Manutenção Adaptativa de Software Embarcado para Telefones Celulares Apoiado por Ferramentas de Automação

Francisco F. P. Lima, Tales P. Nogueira, Antonia D. B. Nogueira, José S. R. Neto, João B. F. Filho, Claudio R. F. Lima, Smaylle J. C. Leite, Windson Viana, Valéria L. L. Dantas, Rossana M. C. Andrade

Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) – Universidade Federal do Ceará (UFC)
Campus do Pici s/n Bloco 942-A – 60.455-900 – Fortaleza – Ceará – Brazil

{fcofabricio, tales, diana, netoreboucas, bosco, claudio, smaylle, windson, valerialelli, rossana}@great.ufc.br

Abstract. *It is well known that the development of software supported by CASE tools improves product quality assurance. In this paper, we describe our experience in the use of automation tools during the process of developing software for embedded systems. They are called: FlexCA, which is designed to provide flexibility in information processing, and Auto Sanity Tool, which is designed for testing. These tools are used in the stage of adaptive maintenance and testing. We also show significant results regarding the increase of production speed of the generated software as well as the decrease in the amount of errors with the use of these tools.*

1. Introdução

Na busca contínua de uma melhoria da qualidade dos seus produtos, as empresas de software necessitam, primeiramente, aperfeiçoar o processo de desenvolvimento. Entre os parâmetros que podem ser utilizados para se medir a qualidade de um processo está a produtividade que pode ser definida pela relação entre quantidade de trabalho realizado e os resultados obtidos. Outro parâmetro importante é a qualidade, a qual assegura que o produto final atende os requisitos necessários de forma confiável e eficiente [9].

Na ultima década, comprovou-se a efetividade do emprego de ferramentas CASE (*Computer Aided Software Development*) [3] ao longo do processo de software com o intuito de melhorar atributos de qualidade do produto. Com essas ferramentas, pode-se diminuir a quantidade de defeitos e propiciar uma estrutura de manutenção mais simples e eficiente.

Com a crescente necessidade de entregas de produtos com mais qualidade e em prazos continuamente menores devido à alta competitividade no mercado, é essencial a concepção de novas ferramentas que incrementem a velocidade do processo de geração de versões de software mantendo bons níveis de qualidade.

Um exemplo de cenário com necessidade de geração de versões são os softwares embarcados em dispositivos móveis, principalmente telefones celulares [4][7]. A grande diversidade de novos modelos e requisitos específicos para países, regiões e operadoras exigem continuamente a geração de novas versões dos softwares embarcados que precisam ser produzidas rapidamente e com alta qualidade.

De acordo com o padrão IEEE 1219 [10], manutenção de software é a modificação de um produto de software após sua entrega com o objetivo de corrigir falhas, melhorar o desempenho ou outros atributos ou adaptar o produto a um ambiente modificado. Adicionalmente, de acordo com o padrão Software Engineering – Maintenance of Software ISO/IEC 14764 [8], manutenção adaptativa é a modificação de software realizada depois de sua entrega com o objetivo de manter o sistema funcionando corretamente em um ambiente ou plataforma distinta [1] podendo considerar restrições de hardware ou sistema operacional [9] e em alguns casos, mudanças de arquitetura podem ser consideradas [6]. No caso deste trabalho, outras mudanças no ambiente estão envolvidas e também são importantes para a aceitação do software. A principal delas é a localização geográfica do dispositivo móvel contendo o software. A geração de versões específicas de um software para países, regiões e operadoras citada anteriormente é, portanto, um exemplo de manutenção adaptativa.

Este artigo relata a experiência da aplicação de duas ferramentas de auxílio à testes ao longo do ciclo de vida de produção de sistemas embarcados para celulares. A primeira ferramenta, chamada de FlexCA, permite desenvolver atividades de manutenção adaptativa e a segunda, chamada de Auto Sanity, é empregada na fase de testes funcionais e de regressão [2], também chamados, no escopo deste trabalho, de testes de sanidade.

