Modularizing Code by Use Cases and Tests for Better Maintainability

Michal Bystrický and Valentino Vranić
Institute of Informatics, Information Systems and Software Engineering
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, Bratislava, Slovakia
michal.bystricky@stuba.sk,vranic@stuba.sk

CCS CONCEPTS

• **Software and its engineering → Abstraction, modeling and modularity**; **Object oriented development**; **Use cases**; **Acceptance testing**; *Object oriented architectures*; *Object oriented languages*;

KEYWORDS

use case; modularization; user acceptance tests; test driven development; Cucumber

ACM Reference format:
Michal Bystrický and Valentino Vranić. 2017. Modularizing Code by Use Cases and Tests for Better Maintainability. In *Proceedings of Programming '17, Brussels, Belgium, April 03-06, 2017,* 1 pages.
DOI: http://dx.doi.org/10.1145/3079368.3079379

1 BACKGROUND

In common object-oriented code, user acceptance tests, which play a very important role in any kind of software development process and which follow procedural modularization, would be scattered and, consequently, hard to maintain. This is of particular importance, since it is known that user acceptance tests immensely improve code comprehension, as can be seen in the Cucumber approach [4].

Use case driven modularization, which has gained some attention so far via various approaches, and which enables to preserve use cases in code, ranging from their structural preserving using aspect-oriented means [5], preserving them at the level of roles [3], up to preserving them literally [2], provides a very good basis for modularizing code according to user acceptance tests.

2 ENVIRONMENT

We developed a modularization environment that enables to work with both use case and test driven code modularization.[1] The environment, which enables to organize code according to use cases and tests simultaneously, and enforces their representations in code, is based on our previous work on literal inter-language use case driven modularization [1]. To achieve this, the environment uses static file processing.

[1]The environment and all results are available at useion.com.

Both use case and test driven modularizations remain compatible while at the same time open new capabilities of having different views on software based on use cases and various tests simultaneously. The environment synchronizes changes in multiple modularizations at the level of static file processing and enforces use case and test representations in code by displaying a percentage of how well use cases and tests are covered by code.

3 EXPERIENCE

In applying our environment to a real world application of 55 use cases built on the OpenCart e-commerce platform, we observed how use case and test driven modularization supported by our environment simplify development and maintenance. Moreover, the environment helps developers respond effectively to common change requests using object-oriented, use case driven, and test driven modularization interchangeably.

ACKNOWLEDGMENTS

This work was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under the grant No. VG 1/0752/14, Research & Development Operational Programme for the project Research of Methods for Acquisition, Analysis and Personalized Conveying of Information and Knowledge, ITMS 26240220039, co-funded by the ERDF, and STU Grant scheme for Support of Young Researchers (M. Bystrický).

REFERENCES

[1] Michal Bystrický and Valentino Vranić. 2016. Development Environment for Literal Inter-Language Use Case Driven Modularization. In *MODULARITY Companion 2016, Companion Proceedings of the 15th International Conference on Modularity, Modularity 2016, Modularity 2016 Demos & Posters.* ACM, Málaga, Spain.
[2] Michal Bystrický and Valentino Vranić. 2016. Literal Inter-Language Use Case Driven Modularization. In *Proceedings of LaMOD'16: Language Modularity À La Mode, workshop, Modularity 2016.* ACM, Málaga, Spain.
[3] James Coplien and Gertrud Bjørnvig. 2010. *Lean Architecture for Agile Software Development.* Wiley.
[4] Shankar Garg. 2015. *Cucumber Cookbook.* Packt Publishing.
[5] Ivar Jacobson and Pan-Wei Ng. 2004. *Aspect-Oriented Software Development with Use Cases.* Addison-Wesley.