Quality Validation through Pattern Detection – A Semantic Web Perspective

David Walsh[1], Philipp Schugerl[1], Juergen Rilling[1], Philippe Charland[2]

[1]*Department of Computer Science and Software Engineering*
Concordia University, Montreal, Canada
{da_wals, p_schuge, rilling}@cse.concordia.ca

[2]*System of Systems Section*
Defence R&D Canada – Valcartier
Quebec, Canada
philippe.charland@drdc-rddc.gc.ca

***Abstract*— Given the ongoing trend towards the globalization of software systems, open networks, and distributed platforms, validating non-functional requirements and quality becomes essential. Our research addresses this challenge from two different perspectives: (1) the integration of knowledge and tool resources through Semantic Web technologies as part of our SE-PAD environment, in order to reduce or eliminate existing traditional information and analysis silos. (2) The ability to reason upon linked resources to infer both explicit and implicit patterns to support the validation of quality aspects. We have applied our SE-PAD environment for the detection of security and design patterns, as well as the violations of secure programming guidelines.**

Keywords-patterns; Semantic Web; software comprehension; software quality

I. INTRODUCTION

Patterns [1, 6] and programming guidelines have been promoted for some time to help improve various quality aspects of the final software product and to detect problematic areas. One approach to detect common coding problems or to enforce best practice coding patterns are manual code reviews. However, the quality of these reviews will largely depend on the expertise of the reviewer, with respect to the patterns and guidelines to be validated, and the thoroughness of the review process itself. The challenge is that these already implemented problems or bad practices could remain undiscovered for years to come, leading to situations where people (1) reuse code with flaws without being aware of them or even worst (2) the same mistakes or bad programming practices are repeated and become recurring patterns. These issues become even more aggravating in our global software economy, with its collaborative workspaces and diversified knowledge distribution among project stakeholders.

The objective of our research is to provide a novel approach that takes advantage of Semantic Web technologies to represent software artifacts and related knowledge resources. This builds the basis to (1) integrate and eliminate some of these existing technology and knowledge silos. (2) Take advantage of Semantic Web technologies for implicit and explicit pattern detection and to ensure that quality guidelines are followed. (3) Demonstrate through various motivating examples and case studies the flexibility and applicability of our approach.

The remainder of the article is organized as follows. Section 2 provides an overview of our SE-PAD tool. Quality validation and pattern detection using SE-PAD are discussed in Section 3. Conclusions and future work are presented in Section 4.

II. THE SE-PAD APPROACH

Capturing programming expertise and best practices is a well-recognized research and application domain. Many forms of patterns (e.g., security, design, anti-patterns) and programming guidelines have been established, describing their benefits, limitations, and potential application contexts. The objective of our research is not to compete with existing specialized tools. Rather, we see SE-PAD as a complementary knowledge integration approach. Furthermore, given the globalization of software development processes, we believe that a standardized representation using Semantic Web technologies [2] will become an enabling factor. As part our SE-PAD approach (Figure 1), we first parse Java code to extract the Abstract Syntax Tree (AST) of the program to be analyzed. The extracted facts are automatically populated [3] in our source code ontology, creating individuals. In a next step, Racer, a DL reasoner [4], is used to automatically infer additional individuals that are defined in the model as part of cardinality restrictions and stored in our model. We use lightweight reasoning to ensure that our implementation meets our scalability and performance requirements. Finally, SPARQL queries are applied to the triple store for knowledge exploration and analysis.

