Avaliação da Qualidade de Produtos J2ME por Meio do Uso de Pacotes de Experimentação♦

Gilcimar Divino de Deus[1], **Auri Marcelo Rizzo Vincenzi**[1], **Fábio Nogueira de Lucena**[1], **Márcio Eduardo Delamaro**[2]

[1]Instituto de Informática – Universidade Federal de Goiás
Caixa Postal 131 – 74001-970 – Goiânia – GO – Brazil

gyngil@gmail.com, auri@inf.ufg.br, fabio@inf.ufg.br

[2]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
Caixa Postal 668 – 13560-970 – São Carlos – SP – Brazil

delamaro@icmc.usp.br

Abstract. The software quality for mobile devices is a concern due to the increasing number of devices available. There are several techniques to evaluate the software quality, and most of them can be applied for testing J2ME applications. An experimentation package aiming at assessing some of the most popular techniques for J2ME software products was proposed. This article presents the results obtained after the second replication of the experimentation package and compares the data with the one obtained from the first replication. The controlled replication of the experiment, provided by the use of the package, allow us to constant update the knowledge database with data from new subjects, increasing the significance of the obtained results.

Resumo. A qualidade de software para dispositivos móveis é cada vez mais preocupante devido ao número crescente de dispositivos disponíveis. Existem diversas técnicas para se avaliar a qualidade de software, e para produtos de software J2ME não é diferente. Um pacote de experimentação que visa avaliar algumas das mais conhecidas técnicas de teste de software foi proposto. O presente artigo apresenta os resultados obtidos após a segunda replicação desse pacote e compara os dados com aqueles da primeira replicação. As replicações desse pacote controlado permitem a atualização constante de uma base de informações consistentes com dados de novos indivíduos, aumentando a confiança nos resultados obtidos.

1 Introdução

No Brasil existem hoje cerca de 145 milhões de celulares [ANATEL 2008]. O poder de processamento, velocidade de transmissão e outras características tecnológicas proporcionam o tratamento de informação por sistemas em dispositivos móveis. É muito importante que projetos sejam desenvolvidos e focados em aprimorar as estratégias de teste e aplicá-las no ambiente móvel.

♦ Trabalho desenvolvido com o apoio do CNPq (processos 478001/2004-5 e 309503/2006-0).

Um dos problemas encontrados na área de dispositivos móveis é a dificuldade de executar os testes dos aplicativos no próprio dispositivo (ambiente real). As fases de desenvolvimento e testes normalmente ocorrem em computadores *desktop*, com o uso de emuladores. É extremamente importante que os aplicativos sejam testados em seu ambiente real, pois devido suas limitações de memória e processamento, erros podem acontecer e serem camuflados pelos emuladores. A ferramenta JaBUTi/ME foi desenvolvida dentro desse contexto e apóia o teste de programas J2ME tanto em emuladores quanto em dispositivos reais [Delamaro et al. 2006].

Este artigo apresenta os resultados coletados após a segunda replicação do pacote de experimentação criado por Deus et al. (2008) e busca analisar e comparar os dados consolidados com os da primeira replicação. Um pacote de experimentação consiste de uma maneira sistemática e controlada de realizar experimentos em várias etapas, permitindo atingir uma quantidade de dados com significância estatística de maneira incremental. Além disso, a disponibilidade de um pacote de experimentação viabiliza que o mesmo estudo possa ser realizado por diferentes pessoas, em locais diferentes e, permite uma comparação desses dados ao longo do tempo. No caso deste artigo, o objetivo é apresentar a base de informações coletada até o momento sobre as três técnicas de teste de software investigadas no pacote de experimentação (*ad hoc*, funcional e estrutural) desenvolvido e a adequação da ferramenta JaBUTi/ME no apóio ao teste de produtos J2ME.

O artigo está organizado com se segue. Na Seção 2 é apresentada uma breve descrição sobre a atividade de teste de software. A Seção 3 traz um resumo da ferramenta JaBUTi/ME, juntamente com os critérios de teste apoiados pela mesma. O pacote de experimentação utilizado nessa segunda replicação é detalhado na Seção 4. Na Seção 5, os dados da segunda replicação são analisados e confrontados com os da primeira. Os trabalhos relacionados a esse assunto são descritos na Seção 6, e finalmente, na Seção 7 é apresentada a conclusão desta segunda replicação juntamente com indicações de trabalhos futuros.

2 Teste de Software

O atividade de teste de software é dividida em quatro etapas, planejamento dos testes, projeto dos casos de testes, execução dos testes e análise dos resultados [Delamaro et al. 2007]. Essas etapas normalmente são distribuídas no processo de desenvolvimento de software em três fases, unidade, integração e de sistema, de modo que a complexidade dos testes possa ser minimizada.

Algumas técnicas são adotadas para ajudar a revelar defeitos em programas de software, as mais conhecidas são *ad-hoc*, funcional, estrutural e baseada em defeitos. Os testes *ad hoc*, também conhecidos como testes aleatórios, utilizam a experiência do testador como indicador de qualidade para o teste, ou seja, quando o testador acreditar que o programa testado está livre de defeitos, ele pára de testá-lo, o requisito de teste é a satisfação do próprio testador. Outra forma de executar o teste *ad hoc* é por meio de algoritmos de geração aleatória de dados de teste.

Já na técnica funcional o programa é testado com a visão do usuário, no qual o componente testado é considerado uma caixa preta, da qual não se conhecem os detalhes de implementação, são fornecidas entradas e avaliados os resultados produzidos. O domínio de teste pode ser muito grande ou infinito e uma das maneiras de

tornar o teste viável é a utilização de critérios funcionais como, análise de valor limite e/ou particionamento de equivalência. No primeiro, são testadas as entradas que representam as fronteiras do componente testado, ou seja, os limites superiores e inferiores. No segundo, as entradas são particionadas de tal forma que todas as entradas sejam representadas de alguma maneira nos subconjuntos ou partições [Beizer 1995, Myers 2004].