O restante deste artigo está organizado da seguinte forma: a Seção 2 descreve as duas ferramentas para manutenção adaptativa e automatização de testes para software embarcado para telefones celulares. Na Seção 3, os resultados obtidos com o uso das ferramentas são relatados. Finalmente, a Seção 4 contém as conclusões e os trabalhos futuros.

2. Manutenção adaptativa de software embarcado apoiado pelas ferramentas propostas

As ferramentas aqui apresentadas foram projetadas e implementadas para auxiliar a execução do processo de *Country Adaptation* (CA), que consiste em realizar alterações no código-fonte de aparelhos celulares (e.g., animação mostrada durante a inicialização e desligamento, papel de parede, informações de conexão de dados) de acordo com o que é pedido pelas operadoras de telefonia móvel. Por exemplo, uma determinada operadora de telefonia pode se interessar em ter sua logomarca configurada como papel de parede padrão quando o celular for vendido. Além disso, a operadora pode exigir que o dispositivo seja vendido com todos os perfis de conectividade, e.g. WAP, GPRS, devidamente configurados.

Essa adaptação é executada através da modificação do código que é posteriormente embarcado no dispositivo móvel. Esse processo pode ser realizado de várias formas dependendo da tecnologia utilizada pelo equipamento. Estas adaptações (novas versões de software) para celulares são lançadas frequentemente, destinadas principalmente a atender especificações requeridas por operadoras para determinados modelos. Dentre essas versões, algumas são rejeitadas, pois as funções básicas não funcionam corretamente ou devido à ocorrência de outros tipos de erros e falhas. Como forma de evitar esses problemas, um conjunto de ferramentas de apoio para a manutenção adaptativa e a automatização de testes foi desenvolvido.

A Figura 1 ilustra o processo de CA, que é iniciado a partir de uma versão do software a ser embarcado no dispositivo e os dados da PRI (*Product Release Instructions*) que contêm as preferências de uma operadora. De posse desses dois elementos, é possível automatizar o processo de geração de uma nova versão através da ferramenta FlexCA.

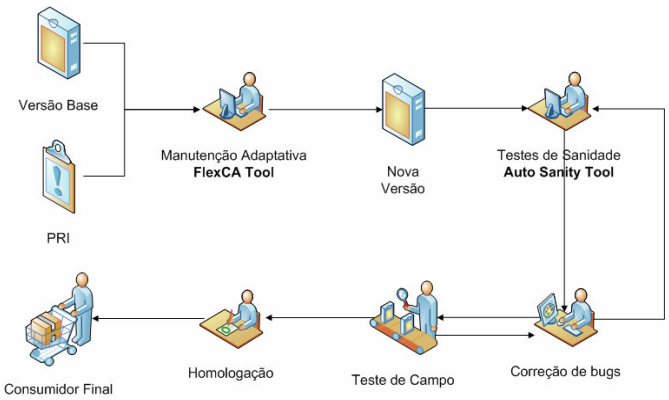

Figura 1. Processo de manutenção adaptativa de software embarcado apoiado por ferramentas de automação

A nova versão passa pelos testes de sanidade aplicados pela ferramenta Auto Sanity. Com os resultados dos testes de sanidade, eventuais falhas são identificadas, corrigidas e, novamente, o software passa pelos testes de sanidade até que todos os testes sejam realizados sem identificar nenhuma falha. Só então a versão pode ser enviada para testes de campo.

Nos testes de campo, onde o dispositivo móvel é testado em situações reais de uso, novas falhas podem ser identificadas, voltando o ciclo para a correção de bugs. Quando o software passa com sucesso em todos os testes de campo, ele pode ser homologado e liberado para ser embarcado nos dispositivos móveis a serem vendidos.

Vale ainda ressaltar que a ferramenta FlexCA é capaz de estar continuamente atualizada com o constante desenvolvimento de tecnologias para celulares devido à sua flexibilidade, enquanto que a Auto Sanity possui uma interface de fácil configuração para suportar novos modelos de aparelhos celulares para testes.

Estas ferramentas foram realizadas em um projeto de parceria entre uma empresa fabricante de celulares e o Departamento de Ciência da Computação da Universidade Federal do Ceará[1]. Uma explicação mais aprofundada sobre o funcionamento das ferramentas é dada nas subseções seguintes.

