# Impacto de Aglomeração de Anomalias de Código em Projetos .Net

**André Hideki Eposhi, Willian Nalepa Oizumi**

Campus Paranavaí – Instituto Federal do Paraná(IFPR)
Paranavaí – PR – Brasil

aheposhi@gmail.com, willian.oizumi@ifpr.edu.br

Apesar dos atributos de qualidade, i.e., a qualidade de um item [IEEE Std 610.12 1990], serem essenciais para o sucesso de todo o projeto de software, muitos desenvolvedores negligenciam a ocorrência de anomalias de código que podem impactar negativamente em diversos atributos de qualidade.

Anomalias de código são estruturas recorrentes de implementação que podem indicar a presença de um problema de *design* [Fowler 1999]. Estudos da literatura têm indicado que aglomerações de anomalias de código, i.e., grupos de anomalias de código inter-relacionadas afetando uma localidade do código fonte [Oizumi et al. 2016], são mais propensas a causar problemas de *design* [Ferreira 2014] [Oizumi et al. 2016]. Porém, ainda, há pouca, evidências sobre quais as combinações de anomalias são mais propensas a afetar negativamente atributos de qualidade.

Sendo assim, nesse trabalho pretende-se analisar aglomerações de anomalias de código heterogêneas que impactam negativamente nos diferentes atributos de qualidade, tais como, manutenibilidade, robustez e segurança. Também avaliar as combinações de anomalias de código que são mais propensas a prejudicar os atributos de qualidade.

Serão utilizadas ferramentas de análise de código estático como apoio para detectar diversos tipos de anomalias de código, tais como, *code smell*s e vulnerabilidades de segurança de software. Como exemplos de *code smells* podemos citar:

- *Feature Envy:* ocorre quando um método acessa os dados de outro objeto mais do que os dados do seu próprio objeto [Fowler 1999].
- *Large Class:* ocorre quando uma classe acumula muitas funcionalidades, atributos e métodos [Fowler 1999].
- *Long Method:* ocorre quando um método contém muitas linhas de código e acumula muitas responsabilidades [Fowler 1999].

Anley (2007) define vulnerabilidade como uma falha em um sistema que permite a um atacante usá-la de uma forma não prevista pelo projetista. Como exemplos de vulnerabilidade de segurança de software podemos citar:

- *Cross Site Scripting:* é o meio mais comum de ataques a cliente web, constituindo uma poderosa arma contra a rede interna das corporações [Dhanjani 2009].

- *SQL Injection:* é um ataque realizado quando um código SQL é inserido ou concatenado aos parâmetros de entrada fornecidos pelo usuário e posteriormente o código é encaminhado ao banco de dados que o interpreta e executa [Clarke 2009].
- *Cross Site Request Forgery:* é um ataque que rompe a integridade da sessão do usuário com um determinado sítio, injetando requisições de rede através do navegador da vítima. Os navegadores Web por meio de suas políticas de segurança permitem que sítios Web enviem solicitações HTTP de qualquer endereço de rede [Barth 2008].

Portanto, serão coletados os dados extraídos das ferramentas de análise de código estático, para assim analisá-los. As aglomerações de anomalias serão avaliadas manualmente e classificadas de acordo com a sua relevância. Dessa forma, pretende-se obter subsídios necessários para determinar quais aglomerações impactam negativamente nos atributos de qualidade e quais as combinações de anomalias de código são mais propensas a prejudicar os atributos de qualidade.

**Referências**

Anley C. (2007), The ShellCoder´s Handbook Discovering and Exploiting Security Holes, Wiley Publishing, Inc, 2nd Edition.

Barth, A., Jackson, C., e Mitchell, J. C., (2008) "Robust defenses for cross-site request forgery". In: Proceedings of the 15th ACM conference on Computer and communications security. (New York, New York, USA). 2008.

Clarke J. (2009), SQL Injection Attacks and Defense, Elsevier Inc, 2nd Edition.

Dhanjani, N. Hacking: The Next Generation, O´Reilly, 1st Edition.

Ferreira, M. (2014) "Detecção de anomalias de código de relevância arquitetural em sistemas multi linguagens". Dissertação de Mestrado, Universidade Pontifícia Católica, Rio de Janeiro, Brasil, 2014.

Fowler, M. (1999) Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional, 1st Edition.

IEEE Std. 610.12 (1990), IEEE Standard Glossary of Software Engineering Terminology. The Institute of Electrical and Electronics Engineers. (New York, New York, USA), 1990.

Oizumi W, Garcia A, Souza, A, S, Cafeo B, e Zao Y. (2016), "Code anomalies flock together: exploring code anomaly agglomerations for locating design problems". In Proceedings of the 38th International Conference on Software Enginer (ICSE´16) (New York, New York, USA). 2016.