UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

Avaliação de cobertura de código de várias versões de Randoop

PROPOSTA DE TRABALHO DE GRADUAÇÃO

Aluno: Vinicius Thiago (vtls@cin.ufpe.br)
Orientador: Paulo Borba (phmb@cin.ufpe.br)
Área: Engenharia de Software

10/06/2021

Resumo

Durante o desenvolvimento de software, a etapa de testes é fundamental para verificar e validar o software em desenvolvimento. Porém, trata-se de uma etapa que demanda um alto custo de tempo e esforço para criar e manter os testes. Considerando a importância dos testes, ferramentas de geração automática de testes surgem como uma solução que reduz o custo e o esforço associados a esta etapa. Entretanto, em alguns casos, essas ferramentas de geração de testes apresentam deficiências gerando testes de baixa qualidade e cobertura. Assim, modificações nessas ferramentas podem ser implementadas para mitigar os efeitos de suas deficiências. Assim, neste trabalho nosso objetivo é avaliar os testes gerados pela ferramenta Randoop e suas modificações, quanto a cobertura de código, número e qualidade dos testes gerados.

Introdução

Durante o desenvolvimento de software, teste é uma atividade essencial responsável por verificar/validar o software em construção. Diferentes tipos de testes podem ser usados para diferentes propósitos. Por exemplo, testes de regressão são testes usados para detectar inesperadas mudanças de comportamento no software inseridas por novas mudanças [1]. Logo, a criação e evolução de suítes de teste para este propósito torna-se uma atividade que demanda um alto custo de tempo e esforço. Para tornar este processo mais dinâmico e menos custoso, estes testes podem ser automatizados e integrados ao projeto sendo executados a cada nova mudança.

Neste sentido, ferramentas de geração automáticas de testes, como Randoop [2] e EvoSuite [4], podem ser usadas como uma alternativa para gerar testes de maneira automática. Essas ferramentas funcionam recebendo uma versão do sistema como entrada e retornam uma suíte de testes explorando a versão recebida. Com isso, mudanças de comportamento inesperadas poderiam ser detectadas mais rápido reduzindo tempo e esforço do time de desenvolvimento. Por exemplo, Silva et al. [3] investigam se testes gerados por essas ferramentas podem ser usados para detectar mudanças de comportamento quando *refactorings* são realizados. Da Silva et al. [1] adota uma abordagem similar, onde utilizam testes gerados para detectar conflitos semânticos [5]. Entretanto, em ambos trabalhos, os autores reportam deficiências que impactam negativamente nas taxas de detecção de mudanças de comportamento pelos testes gerados.

Estas deficiências ocorrem porque as ferramentas apresentam dificuldades para gerar testes que explorem todos os diferentes estados do código em análise. Por exemplo, nos casos em que o código em análise apresenta dependências a objetos mais complexos, com muitas dependências externas (APIs, banco de dados etc), as ferramentas geram testes com uma baixa qualidade. Isso impacta negativamente na cobertura do código. Logo, se o código em análise não pode ser totalmente coberto e não tem entradas boas o suficiente, os testes gerados terão uma baixa qualidade e as mudanças não serão detectadas.

Com base nestas deficiências relatadas, mudanças poderiam ser implementadas em Randoop com o intuito de melhorar o processo de geração de testes, e posteriormente, a cobertura de código alcançada por testes gerados automaticamente. Assim, neste trabalho, nós alteramos a ferramenta Randoop forçando ela a gerar objetos da classes envolvidas nos métodos de testes criados. Em seguida, nós avaliamos as vantagens e desvantagens observadas após estas mudanças. Para tanto, nós realizamos um estudo empírico avaliando a cobertura, o número e a qualidade dos testes gerados por Randoop e suas variações para diferentes versões de um projeto.

Objetivos

Esse trabalho busca avaliar as vantagens e desvantagens observadas na geração de testes pela ferramenta Randoop e suas diferentes versões propostas. Para tanto, nós avaliaremos para cada versão sua cobertura de código, número e qualidade dos testes gerados.

Metodologia

Neste trabalho será realizado um estudo empírico com o objetivo de avaliar as vantagens e desvantagens observadas na geração de testes pela ferramenta Randoop e suas diferentes versões. Randoop é uma ferramenta de geração automática de testes que recebe uma versão do sistema como entrada e retorna uma suíte de testes explorando a versão recebida. Entretanto, esta ferramenta apresenta algumas deficiências que nós investigamos aqui desenvolvendo diferentes versões de Randoop. Assim, nosso objetivo é avaliar as vantagens e desvantagens destas novas versões de Randoop. Para realizar nosso estudo, nós selecionaremos projetos e seus diferentes cenários de merge composto por 4 versões (commits Base, Right, Left e Merge). Em seguida, testes automatizados serão gerados por Randoop e suas diferentes versões para cada commit do cenário de merge descrito. Por fim, nós iremos avaliar o impacto na cobertura, usando uma ferramenta para gerar a cobertura de código, número e qualidade dos testes gerados.

Cronograma

	Período						
Atividade	Fevereiro	Março	Abril	Maio	Junho	Julho	Agosto
Revisão bibliográfica	X	X					
Implementação	X	X	x	x			
Estudo do algoritmo	X	X	x	x			
Experimentos	X	X		x	x		
Avaliação dos resultados	X	X			x		
Escrita do TG						x	x
Preparação da apresentação						x	x

Referências

[1] SILVA, Leuson; BORBA, Paulo; MAHMOOD, Wardah; BERGER, Thorsten; MOISAKIS, João. **Detecting Semantic Conflicts via Automated Behavior Change Detection, 2020**

[2] PACHECO, Carlos; LAHIRI, Shuvendu; ERNST, Michel; BALL Thomas. **Feedback-directed Random Test Generation, 2007**

[3] SILVA, Indy; ALVES, Everton; ANDRADE, Wilkerson. **Analyzing Automatic Test Generation Tools for Refactoring Validation, 2017**

[4] FRASER, G; ARCURI, A. **Evosuite: Automatic Test Suite Generation for Object-Oriented Software, 2011**

[5] SARMA, Anite; HOEK, André. **Palantír: Early Detection of Development Conflicts Arising from Parallel Code Changes, 2012**

Possíveis Avaliadores

Prof. Marcelo dAmorim (damorin@cin.ufpe.br)

Prof. Breno Alexandro Ferreira de Miranda (bafm@cin.ufpe.br)

Assinaturas

Recife, 10 de Junho de 2021

(Aluno)

(Orientador)