2.1. FlexCA

A Figura 2 mostra a arquitetura básica da ferramenta. O processo de CA necessita da presença de uma fonte de dados onde as configurações da operadora estejam especificadas. Essa fonte de dados, chamada de PRI, pode ser um arquivo ou um banco de dados relacional. As partes do software que são modificadas dependem da tecnologia do modelo. Para algumas tecnologias é necessário alterar somente arquivos binários; para outras, algumas configurações podem ser colocadas em arquivos INI, XML ou SQL.

[1] O projeto foi patrocinado por um fabricante de celulares sob o incentivo da Lei de Informática nº 8248/91

A lógica de manutenção é implementada em *scripts* que se baseiam em um *template* pré-determinado. Esses *scripts* obtêm os dados da fonte de PRI e alteram os arquivos necessários. A Figura 3 mostra o editor de *scripts* na ferramenta. No início, um conjunto de *scripts* é implementado para cada tecnologia e reutilizado nos processos de CA para cada modelo de aparelho celular daquela tecnologia. Essa implementação inicial demanda um tempo considerável comparado com um processo de CA para um modelo de forma isolada. Entretanto, com a possibilidade de reutilização de *scripts* em outras CAs e modelos, esse tempo é justificável.

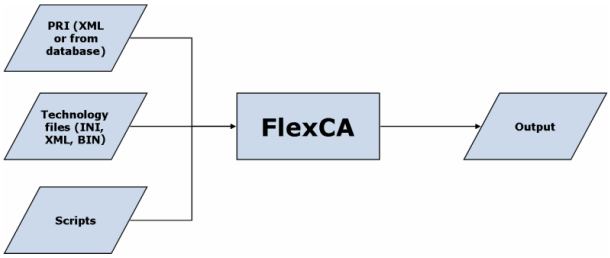

Figura 2. Arquitetura da ferramenta FlexCA

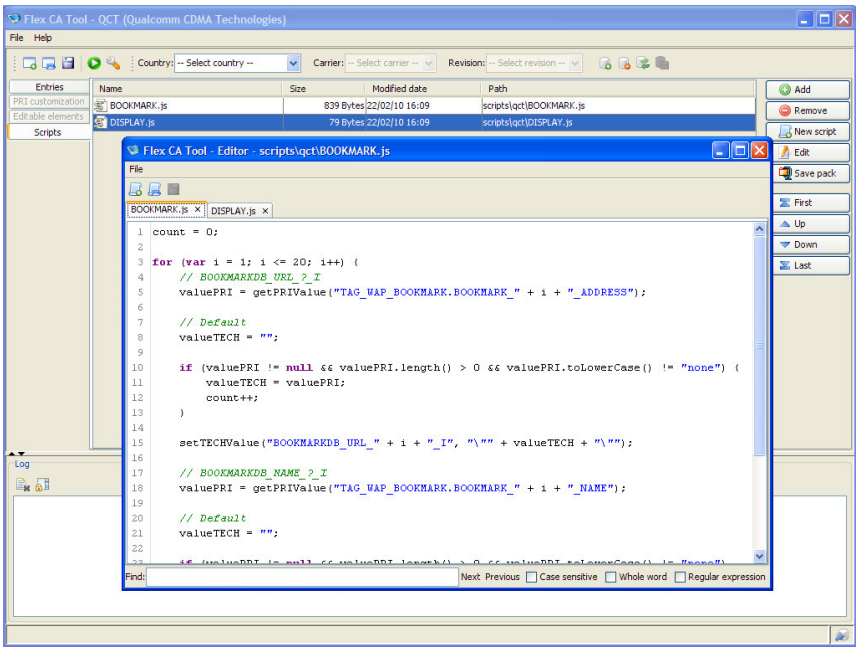

Figura 3. Edição de um script na FlexCA

2.2. Auto Sanity

Depois de realizado o processo de CA, as versões são verificadas e validadas pela equipe de desenvolvimento antes de serem testadas por uma equipe especializada em campo. Esses testes realizados pela equipe de desenvolvimento podem ser vistos como testes de unidade para uma CA e como testes de regressão, para verificar se durante o processo de CA nenhuma parte importante do software foi danificada. A Auto Sanity é a ferramenta que proporciona a automatização desses testes através da manipulação (criação e edição) de *scripts* de casos de teste, da execução desses *scripts* e da geração de relatórios de resultado dos testes das versões do software geradas pela FlexCA.