A técnica estrutural, também conhecida como teste caixa-branca, estabelece as entradas de teste (casos de testes) baseadas na implementação do componente e auxilia na detecção de defeitos de lógica e programação. Os critérios dessa técnica buscam exercitar partes do código e não da especificação. Condições, laços, definições e usos de variáveis são itens exercitados por tais critérios de teste os quais, em geral, derivam os requisitos de teste de uma representação do programa conhecida como grafo de fluxo de controle ou grafo do programa. Os critérios mais conhecidos do teste estrutural são os baseados em fluxo de controle e baseados em fluxo de dados [Myers 2004].

Nesse trabalho serão abordadas as técnicas, *ad hoc*, funcional (focando principalmente na análise de valor limite e particionamento de equivalência), e os critérios da técnica estrutural apoiados pela JaBUTi/ME (principalmente Todos-Nós e Todas-Arestas), descritos brevemente a seguir.

3 JaBUTi/ME e Dispositivos Móveis

A atividade de teste, sem a existência de uma ferramenta, aumenta a chance de enganos serem cometidos pelo testador, além de não proporcionar produtividade na execução dos testes e análise dos resultados. Diversas ferramentas foram disponibilizadas, sendo que cada uma apóia a utilização de um ou mais critérios de teste. A ferramenta JaBUTi (Java Bytecode Understanting Testing) [Vincenzi et al. 2003] é uma delas e explora alguns critérios de cobertura de código (técnica estrutural), aplicados na estrutura do programa, os quais auxiliam tanto na criação de casos de teste que exercitem trechos do programa ainda não executados, quando na avaliação da qualidade de conjuntos de teste já existentes, considerando os critérios apoiados.

Dentre os diversos recursos oferecidos pela JaBUTi, um dos mais importantes é o apoio na cobertura de programas Java com base no código objeto, ou seja, a JaBUTi realiza todas as computações para o teste de programas Java diretamente no *bytecode*. Além da visualização do *bytecode*, a ferramenta oferece ainda a visualização do código fonte quando o mesmo encontra-se disponível.

Uma das plataformas exploradas pela nova versão da JaBUTi, é voltada para dispositivos móveis Java, também conhecida como J2ME. Em sua nova versão a ferramenta é conhecida como JaBUTi/ME (Java Bytecode Understanting Testing/Micro-Edition) [Delamaro et al. 2006]. Que explora os mesmos recursos da versão inicial e complementa com recursos que apóiam o teste de programas nos dispositivos móveis reais ou em emuladores, como o Wireless Toolkit da Sun Microsystems [Microsystems 2009]. Dentre alguns dos recursos configuráveis dessa versão estão os diferentes mecanismos de instrumentação de código oferecido, viabilizando que a aplicação real se comunique com o servidor de teste dependendo das restrições de conectividade e memória impostas pelos diferentes tipos de dispositivos móveis, conforme ilustrado na Figura 1.

A instrumentação do programa é uma atividade essencial para a aplicação do teste estrutural. É através dela que é possível capturar as informações de cobertura durante a execução dos testes. No caso da JaBUTi/ME, instrumentar o código corresponde a inclusão de uma chamada a um método especial em pontos específicos do bytecode. Esse método é responsável por identificar e armazenar a informação sobre qual trecho de código foi executado, enviando posteriormente essas informações para o servidor de testes.

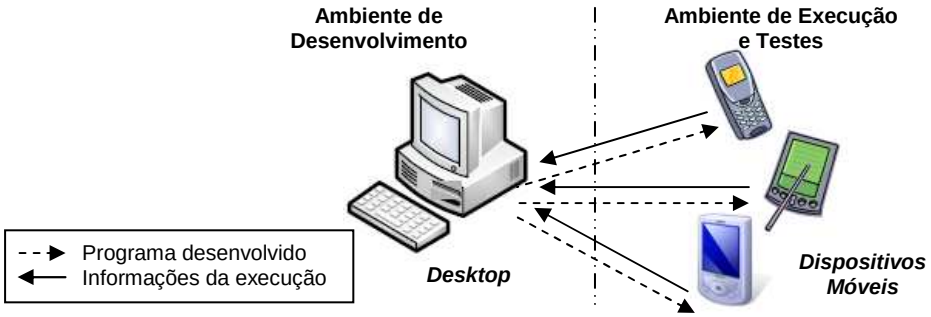


Figura 1 – Ambiente de desenvolvimento cruzado.

A ferramenta JaBUTi/ME foi desenvolvida no contexto de um projeto de pesquisa denominado "Utilização de Teste Estrutural para Programas Java em Dispositivos Móveis", que teve o apoio do CNPq (478001/2004-5). Uma vez desenvolvida a ferramenta o objetivo é a realização de uma série de estudos experimentais visando comprovar sua utilidade prática no teste de programas para essa área de aplicação. Nesse contexto, o foco desse trabalho se volta para a execução de testes em dispositivos móveis utilizando a JaBUTi/ME, fazendo uso de pacotes de experimentação.

4 Pacotes de Experimentação

Um pacote de experimentação permite que os experimentos sejam conduzidos de forma sistemática, viabilizando sua replicação. A cada replicação de um determinado pacote de experimentação, novos dados são coletados para uma nova amostra de participantes, melhorando a representatividade estatística da base de dados para o pacote em questão.

A intenção desta segunda replicação do pacote de experimentação é aumentar o grau de confiança dos dados coletados na primeira replicação [Deus et al. 2008]. Nesta seção é descrito como o segundo estudo experimental utilizando a ferramenta JaBUTi/ME foi conduzido, a fim de avaliar as técnicas *ad hoc*, funcional e estrutural e a adequação das mesmas no teste de aplicativos desenvolvidos para dispositivos móveis, que obedecem à especificação J2ME, além de avaliar os benefícios que os critérios apoiados pela ferramenta oferecem ao testador no desenvolvimento de casos de teste.

O objetivo final dos estudos é contribuir para o desenvolvimento de uma estratégia de teste incremental (com o apoio de uma ou mais ferramentas de teste) que possa ser empregada para garantir a qualidade de produtos de software e sistemas de informação que utilizem dispositivos móveis. Considera-se que com a crescente demanda por produtos de software para tais dispositivos, os resultados obtidos com esses estudos podem contribuir de maneira significativa na avaliação das técnicas de testes e melhoria da qualidade dos produtos finais J2ME utilizados em todo mundo.

