Maintaining SOA Systems of the Future
How Can Ontological Modeling Help?

Bilal Gonen[1], Xingang Fang[1], Eman El-Sheikh[1], Sikha Bagui[1], Norman Wilde[1],
Alfred Zimmermann[2] and Ilia Petrov[2]
[1]*Department of Computer Science, University of West Florida, 11000 University Parkway, Pensacola, FL 32514, U.S.A.*
[2]*Faculty of Informatics, Reutlingen University, Alteburgstrasse 150, D-72762 Reutlingen, Germany*

Keywords: Services Oriented Architecture (SOA), Ontology, Knowledge Modelling, Semantic Browsing, Software Maintenance, Software Evolution.

Abstract: Many future Services Oriented Architecture (SOA) systems may be pervasive SmartLife applications that provide real-time support for users in everyday tasks and situations. Development of such applications will be challenging, but in this position paper we argue that their ongoing maintenance may be even more so. Ontological modelling of the application may help to ease this burden, but maintainers need to understand a system at many levels, from a broad architectural perspective down to the internals of deployed components. Thus we will need consistent models that span the range of views, from business processes through system architecture to maintainable code. We provide an initial example of such a modelling approach and illustrate its application in a semantic browser to aid in software maintenance tasks.

1 INTRODUCTION

As computing resources have become pervasive, with powerful networked computers in everything from smart phones to smart cars, a new class of information systems is emerging. Perhaps we can best begin with an example.

> *Consider the driver of a future intelligent car. A warning light appears on his dashboard as he is travelling on an expressway. He selects "investigate" and the car's maintenance application looks up the error code and identifies which engine component triggered the warning. It then consults several online databases that pull together information from both the car's manufacturer and service histories of other vehicles that use the same component.*
>
> *The application sends a summary of the results to the driver's smart phone and recommends an immediate replacement of the component. It identifies several vendors and checks their inventory. The driver can choose to schedule service at a repair shop or to order the component online and swap components himself (Zimmermann, 2014).*

Such systems, intended to support many everyday situations and tasks, have sometimes been called *SmartLife* applications. Their development will involve advanced cloud infrastructures and services, new tools and methods for software development, and Big Data and open data innovation. These are major current research objectives of the National Science Foundation (National Science Foundation, 2014) and the European Union (European Commission, 2013).

In many cases the most reasonable structure for such applications will use Services Oriented Architecture (SOA) and thus the application will be implemented by orchestrating loosely-coupled services running on many nodes and communicating via message passing. SOA applications often follow the Web Services standards so that orchestration is done with Business Process Execution Language (BPEL), service interfaces are specified using Web Services Description Language (WSDL) and messages use XML described by an XML Schema Definition (XSD).

Development of such applications will be challenging, but as we look to the future their ongoing *maintenance* may be even more so. For some time researchers have pointed out the difficulties of maintaining SOA (e.g. Gold, 2004; Lewis, 2008). But at least in most cases earlier SOA

Gonen B., Fang X., El-Sheikh E., Bagui S., Wilde N., Zimmermann A. and Petrov I..
Maintaining SOA Systems of the Future - How Can Ontological Modeling Help?.
DOI: 10.5220/0005132903760381
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2014), pages 376-381
ISBN: 978-989-758-049-9
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

systems have been implemented within the context of some organizational framework, whether a corporation or a government, that can provide the necessary governance to manage change. However these new SOA applications look to require integration of services and data from multiple partners with little organizational nexus.

The essential problem of software maintenance has always been *program comprehension*: changes made to imperfectly understood software can have disastrous results. Maintainers are often not the same as the original software developers so they require time and effort to grasp the details that are relevant for each change. In the case of these new applications, maintainers may face events such as unexpected partner-forced changes or the sudden appearance of a security risk of unknown scope. Maintainers will need to respond quickly and accurately to such events, which will require rapid analysis of both their own code and its place within the wider application.

In this position paper we argue that maintenance of future SOA systems will be a serious challenge. While there will be no simple solution to this challenge, ontological models may be able to ease the maintainer's task. Such models must cover both high level business and architectural views of the whole application as well as a lower, code-focused view, since ultimately most maintenance tasks require changing code.

