The Journal of Open Source Software

s4rdm3x: A Tool Suite to Explore Code to Architecture Mapping Techniques

Tobias Olsson[1], **Morgan Ericsson**[1], **and Anna Wingkvist**[1]

1 Department of Computer Science and Media Technology, Linnaeus University, Sweden

DOI: 10.21105/joss.02791

Software

- Review
- Repository
- Archive

Editor: George K. Thiruvathukal

Reviewers:

- @kinow
- @xirdneh

Submitted: 03 August 2020
Published: 07 February 2021

Summary

Architectural drift and erosion, where the implementation starts to deviate from the intended software architecture or the rules set by it, are common problems in long-lived software systems. This can be avoided by using techniques, such as Reflexion modeling (Murphy et al., 1995), to validate that the implementation conforms to the indented architecture. Unfortunately, such techniques require a mapping from source code modules (e.g., classes) to elements of the architecture, something that is not always available or up to date. This is a known problem when, e.g., companies want to adopt static architecture conformance checking; the effort to manually create or bring this mapping up to date is just too time-consuming and error-prone (Ali et al., 2017; Bittencourt et al., 2010).

The `s4rdm3x` tool suite is designed for researchers and practitioners to study and evaluate algorithms that perform part of the mapping automatically, such as orphan-adoption clustering (Christl et al., 2007) or information retrieval techniques (Bittencourt et al., 2010). It includes a graphical user interface to define and run experiments with mapping algorithms and their parameters, and visualize and explore the results of these experiments. The experiments can be executed locally or in a remote high-performance computing environment. The tool suite includes reference implementations of state of the art mapping algorithms and a set of Java systems with validated mappings between classes and architecture elements. The tool suite is extensible, so it is easy to add new mapping algorithms and visualizations to explore their performance.

Statement of Need

To faciliate the further development and evaluiation of mapping techniques the software provides reference implementations of the current state-of-the-art mapping techniques and the means to implement new techniques and run experiments. It includes the HuGMe orphan adoption clustering method (Christl et al., 2007), and four attraction functions to decide which architectural element a source code module should be mapped to: `CountAttract` (Christl et al., 2007), `IRAttract`, `LSIAttract` (Bittencourt et al., 2010) and `NBAttract` (Tobias Olsson et al., 2019). There is also a reference implementation of our novel technique to create a textual representation of source code dependencies at an architectural level; Concrete Dependency Abstraction (CDA). It also contains a set of validated mappings between source code classes and architectural elements that are often used in software architecture erosion research. These systems have either been recovered from replication packages (Brunet et al., 2012; Lenhard et al., 2019) or the SAEroCon workshop repository.

The Journal of Open Source Software

The s4rdm3x Tool Suite

S4rdm3x is an extensible suite of tools for source code analysis, architecture definition, mapping of source code modules to architecture elements, experiment definitions, and exploratory and visual analysis. The suite consists of an *extensible core* and two tools, a *graphical editor* to create and visualize mapping experiments and a *command-line tool to run experiment* at scale.

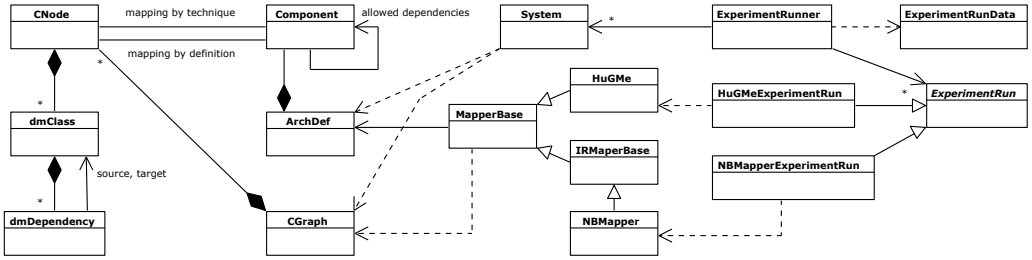

Figure 1: Overview of the s4rdm3x Core showing the implementation metamodel, and the mapper and experiment subsystem