4.1 Pacote de Experimentação para a JaBUTi/ME

Seguindo o processo de estudo experimental de Wohlin et al. (2000), este experimento está organizado da seguinte forma:

- **Definição:** Teste Estrutural de Software J2ME em Dispositivos Móveis Utilizando a JaBUTi/ME.
- **Contexto:** Este experimento faz parte da Engenharia de Software, especificamente da disciplina de Teste de Software. Será utilizada uma ferramenta específica, JaBUTi/ME, criada para teste estrutural de programas Java e adaptada para o contexto de dispositivos móveis.
- **Hipóteses:** As seguintes hipóteses poderão ser validadas ou não após a realização do experimento.
    - *Hipóteses nulas:*

      $H_{0,1}$ - A técnica estrutural com a ferramenta JaBUTi/ME revelou a mesma quantidade de defeitos que a técnica *ad hoc* ou que a técnica funcional;

      $H_{0,2}$ - A técnica estrutural com a ferramenta JaBUTi/ME apresentou percentual equivalente de cobertura da técnica *ad hoc* ou funcional;

      $H_{0,3}$ - A técnica estrutural com a ferramenta JaBUTi/ME não contribuiu na criação de novos casos de teste.
    - *Hipóteses alternativas:*

      $H_{1,1}$ - A técnica estrutural com a ferramenta JaBUTi/ME revelou quantidade de defeitos superior que a técnica *ad hoc* e/ou que a técnica funcional;

      $H_{1,2}$ - A técnica estrutural com a ferramenta JaBUTi/ME apresentou maior percentual de cobertura da técnica estrutural em relação à *ad hoc* e/ou funcional;

      $H_{1,3}$ - A técnica estrutural com a ferramenta JaBUTi/ME contribuiu na criação de novos casos de teste, não identificados antes pelas técnicas *ad hoc* e/ou funcional.
- **Variáveis dependentes:**
    - *Complexidade dos programas;*
    - *Quantidade de defeitos revelados;*
    - *Percentual de cobertura;*
    - *Quantidade de casos de testes novos;*
- **Variáveis independentes:**
    - *Técnica ad hoc;*
    - *Técnica funcional;*
    - *Técnica estrutural;*
    - *Programas selecionados;*
- **Participantes:** 20 pessoas (estudantes e/ou professores de graduação ou pós-graduação) com conhecimento de informática e programação Java serão utilizados. Todos os estudantes responderão um questionário para traçar o perfil dos participantes para fins de análise e interpretação. O único pré-requisito para participar do experimento será o participante ter no mínimo noções de programação em linguagem Java, reconhecendo comandos, estruturas condicionais, laços e etc. Não será exigida experiência em teste de software.

- **Projeto Experimental:** Foram selecionados quatro programas J2ME para a realização do experimento. Será utilizada a técnica *fatorial-fracional randomizado*[1] [Felizardo 2003] para a criação de grupos e distribuição dos programas de modo que cada indivíduo aplica determinada técnica de teste em um programa diferente. Três programas serão utilizados pelos participantes na execução do experimento e o quarto programa será utilizado para treinamento nas técnicas funcional e estrutural. A identificação (nomes) dos participantes não é relevante para o objetivo do experimento. Será adotado o agrupamento dos participantes apenas como forma de dividir o mesmo programa para determinado número de estudantes, não necessariamente eles devem estar próximos fisicamente, e as informações coletadas serão avaliadas individualmente. Vale ressaltar que os programas serão divididos de forma uniforme entre os grupos.

O experimento será realizado em três dias programados sendo que para cada técnica de teste será ministrada uma hora de treinamento da técnica e, posteriormente, os participantes terão uma hora e meia para a aplicação prática da mesma em um dos programas selecionados. O laboratório utilizado possui 20 máquinas *desktop* com Sistema Operacional Linux, Java 6.0, Eclipse, Wireless Took Kit 2.5, EclipseME e a ferramenta JaBUTi/ME instalados.

Os programas foram selecionados de repositórios de software livre *http://www.sourceforge.net* e *http://code.google.com*. Cerca de 20 (vinte) programas foram pré-selecionados. O critério de seleção utilizado para escolha dos programas foi existência do código fonte e a complexidade dos programas, sendo que os de maior complexidade foram escolhidos.

Para viabilizar a coleta de dados de execução de todos os programas, estes foram instrumentados previamente utilizando-se a ferramenta JaBUTi/ME. Desse modo, mesmo quando são utilizadas as técnicas *ad hoc* ou funcional para a geração dos conjuntos de teste, a execução dos testes é "monitorada" e a cobertura dos mesmos pode ser posteriormente avaliada em relação aos critérios estruturais implementados pela JaBUTi/ME. Vale ressaltar que a mesma ferramenta foi utilizada para avaliação das três técnicas aplicadas. A Figura 2 ilustra esse processo de execução das aplicações instrumentadas e como a coleta das informações de cobertura é realizada. Além disso, a quantidade de defeitos identificados pelos casos de teste criados a partir de cada critério também será coletada por meio de formulários que devem ser preenchidos pelos participantes.

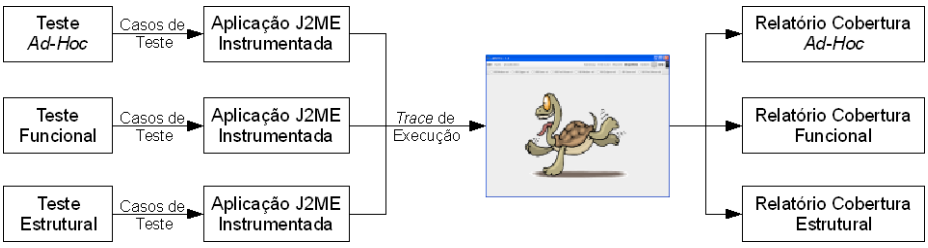


Figura 2 – Esquema de monitoramento adotado.

- **Instrumentação:** Nessa etapa serão preparados os formulários, programas, ambiente de laboratório para a realização do experimento.

[1] O termo "randomizado" foi utilizado para ser consistente com sua referência [Felizardo 2003].

Foram preparados cinco formulários para coleta de informações, são eles: Formulário 1 – Perfil do Participante, Formulário 2 – Definição dos Grupos, Formulário 3 – Casos de Teste, Formulário 4 – Sugestões e Formulário 5 – Avaliação do Curso. Os formulários desenvolvidos, bem como todos os resultados coletados estão disponíveis em *http://www.inf.ufg.br/~auri/curso*.

