Universidade Federal de Pernambuco

Graduação em Ciência da Computação
Centro de Informática
2018.2

AutoTestCoverage^{C}: Uma ferramenta para obtenção de cobertura de código para componentes Android sem uso de instrumentação

---

Proposta de Trabalho de Graduação

Aluno: João Luiz de Andrade Neto (jlan@cin.ufpe.br)
Orientador: Alexandre Cabral Mota (acm@cin.ufpe.br)

Recife, 22 de Agosto de 2018

Sumário

1. **Introdução……3**
2. **Objetivos……..4**
3. **Cronograma…….5**
4. **Possíveis avaliadores……...5**
5. **Referências……..6**
6. **Assinaturas……..7**

1. Introdução

Quando falamos da indústria de desenvolvimento de software, nos dias atuais, logo conseguimos ter em mente que diversas funcionalidades são diariamente implementadas e colocadas em produção para centenas ou milhares de usuários. Isso significa que constantemente várias linhas de código são modificadas dentro dos produtos da indústria. Com essa realidade, fica fácil de se deparar com falhas e diminuição da experiência do usuário, no nosso dia a dia. Com o objetivo de evitar que esses erros cheguem até o usuário, as empresas exercitam testes com o objetivo de melhorar a qualidade de seus produtos [1, 2], ao mesmo tempo, que buscam diminuir as interferências na usabilidade de seus usuários.

Com o interesse de propor produtos de melhor qualidade a seus clientes, empresas optam por implantar um processo de teste, desde a criação, seleção e execução de testes. Isto requer tempo e dinheiro, esperando assim que tal investimento traga um resultado positivo na diminuição dos defeitos escapados (chamamos assim um defeito que não foi encontrado pelo time de teste, mas sim pelo usuário final) [3]. Mas um questionamento interessante se dá em como podemos medir se de fato os testes selecionados para uma determinada campanha estão exercitando as regiões de código modificadas.

Uma métrica conhecida como cobertura de código é bastante utilizada para medir a quantidade de código que foi testada por um conjunto de caso de testes [4], podendo esses dados de cobertura, a princípio, serem obtidos com o auxílio da instrumentação de código-fonte (trata-se da habilidade de monitorar ou medir a performance de um produto, diagnosticando erros e capturando dados de cobertura) [5]. Assim sendo, torna-se possível utilizar diferentes métricas para medir cobertura, tais como, métodos, classes, instruções, arquivos, dentre outros.

Entretanto, algumas vezes ocorre de termos um cenário onde possuímos acesso de leitura ao código-fonte de uma aplicação, mas não podemos realizar instrumentação nela. Este é exatamente o caso da nossa parceira industrial: a Motorola Mobility. O processo de captura dos dados e a análise dessas informações providas pelos casos de teste, se torna mais desafiador nesse caso. Nesse documento mostraremos como viabilizamos essas informações para nosso parceiro de maneira a melhorar a seleção dos casos de teste utilizados.

2. Objetivos

Este trabalho tem como principais objetivos:

- Identificar as áreas modificadas do código-fonte de uma aplicação dadas duas versões desta aplicação;
- Obter dados de cobertura a partir da execução de casos de testes sem o uso da técnica de instrumentação de código;
- Calcular o percentual de cobertura mediante os dados de cobertura dos testes em relação às áreas que deveriam ser cobertas;
- Armazenar os dados de cobertura obtidos, provenientes de uma campanha de testes, juntamente com os respectivos testes em um gerenciador de testes.

Para atingir os quatro objetivos anteriores, pretendemos construir uma ferramenta de software chamada de AutoTestCoverage^{C}. A ferramenta AutoTestCoverage^{C} visa analisar aplicativos Android [6]. Ela foi divida em dois módulos. O primeiro deles deve identificar as áreas do código-fonte que devem ser cobertas a partir de uma campanha de testes. Iremos desenvolver esta parte em Python [7] com o framework Django [8]. O segundo módulo é o relacionado à obtenção dos dados de cobertura sem uso de instrumentação. Será desenvolvido em Java [9] mas lidará com código Android. Após a obtenção da cobertura de uma campanha de testes, este resultado será armazenado na ferramenta Solr [10] (ferramenta utilizada para armazenar grandes quantidade de texto, visando auxiliar buscas textuais*)*, para que assim, os dados possam ser acessados pelo gerenciador de testes.

3. Cronograma

Nessa sessão a tabela 1 descreve todo o cronograma das atividades previstas a serem realizadas durante o desenvolvimento desse trabalho de graduação, ao longo do semestre.

Atividade	Agosto	Setembro	Outubro	Novembro	Dezembro
Elaboração da Proposta					
Pesquisa Bibliográfica					
Testes e Experimentos					
Análise dos Resultados					
Elaboração do Relatório					
Preparação para Apresentação de Defesa					

4. Possíveis avaliadores

Os possíveis avaliadores para o resultado a ser obtido no final de todas as etapas descritas nesse documento são:

- Juliano Manabu Iyoda;
- Leopoldo Teixeira

5. Referências

[1] - Jaime Jorge, Research on code coverage, *Research on code coverage*, Medium, Jan 5, 2017. https://blog.codacy.com/research-on-code-coverage-44c1f1c3d8b3;

[2] - Sheikh Umar Farooq, S. M. K. Quadri, Nesar Ahmad, Software Measurements And Metrics: Role In Effective Software Testing, International Journal of Engineering Science and Technology (IJEST), Acesso: 19/03/2018;

[3] - Mary Ann Vandermark, Defect Escape Analysis: Test Process Improvement, IBM Software Group, Tivoli Software January 23, 2003, Acesso: 19/03/2018;

[4] - Marco Antônio Rocha, Cobertura: Uma Métrica para Aferir Testes, http://matera.com/br/2014/01/16/cobertura-uma-metrica-para-aferir-testes/, Acesso: 19/03/2018;

[5] - IBM Website, Source code instrumentation overview https://www.ibm.com/support/knowledgecenter/SSSHUF_8.0.0/com.ibm.rational.testrt.doc/topics/cinstruovw.html, Acesso: 19/03/2018;

[6] - Google, **Android**, https://www.android.com/, Acesso: 21/03/2018;

[7] - **Python**, https://www.python.org/, Acesso: 21/03/2018;

[8] - **Django**, https://www.djangoproject.com/, Acesso: 21/03/2018;

[9] - **Java**, https://www.java.com/, Acesso: 21/03/2018;

[10] - Apache,**Solr,** http://lucene.apache.org/solr/, Acesso: 23/03/2018;

6. Assinaturas

João Luiz de Andrade Neto
Orientando

Alexandre Cabral Mota
Orientador