The 16th IEEE International Conference on Program Comprehension

Semantic Technologies in System Maintenance

Juergen Rilling
Concordia University
Department of Computer Science and Software Engineering
Montréal, Québec, Canada
rilling@cse.concordia.ca

René Witte
University of Karlsruhe
Institute for Program Structures and Data Organization (IPD)
Karlsruhe, Germany
witte@ipd.uka.de

Dragan Gašević
Athabasca University
School of Computing and Information Systems
Athabasca, Alberta, Canada
dragang@athabascau.ca

Jeff Z. Pan
The University of Aberdeen
Department of Computing Science
Aberdeen, Scotland, UK
jpan@csd.abdn.ac.uk

Abstract

This paper gives a brief overview of the International Workshop on Semantic Technologies in System Maintenance. It describes a number of semantic technologies (e.g., ontologies, text mining, and knowledge integration techniques) and identifies diverse tasks in software maintenance where the use of semantic technologies can be beneficial, such as traceability, system comprehension, software artifact analysis, and information integration.

1. Introduction

While software is a technical category designed to perform specific tasks by using computer hardware, it is also a social category that nowadays is used in almost every aspect of a human's life. Software is also a knowledge repository, where knowledge is largely related to the application domain, and not to software as an entity. Software engineers and maintainers need to be able to share and interrelate knowledge stored in software with the knowledge about all relevant aspects surrounding and influencing software maintenance (e.g., domain knowledge, new requirements, policies, and the contexts, in which people use and interact with software) in order to bring *system maintenance* to more advanced levels. This "System Maintenance 2.0" requires the use of semantically rich representations of knowledge coupled with advanced techniques for knowledge capturing, processing, and integration.

Semantic technologies have more recently become prominent with research in the Semantic Web and Web 2.0, both of which have seen first applications in software maintenance. Beyond these, many well-known knowledge management technologies, such as text and data mining, are also becoming increasingly important in the software domain.

In this paper, we discuss some of the most relevant aspects for the use of semantic technologies in solving system maintenance tasks. In order to bring together researchers from different communities to explore this rather timely research topic, we organized the *International Workshop on Semantic Technologies in System Maintenance (STSM),* collocated with the 16th IEEE International Conference on Program Comprehension (ICPC).

2. Semantic Technologies

In this section, we briefly introduce some of the most relevant aspects that are contributing to this synergistic space among semantic technologies and their application in system maintenance.

2.1 Ontologies and Reasoning

The term "ontology" originates from philosophy, where it denotes the study of existence. In computer science, the most common definition has been provided by Gruber [1]: *"An ontology is an explicit specification of a conceptualization."* Ontologies are typically used as a formal and explicit way of specifying the concepts and relationships in a domain of discourse. Ontologies can overcome portability, flexibility, and information sharing problems associated with databases. Compared to relational approaches, which assume complete knowledge (closed world assumption), ontologies support the modeling of incomplete knowledge (open world assumption) and extendibility of the ontological model.

Semantic Web technologies allow for machine understandable Web resources that can be shared and processed by both software tools (e.g., search engines) and humans. Ontologies are an important foundation of "Semantic Web"-enabled technologies, as they allow for both sharing knowledge between different agents and creating common terminologies for understanding. They are also an important step towards enrichment of services and content of the next generation of the Internet (so-called "Web 3.0").

The Web Ontology Language (OWL) is a standard put forward by the W3C (see www.w3.org/TR/owl-features/). It provides for creating machine-understandable information to enable the automatic processing and integration of Web resources. The sub-language OWL-DL is based on Description Logics (DLs) and allows for enriching platforms with *reasoning* services provided by DL-based knowledge representation systems. Unlike many logic programming approaches that cannot guarantee completeness, DL reasoning services are proven to be sound, complete, and terminating. Moreover, DL reasoning is automatic and does not require the development of logic programs to extract the desired inferences. DL reasoning is usually performed on demand and triggered by relevant queries to the knowledge base.

2.2 Text Mining and NLP

Dealing with the overwhelming amount of information readily available today is one of the biggest challenges in computer science. Database and information system technology together with information retrieval (IR) enables users to quickly obtain vast amounts of information in textual form. However, a serious bottleneck remains by reading, interpreting, and using the collected information. While a completely automated understanding of natural language is still impossible, there now exists a robust set of language technologies that can support specific tasks through semantic analyses based on language technologies, such as natural language processing (NLP).

This is addressed by the emerging research field of *Text Mining* [2], which developed from the observation that most knowledge today – more than 80% of the data stored in databases – is hidden within documents written in natural languages, and thus cannot be automatically processed by traditional data mining systems.

Text mining is a highly interdisciplinary field, which builds on foundations and technologies from information systems, natural language processing, and artificial intelligence. While already in wide use in domains like news analysis and biomedical knowledge extraction from research papers, text mining for the software engineering domain, using documents like user manuals, requirements specifications or bug reports, is still in its infancy.