Dos programas pré-selecionados, 4 (quatro) foram escolhidos para a realização do experimento com base na complexidade dos mesmos, sendo que os mais complexos foram selecionados. A Tabela 1 apresenta os três que serão utilizados para a coleta de dados, uma estatística baseada na média da complexidade ciclomática máxima de seus métodos, e uma breve descrição de cada um deles.

Os programas P1, P2 e P3 serão utilizados pelos participantes na aplicação das técnicas de teste *ad hoc*, funcional e estrutural. A ordem e distribuição estão definidas na Tabela 2. Um quarto programa, chamado BMI, que calcula o Índice de Massa Corporal com base no peso e altura de um indivíduo e o classifica de acordo com os níveis de obesidade, será utilizado para o treinamento dos participantes nas técnicas funcional e estrutural.

Tabela 1 – Programas Selecionados e Complexidade.

Id.	Nome	Complexidade	Descrição
P1	AntiPanela	3,87	Faz o cadastro de jogadores e realiza o sorteio das equipes com base no número de jogadores evitando a formação de "panelas".
P2	CarManager	5,52	Controla e gerencia os gastos de combustível de um veículo automotor.
P3	CódiceFiscale	6,17	Verifica a validade ou gera o "Codice Fiscale" italiano, semelhante ao CPF brasileiro.

Para a seleção dos programas J2ME houve a necessidade dos códigos fontes dos programas, para que defeitos fossem inseridos nos mesmos. Uma média de 10 (dez) defeitos foram inseridos artificialmente em cada um dos programas escolhidos com base no conceito de mutação [Ma et al. 2005]. Os defeitos inseridos foram de inicialização de variáveis, cálculos, fluxo de controle, interface e estrutura de dados. Após a inserção dos defeitos, os programas foram compilados e instrumentados utilizando os recursos oferecidos pela JaBUTi/ME, viabilizando o rastreamento da execução dos casos de teste e, posteriormente, a análise de cobertura dos mesmos em relação aos critérios apoiados pela ferramenta.

- **Avaliação:** A avaliação será realizada em todos os programas baseado no Formulário 3 – Casos de Teste, que contém as informações sobre os casos de testes executados (defeitos encontrados), e verificando se o arquivo *trace* está sendo produzido para análise de cobertura.
- **Preparação:** Serão distribuídos os materiais e as instruções para a participação no experimento. O programa BMI foi o escolhido para ser utilizando apenas durante o treinamento de todas as técnicas e para exemplificar como é a execução dos demais programas no emulador do dispositivo móvel.
- **Execução:** A tarefa de executar o planejado, no tempo estimado e documentar qualquer desvio ocorrido que possa alterar ou afetar o objetivo do experimento. Também serão explicadas aos alunos as especificações dos programas a serem testados, a fim de ambientá-los com as funcionalidades que os mesmos oferecem.

- **Validação dos Dados:** Ao final da aplicação de cada técnica pelo participante, todo o projeto (incluindo o arquivo *trace*) deverá ser nomeado e enviado à comissão organizadora, que irá assegurar que os dados gerados pelos programas instrumentados de cada participante foram recebidos corretamente.
- **Análise e Interpretação:** Logo após a coleta dos dados do experimento e da replicação, as informações serão cruzadas e analisadas a fim de avaliar as hipóteses definidas no pacote de experimentação.
- **Apresentação e Empacotamento:** Os dados e conclusões resultantes dos experimentos serão divulgados por meio de artigos e uma dissertação de mestrado [Deus 2009] agrupando os dados de todas as replicações realizadas. Este artigo é referente aos dados coletados após a segunda replicação do pacote de experimentação.

5 Análise dos Dados

A primeira aplicação desse pacote de experimentação foi realizada no mês de agosto de 2008 [Deus et al. 2008]. Este artigo trata da segunda replicação do pacote de experimentação, realizado entre 17 a 20 de novembro de 2008, visando a fortalecer os dados e aumentar o tamanho da amostra para todo o experimento.

Os estudantes iniciaram respondendo ao questionário de perfil do participante, sendo que a idéia foi apenas coletar as informações da experiência profissional e acadêmica dos indivíduos. Uma introdução sobre teste de software relatando a importância de testar, o papel do testador, principais tipos de testes, teste de unidade, integração e sistema foram explicados e discutidos pelos participantes do curso. Logo em seguida os alunos foram divididos em 6 (seis) grupos, dos quais os integrantes de cada grupo foram selecionados de forma aleatória, por meio de sorteio. Definidos os grupos, foi distribuído a cada grupo um programa J2ME e sua respectiva especificação de requisitos. O objetivo do primeiro dia do experimento era encontrar o maior número de defeitos existentes nos programas de acordo com o conhecimento que cada um já possuía sobre construção e teste de software, ou seja, utilizando a técnica *ad hoc*. A distribuição dos programas em seus respectivos grupos é apresentada na Tabela 2.

Tabela 2 – Distribuição dos Grupos, Programas e Técnicas.

Técnica / Grupo	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**
ad hoc	P1	P3	P2	P1	P3	P2
Funcional	P2	P1	P3	P2	P1	P3
Estrutural	P3	P2	P1	P3	P2	P1
G – Grupo; P – Programa;						

Após a execução da técnica *ad hoc*, os participantes receberam um treinamento sobre os critérios da técnica de teste funcional. Novos programas e suas especificações foram distribuídos (de acordo com as Tabelas 1 e 2) a cada grupo e novamente foi solicitado para que eles aplicassem os conhecimentos adquiridos da técnica funcional na realização dos testes do segundo programa.

Depois da execução dos testes utilizando a técnica funcional, foi ministrado um treinamento aos participantes sobre a técnica estrutural e a utilização da ferramenta JaBUTi/ME. Em seguida uma terceira e última distribuição dos programas entre os grupos (conforme Tabela 2) foi realizada, e os participantes aplicaram os conceitos da técnica estrutural na execução dos testes.

Durante a execução dos testes para qualquer uma das técnicas, os participantes registraram as não-conformidades encontradas. Para finalizar o experimento foi solicitado aos estudantes que respondessem um formulário com propostas de melhoria e a avaliação individual do curso. É importante ressaltar que todos os participantes testaram, obrigatoriamente, os três programas do experimento por meio de diferentes critérios de teste.

