Evento	Salão UFRGS 2019: SIC - XXXI SALÃO DE INICIAÇÃO CIENTÍFICA DA UFRGS
Ano	2019
Local	Campus do Vale - UFRGS
Título	Análise de Anomalias e Cobertura de Testes em Códigos Orientados a Objeto
Autor	FELIPE BERTOLDO COLOMBO DE SOUZA
Orientador	ERIKA FERNANDES COTA

Análise de Anomalias e Cobertura de Testes em Códigos Orientados a Objeto

Orientando: Felipe Bertoldo Colombo de Souza
Orientadora: Érika Fernandes Cota
Universidade Federal do Rio Grande do Sul

O paradigma de orientação a objetos trouxe grandes vantagens em relação ao paradigma imperativo com, por exemplo, polimorfismo, herança e encapsulamento. Amplamente usada na atualidade, a orientação a objetos requer técnicas de teste que atendam às suas especificidades, garantindo que, além das unidades, as interações entre as classes também sejam consideradas.

Ammann e Offutt[1] definem anomalias próprias desse paradigma, referentes ao uso de herança e polimorfismo. Essas anomalias são difíceis de serem testadas de forma automática uma vez que, muitas vezes, elas têm carga semântica, o que depende da visão do programador para a verificação do código.

Baseada nessa dificuldade de verificação automática, um analisador que identifica potenciais situações anômalas e as indica ao programador foi proposto anteriormente por outro grupo e implementado para a linguagem C++. O analisador explora um código intermediário, gerado a partir do código fonte em C++, de maneira a verificar essas situações específicas e avisar ao programador sobre situações suspeitas, ou seja, interações entre classes que podem levar à ocorrência de falhas. Optou-se, naquele trabalho, pelo uso de um código intermediário aceito pelo compilador LLVM, sob a hipótese de que o analisador pudesse ser útil para códigos em outras linguagens de programação além do C++. Essa hipótese, porém, não havia ainda sido validade e foi objeto de pesquisa desse bolsista.

A partir da questão da aplicabilidade do analisador para outras linguagens, estudamos a linguagem Swift, que faz uso do LLVM para sua compilação, o que facilitaria a geração do código intermediário que o analisador espera. Entretanto, o arquivo intermediário gerado para o código Swift possui estrutura bastante diferente do arquivo gerado para C++, o que inviabiliza a utilização da mesma ferramenta para a análise do código.

No entanto, a análise sobre o Swift ainda se mostrou valiosa, porque nem todas as anomalias podem ocorrer, uma vez que a linguagem foi implementada com características que já a protegem de determinados erros, o que não era visto em linguagens mais antigas, como o próprio C++. Seguindo essa linha, estamos fazendo uma análise comparativa entre o Swift e C++ (referenciando uma linguagem antiga e uma mais moderna) de modo que possamos identificar o que mudou - quais possibilidades de anomalias já estão "naturalmente" eliminadas - e quais anomalias ainda podem ocorrer, construindo uma visão crítica que pode auxiliar os programadores no momento da concepção do código, de modo a criar casos de teste que cubram as potenciais anomalias.

Para o futuro, essa análise pode ser feita em mais linguagens modernas, a fim de comprovar o quão preocupantes essas anomalias são nos tempos atuais. Isso é importante na análise custo X benefício desse tipo de análise automática específica, uma vez que as diferentes linguagens podem requerer ou não a importância dessa verificação.

[1] Paul Ammann , Jeff Offutt, Introduction to Software Testing, Cambridge University Press, New York, NY, 2008