2.3 Models & Metamodels

Model-driven engineering (MDE) is a new software engineering discipline in which the process heavily relies on the use of models [3, 4]. A model is a set of statements about some system under study. Models are usually specified by using modeling languages (e.g., UML), while modeling languages are defined by metamodels. A metamodel is a model of a modeling language. That is, a metamodel makes statements about what can be expressed in the valid models of a certain modeling language.

The OMG's Model Driven Architecture (MDA) is one possible architecture for MDE. The MDA introduces an approach that distinguishes between three different types of models, namely, computation-independent models (CIMs), platform-independent models (PIMs), and platform-specific models (PSMs). The important consequence is that, one can deploy the same system design (PIM) to many different platforms (PSM). One important characteristics of MDA is its organization, that is, four layers, including, M1 layer or model; M2 layer or metamodel; and M3 layer or metametamodel layer. The relations between different MDA layers can be considered as instance-of or conformant-to, which means that a model is an instance of a metamodel, and a metamodel is an instance of a metametamodel.

In this context, *model transformations* represent the central operation for processing models in MDE. Model transformations are the process of producing one model from another model of the same system. In fact, a model transformation means converting an input model, which conforms to one metamodel, to another model, which conforms to another metamodel. In November 2005, the OMG published the final specification of the MOF2 Query View Transformation (QVT) standard. While QVT covers the important scope of model-to-model transformations, to be able to fully support round-trip engineering, model-to-text and text-to-model transformations are the on-going research challenge in the MDE community.

2.4 Information Integration and Uncertainty

Models for representing *uncertain, incomplete,* or *inconsistent* information have a long tradition in database and information system research [5]. These works stem from the observation that data derived from the "real world" is rarely 100% complete, accurate, and consistent. By explicitly representing these *imperfections* within the underlying data models, reality can be more closely matched by the "mini world" models. This allows to capture more information, as transforming incomplete to precise data invariably leads to data loss (e.g., when only one of

two pieces of inconsistent information can be kept), and therefore should reduce development costs and improve overall system performance.
On the theoretical side, the various approaches differ in the knowledge representation formalism employed (e.g., fuzzy set theory or probabilistic models). A challenge until today is the question how these models can be integrated into existing database and software systems, without requiring extensive changes throughout the employed technologies. Closely related is the field of *information fusion* (also known as *information integration*), which emerged from the task of combining data from heterogeneous sources for data warehousing and data mining.

2.5 Semantic Desktops and Web 2.0 & 3.0

Recent research on ontologies suggests that ontologies are not just about symbols representing knowledge, but also about social interactions of the ontology users [6]. This notion has considerable influence on the adoption of Semantic Web technologies, as the construction, use, and evolution of ontologies is notably a difficult task. On the other hand, the Web 2.0 movement focuses on creating new knowledge through collaboration and social interactions of individuals on the Web (e.g. wikis, blogs, etc.). Collaborative tagging systems such as del.icio.us, Flickr, or BibSonomy provide intuitive ways for users to annotate Web resources. These systems use tags to reflect personal assertions about resources, and leverage these terms for recommending content to other members in the community, as well as for building a shared community vocabulary (called a folksonomy). However, Web 2.0 technologies in general, and collaborative tagging in particular, suffer from the problems of ambiguity in their tags' meanings and the lack of semantics (e.g., synonymy), the lack of a coherent categorization scheme, and the needed time and size of the community in which they will be used [7]. This can obviously be addressed by ontologies, clearly explaining why Semantic Web and Web 2.0 are complementary approaches that create Web 3.0 [8].

Looking from a software maintenance perspective, Web 3.0 brings many promising opportunities. For example, the collaborative nature of software engineering has more recently been addressed by introducing Wiki systems into the SE process. Semantic Wiki extensions like *Semantic MediaWiki* or *IkeWiki* add formal structuring and querying extensions based on RDF/OWL metadata. Moreover, local desktops of single users also benefit from the use of Web 3.0 technologies. This leads to the vision of the *Semantic Desktop*, where Web 3.0 is used to share knowledge resources (e.g., emails, blogs, and discussion forum posts) in a manner contextualized to the current needs of the users. From the software maintenance perspective, this opens further opportunities for a better integration of software artifacts and knowledge resources by leveraging the working *context* of software maintainers.

3. System Maintenance Challenges

The demand for software support of business processes is constantly increasing, leading to extensions of existing or the development of new systems. Either way, software maintainers have to deal with an ever-increasing code base. While the semantic technologies introduced above can facilitate their work, they need to be adapted to the specific application domain of system maintenance, which is a non-obvious task due to the complexity of both areas. A necessary first step is to communicate the specific challenges in system maintenance to developers of semantic technologies, which is one of the ideas behind our STSM workshop.

