Área: CV ( ) CHSA ( ) ECET ( )

UMA FERRAMENTA PARA GERAÇÃO AUTOMÁTICA DE RELATÓRIO DE MUDANÇAS

Cleiton dos Santos Moura(bolsista, PIBITI/UFPI), Pedro de Alcântara dos Santos Neto (Orientador, Depto. de Informática e Estatística – UFPI)

Introdução

Assim como quase todos os tipos de produtos, *software*s evoluem constantemente com o decorrer do tempo. Tal evolução, também chamada de manutenção, é muito comum em sistemas em operação (IEEE, 1998). De acordo com muitos autores, a manutenção corresponde à parte mais onerosa do desenvolvimento de *software*, podendo representar até 80% do custo total do desenvolvimento (AZIZ et al., 2009).

Por conta da importância da manutenção, e pelo fato dela sempre estar associada a mudanças no *software*, foi criada uma área específica da Engenharia de *Software* denominada Gerência de Configuração de *Software* (GCS) (IEEE, 1990). A GCS é definida como uma disciplina que visa identificar e documentar as características de itens de configuração, controlar as suas alterações, armazenar e relatar as modificações aos interessados e garantir que foi feito o que deveria ter sido feito. Uma das tarefas da GCS é a geração de relatórios que detalhem as mudanças que foram feitas em um *software*, o que exige um tempo significativo para analisar o status do *software* antes e após as mudanças.

Esse relatório de mudanças, ou relatório de alteração, tem um papel importante porque, além de ser exigido pela GCS, todo e qualquer usuário gostaria de saber o que foi modificado de uma versão para outra de um *software*. Isso evita erros durante a operação do sistema, além da economia de recursos, pois uma vez já sabendo o que foi alterado, muitos chamados técnicos poderiam ser evitados. Porém, manter um relatório de alteração é uma tarefa cansativa e bastante propensa a erros. Se imaginarmos uma grande equipe trabalhando em um mesmo *software*, pode ficar bastante difícil sua execução.

Dado o cenário descrito, é plenamente justificável a necessidade de uma ferramenta que auxilie a automação da geração do relatório de mudanças de *software*. Na literatura só foi encontrada uma ferramenta que auxilie na criação do relatório de alteração, a JDiffChaser ferramenta que realiza o relatório de alteração baseado em telas capturadas das diferentes versões de uma aplicação. Mas ela só realiza relatório de aplicações feitas em aplicações Desktop Java, tendo assim um uso muito limitado. Diferentemente da JDiffChaser, a TChangeReport realiza no momento relatórios de Aplicações Web, mas como ela possui um mecanismo que aceita como entrada testes criados com qualquer ferramenta de teste funcional, ela se torna uma ferramenta bastante flexivel.

Metodologia

Este trabalho foi dividido em cinco fases: Teórica, Análise, Desenvolvimento, Experimentação e Fase Final. Durante a Fase Teórica foi realizado a pesquisa sobre trabalhos que abordavam testes funcionais de *software* e a utilização desses testes para geração de artefatos. Na Fase de Análise foi avaliada a forma de organização dos testes, para obter convenções de como eles são organizados e foram analisadas também formas de obter dados dos testes.

Na Fase de Desenvolvimento foi implementada a ferramenta TChangeReport, que realiza de forma semiautomática a geração do Relatório de Alteração, por meio do reuso de testes funcionais.

Na fase de Experimentação foi realizado um estudo experimental com intuito de avaliar o tempo gasto para criação de relatórios com e sem o uso da ferramenta, além de analisar a qualidade dos relatórios gerados.

Por fim, na Fase Final foi escrito um documento e disponibilizado na página http://www.ufpi.br/pasn a documentação da Ferramenta. Além disso, nessa fase foi iniciado o processo de registro de propriedade intelectual da ferramenta junto ao Núcleo de Inovação e Transferência de Tecnologia-NINTEC da UFPI. Tal processo ainda não foi concluído. Também foi desenvolvido um artigo que descreve a ferramenta, tendo sido obtido sua aceitação na sessão de ferramentas do maior congresso nacional na área de Engenharia de *Software*, o Congresso Brasileiro de *Software* – CBSoft 2011, que deverá acontecer no final de setembro. Nossa ferramenta concorrerá ao prêmio de melhor ferramenta para apoio à Engenharia de Software de 2011.

Resultados e Discussão

Pelo fato de *software*s sofrerem alterações durante todo ciclo de vida, sempre é necessário atualizá-lo. Isso também acontece para os testes, que devem ser atualizados juntamente com o software, para que ele reflita as novas regras desenvolvidos.. Como os testes possuem grande quantidade de informações a respeito do comportamente de um software, precebeu-se que eles poderiam ser utilizados para se inferir as alterações realizadas em diferentes versões do *software*. Baseado nesta constatação criou-se a ferramenta TChangeReport, que utiliza testes funcionais para gerar relatórios de mudanças.

A ferramenta TChangeReport foi desenvolvida utilizando a tecnnologia J2SE[1], e está sob a licença GNU (*General Public Licence*). A ferramenta foi construida a partir do arcabouço TWork (*Test extractor based frameWORK*), que é um arcabouço que auxilia na criação de artefatos, a partir do reuso de testes funcionais.