Assim, a Auto Sanity proporciona resultados confiáveis, livres de erros humanos e que utilizam poucos recursos para sua execução, como tempo e pessoas envolvidas.

Assim como a FlexCA, a Auto Sanity se baseia em *scripts*. Esses *scripts* são divididos em três categorias: *Commands*, *Test Cases* e *Test Sets*.

Os *Commands* constituem-se na parte principal da ferramenta. É através deles que a lógica dos testes é executada e o resultado é obtido. Cada *Command* é implementado utilizando um *script* escrito na linguagem Boo [5] que obedece a um determinado padrão. A Figura 4 mostra um exemplo de um Command, chamado *Navigate*. O método *Run* é o ponto de entrada para o *Command* e, para a sua execução, são passados dois conjuntos de objetos: um conjunto formado pelos módulos de serviços, que são fornecidos pelo núcleo da ferramenta e executam tarefas específicas para a realização de testes, e o conjunto de parâmetros do *Command*, que são especificados no *script* de teste que o utiliza.

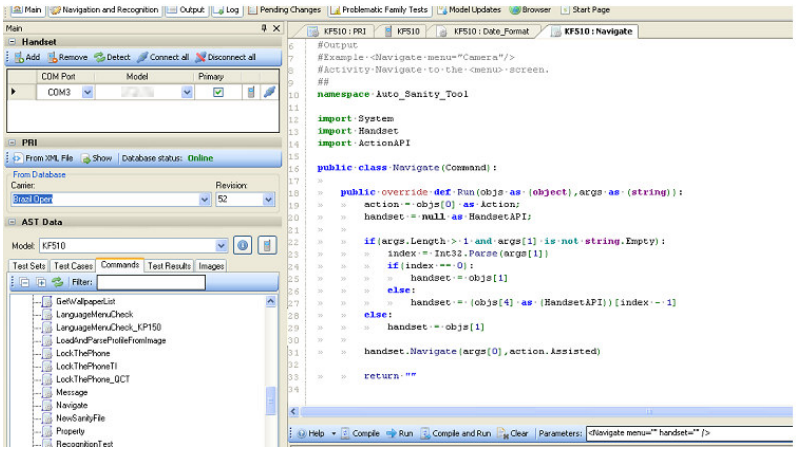

Figura 4. Edição de um Command

Os *Test Cases* são formados por ações que representam os passos necessários para a execução de um teste. Tais ações são representadas pelos *Commands*. Já os *Test Sets* são conjuntos de *Test Cases* que representam o conjunto de testes necessários para um determinado modelo de telefone. As Figuras 5 e 6 mostram exemplos desses *scripts*.

Assim como a FlexCA, um conjunto de *scripts* tem que ser implementado no início do desenvolvimento de versões do software para cada modelo. Entretanto, na Auto Sanity, o nível de reutilização é maior, visto que os casos de teste sofrem poucas variações para diversas tecnologias e modelos. Porém, ocasionalmente, faz-se necessário alterar algumas configurações para atender características de certos modelos, e.g. as dimensões da tela.