5.1 Perfil do participante

Em relação ao perfil dos participantes da segunda replicação observam-se pela Figura 3 que os indicadores foram todos maiores que os da primeira replicação. Por exemplo, a idade média, experiência profissional e acadêmica, e em linguagens de programação tiveram aumento significativo.

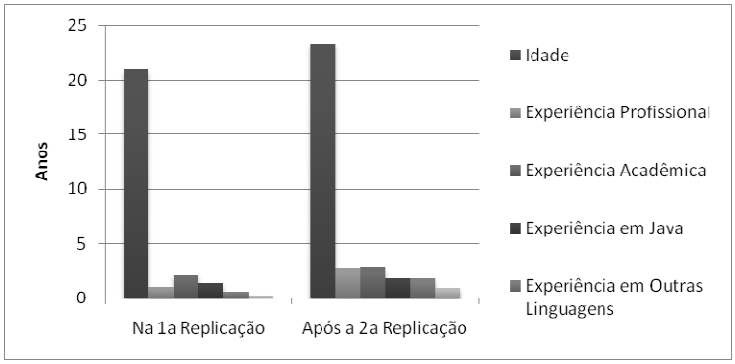
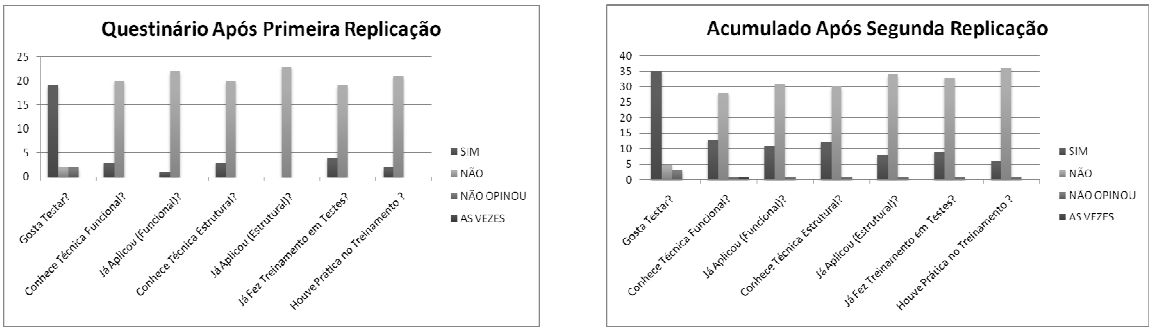


Figura 3 – Média Acumulada das Replicações.

Tento em vista que os participantes da segunda replicação tinham mais experiência em desenvolvimento de software do que os da primeira, os resultados que identificavam quais técnicas os participantes conheciam e já haviam aplicado com atividades práticas também tiveram melhores indicadores. Estas informações podem ser avaliadas no gráfico da Figura 4, que mostra as questões dos formulários e os valores acumulados da primeira (a) e da segunda (b) replicações.



(a) (b)

Figura 4 – Resultado Acumulado dos Questionários.

5.2 Análise dos dados coletados

O pacote de experimentação foi preparado para capturar informações de cobertura dos programas testados (independente da técnica aplicada). A ferramenta JaBUTi/ME foi utilizada para leitura desses dados dos testes executados pelos participantes. A ferramenta apóia quatro critérios de testes, sendo eles: Todos-Nós, Todas-Arestas,

Todos-Usos e Todos-Potenciais-Usos [Vincenzi et al. 2003]. Apesar de o treinamento ter focado nos dois primeiros critérios, conhecidos como critérios de fluxo de controle, (Todos-Nós e Todas-Arestas), todos os critérios citados acima foram analisados (inclusive fluxo de dados), buscando aferir qual a cobertura dos testes produzidos em relação a esses critérios de teste. Um ponto importante a ser observado é que os requisitos de teste (produzidos pelos critérios citados) que não são executáveis não foram identificados. E o tempo para a execução dos testes foi controlado no período de uma hora e meia. Portanto, pode ser que a cobertura máxima de 100% não seja obtida em função da presença desses requisitos e/ou da restrição de tempo. Entretanto, como o objetivo é comparar qual conjunto de teste cobriu mais requisitos, uma porcentagem maior de cobertura e mesmo que inferior a 100% é suficiente para estabelecer essa relação. As Tabelas 3 e 4 apresentam a síntese desses dados.

Tabela 3 – Média de cobertura por critério e casos de teste [Deus et al, 2008]

	AntiPanela			CarManager			CodiceFiscale		
Critérios de Teste/Técnica	**ad hoc**	**Funcional**	**Estrutural**	**ad hoc**	**Funcional**	**Estrutural**	**ad hoc**	**Funcional**	**Estrutural**
Todos-Nós	76,38	80,25	73,43	59,00	51,13	62,80	49,63	49,67	66,00
Todas-Arestas	65,38	71,00	62,86	46,43	39,13	49,40	38,38	38,00	51,60
Todos-Usos	65,75	70,75	62,71	49,71	43,00	53,20	43,00	42,83	57,60
Todos-Potenciais-Usos	66,38	69,75	63,86	42,43	36,13	47,40	34,38	34,00	35,00
Número de Casos de Teste	14,25	13,00	9,71	7,33	10,11	10,50	11,13	8,50	14,60
Defeitos Revelados	9,25	7,00	5,57	4,33	6,88	6,75	6,00	6,17	6,40

Os dados acumulados com a execução da segunda replicação revelaram que a técnica estrutural passou a ter maior percentual de cobertura em todos os programas (AntiPanela, CarManager e CodiceFiscale). Não mais como na primeira replicação que apenas os dois programas mais complexos possuíam a maior cobertura. Apesar da média da técnica estrutural ter sido superior, os desvios-padrões da técnica estrutural aumentaram passando, em média, de 8 para 13, como o aumento não foi significativo mantém a confiança na média, vide Figura 3 (b).

Na primeira replicação nos programas CarManager e CodiceFiscale, a técnica estrutural mostrou ser mais eficaz na cobertura de código dos programas em todos os critérios de teste apoiados pela ferramenta JaBUTi/ME, com os dados da segunda replicação isso se estendeu ao programa AntiPanela, conforme Tabela 3 e Tabela 4.

