RTS é utilizada a ferramenta Mantis [6], que consiste de uma ferramenta web de bugtracking livre e open source. O Mantis possui como função o registro, atribuição e acompanhamento de não conformidades através de uma interface web como mostra a figura 8. O Mantis possui a vantagem de possuir seu workflow configurável através de uma máquina de estados definida, onde define fluxo em que o registro irá percorrer de acordo com sua categoria e estado.

Figura 8 – Lista de ocorrência de erro do Mantis.

Além do Mantis colaborar com o processo de teste, também colabora com o processo de homologação, onde o cliente registra suas requisições (bugs, mudanças, sugestões de melhorias ou pendências) de feedback à fábrica de software. O Mantis corrobora, ainda, com a geração de métricas apresentadas ao final do ciclo do processo a fim de caracterizar a melhoria contínua das práticas, corrigir falhas e ainda medir o desempenho quantitativo/qualitativo da equipe responsável pelos testes.

6 RESULTADOS OBTIDOS

Com a evolução e a institucionalização do processo descrito na seção 4, bem como a utilização das ferramentas descritas na seção 5, alguns resultados já foram verificados. Vale ressaltar que essas melhorias não se resumem tão somente ao aperfeiçoamento do processo de teste, e sim à aplicação do programa de melhoria ao processo organizacional da fábrica de software discutida na seção 2.

Para a avaliação da melhoria foram coletadas métricas durante todo o processo de desenvolvimento de dois módulos de complexidade similar, as quais, do seu início até sua homologação pelo cliente passaram-se 2 meses cada um. As métricas foram coletadas ao término de cada fase (desenvolvimento, testes, levantamento de requisitos, homologação) pelo fato das mesmas já serem maduras e completas ao término das fases. Essas coletas eram feitas pelos coordenadores (líderes) de cada fase, em função de possuírem um monitoramento da execução das atividades de cada fase e tempo de realização das mesmas. Como instrumentos de coleta foram utilizados registros das ferramentas de bugtraking, monitoramento de andamento com base no cronograma de atividades, relatórios de execução de testes e relatórios de testes regressivos automatizados. As métricas analisadas pelos coordenadores eram repassadas aos gerentes, onde eram verificados os indicadores, e, dependendo de cada métrica, decisões eram tomadas a fim de prover melhorias ao processo de desenvolvimento e aos produtos resultantes da execução deste processo.

Para a avaliação da qualidade do produto foram colhidos registros de defeitos na ferramenta de bugtracking tanto pela equipe do fornecedor quanto pelo cliente. Seguem abaixo os resultados:

- **Aumento da produtividade:** Com a utilização de ferramentas em determinados momentos no processo obteve-se um ganho de tempo considerável na execução das atividades. O teste regressivo, por exemplo, foi uma das atividades que o ganho de produtividade foi mais acentuado, antes da adoção da ferramenta Selenium a equipe levava de 3 a 4 dias para realizar o teste regressivo funcional em todo o sistema, com a ferramenta essa atividade dura em média de 20 a 30 minutos;
- **Aumento da qualidade do produto:** por conta da contratação de uma equipe mais qualificada para a área de teste de software e o aumento das atividades que compõem o fluxo de funcionamento dos testes, é provável que tenha contribuído para que o sistema tenha se tornando mais estável e consideravelmente com menos defeitos [2]. O aumento da qualidade do produto deu-se também pela institucionalização das áreas de garantia da qualidade e controle da qualidade, com isso, uma maior dedicação e expertise voltados para área de teste, porém, é facilmente perceptível que o aumento da qualidade não se resume somente a essas práticas e sim ao conjunto de melhorias providos ao processo;
- **Sistematização do processo:** com a sistematização de parte do processo erros relacionados a desvio de processo ou perda de sincronia entre as equipes foi diminuído em media 40% (informação coletada pela equipe de garantia da qualidade) para alguns procedimentos e atividades. A atividade de registro de não conformidades, por exemplo, que possui seu workflow controlado pela ferramenta Mantis através de uma máquina de estados definida, onde dependendo de categoria o registro é atribuído ao responsável pela próxima tarefa relacionada a não conformidades;
- **Diminuição de descoberta de defeitos em homologação:** com o aumento da abrangência dos testes em diferentes contextos (tipos de testes), tipos de defeitos que antes não eram detectados no período de teste e sim em homologação passaram a ser encontrados na fase correta (testes), por exemplo, conformidade com os requisitos, conformidade ao padrão de interface, análise de desempenho, entre outros. Hoje apenas 15% dos defeitos de desenvolvimento do projeto são descobertos pelo cliente em fase de homologação frente aos 28% medidos no processo inicial. As métricas de quantidade de defeitos encontrados foram calculadas através da coleta de número de defeitos não conhecidos encontrados em fase de homologação frente ao total de defeitos do módulo antes melhorias e pós-melhorias;
- **Ganho da qualidade do produto:** Com a sistematização de algumas atividades do processo e ganho de produtividade, além de melhorias realizadas no processo, o produto apresentou uma melhora considerável na qualidade, apenas 15% dos defeitos conhecidos do sistemas (erros relatados em homologação pelo cliente e não relatado pela equipe de teste) são encontrados em fase de homologação frente aos 30% encontrados anteriormente nessa mesma fase. Essa métrica colhida foi obtida comparando dois módulos de mesma complexidade (contratos e propostas) onde foi pego o número de erros registrado pelo cliente e não registrado pela equipe de teste no módulo de propostas (antes das melhorias) e também o número de erros registrados pelo cliente e não registrados pela equipe de testes para o módulo de contratos (pós-melhorias);