Figura 5. Execução de um Test Set

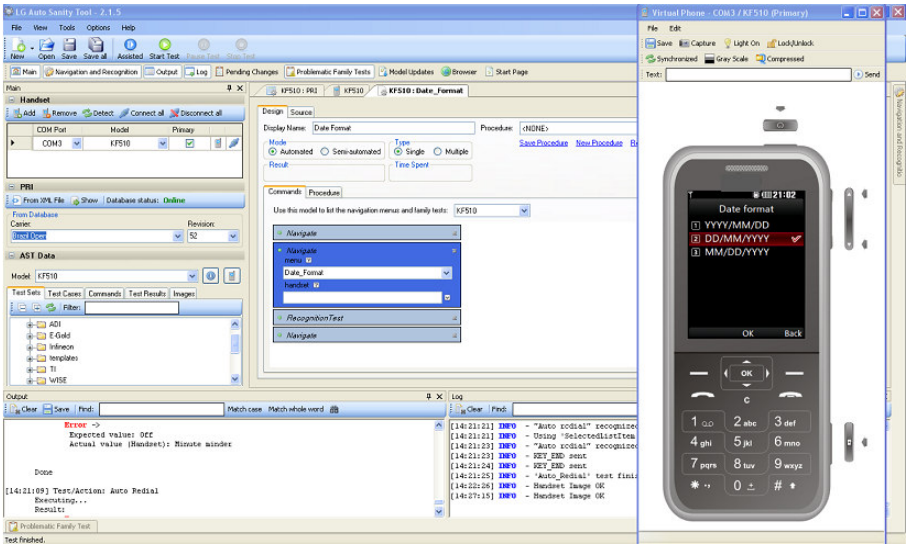

Figura 6. Criação de um Test Case

3. Testes e Resultados

As ferramentas foram testadas em uma empresa fabricante de equipamentos celulares atuante no mercado brasileiro. Para tanto, foi elaborado e executado um plano de testes com o objetivo de verificar o funcionamento correto das ferramentas de acordo com os requisitos das tecnologias testadas. Nessa medição, foram usados nove modelos de aparelhos celulares de três tecnologias distintas. O tempo gasto durante os testes foi relatado pelos próprios desenvolvedores e não foi considerado o tempo de criação dos *scripts* iniciais, visto que alguns desses *scripts* já são incorporados nas ferramentas. Uma melhor avaliação das ferramentas terá que ser feita posteriormente à medida que o uso delas for mais abrangente e os usuários ficarem mais acostumados com a criação e edição de *scripts*.

Como resultado da aplicação da FlexCA e da Auto Sanity no processo de manutenção adaptativa de software, ganhos significativos foram alcançados, tanto em tempo de execução quanto na quantidade de erros reportados. A Figura 7 apresenta os resultados da utilização da FlexCA no processo de CA, onde foi realizada manutenção adaptativa de acordo com a tecnologia de celular empregada.

Para uma determinada tecnologia, o tempo gasto na execução do processo de CA para uma versão do software era de 60 minutos. Após a adoção da ferramenta, esse tempo caiu para cerca de 15 minutos, resultando em um ganho de 75%. Além disso, o número de erros relatados devido a erros humanos durante o processo que era de cinco erros por versão em média, caiu para zero.

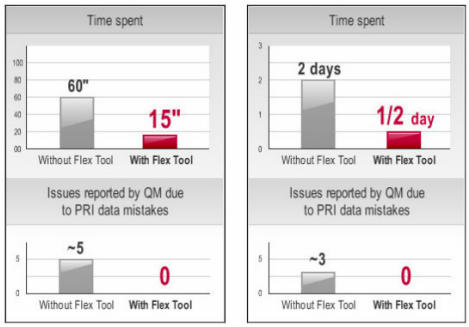

Figura 7. Gráficos comparativos dos resultados alcançados com a aplicação da ferramenta FlexCA em duas plataformas diferentes

Ao utilizar outra tecnologia, o tempo gasto para liberar uma versão do software era de dois dias antes da adoção da FlexCA. Ao usar a ferramenta, este tempo diminuiu em quatro vezes, resultando na metade de um dia para a execução de todo o processo de CA. O número de erros reportados para esta tecnologia também caiu para zero. Antes do uso da ferramenta, três ocorrências de erro, em média, eram relatadas.

Na Figura 8, podem ser vistos os resultados da aplicação da ferramenta Auto Sanity no processo de testes de sanidade. O processo de testes foi realizado com três tecnologias diferentes e por um testador com conhecimentos básicos sobre a ferramenta e as plataformas. Como pode ser notado nos gráficos, o tempo médio gasto para realizar os testes era de 90 minutos antes da implantação da ferramenta. Depois que a Auto Sanity foi adotada, esse tempo caiu para, em média, 24 minutos. Com relação à quantidade de erros relatados pela equipe de gerenciamento da qualidade, que em média era de dois erros por versão, esse número caiu para zero com a automatização dos testes através da Auto Sanity.