3.1 Software Comprehension

One of the major challenges for software engineers while performing maintenance tasks is the need to comprehend a multitude of often disconnected artifacts. These artifacts originate typically from the software development process and are typically revisited and modified multiple times during a system's life cycle.

As a result, maintainers often face the challenge to identify and comprehend different representations and interrelationships that exist among the software artifacts and knowledge resources involved in a particular maintenance task. From a maintainer's perspective, exploring and linking these artifacts and knowledge resources becomes a key challenge. What is needed are techniques and representations that allow maintainers to emerge, share and collaborative in the available knowledge and resource space.

Traditionally, the focus was on supporting maintainers with an analysis of the static and dynamic aspects of software systems, mainly derived from source code or analyzing run-time behavior. However, other artifacts, such as documentation, contain relevant semantic information that is normally difficult or even impossible to extract only from source code.

Thus, an increasingly important aspect of software comprehension becomes the extraction of semantic knowledge from artifacts other than the source code.

3.2 Traceability Recovery

While performing software maintenance, software engineers spend a large amount of effort on synthesizing and integrating information from various sources to establish links among these artifacts. Existing research in software traceability has mainly focused in the past on reducing the costs associated with this manual effort by developing automatic assistance in establishing traceability links among software artifacts during the software development process. Given the complexity of software, the number of systems already developed, and the money spent on keeping these systems maintained, (re-)establishing traceability among existing software artifacts becomes an important maintenance aspect. Among the existing traceability challenges are: (1) Establish links between incomplete, inconsistent and often not well defined artifacts and (2) the need to evolve existing links, to ensure quality and trustworthiness among them.

Research on re-establishing traceability between source code and documents has mainly focused on connecting documents and source code using Information Retrieval (IR) techniques. However, these approaches intrinsically ignore structural and semantical information that can be found in both documents and source code, limiting therefore both their precision and applicability. As a result, re-establishing, maintaining, and validating traceability links between existing software artifacts remains a major challenge.

3.3 Distributed Processes

Software is used to implement solutions that are expected to change periodically to adapt to ever changing environments. The efficient management and execution of these changes is critical to software quality and software evolution. Managing and supporting software maintenance creates ongoing challenges, due to the variations and interrelationships that exist among software artifacts, tool resources, maintenance processes, and tasks. Maintainers are often left with no or only limited guidance on how to complete a particular task within a given maintenance and organizational context, using a set of available resources (e.g., tools, artifacts). Existing tool and technique integration approaches face the same ongoing challenge due to a lack of integration standards that would allow for sharing services and knowledge among them. Only little work exists in examining how different tools and techniques work together. The situation is further complicated by the ever increasing globalization of both software development and maintenance processes, resulting in an additional need to also support *collaborative* software maintenance and knowledge sharing processes.

4. Conclusions

While our survey discusses some of the most relevant research areas, our list of technologies has definitely not covered all possible approaches that contribute to the core set of semantic technologies. Furthermore, the list of system maintenance challenges is provided just to stimulate community discussions and to start further exploration and leveraging of semantic technologies in system maintenance. As a first step toward establishing a research community in this area, the first international STSM workshop is focusing on the application of semantic technologies to system maintenance, including (but not limited to) the following topics:

- Traceability Link Recovery
- Reverse Engineering
- System Comprehension
- Processes and Process Modeling
- Outsourcing and Off-Shoring
- Software Development and Maintenance Life-cycle
- Software Artifact analysis and integration (e.g., requirements, source code, documents, emails, bug reports)
- Integration of semantic technologies in software maintenance tools

Results from the workshop will be published online under http://megaplanet.org/stsm2008/.

5. References

[1] Gruber, T.R.: “A Translation Approach to Portable Ontology Specifications”, *Knowledge Acquisition*, 5(2):199-220, 1993.

[2] Feldman, R. & Sanger, J. 2006. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, Cambridge University Press.

[3] Bézivin, J. “On the unification power of models,” *Soft. and Sys. Modeling*, vol. 4, no. 2, pp. 171-188, 2005.

[4] Favre, J., M., "Towards a Basic Theory to Model Model Driven Engineering", *In Proc. of the UML2004 Int. Workshop on Software Model Engineering*, 2004.

[5] Bouchon-Meunier, B., Yager, R.R., and Zadeh, L.A., 1999. *Information, Uncertainty and Fusion,* Springer.

[6] Mika, P. (2005). “Ontologies Are Us: A Unified Model of Social Networks and Semantics,” *In Proceedings of the 4th Int’l Semantic Web Conf.,* pp. 522-536.

[7] Mikroyannidis, A., “Toward a Social Semantic Web,” *Computer*, vol. 40, no. 11, 2007, pp. 113-115.

[8] Lassila, O. and Hendler, J. 2007. Embracing "Web 3.0". *IEEE Internet Computing* 11, 3 (May. 2007), 90-93