As an example, we describe how an existing SOA ontology from the Open Group (2010) can be extended to a SOA Evolution Ontology that better meets the needs of a software maintainer. The Open Group's ontology describes business processes, services and their interfaces in a fairly abstract manner. The maintainer needs that description, but he also needs to deal with concrete implementation details as may be found in design rationale, detailed interface specifications and in code.

We show how the resulting ontology can support *semantic browsing* to help a maintainer quickly acquire the information he needs for a particular maintenance task.

2 ARCHITECTURAL MODELING AND THE ESARC CUBE

One methodology to help address the range of issues facing SmartLife evolution is the ESARC Cube, developed to support the assessment of the maturity of SOA toolsets (Zimmermann, 2011). ESARC, the Enterprise Services Architecture Reference Cube, is an architecture reference model, which identifies an integral view for the main interweaved architecture domains such as: Architecture Governance, Architecture Management, Business and Information Architecture, Information Systems Architecture, Technology Architecture, Operation Architecture, and Cloud Services Architecture. ESARC provides a coherent aid for the examination, comparison, classification, quality evaluation and optimization of architectures. ESARC abstracts from any specific concrete business scenarios or technologies. The Open Group Architecture Framework (Open Group, 2009) provides the basic blueprint and structure for the extended service-oriented enterprise software architecture domains, views and viewpoints.

This approach for architectural modelling focuses on metamodels as abstractions for architectural elements and relates them to architecture ontologies (Zimmermann, 2013). Metamodels define models of models and are used within ESARC to define generic architecture model elements and their relationships. Architecture ontologies represent a common vocabulary that is based on explicitly defined concepts for enterprise architects who need to share their knowledge. Ontologies include the ability to automatically infer transitive knowledge. The metamodel of the Business and Information Reference Architecture consists of ESARC-specific concepts, which are derived as specializations from generic concepts such as Element and Composition from the previously mentioned Open Group's SOA Ontology (Open Group, 2010). There are exemplary metamodels and related ontologies for the following main architecture domains of ESARC: Business and Information Reference Architecture, Information Systems Reference Architecture, and the Technology Reference Architecture.

From the point of view of a software maintainer the most relevant of these will be the Information Systems Reference Architecture and the Technology Reference Architecture. These describe the code and deployment issues most likely to be relevant in making a specific change. However the maintainer must also be aware of the business processes, business rules and the organizational concerns at the higher levels. He thus must gather and use a wide range of information, from his own organization, from partner service provider organizations, and from infrastructure vendors (database management systems, enterprise service middleware, etc.). There will be many opportunities for confusion and misunderstanding as the maintainer tries to integrate

this range of sources. Our hypothesis in this paper is that ontologies could provide a significant aid to software comprehension, provided they consistently integrate information from different layers of the ESARC Cube.

3 RELATED WORK

As well as the ESARC Cube background, the research described in this paper draws upon earlier work in three areas: SmartLife research, software maintenance research on program comprehension, and semantic web research related to the study of software artefacts.

Applications and research focused on SmartLife were introduced by Hitachi (2013), SmartLife tracking and rescuing disaster management (Nagashree, 2012), work-life innovation (Mitchell, 2012), smart public information system for public transport (Patinge, 2012), smart energy systems (B.A.U.M. Consult, 2012), which includes IT for smart grids, supply/demand energy coordination, IT for smarter buildings, security, green IT, and IT for novel energy forms. Smart grids are advanced electricity systems of networks, which enable a two-way exchange of electricity power and information between suppliers and consumers based on intelligent communication, monitoring information and management systems.

In the software maintenance literature most of the research on understanding SOA applications has focused on dynamic analysis approaches that start from message logs or traces of execution (e.g. De Pauw, 2005). However, some research has looked at static analysis of the artefacts that describe services, such as WSDL interface descriptions and XSD data schemas (Coffey, 2012; Goehring, 2013).

Research has also been reported on applying semantic web techniques for maintaining traditional (non-SOA) software systems. This research focused on providing ontological support for software artefacts such as source code and documentation. In work reported by Witte (2007), customized ontologies were populated automatically from source code and documentation, and then queried to provide support for source code security analysis, for traceability links between source code and documentation and for architecture analysis. In work by Hyland-Wood (2008), an ontology was developed to describe the relationship between object-oriented software components. However, very little research has been reported on the application of semantic techniques for maintenance and evolution of SOA systems in particular.