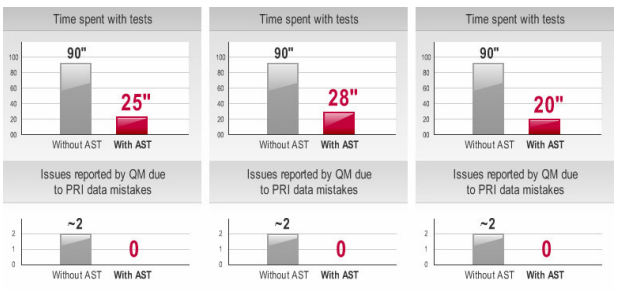

Figura 8. Gráficos comparativos dos resultados alcançados com a aplicação da ferramenta Auto Sanity em três tecnologias diferentes

4. Considerações Finais

Este trabalho apresentou as características e a avaliação de duas ferramentas para auxiliar o processo de manutenção adaptativa de software embarcado, chamadas FlexCA e Auto Sanity. As ferramentas são empregadas em momentos diferentes do ciclo de desenvolvimento. A primeira automatiza o processo de manutenção adaptativa e tem uma arquitetura bastante flexível para que possa acompanhar a evolução das tecnologias de celulares. A segunda auxilia a realização de testes, onde as funções básicas do software são testadas com o objetivo de identificar falhas antes dos testes de campo.

Com os resultados alcançados, nós demonstramos que as ferramentas em questão foram responsáveis por um significativo aumento na qualidade dos sistemas desenvolvidos através da medição de indicadores como o tempo gasto de produção, que diminuiu drasticamente, e a quantidade de falhas humanas no processo que caiu para zero.

Como sugestão para trabalhos futuros, com relação à FlexCA, melhorias com relação à arquitetura da ferramenta serão realizadas a fim de torná-la ainda mais flexível e de fácil utilização. Entre essas melhorias, está incluído o gerenciamento de vários bancos de dados e a utilização de uma base de dados interna para manter preferências dos usuários. Para a Auto Sanity, os trabalhos futuros incluem melhorias na estabilidade e usabilidade, integração com equipamentos de teste de rede e a funcionalidade de testar telefones celulares remotamente.

Referências

[1] Abran, J. Moore, P. Bourque, R. Dupuis, and L. Tripp. Guide to the Software Engineering Body of Knowledge - SWEBOK, IEEE-Computer Society Press: www.swebok.org, Los Alamitos, USA (2001)

[2] Beizer, B. Software Testing Techniques. Van Nostrand Reinhold, New York, 2nd. ed. 1990. (1990)

[3] Carnegie Mellon University. Computer Aided Software Engineering (CASE) Environments. [Online]. Available: http://www.sei.cmu.edu/legacy/case/case_whatis.html. (2007)

[4] Carvalho, W. V. and Andrade, R. M. C. “XMobile: a MB-UID Environment for Semi-Automatic Generation of Adaptive Applications for Mobile Devices”. Journal of Systems and Software, pp 382-394, (2007)

[5] Codehaus Foundation. [Online] Available: http://boo.codehaus.org/. (2009)

[6] Harrison, W. Using service specific proxies to migrate web services to the "Wireless web": an illustrative case study of adaptive maintenance. International Conference on Software Maintenance, pp. 300-309. (2002)

[7] Rocha, L. S., Castro, C. E. P. L., Machado, J. e Andrade, R. M. C. Utilizando Reconfiguração Dinâmica e Notificação de Contextos para o Desenvolvimento de Software Ubíquo. Anais do Simpósio Brasileiro de Engenharia de Software, pp. 219 - 235, (2007)

[8] Software Engineering - Software Maintenance. ISO/IEC Std. 14764. (2000)

[9] Sommerville, I. Software Engineering. 8th ed., Addison-Wesley Pub. Lim. (2006)

[10] Standard for Software Maintenance. IEEE Std. 1219. (1998)