On Controlling the Attack Surface of Object-Oriented Refactorings

Sebastian Ruland[1], Géza Kulcsár[1], Erhan Leblebici[1], Sven Peldszus[2], Malte Lochau[1]

Abstract: The results of this work have originally been published in [Ru18]. Refactorings constitute an effective means to improve quality and maintainability of evolving object-oriented programs. Search-based techniques have shown promising results in finding near-optimal sequences of behavior-preserving program transformations that (1) maximize code-quality metrics and (2) minimize the number of code changes. However, the impact of refactorings on non-functional properties like security has received little attention so far. To this end, we propose, as a further objective, to minimize the *attack surface* of object-oriented programs (i.e., to maximize strictness of declared accessibility of class members). Minimizing the attack surface naturally competes with applicability of established refactorings like *MoveMethod*, frequently used for improving code quality in terms of coupling/cohesion measures. Our tool implementation is based on an EMF meta-model for JAVA-like programs and utilizes *MOMoT*, a search-based model-transformation and optimization framework. Our experimental results gained from applying different accessibility-control strategies to a collection of real-world JAVA programs show the impact of attack surface minimization on design-improving refactorings. We further compare the results to those of existing refactoring tools.

Keywords: Object-Oriented Refactorings; Search-based Refactorings; Attack Surface; Model Transformation

1 Summary

One major challenge in designing object-oriented programs is to solve the *Class-Responsibility-Assignment (CRA)* problem, by identifying candidates for classes and assigning related *responsibilities* (i.e., program data and operations) to them. In modern software development practices, especially in agile development, these responsibilities frequently change over time. Therefore, program-evolution phases are accompanied by refactorings (i.e., behavior-preserving code transformations) to update CRA decisions. Hence, refactorings help to maintain or even improve code quality over time [Fo00]. In this regard, the different, and often contradicting, objectives of refactorings are (1) to optimize code-quality metrics (e.g., coupling and cohesion or anti-pattern avoidance), and (2) to preserve the initial program design as much as possible, by minimizing the number of changes executed by the refactoring [Ca16]. A manual search for refactorings constituting

[1] TU Darmstadt, Real-Time Systems Lab, Magdalenenstr. 4, 64289 Darmstadt, Germany, sebastian.ruland@es.tu-darmstadt.de, malte.lochau@es.tu-darmstadt.de
[2] University of Koblenz-Landau, Institute for Software Technology, Universitätsstraße 1, 56070 Koblenz, Germany, speldszus@uni-koblenz.de

doi:10.18420/SE2020_26

a reasonable trade-off between both objectives is often impractical in case of real-world programs, as each refactoring consists of sequences of interdependent code transformations accompanied by complex applicability constraints. Due to the large search space and the multitude of contradicting objectives, search-based optimization techniques constitute a promising approach to find applicable refactorings. There exists recent work considering various semi-automated approaches for specific refactorings, where validity of possible refactorings is mostly concerned with purely functional behavior preservation [Ou16].

In contrast, our proposed methodology, called GOBLIN, further takes into account *attack-surface metrics* as well-known indicator for program security. In particular, the *attack surface* of a program comprises all conventional ways of entering a software by users/attackers. Therefore, a large attack surface increases the danger of vulnerability exploitation. Hence, we consider minimization of the attack surface (i.e., granting least privileges to class members) as an additional *non-functional* optimization objective during refactoring. To this end, GOBLIN utilizes search-based optimization techniques aiming to find (near-)optimal sequences of refactorings for JAVA-like programs, where we additionally take accessibility constraints into account. We apply already established refactoring operations, namely *move-method* refactorings, and optimize different code-quality metrics as well as the impact on the attack surface. Thus, we have to strengthen or weaken accessibility declarations on demand to ensure correctness of accessibility modifiers after refactoring. As further objectives for code optimization, we consider *elimination of design flaws* (i.e., by optimizing coupling and cohesion measures and avoiding anti-patterns such as the *The Blob*)and *preservation of the original program design* (i.e., minimizing the number of changes). We evaluated our approach by applying GOBLIN to a collection of real-world programs. Besides showing the applicability of our tool, the evaluation results further provide in-depth insights into the interplay between traditional code-quality metrics and attack-surface metrics.

Acknowledgments. This work was funded by the Hessian LOEWE initiative within the Software-Factory 4.0 project.

Bibliography

[Ca16] Candela, Ivan; Bavota, Gabriele; Russo, Barbara; Oliveto, Rocco: Using Cohesion and Coupling for Software Remodularization: Is It Enough? ACM Trans. Softw. Eng. Methodol., 25(3):24:1–24:28, June 2016.

[Fo00] Fowler, Robert: Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2000.

[Ou16] Ouni, Ali; Kessentini, Marouane; Sahraoui, Houari A.; Inoue, Katsuro; Deb, Kalyanmoy: Multi-Criteria Code Refactoring Using Search-Based Software Engineering: An Industrial Case Study. ACM Trans. Softw. Eng. Methodol., 25(3):23:1–23:53, 2016.

[Ru18] Ruland, Sebastian; Kulcsár, Géza; Leblebici, Erhan; Peldszus, Sven; Lochau, Malte: Controlling the Attack Surface of Object-Oriented Refactorings. In: Fundamental Approaches to Software Engineering. Springer International Publishing, Cham, pp. 38–55, 2018.