4 EXTENDING THE OPEN GROUP ONTOLOGY

The starting point for our SOA Evolution Ontology is the Open Group SOA Ontology (Open Group, 2010). This was developed in order to facilitate understanding of SOA applications and improve communications between business and information technology experts. The Open Group SOA Ontology seemed an appropriate point of departure given its earlier use with ESARC and the maintainer's need to comprehend the system at multiple levels. The Open Group SOA Ontology is defined in the web ontology language (OWL) and is ready for extension and population for specific applications. In the ontology, 15 classes and 30 object properties are defined. The class hierarchy is shown in Figure 1.

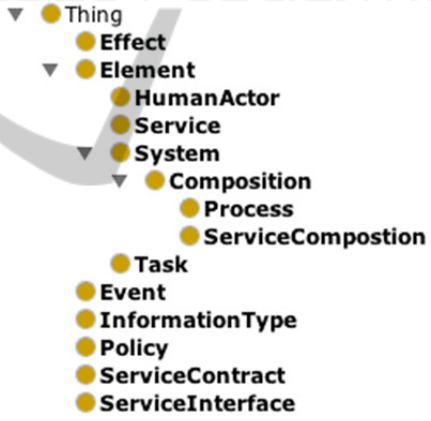

Figure 1: The Open Group SOA Ontology class hierarchy.

To develop the SOA Evolution Ontology, the Open Group SOA ontology was extended to improve support for software maintainers while preserving consistency with any existing higher level models. For example, the Service class was subclassed into InternalService and ExternalService. The maintainer must approach very differently those internal services whose code may be modified, since it is owned by his organization, and those external services which he can only invoke through an interface. Since tracing data usage is a very common task in software maintenance, a DataItem class was added to record the fields in each message. As a final example, a ProcessingModule class was added to allow Service instances to be linked to code artefacts.

As an example of the use of the extended ontology we populated it to provide a description of WebAutoParts, a hypothetical online automobile parts dealer that has been used in previous research studies (Wilde, 2012). WebAutoParts models an Internet start-up company using a SOA strategy for rapid development. A small amount of BPEL code is used to orchestrate real commercial web services from well-known vendors such as Amazon. These are represented by their WSDL and XSD interface descriptions. WebAutoParts might be typical not of a whole SmartLife application, but of one of its major components.

5 USING SEMANTIC SEARCH TO SUPPORT MAINTENANCE TASKS

Software maintainers inevitably spend a great deal of their time searching for the detailed information they need to do their jobs. Finding information and understanding relationships among software components and documentation is critical for software engineers to make timely decisions during the software maintenance process. As we have argued, maintainers of future SOA applications will probably confront an even greater diversity of information sources as compared with those found for conventional software.

Most search systems support information access through keyword-based search that often returns ambiguous results putting the burden on the user to select and filter a large volume returned results. In contrast, semantic search, which has been a focus of the Semantic Web initiative (Semantic Web, 2014), improves information retrieval on the web by giving machines the ability to reason about web content to better serve user needs.

As a first application of the SOA Evolution Ontology we are applying semantic web ideas to explore its use in a Semantic Browser (Gonen 2011). This specialized browser would support navigating the large repositories of textual, semi-structured artefacts describing a SOA system. These artefacts are annotated through semantic labels which support discovery of the semantic relations between different artefacts. Textual artefacts include natural language design rationale, design and code documentation, semi-formal service interface specifications (e.g. WSDLs), BPEL orchestration code, etc.

To use our current Semantic Browser, still under development, the classes of the SOA Evolution Ontology are first populated for a specific application such as WebAutoParts, thus creating a set of related "individuals" (instances) for each class. The original artefacts are then annotated so that, each time the name of an individual appears, a semantic link is added showing the relations to other individuals. These semantic links allow users to navigate the artefacts by following these named relations.