- **Descobertas de defeitos prematuramente:** em função de o teste de Software começar a acontecer juntamente com o início da codificação do produto, os defeitos acabam sendo relatados mais "cedo", ou seja, antes mesmo de começar a execução dos testes de fato, e também, antes do cliente receber o sistema, de maneira a antecipar as correções destes defeitos tanto na especificação quanto na implementação dos requisitos. Isso se dá em função da maioria das inconsistências ser corrigidas ainda na iteração de desenvolvimento do próprio módulo funcional, de modo que não consuma o tempo alocado para o desenvolvimento de outros módulos tanto para o desenvolvedores, analistas de requisitos, analistas de testes e Projetistas. Com essas ações o time de desenvolvimento tem a oportunidade de corrigir ou adequar determinadas falhas ou mudanças ainda em tempo de implementação, o que leva também à diminuição de custos e uma quantidade menor de defeitos em fase de homologação;
- **Diminuição dos custos:** a descoberta prematura de defeitos nos módulos funcionais faz com que os mesmos tornem-se menos custosos para a fábrica de software tanto no sentido financeiro quanto no sentido moral [3]. No sentido financeiro, pelo fato de os defeitos ser descobertos após a entrega do produto e os envolvidos na correção já estar alocados em outras tarefas, tendo que parar a realização destas para corrigir os defeitos encontrados, com isso, o tempo de desenvolvimento das novas tarefas era atrasado necessitando de mais esforço alocado para desenvolvê-las, ou seja, gerando mais custos. No sentido moral, pelo fato de um maior número de defeitos ser descobertos antes da entrega do produto para o cliente minimizando os registros de não conformidades na ferramenta de bugtracking. A análise de redução de custos é baseada na lei de 10 descrita por Myers [4]. Outro fato importante que contribuiu para a diminuição dos custos é que o SLA de descobertas de defeitos estabelecido entre fornecedor e cliente, conforme discutido na seção 2, é alcançado com uma redução média (dependendo da complexidade e tamanho do módulo) de 50% do tempo de homologação. Esta métrica foi coletada comparando o tempo de homologação entre dois módulos de mesmo nível de complexidade um antes da melhoria e outro depois;
- **Concepção e elaboração de um conjunto de ferramentas livres de apoio ao processo de teste de software:** Com estudos, tunning e testes das ferramentas apresentadas, projetos e/ou fábricas com similaridades de processo, atividades e procedimentos podem também utilizar a suíte de ferramenta apresentada, ou tomar como referência para decisão de utilização;
- **Controle das atividades planejadas:** a utilização de ferramentas em determinadas atividades, torna o controle de andamento e estimativa de tempo mais exatos, uma vez a ferramenta dando visibilidade de em que ponto, momento ou responsável está determinado procedimento. Esse sentimento foi elencado em entrevista com alguns coordenadores das equipes de teste e homologação.
- **Melhora da produtividade dos analistas de requisitos:** com os Analistas de Requisitos dedicados apenas a sua função (definida no processo) e não mais despendendo tempo em testes, os mesmos aumentaram sua produtividade na elicitação e especificação de requisitos, dispondo de mais tempo para reuniões com cliente e equipe técnica de desenvolvimento para a especificação mais precisa dos requisitos. A informação de produtividade foi relatada pela gerência de requisitos quanto aos resultados alcançados;

- **Formação de uma equipe de testes:** formou-se uma equipe qualificada e dedicada somente para a disciplina de testes. A equipe atualmente é formada por um líder de testes e quatro analistas de testes. O líder de testes é responsável basicamente por gerenciar recursos e demandas passadas à equipe, bem como definir juntamente com os analistas os tipos de testes a ser realizados nos módulos funcionais do projeto. Já os analistas de testes são responsáveis por gerar o documento projeto de teste, a execução dos testes e os registros de não conformidades;
- **Redução da quantidade de ciclo de testes:** com alguns ciclos gerados no novo processo (pós-melhorias), foi percebido que, com a adoção das boas práticas discutidas na seção 4, o número de ciclo de testes, para o atendimento ao SLA (seção 2), foi diminuído. No caso dos módulos analisados, a diminuição foi de 5 ciclos para 3 ciclos, entendendo que para esses módulos cada ciclo corresponde a um esforço de 80 horas/homem de trabalho, ou seja, obteve uma redução total de 160 horas/homem, levando em consideração que a complexidade dos módulos e a experiência da equipe envolvida eram as mesmas. Com essa redução, o esforço da equipe de testes e o tempo que um módulo é liberado para homologação é menor que o antes verificado. Assim, a equipe pode dar vazão em outros módulos ao invés de realizar mais ciclos nos mesmos.

