Cataloging Dependency Injection Anti-patterns in Software Systems

Rodrigo Laigner[1], **Marcos Kalinowski**[2]

[1]Department of Computer Science (DIKU) – University of Copenhagen
Copenhagen, Denmark

rnl@di.ku.dk

[2]Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro – Rio de Janeiro, RJ – Brazil

kalinowski@inf.puc-rio.br

Abstract. *Dependency Injection (DI) is a prevalent technique employed in software systems. By delegating the dependency provision to an independent agent (i.e., a DI framework), developers often benefit from high-modular components. However, the misuse of the DI technique leads to higher maintenance efforts. As the literature presents no comprehensive characterization of bad DI implementation practices, we put forth an investigation that led to the proposition and evaluation of a novel catalog of DI anti-patterns and refactorings. The proposed anti-patterns appear frequently in both open-source and industry projects. Furthermore, practitioners confirm the relevance and usefulness of the catalog.*

Resumo. *Injeção de Dependência (ID) é uma técnica predominante em sistemas de software. Ao delegar o provisionamento de dependências para um agente independente (ou seja, um framework de ID), os desenvolvedores muitas vezes se beneficiam de componentes altamente modulares. No entanto, o uso incorreto da técnica de ID leva a maiores esforços de manutenção. Como a literatura não apresenta uma caracterização abrangente de más práticas de implementação de ID, apresentamos uma investigação que levou à proposição e avaliação de um novo catálogo de antipadrões e refatorações de ID. Os antipadrões propostos aparecem com frequência em projetos de código-fonte aberto e da indústria. Além disso, profissionais confirmam a relevância e a utilidade do catálogo.*

1. Background and Motivation

Dependency injection (DI) is a prevalent and industry-strength technique employed in software systems today. DI allows decoupling classes from their dependencies (through an interface-oriented design) by outsourcing the dependency provision process to an independent agent, the so-called DI container [Fowler 2004]. DI has become a traditional practice in industry as characterized by the existence of several popular dependency injection frameworks, such as Spring and Guice, and the embedded support of dependency injection in modern frameworks such as .NET Core from Microsoft. Although the DI practice is popular, the implementation of DI is not trivial and demands in-depth knowledge on object-oriented design. Most importantly, improper dependency injection usage may actually hinder the effective achievement of its main goal, loose-coupling.

While the technical and white literature conjecture about the existence of dependency injection anti-patterns and smells, these propositions do not provide a comprehensive analysis of the state of the practice For instance, the practical relevance of these propositions is unknown (e.g., the degree of occurrence in real projects and whether design principles are harmed), as well as their generalizability. Moreover, there is no evidence on the acceptance and perceived usefulness from the developers' point of view. Therefore, developers' needs are insufficiently met when reasoning about the benefits and trade-offs when it comes to adoption of dependency injection in the source code. Given this clear gap, the goal of this work is to provide a framework of ideas and tools to effectively address dependency injection in software systems.

2. Methodology

We started by reviewing suggestions of dependency injection anti-patterns and discussing their shortcomings. Next, we put forth two methodological approaches to derive an initial proposition of DI anti-patterns, inductive and deductive. We hypothesized about the existence of dependency injection practices that cause harm to modularity in light of object-oriented design principles, and provided a comprehensive set of anti-patterns that serves as an easy to use guideline for software developers in the wild. To validate our proposition, we built a static analysis tool to automatically identify instances of the proposed dependency injection anti-patterns in software systems. The tool showed high efficacy and it revealed that the DI anti-patterns appear frequently in both open-source and industry projects. Next, to empirically strengthen our contribution, we surveyed several expert developers on the benefits of each proposed anti-pattern and collected feedback on their practical relevance and usefulness. Lastly, for each anti-pattern, we also devised a respective refactoring suggestion.

3. Results and Concluding Remarks

The initial results of this work have been presented and awarded at SBES 2019 [Laigner et al. 2019] and an extended version with the complete results is currently under the revision at the Journal of Systems and Software [Laigner et al. 2021]. The catalog contains DI anti-patterns that occur in practice and are considered useful. The corresponding refactorings also show large acceptance from developers. Sharing it with practitioners can help them to avoid and to remove such anti-patterns, improving the quality of their code. Furthermore, given that dependency injection is a widely adopted practice in real-world settings, we hope that our work raises the awareness of the research community, helping to steer future investigations on the topic.

Referências

Fowler, M. (2004). Inversion of control containers and the dependency injection pattern.

Laigner, R., Kalinowski, M., Carvalho, L., Mendonça, D. S., and Garcia, A. (2019). Towards a catalog of java dependency injection anti-patterns. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering, SBES 2019, Salvador, Brazil, September 23-27*, pages 104–113.

Laigner, R., Mendonça, D., Garcia, A., and Kalinowski, M. (2021). Cataloging dependency injection anti-patterns in software systems. Under revision at the Journal of Systems and Software (JSS) - e-print available at https://arxiv.org/abs/2109.04256.