As an example, consider a scenario of a software maintainer trying to diagnose a problem with the US Postal Service shipping costs computed by WebAutoParts. He needs to discover where in the system shipping costs are computed and particularly what data is involved. In the software maintenance literature this kind of problem has traditionally been known as "concept location" or "concept assignment" (Biggerstaff, 1993). Suppose the maintainer has located one identifier as a starting point "GetUSPSRate".

The software maintainer performs an initial query on "GetUSPSRate," and is offered several files containing that term as shown in Figure 2. One such file is "OrderProcessing.bpel" and the maintainer selects this file.

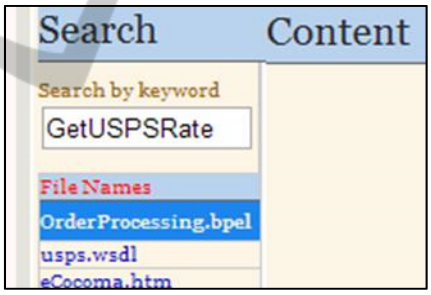

Figure 2: Search using Semantic Browser.

The Semantic Browser displays the file contents (Figure 3). The named individuals, which exist in the ontology, appear highlighted and underlined. The software engineer clicks on the "GetUSPSRate" in the text, and its relationships, such as "is a" and "has interface" are displayed. Immediately the maintainer would discover that "GetUSPSRate" is an ExternalService and that fact will condition possible maintenance fixes.

The software maintainer then selects the "has interface" relationship from the list, and is offered a list of interfaces, based on the ontology. He then selects "USPS.GetUSPSRate.Interface" from the list, and is offered all of the files that contain that term as shown in Figure 3. He may click on any of those file names to navigate to documentation that will help explain how this service is invoked. So he is well on

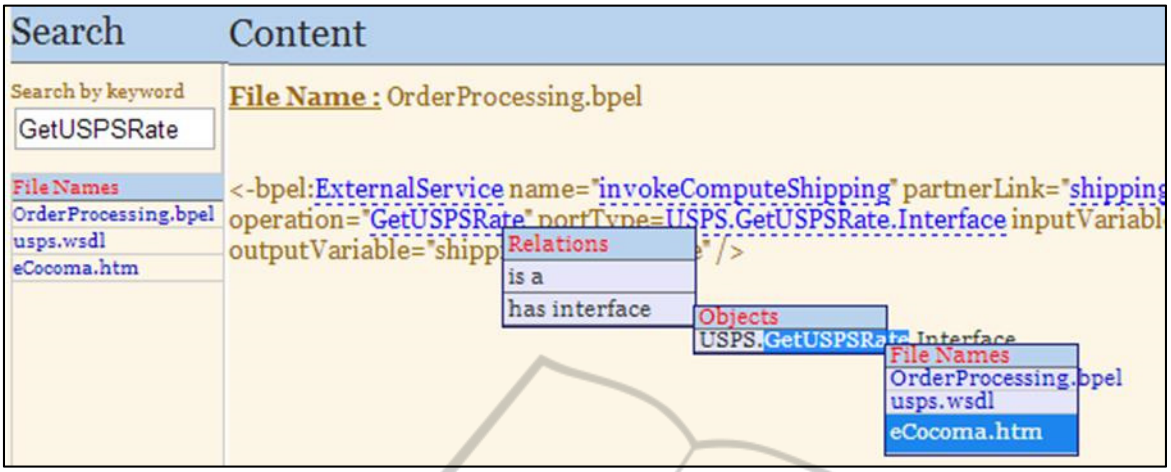

Figure 3: Sample search results using Semantic Browser.

his way to understanding what he may need to fix.

6 CONCLUSIONS AND FUTURE WORK

In this paper we addressed the problem of the evolution of future SOA applications. Software maintainers of such SOA systems will confront great challenges to keep them in continuous service in the face of a rapidly changing environment, continually emerging security risks, and a dynamic mix of partner organizations. We argued that an ontology-based approach could ease the difficulties of maintenance and introduced a SOA Evolution Ontology that can be populated with information about any specific SOA-based system. This ontology extends the Open Group SOA Ontology so it should be compatible with other tools based on this standard. As a first application of the SOA Evolution Ontology we proposed a Semantic Browser to aid a maintainer in navigating the many artefacts that describe a SOA system. We illustrated the approach by populating the SOA Evolution Ontology to model WebAutoParts, a small SOA system which could be typical of SmartLife components.