Para geração dos relatórios a partir da ferramenta foram criados padrões e convenções para o uso da ferramenta, que basicamente descrevem como deve ser a organização dos testes em um projeto.

A Figura 1 exibe o fluxo de criação do relátorio de mudanças com o auxilio da TChangeReport. O primeiro passo no fluxo é a criação dos testes a partir do uso de uma ferramenta de testes funcionais, seguido de sua organização baseando-se nos padrões e convenções pré-estabelecidos. No passo 3 deve-se entrar com informações que não podem ser inferidas diretamente pelos testes funcionais. No passo 4 a ferramenta gera o relatório em formato formato *Rich Text Format* (RTF) e, se necessário, esse pode ser ajustado manualmente, pois o documento pode conter algumas palavras ou frases não claras ou não objetivas, que dificultam o entendimento pelo usuário final.

A ferramenta possui 4 componentes principais: *ModelBuilder*, *SystemFeature*, *FeatureChange* e *ReportBuilder.* A *ModelBuilder* recebe um *TestSuite* e cria, para cada procedimento de teste contido nesse modelo, uma entidade *SystemFeature*, que representa uma funcionalidade (ou caso de uso) de uma aplicação. A *FeatureChange* recebe as mudanças entre as *SystemFeatures* relatando quais

[1] *Java 2 Standard Edition*

campos sofreram alterações. A classe *ReportBuilder* recebe as *SystemFeature* adicionadas e removidas, as *FeatureChange* e gera os textos que iram compor o relatório.

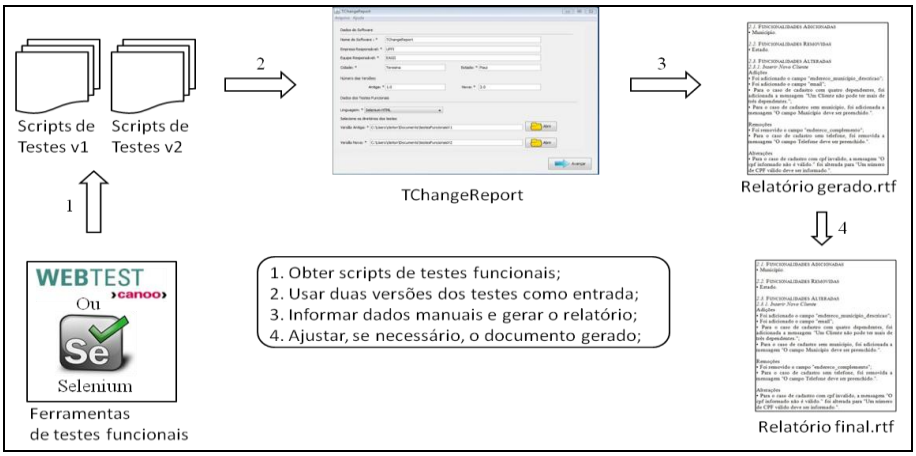

Figura 1: Fluxo de criação do Relatório de Alteração com a TChangeReport.

Com intuito de avaliar a ferramenta criada, foi realizado um estudo experimental com o intuito de avaliar o tempo gasto para criação do relatório de alteração com e sem o uso da ferramenta, além de avaliar a qualidade do trabalho realizado. Os resultados obtidos indicam que o uso da ferramenta gerou um ganho significativo em termo de tempo, gerando uma redução média de 45% do tempo gasto, com uma qualidade superior, uma vez que todos os relatórios gerados pela ferramenta foram aprovados por um membro externo ao estudo e apenas 60% dos relatórios gerados sem o apoio da ferramenta foram aprovados.

Conclusões

Neste trabalho descrevemos uma ferramenta para automação de relatórios de alteração, a TChangeReport. Essa ferramenta é inovadora, pois utiliza os testes funcionais para sua execução, algo não encontrado nos trabalhos relacionados. Após um estudo experimental sobre o uso da ferramenta em um ambiente controlado, verificamos que sua utilização pode reduzir o esforço na criação do relatório de alteração e ainda melhorar a qualidade do documento gerado.

Como trabalhos futuros, estamos planejando outro estudo experimental, mas em um ambiente industrial. Pretende-se também aperfeiçoar a ferramenta, tornando-a uma aplicação Web, buscando aumentar a flexibilidade e sua escalabilidade. Além disso, devemos iniciar a realização de estudos para incrementar os resultados obtidos com seu uso, a partir da introdução de técnicas de processamento de linguagem natural e inteligência computacional.

Referências Bibliográficas

AZIZ, J. AHMED, F.; LAGHARI, M. S. Empirical Analysis of Team and Application Size on Software Maintenance and Support Activities. **2009 International Conference on Information Management and Engineering**, p. 47-51. Ieee. doi: 10.1109/ICIME.2009.51, 2009.

IEEE. IEEE Standard Glossary of Software Engineering Terminology. **IEEE Std 610.12-1990**, p. 1. doi: 10.1109/IEEESTD.1990.101064, 1990.

IEEE. IEEE Standard for Software Test Documentation. **IEEE Std 829-1998**, p. i. doi: 10.1109IEEESTD.1998.88820, 1998.

Palavras-chave: Teste de *Software*. Relatório de Mudanças. Geração de Documentos