Software Engineering and the Semantic Web: A match made in heaven or in hell?

Bernstein, Abraham

Abstract: The Semantic Web provides models and abstractions for the distributed processing of knowledge bases. In Software Engineering endeavors such capabilities are direly needed, for ease of implementation, maintenance, and software analysis. Conversely, software engineering has collected decades of experience in engineering large application frameworks containing both inheritance and aggregation. This experience could be of great use when, for example, thinking about the development of ontologies. These examples — and many others — seem to suggest that researchers from both fields should have a field day collaborating: On the surface this looks like a match made in heaven. But is that the case? This talk will explore the opportunities for cross-fertilization of the two research fields by presenting a set of concrete examples. In addition to the opportunities it will also try to identify cases of fools gold (pyrite), where the differences in method, tradition, or semantics between the two research fields may lead to a wild goose chase.

Software Engineering and the Semantic Web: A match made in heaven or in hell?

Abraham Bernstein*

Dynamic and Distributed Information Systems Group
Department of Informatics, University of Zürich
Binzmühlestrasse 14, 8050 Zürich, Switzerland
<lastname>@ifi.uzh.ch
http://www.ifi.uzh.ch/ddis/bernstein.html

Abstract. The Semantic Web provides models and abstractions for the distributed processing of knowledge bases. In Software Engineering endeavors such capabilities are direly needed, for ease of implementation, maintenance, and software analysis.
Conversely, software engineering has collected decades of experience in engineering large application frameworks containing both inheritance and aggregation. This experience could be of great use when, for example, thinking about the development of ontologies.
These examples—and many others—seem to suggest that researchers from both fields should have a field day collaborating: On the surface this looks like a match made in heaven. But is that the case?
This talk will explore the opportunities for cross-fertilization of the two research fields by presenting a set of concrete examples. In addition to the opportunities it will also try to identify cases of fools gold (pyrite), where the differences in method, tradition, or semantics between the two research fields may lead to a wild goose chase.

Keywords: Software Engineering, Semantic Web, Software Analysis, Knowledge Representation, Statistics, Ontologies

The Semantic Web at use for Software Engineering Tasks and its Implications

Semantic Web technologies have successfully been used in recent software engineering research. Dietrich [2], for example, proposed an OWL[1] ontology to model the domain of software design patterns [4] to automatically generate documentation about the patterns used in a software system. With the help of this

* I would like to thank the participants of the Semantic Web Software Engineering Workshop (SWSE) at the International Semantic Web Conference (ISWC) 2009 for their valuable comments on an earlier version of this presentation. Partial support for some of the work presented in this talk was provided by Swiss National Science Foundation award number 200021-112330.

[1] http://www.w3.org/TR/owl2-overview/

ontology, the presented pattern scanner inspects the abstract syntax trees (AST) of source code fragments to identify the patterns used in the code.

Highly related is the work of Hyland-Wood *et al* [6], in which the authors present an OWL ontology of *Software Engineering Concepts (SECs)*. Using SEC, it is possible to enable language-neutral, relational navigation of software systems to facilitate software understanding and maintenance. The structure of SEC is very similar to the language structure of Java and includes information about classes and methods, test cases, metrics, and requirements of software systems. Information from versioning and bug-tracking systems is, however, not modeled in SEC.

Both, Mäntylä *et al* [7] and Shatnawi and Li [8] carried out an investigation of *code smells* [3] in object-oriented software source code. While the study of Mäntylä additionally presented a taxonomy (i.e., an ontology) of smells and examined its correlations, both studies provided empirical evidence that some code smells can be linked with errors in software design.

Happel *et al* [5] presented the *KOntoR* approach that aims at storing and querying metadata about software artifacts in a central repository to foster their reuse. Furthermore, various ontologies for the description of background knowledge about the artifacts such as the programming language and licensing models are presented. Also, their work includes a number of SPARQL queries a developer can execute to retrieve particular software fragments which fit a specific application development need.

More recently Tappolet *et al* [9] presented EvoOnt,[2] a set of software ontologies and data exchange formats based on OWL. EvoOnt models software design, release history information, and bug-tracking meta-data. Since OWL describes the semantics of the data, EvoOnt (1) is easily extendible, (2) can be processed with many existing tools, and (3) allows to derive assertions through its inherent Description Logic reasoning capabilities. The contribution of their work is that it introduces a novel software evolution ontology that vastly simplifies typical software evolution analysis tasks. In detail, its shows the usefulness of EvoOnt by repeating selected software evolution and analysis experiments from the 2004-2007 Mining Software Repositories Workshops (MSR). The paper demonstrates that if the data used for analysis were available in EvoOnt then the analyses in 75% of the papers at MSR could be reduced to one or at most two simple queries within off-the-shelf SPARQL[3] tools. In addition, it presents how the inherent capabilities of the Semantic Web have the potential of enabling new tasks that have not yet been addressed by software evolution researchers, e.g., due to the complexities of the data integration.

This semi-random[4] selection *examples clearly illustrate the usefulness of Semantic Web technologies for Software Enginering.*

[2] `http://www.ifi.uzh.ch/ddis/research/evoont/`
[3] `http://www.w3.org/TR/rdf-sparql-query/`
[4] All of the examples are focused on Semantic Web enabled software analysis.

Conversely, decades of experience in engineering large application frameworks containing both inheritance and aggregation provides a sound foundation for the usefulness of *Software Engineering techniques to Semantic Web research.*

These insights seem to suggest that researchers from the fields should have a field day collaborating: On the surface this looks like a match made in heaven. But is that the case?

The main purpose of this talk is to *identify the opportunities for cross-fertilization between the two research fields* by presenting a set of concrete examples.

To contrast these opportunities it will also try to *identify possible barriers and/or impediments to collaboration,* where the differences in method, tradition, or semantics between the two research fields may lead to a wild goose chase.

References

1. S. Demeyer, S. Tichelaar, P. Steyaert, FAMIX 2.0—the FAMOOS inf. exchange model, Tech. rep., University of Berne, Switzerland (1999). `http://www.iam.unibe.ch/~famoos/FAMIX/Famix20/Html/famix20.html`
2. J. Dietrich, C. Elgar, A formal description of design patterns using OWL, in: Proceedings of the Australian Software Engineering Conference, Brisbane, Australia, 2005.
3. M. Fowler, Refactoring, Addison-Wesley, 1999.
4. E. Gamma, R. Helm, R. E. Johnson, Design Patterns. Elements of Reusable Object-Oriented Software, Addison-Wesley Longman, 1995.
5. H.-J. Happel, A. Korthaus, S. Seedorf, P. Tomczyk, KOntoR: An Ontology-enabled Approach to Software Reuse, in: Proceedings of the 18th Internatinal Conference on Software Engineering and Knowledge Engineering (SEKE), San Francisco, CA, 2006.
6. D. Hyland-Wood, D. Carrington, S. Kapplan, Toward a Software Maintenance Methodology using Semantic Web Techniques, in: Proceedings of the 2nd ICSM International Workshop on Software Evolvability (SE), 2006, pp. 23–30.
7. M. Mäntylä, J. Vanhanen, C. Lassenius, A Taxonomy and an Initial Empirical Study of Bad Smells in Code, in: Proceedings of the International Conference on Software Maintenance (ICSM), 2003, pp. 381–384.
8. R. Shatnawi, W. Li, A Investigation of Bad Smells in Object-Oriented Design Code, in: Proceedings of the 3rd International Conference on Information Technology: New Generations, 2006, pp. 161–165.
9. J. Tappolet, C. Kiefer, A. Bernstein, Semantic web enabled software analysis, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 8, July 2010.