Tabela 4 – Nova Média de cobertura por critério e casos de teste.

	AntiPanela			CarManager			CodiceFiscale		
Critérios de Teste/Técnica	**ad hoc**	**Funcional**	**Estrutural**	**ad hoc**	**Funcional**	**Estrutural**	**ad hoc**	**Funcional**	**Estrutural**
Todos-Nós	70,92	65,29	84,44	61,38	53,00	64,73	44,42	52,82	65,67
Todas-Arestas	60,42	55,21	74,67	48,15	40,40	50,82	34,25	40,64	52,08
Todos-Usos	59,42	55,00	74,00	52,46	44,20	54,64	38,08	45,36	57,75
Todos-Potenciais-Usos	59,17	55,93	73,11	44,00	36,73	47,09	31,08	36,64	41,83
Número de Casos de Teste	13,82	9,71	13,78	8,45	10,53	10,44	9,92	8,00	13,08
Defeitos Revelados	9,27	5,50	6,11	5,27	5,80	4,44	5,00	5,27	5,75

A utilização da ferramenta JaBUTi/ME ajudou a garantir que mais trechos desses programas fossem executados. Em uma visão real, seria o mesmo que informar ao cliente que está supostamente requisitando um software, que todos os programas tiveram mais partes executadas, pelo menos uma vez, e avaliados os resultados obtidos pelos programas.

Embora o emprego da técnica funcional em todos os programas tenha diminuído a quantidade média de casos de testes de 12,9 para 7,3, continuou sendo desta técnica os maiores números de casos de testes criados, o desvio padrão médio também reduziu de 7,3 para 2,6, aumentando a proximidade dos dados coletados da média.

Vale ressaltar que o experimento tinha tempo controlado, ou seja, todos os participantes tiveram uma hora e meia para a execução dos testes, independente da sua complexidade. Portanto, um programa mais complexo como CarManager e CodiceFiscale em ambiente real de testes seria testado com mais tempo, principalmente, porque a aplicação de uma técnica (funcional ou estrutural) e uma ferramenta (JaBUTi/ME) foram utilizadas para a maioria dos participantes pela primeira vez na prática. Além disso, o experimento com o tempo de uma hora e meia mostra a tendência de qual técnica se mostra mais eficaz na localização e cobertura dos programas J2ME. Todas essas informações estão detalhadas na Tabela 4, e nas Figuras 5 e 6.

Como é possível observar nas Tabelas 3 e 4, a técnica estrutural passou a ter maior cobertura após o crescimento da amostra, resultante do acúmulo dos dados da segunda replicação. Além disso, no programa AntiPanela houve um aumento na quantidade de casos de testes e de defeitos revelados. Os novos valores, Tabela 4, ficaram muito próximos dos valores da primeira replicação, Tabela 3. Ou seja, houve um crescimento no programa de pior desempenho da técnica estrutural na primeira replicação e mantiveram-se estáveis todas outras médias.

Acumulando os dados apresentados por Deus et al. (2008) com a segunda replicação, a aplicação da técnica estrutural em conjunto com a ferramenta JaBUTi/ME mostrou ser viável na criação de uma maior cobertura de código (Figura 5 (a), (b) e (c)), uma maior ou igual quantidade de casos de testes criados (Figura 6 (a)) e houve uma maior revelação de defeitos (Figura 6 (b)), para o programa de maior complexidade. Alguns os defeitos inseridos eram diversos, alguns, por exemplo, de interface, que são mais difíceis de serem descobertos pela técnica estrutural. O resultado da segunda replicação mostrou que a técnica estrutural teve um aumento médio de 6,1% no critério Todos-Nós e 8,4% no critério Todas-Arestas em relação a primeira replicação. Essa segunda replicação foi de extrema importância para fortalecer os dados do pacote de experimentação, mostrando a real utilidade da técnica e comparando-as com as outras duas. E novamente os números apresentados mostraram que os testadores tiveram maior facilidade em construir os casos de testes, e em alcançar maiores coberturas de código de software utilizando a técnica estrutural e a ferramenta JaBUTi/ME.

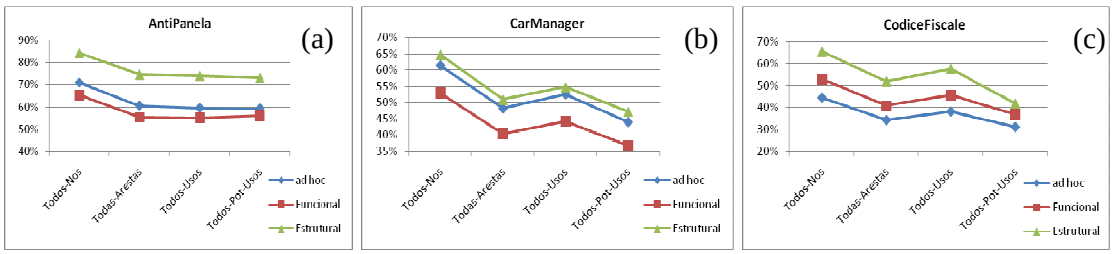


Figura 5 – Percentual de cobertura de critérios por programa.

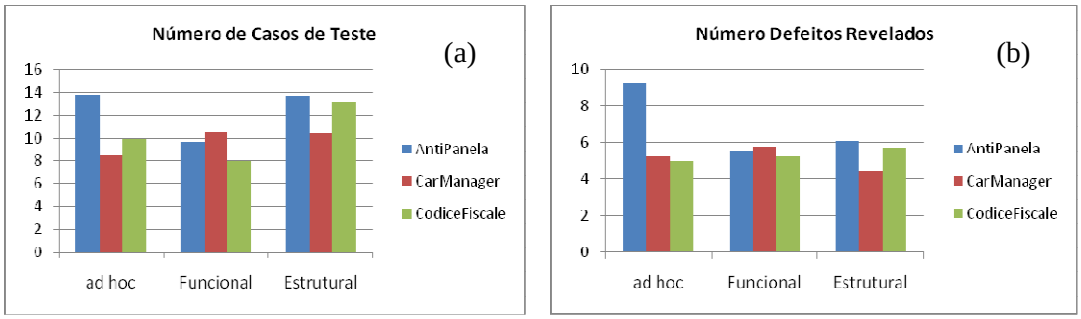


Figura 6 – Número de Defeitos e de Casos de Teste por programa.

