An Evolutionary Analysis of Software-Architecture Smells

Philipp Gnoyke[1] Sandro Schulze[2] Jacob Krüger[3]

Abstract: In this extended abstract, we summarize our paper with the homonymous title published at the International Conference on Software Maintenance and Evolution (ICSME) 2021 [GSK21].

Keywords: software maintenance; software evolution; architecture smells; software quality; technical debt; empirical study

Background: Since user and technological requirements evolve over time, software systems must be maintainable to efficiently adapt; or they are at risk of becoming irrelevant. Design principles aim to keep software understandable, changeable, extensible, reusable, and testable. Violations of such principles on an architectural level are described as *Architecture Smells* (ASs), which often indicate degrading system quality. The amount and severity of smells in a system can furthermore be portrayed with *Technical Debt* (TD), which refers to *interest* in the form of reduced maintainability. Postponing or abandoning quality assurance can lead to *technical bankruptcy* if TD renders a system's evolution economically or practically impossible. In our study, we focused on three types of ASs: First, a *Cyclic Dependency* (CD) is a set of components (classes or packages) that depend on each other, so that their dependency graph forms a cycle. Second, a *Hub-Like Dependency* (HD) represents a component with a high number of incoming and outgoing dependencies. Finally, an *Unstable Dependency* (UD) describes a component that depends on less stable components than itself.

Objective: Only few studies have analyzed the evolution of ASs, as well as their long-term influence on software degradation and TD. A noteworthy study has been performed by Sas et al. [SAAF19], upon which we built. We aimed to expand the current understanding of ASs by providing a more precise conceptual representation of the evolution of ASs and TD, as well as by interpreting empirical observations. For this purpose, we analyzed how the number and composition of ASs in software projects evolve over time. Additionally, we studied how the evolution of ASs relates to the evolution of TD, and how the former impacts the latter. Finally, we identified factors influencing the lifespan of ASs.

Method: We propose an improved technique for tracking ASs over the version history of systems. This especially concerns CDs, for which we introduce the distinction between *subcycles* and *supercycles*. Since supercycles can merge or split between versions, we track their evolution in a novel way with *cyclic dependency evolution graphs*. Furthermore, we propose to distinguish *intra-* and *inter-version smells*, with the former being a smell

[1] Otto-von-Guericke University Magdeburg, Germany philipp.gnoyke@t-online.de
[2] University of Potsdam, Germany sandro.schulze@uni-potsdam.de
[3] Ruhr-University Bochum, Germany jacob.krueger@rub.de

instance in a particular version and the latter being a set of related intra-version smells over multiple versions. We define several properties for both types to assess their severity and growth/lifetime patterns. To empirically evaluate our new concepts and answer our research questions, we analyzed the evolution of ASs and TD in 14 open-source systems from the *Qualitas Corpus* with a total of 485 versions. For detecting intra-version smells and calculating properties, we used a modified version of *Arcan* [Ar17]. Moreover, we implemented our own tool *ASTDEA* to track and compute properties of inter-version smells.

Results: First, we found that TD and the number of ASs tend to increase along the code size of systems. When looking at their relative sizes, they remained mostly stable and decreased in some systems. So, we could not confirm exponential growth patterns. However, we found many ASs that, after being introduced, persisted for the entire observation period. Second, we noted that some AS types like CDs had a greater impact on TD and system degradation, while not always corresponding to their share on the number of ASs. Rather, aspects like the complexity of ASs seem to be more pivotal regarding their impact. Finally, we observed no universal patterns between the lifespan and certain properties of all AS types. However, especially complex CDs among classes tended to persist longer. For UDs, we observed this longevity in case that they overlapped with other smells or had a large difference in stability.

Conclusion: Our results indicate that it is in the interest of practitioners to remove ASs early on to prevent their manifestation, which especially holds true for CDs. Our improved technique for conceiving AS and TD evolution helps practitioners to identify and contend system degradation, as well as researchers to inspire new studies on ASs and TD.

Data Availability: Our replication package[4] contains the source code and compilation of ASTDEA and our modified version of Arcan, the raw output dataset, the queries we used to aggregate information from it, as well as additional diagrams and evaluations.

Bibliography

[Ar17] Arcelli Fontana, Francesca; Pigazzini, Ilaria; Roveda, Riccardo; Tamburri, Damian; Zanoni, Marco; Di Nitto, Elisabetta: Arcan: A Tool for Architectural Smells Detection. In: Proceedings of the International Conference on Software Architecture Workshops (ICSAW). IEEE, 2017.

[GSK21] Gnoyke, Philipp; Schulze, Sandro; Krüger, Jacob: An Evolutionary Analysis of Software-Architecture Smells. In: Proceedings of the International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2021.

[SAAF19] Sas, Darius; Avgeriou, Paris; Arcelli Fontana, Francesca: Investigating Instability Architectural Smells Evolution: An Exploratory Case Study. In: Proceedings of the International Conference on Software Maintenance and Evolution (ICSME). IEEE, 2019.

[4] `https://figshare.com/s/fa17e81cf4f27c84d059`