The *core* provides Java bytecode analysis to extract a dependency graph (and naming information) as well as loading an architectural definition and source to module mapping. Figure 1 provides an overview of the important classes in the core and their main dependencies; the three leftmost classes represent source code modules, and their implemented dependencies contained as a graph and the architecture is represented by components and their allowed dependencies. There is a rich set of dependencies that are extracted from (Java) byte code, including the possibility to include implicit dependencies found via hard-coded constants. The `MapperBase` class is used to implement different mapping strategies as subclasses. `MapperExperiment` provides functionality to set up and run mapping experiments using combinations of random parameters at different intervals. An experiment is implemented as a subclass of `MapperExperiment` that instantiates the corresponding subclasses of `MapperBase`. This means that the mappers are not exclusive for experiments and can be reused in other situations, e.g., in a tool that performs semi-automatic mapping as part of a reflexion modeling approach.

The *graphical editor* is used to define, visualize, analyze, and compare how well mapping algorithms perform with different parameters and initial sets of known mappings. It supports a range of visualizations and can be extended with new ones. The editor uses an Immediate Mode GUI approach, where the application renders the graphical primitives it needs (e.g., lines, rectangles, and points) every frame, an approach often used in computer games and tools used for computer game development since it offers fine-grained control over the visualization. This fine-grained control makes it possible to extend the editor with custom visualizations. The GUI uses OpenGL to provide hardware-accelerated rendering.

The graphical editor can be used to run experiments, but these generally require a large number of combinations of, e.g., parameters, initial sets, and systems, so they can take a long time to run. The suite includes a command-line tool that runs these combinations in parallel on many-core machines. The command-line tool can read experiment definitions in XML exported from the graphical editor and save the results in a format that can be imported and visualized.

S4rdm3x is implemented in Java and depends on ASM, Weka, and Dear JVM ImGui.

Applications

The `S4rdm3x` tool suite has been used in research studies on orphan adoption (T. Olsson et al., 2018; Tobias Olsson et al., 2019) and as a continuous integration tool-chain for static architecture conformance checking of student project submissions.

Acknowledgments

This work is supported by the Linnaeus University Centre for Data Intensive Sciences and Applications (DISA) High-Performance Computing Center.

References

Ali, N., Baker, S., O'Crowley, R., Herold, S., & Buckley, J. (2017). Architecture consistency: State of the practice, challenges and requirements. *Empirical Software Engineering*, *23*(1), 1–35.

Bittencourt, R. A., Jansen de Souza Santos, G., Guerrero, D. D. S., & Murphy, G. C. (2010). Improving automated mapping in reflexion models using information retrieval techniques. *IEEE Working Conference on Reverse Engineering*, 163–172. https://doi.org/10.1109/WCRE.2010.26

Brunet, J., Bittencourt, R. A., Serey, D., & Figueiredo, J. (2012). On the evolutionary nature of architectural violations. *Working Conference on Reverse Engineering*, 257–266. https://doi.org/10.1109/wcre.2012.35

Christl, A., Koschke, R., & Storey, M.-A. (2007). Automated clustering to support the reflexion method. *Information and Software Technology*, *49*(3), 255–274. https://doi.org/10.1016/j.infsof.2006.10.015

Lenhard, J., Blom, M., & Herold, S. (2019). Exploring the suitability of source code metrics for indicating architectural inconsistencies. *Software Quality Journal*, *27*(1), 241–274. https://doi.org/10.1007/s11219-018-9404-z

Murphy, G. C., Notkin, D., & Sullivan, K. (1995). Software reflexion models: Bridging the gap between source and high-level models. *ACM SIGSOFT Software Engineering Notes*, *20*(4), 18–28.

Olsson, Tobias, Ericsson, M., & Wingkvist, A. (2019). Semi-automatic mapping of source code using naive bayes. *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, 209–216. https://doi.org/10.1145/3344948.3344984

Olsson, T., Ericsson, M., & Wingkvist, A. (2018). Towards improved initial mapping in semi automatic clustering. *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, 51:1–51:7. https://doi.org/10.1145/3241403.3241456