Pelo menos mais uma replicação do experimento está planejada para confirmar os padrões obtidos na primeira e na segunda replicações, validando as hipóteses levantadas em [Deus et al. 2008] de que a ferramenta JaBUTi/ME contribuiu para facilitar os testes de todos dos programas J2ME do experimento. Nessa replicação, dos defeitos inseridos intencionalmente, a técnica funcional revelou uma média de 19%, a técnica *ad hoc* 27% e a estrutural 23%. Além dos defeitos inseridos artificialmente, alguns defeitos naturais (que já estavam presentes no código das aplicações escolhidas) foram revelados. Destes defeitos naturais, a técnica *ad hoc* revelou 1,4 novos defeitos, enquanto a técnica funcional registrou 0,9 e a técnica estrutural apresentou 0,8 novos defeitos.

Comparando as coberturas apresentadas nos gráficos da Figura 5, é possível observar que a técnica estrutural apresentou melhor cobertura em todos os programas AntiPanela (a), CarManager (b) e CodiceFiscale (c). As informações de cobertura foram reforçadas e melhoradas nessa replicação do experimento.

5.3 Lições aprendidas

Alguns dados foram perdidos durante a execução do experimento, as causas mais comuns foram: a) o participante não salvou o arquivo do Formulário 3 – Casos de Teste corretamente para envio aos organizadores do curso; b) o participante não inicializou corretamente os programas, tornando impossível a captura das informações de cobertura; Já o acumulado de perda de informações ficou em 26% para o programa AntiPanela, 14% no CarManager e 26% no CodiceFiscale.

Apesar de ser ressaltada a importância de seguir corretamente todos os passos e executar corretamente o experimento, infelizmente acontecem desvios e o controle de cerca de 20 pessoas executando procedimentos simultaneamente em um laboratório de informática é complexo, e perdas nos dados são inevitáveis.

Na primeira replicação [Deus et al. 2008] houve uma abstenção no último dia do curso de 41% dos participantes, na segunda replicação esse valor ficou em 10%. Visto que o curso não havia taxa para a matrícula e o único incentivo do participante era o aprendizado, aqueles que desistiram não tinham esse objetivo ou tiveram algum problema particular. Apesar de ter sido criado para a segunda replicação um termo de compromisso para a participação nos três dias do curso, não foi possível ter uma taxa inferior ao valor referido acima na segunda replicação. Esse aumento na permanência dos alunos pode ser justificado pelas sugestões feitas na primeira replicação que solicitaram o curso para dias úteis ao invés de sábados, e redução da quantidade de estudantes no laboratório também facilitou o acompanhamento feito pelos instrutores.

Na segunda replicação, outro problema enfrentado foi que duas máquinas do laboratório não estavam funcionando nos dias do experimento, e foi exatamente a mesma quantidade de abstenções registradas. Uma melhoria feita para essa replicação na especificação de cada programa e a explicação do instrutor sobre cada um dos programas, com exemplos de execução e análise do programa J2ME. Isso contribuiu muito para uma maior interpretação dos requisitos do programa pelos participantes do curso. Uma dificuldade para trabalhar com o sistema operacional *Linux* foi detectada pelos instrutores do curso, que tiveram que acompanhar mais de perto a execução dos comandos que antecedem a execução dos testes.

Dentre as sugestões apresentadas no Formulário 4 – Sugestões, 30% solicitaram a apresentação de outras ferramentas, inclusive de outras linguagens para que eles possam ter mais opções ao executar os testes. 25% informaram que gostariam de ter mais tempo para praticarem o aprendizado das técnicas, ou seja, assumiram que só não foi possível encontrar mais defeitos nos programas devido à restrição de tempo. Já 15% sugeriram para não utilizarem programas J2ME.

As informações coletadas no Formulário 5 – Avaliação do Curso revelaram que 100% dos participantes alegaram ter agregado conhecimento na área de testes com o curso. Já 88% revelaram que se sentem seguros para a aplicação das técnicas apresentadas. No formulário foi solicitado que os participantes avaliassem por meio de uma nota o conhecimento adquirido durante o curso, a média acumulada ficou em 7,9 e a nota geral para o curso ficou em 8,5. Ou seja, a grande maioria dos participantes aprovou e elogiou a iniciativa, visto que, as técnicas de testes não são muito difundidas e dificilmente encontra-se um curso de testes gratuito como o realizado.

Muitos comentários foram feitos durante o curso pelos participantes, os mais importantes foram: 1) a visão de carência de profissionais qualificados em testes; 2) a dificuldade para testar software; 3) a falta de ferramentas de testes. Outros comentários foram os elogios aos instrutores do curso e à ferramenta JaBUTi/ME, sendo que muitos se mostraram interessados em continuar estudando-a e aplicá-la em programas acadêmicos e profissionais.

6 Trabalhos Relacionados

No contexto de experimentos de teste para dispositivos móveis utilizando J2ME, até o momento e que seja de conhecimento dos autores, apenas um artigo foi publicado por [Deus et al. 2008]. A partir do pacote de experimento realizou-se uma segunda replicação, que é o artigo em questão, e os dados puderam ser comparados e analisados conforme descrito nas seções acima. Outro artigo relacionado ao tema que pode ser citado é o estudo de caso apresentado em [Delamaro et al. 2006], porém o mesmo foi executado sem a utilização de pacotes de experimentação e serviram apenas para avaliar superficialmente as funcionalidades da JaBUTi/ME.

7 Conclusão e Trabalhos Futuros

Após a segunda replicação desse experimento, foi possível assegurar a importância e a complexidade da disciplina de testes na Engenharia de Software. Existem diferentes técnicas e critérios cada um focado em buscar defeitos em tipos ou partes de aplicações. Dentre os critérios mais conhecidos encontram se a “Analise de Valor Limite”, o “Particionamento de Equivalência”, “Todos-Nós”, “Todas-Arestas”, “Todos-Usos” e “Todos-Potenciais-Usos”.