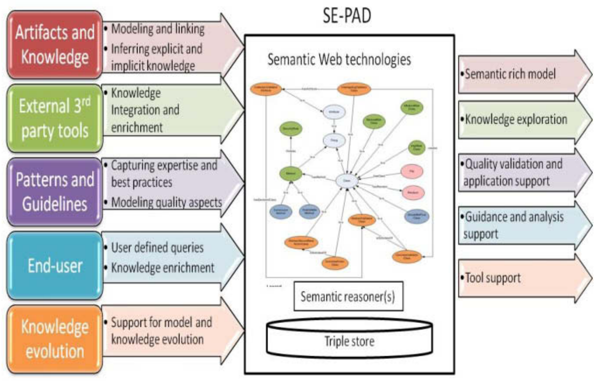
SE-PAD
Artifacts and Knowledge
• Modeling and linking
• Inferring explicit and implicit knowledge
External 3rd party tools
• Knowledge Integration and enrichment
Patterns and Guidelines
• Capturing expertise and best practices
• Modeling quality aspects
End-user
• User defined queries
• Knowledge enrichment
Knowledge evolution
• Support for model and knowledge evolution
Semantic Web technologies
Semantic reasoner(s)
Triple store
•Semantic rich model
•Knowledge exploration
•Quality validation and application support
•Guidance and analysis support
•Tool support

Figure 1. A general overview of our SE-PAD approach

III. QUALITY AND PATTERN DETECTION

In what follows, we discuss how we use Semantic Web technologies in our SE-PAD approach to support the modeling and detection of patterns and the validation of coding guidelines. Our approach focuses on the integration of knowledge and resources, which are essential aspects to eliminate current information and analysis tools silos and support a more global quality validation perspective.

Programming Guidelines. As part of our SE-PAD approach, we support pre-defined and user-defined semantic source code queries, similar to [5]. A major difference between our and existing approaches is that we use ontologies for the modeling of the source code facts and analysis results. Given our ontological representation, we can furthermore use semantic reasoners to infer both explicit and implicit knowledge from the existing models. Results from these validations can be used to enrich the source code ontology and perform additional validation and pattern detection.

Pattern Detection. Patterns represent existing solutions in a structured way and allow knowledge from these already existing solutions to be reused. Patterns can also speed up the development process by providing tested and proven development paradigms that might often not become visible until later in the implementation or the evolution of a system. Different domain specific patterns exist (e.g., design patterns [6], security patterns [3]). As part of our SE-PAD environment, we have formalized a subset of the GoF [6] design patterns that can be applied to improve system quality.

Tool Integration. The objective of our approach is to highlight the flexibility of Semantic Web technologies to support different forms of pattern detection and knowledge integration. As part of our approach, we support the import of results from external tools (e.g., FindBugs [5]) and make them an integral part of our source code model. For example, methods that are reported by FindBugs to contain malicious code vulnerabilities are populated as part of new concepts in our source code ontology. As part of a further analysis, classes can now be automatically classified based on their ratio of malicious code vulnerabilities.

Cross Artifact Analysis. Our source code ontology can be combined with knowledge from other software related artefacts. As part of this cross artefact analysis, the version control ontology is linked to our enriched source code ontology through common concepts. The linked ontologies can then be queried for further knowledge exploration.

IV. CONCLUSIONS

In this research, we focused on the integration of resources and knowledge related to patterns within a common ontological representation. The use of Semantic Web technologies as part of our SE-PAD tool implementation does not only support the integration of knowledge resources at various abstraction and semantic levels. It also provides the foundation for an evolving knowledge base that supports the extension of new resources and patterns.

As part of future work, we plan to further enrich our ontologies with additional knowledge resources and perform additional evaluations of our SE-PAD tool.

Acknowledgement:
This research was partially funded by DRDC Valcartier (contract no. W7701-081745/001/QCV).

References

[1] B. Blakley, C. Heath, and Members of the Open Group Security Forum, *Security design patterns*, 2004.
[2] T.R. Gruber, “A Translation Approach to Portable Ontology Specifications,” *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.
[3] Jena Semantic Web Framework, http://jena.sourceforge.net/index.html.
[4] Racer Systems, http://www.racer-systems.com/index.phtml.
[5] FindBugs, http://findbugs.sourceforge.net/.
[6] E. Gamma, R., et al., *Design Patterns: Elements of Reusable Object-oriented Software*, Addison-Wesley Professional, 1st Edition, 1994.