Os pontos de melhorias citados foram os mais visíveis, porém as melhorias não se resumem a estas podendo ser percebidas em vários outros contextos como: arquitetura, usabilidade, padrão de interface etc.

7 CONCLUSÃO

Com a melhoria nos processos e utilização de ferramentas associada à configuração, adequação e otimização das mesmas, percebeu-se um ganho de produtividade da equipe alocada para o projeto, aumento de qualidade do produto e principalmente aumento da satisfação do cliente. Porém, a evolução tem continuidade, a intenção é incorporar novas ferramentas que consigam trazer mais controle, produtividade, qualidade e menor custo associado ao produto.

Algumas lições aprendidas podem ser destacadas. A primeira remete ao fato de sempre que puder optar por uma ferramenta livre é interessante que seja feito, desta maneira o investimento destinado a uma ferramenta proprietária pode ser convertido em outros tipos de investimento como, por exemplo, contratação de mão de obra, treinamentos etc. A segunda diz respeito ao fato de que ferramentas livres de código aberto possuem um vantagens de poder ser modificadas de acordo com as necessidades do projeto. A terceira retrata o fato de que para algumas tarefas como, por exemplo, teste regressivo, a utilização de ferramentas juntamente com mão de obra especializada geram resultados melhores (tempo, qualidade e custo) do que a utilização de apenas ferramentas ou de apenas de pessoas.

Como trabalhos futuros, a ideia é realizar testes com outras ferramentas de maneira a otimizar atividades do processo que ainda não possuem apoio ferramental e fazer com que a suíte de ferramentas se comuniquem entre si de maneira sistematizada e não apenas manual.

Este artigo apresenta a utilização de uma suíte de ferramentas livres para utilização em um processo de teste de software, o qual se encontra em definição para compor os resultados dos estudos provenientes de uma dissertação de mestrado do PPGCC/UFPA – Programa de Pós-Graduação em Ciência da Computação, e o primeiro de vários em busca da avaliação MPS.BR por parte da organização cujo relato foi extraído.

REFERÊNCIAS

[1] ARAUJO, R; CAPELLI, C; GOMES JR, A G; PEREIRA, M; IENDRIKE, H. S; IELPO, D; TOVAR, J. A. A Definição de processos de software sob o ponto de vista da gestão de processos de negócio. In: **VI Simpósio Internacional de Melhoria de Processos de Software** (SIMPROS), Rio de Janeiro, 2004.

[2] **BrOffice**. Disponível em: http://www.broffice.org/. Acesso em: dez. 2009.

[3] BRUNELLI, M. V. Q. **A utilização de uma metodologia de teste no processo da melhoria da qualidade do software**. Campinas: 2006. Biblioteca Unicamp.

[4] CRESPO, A. N.; SILVA, O. J. ; BORGES, C. A.; SALVIANO, C. F.; ARGOLO, M. T; JINO, M. Uma metodologia para teste de software no contexto da melhoria de processo. In: **III Simpósio Brasileiro de Qualidade de Software** (SBQS), Brasília, 2004.

[5] CVS – **Concurrent version system**. Disponível em: http://www.nongnu.org/cvs/. Acesso em: dez. 2009.

[6] MANTIS – **Oficial site.** Disponível em: http://www.mantisbt.org/. Acesso em: dez. 2009.

[7] MYERS, G. **The art of software testing**. New York: Wiley, 1979.

[8] MOLINARI, L. **Teste de software**. [s.l]: Editora Érica, 2003.

[9] SELENIUM - **Selenium HQ**. Disponível em: http://seleniumhq.org/. Acesso em: dez. 2009.

[10] MPS.BR - Associação para Promoção da Excelência do Software Brasileiro - SOFTEX, **MPS.BR - Guia Geral**, versão 1.3. Disponível em: www.softex.br/mpsbr. Acesso em: dez. 2009.

[11] MPS.BR - Associação para Promoção da Excelência do Software Brasileiro - Guia de Implementação – Parte 4: Nível D, versão 1.2. dez. 2009. Disponível em: http://www.softex.br/mpsbr. Acesso em: dez. 2009.