Foi possível observar também que cada técnica possui seu foco, e que elas podem ser utilizadas em conjunto para encontrar uma maior quantidade de defeitos nos programas. As técnicas apresentadas auxiliam o testador a selecionar valores do domínio de entrada de forma sistemática, otimizando a criação dos casos de teste e buscando aumentar a detecção de defeitos.

Considerando o acumulado dos dados com a segunda replicação do pacote de experimentação foi possível observar que a ferramenta JaBUTi/ME na técnica estrutural auxiliou os testadores a obterem melhores resultados em todos os programas. Dentre os dados coletados é possível citar, uma quantidade maior ou igual de casos de testes gerados relacionados com as outras técnicas avaliadas, uma maior cobertura de código em todos os critérios (“Todos-Nós”, “Todas-Arestas”, “Todos-Usos” e “Todos-Potenciais-Usos”) e uma maior quantidade de defeitos encontrados no programa de maior complexidade. Ao resultar em uma maior quantidade de casos de teste que abrangem mais subdomínios de entrada, é possível inferir que a ferramenta dá suporte ao usuário a observar trechos de código não executados e criar casos de testes que os exercitem, proporcionando uma maior cobertura como aconteceu. Como nessa segunda replicação a abstenção dos participantes foi menor, foi possível observar melhores resultados da técnica estrutural e da ferramenta JaBUTi/ME do que as apresentadas na primeira replicação. Além de fortalecer as outras informações sobre esse critério no teste de software.

A técnica *ad hoc* continua, como na primeira replicação, melhor para revelar defeitos e obter uma boa cobertura nos programas mais simples e intuitivos. Para os que exigem maior esforço de análise de requisitos essa técnica não obteve resultado tão relevante. A técnica estrutural, para alguns programas, mostrou-se ótima para a criação de casos de testes, porém esses casos de testes não foram eficazes para revelar defeitos, uma das características dos critérios vistos nessa técnica é reduzir o número de caso de testes, e isso não ocorreu e os defeitos não apareceram com a aplicação dessa técnica, tanto que ela não revelou uma maior quantidade de defeitos em nenhum dos programas testados.

Enfim, muitas informações da primeira replicação foram fortalecidas, e mais replicações devem acontecer, permitindo uma análise estatística com uma amostra maior, contendo mais informações sobre os testes executados. Outra replicação (trabalho futuro) está prevista para acontecer utilizando os mesmos programas em seu ambiente real, ou seja, no próprio dispositivo móvel. O objetivo dessa replicação é constatar que os resultados da técnica e os defeitos encontrados, também têm a mesma validade para o ambiente real do software e não só em emuladores.

Algumas lições aprendidas na primeira replicação do experimento trouxeram contribuições para a melhoria do pacote de experimentação e culminaram em uma melhor qualidade dos dados coletados e também no aprendizado dos participantes.

Como trabalhos futuros destacam-se: 1) a necessidade de mais replicações do pacote de experimentação visando a melhorar cada vez mais a representatividade estatística dos dados; 2) a avaliação da ferramenta JaBUTi/ME no teste de programas J2ME em dispositivos reais; 3) o desenvolvimento de uma estratégia de teste incremental para produtos J2ME, combinando diferentes critérios de teste de uma maneira coordenada, visando a reduzir o tempo e o custo da atividade de teste aumentando a sua eficácia em detectar defeitos; e 4) avaliar o impacto no pacote de experimentação se produtos J2ME mais complexos forem utilizados.

Referências

ANATEL (2008), Agência Nacional de Telecomunicações, http://www.anatel.gov.br, Novembro.

Beizer, B (1995). Black-Box Testing: Techniques for Functional Testing of Software and Systems. John Wiley & Sons.

Deus, G. D., Vincenzi, A. M. R. e Delamaro, M. E. (2008) *Criação de um Pacote de Experimentação para a Avaliação de Critérios de Teste Estruturais em Produtos J2ME*. V ESELAW - Experimental Software Engineering Latin American Workshop.

Deus, G. D. (2009). Avaliação de Técnicas de Teste para Dispositivos Móveis Por Meio de Experimentação. Dissertação de Mestrado. Instituto de Informática - UFG. (mestrado em andamento)

Delamaro, M. E., Maldonado, J. C, e Jino, M. *Introdução ao Teste de Software*. (2007) Elsevier, Rio de Janeiro, RJ.

Delamaro, M. E., Vincenzi, A. M. R. e Maldonado, J. C. (2006) A Strategy to Perform Coverage Testing of Mobile Applications. *I International Workshop on Automation of Software Test - AST'2006*, ACM Press, 2006, 118-124.

Felizardo, K. R. (2003) *COTEST – Uma Ferramenta de Apoio à Replicação de um Experimento Baseado em Código Fonte*. Dissertação de Mestrado, Universidade Federal de São Carlos - UFSCar.

Höhn, E. N. (2004) *Técnicas de leitura de especificação de requisitos de software: estudos empíricos e gerência de conhecimento em ambientes acadêmico e industrial*. Dissertação de mestrado, ICMC/USP.

Howden, W. E. (1987) *Software Engineering and Technology: Functional Program Testing and Analysis*. McGrall-Hill Book Co.

Lott, C. M. e Rombach, H. (1996) Repeatable Software Engineering for Comparing Defect-Detection Techniques. *Journal of Empirical Software Engineering*, 1, 241-277.

Ma, Y., Offutt, J. e Kwon, Y. R. (2005) MuJava: an automated class mutation system. *Software Testing, Verification and Reliability*, John Wiley and Sons Ltd., 15, 97-133.

Microsistems, S (2009). Sun Java Wireless Toolkit 2.5.2 for CLDC. Disponível em: http://java.sun.com/. Acesso em: 05/04/2009.

Myers, G. J.; Sandler, C.; Badgett, T. e Thomas, T. M (2004). The Art of Software Testing Wiley, Nova York.

Vincenzi, A. M. R., Wong, W. E., Delamaro, M. E. e Maldonado, J. C. (2003) JABUTI: A Coverage Analysis Tool for Java Programs. *XVII SBES - Simpósio Brasileiro de Engenharia de Software*, Sociedade Brasileira de Computação (SBC), 79-84.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. e Wesslén, (2000) *A. Experimentation in Software Engineering – An Introduction*. Kluwer Academic Publishers, Norwell, MA, EUA.