In the short run, future work will include additional evaluation of the SOA Evolution Ontology and the Semantic Browser both within our academic environments and in cooperation with industrial and other scientific partners.

The longer run vision for SmartLife maintenance is that an "ecosystem" of ontologies will emerge to describe these heterogeneous and complex software systems (Zimmermann, 2014). Hopefully, consistent modelling approaches can be found to bridge architectural levels and address the different concerns of business experts, developers and maintainers. The task of supporting the evolution of such systems will always be challenging, but such models could greatly ease the burden on software maintainers.

ACKNOWLEDGEMENTS

Work described in this paper was partially supported by the University of West Florida Foundation under the Nystul Eminent Scholar Endowment, and by the SOA Innovation Lab, Germany.

REFERENCES

B.A.U.M. Consult, 2012. Smart Energy made in Germany: Interim results of the E-Energy pilot projects towards the Internet of Energy, http://www.e-energy.de/documents/E-Energy_Interim_results_Feb_2012.pdf.

Biggerstaff, T. J., Mitbander, B. G., and Webster, D., 1993. The concept assignment problem in program understanding. *In Proceedings of the 15th International Conference on Software Engineering (ICSE '93)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 482-498.

Coffey, J. W., Reichherzer, T., Owsnick-Klewe, B. and Wilde, N., 2012. Automated Concept Map Generation from Service-Oriented Architecture Artifacts, CMC 2012, *Fifth International Conference on Concept Mapping, Malta*, Sept. 17-20, 2012.

De Pauw, W., Lei, M., Pring, E., Villard, L., Arnold, M. Morar, J. F., 2005. Web Services Navigator: Visualizing the execution of Web Services, *IBM Systems Journal*, Volume 44, Number 4, Page 821, DOI:10.1147/sj.444.0821.

European Commission, 2013. Horizon 2020 Work Programme 2014-2015, 5i. Information and Communication Technologies, http://ec.europa.eu/research/participants/portal/doc/call/h2020/common/1587758-05i._ict_wp_2014-2015_en.pdf.

Goehring, G., Reichherzer, T., El-Sheikh, E., Snider, D., Wilde, N., Bagui, S., Coffey, J., White, L. J., 2013. A Knowledge-Based System Approach for Extracting Abstractions from Service Oriented Architecture Artifacts. *(IJARAI) International Journal of Advanced Research in Artificial Intelligence*, Vol. 2, No.3, 2013, pp. 44-52.

Gold, N., Bennett, K., 2004. Program Comprehension for Web Services, *12th IEEE International Workshop on Program Comprehension (IWPC'04)* p. 151.

Gonen, B. 2011. Traversing Documents by Using Semantic Relationships. *The 2011 International Conference on Semantic Web and Web Services*, Las Vegas, Nevada, USA, July 2011.

Hitachi, 2013. Hitachi's Vision for Smart Cities, http://www.hitachi.com/products/smartcity/download/pdf/whitepaper.pdf.

Hyland-Wood, D., Carrington, D., Kaplan, S., 2008. Towards a software maintenance methodology using Semantic Web techniques and paradigmatic documentation modelling, *IET Software*, 2008, 2(4), pp. 337-347.

Lewis, G. A., Smith, D. B., 2008. Service-Oriented Architecture and its implications for software maintenance and evolution. *Proceedings Frontiers of Software Maintenance. IEEE Computer Society: Washington, DC*, pp 1-10.

Mitchel, S., Spencer, P., 2012. Work-Life innovation. The Role of Networked Technologies, Cisco Internet Business Solution Group, http://www.cisco.com/web/about/ac79/docs/ps/WLI-and-Technology_020312FINAL.pdf.

Nagashree, C., Kavya Rao, B., Jyothi Lobo, M., Harshitha, B. S., Antony, P. J., 2012. Smart Life Tracking and Rescuing Disaster Management System, *International Journal of Computer Applications* (0975-8887), Volume 45, No. 23, May 2012, pp. 10-17.

National Science Foundation, 2014. Big Data Research Initiative, http://www.nsf.gov/cise/news/bigdata.jsp.

Open Group, 2009. *TOGAF - The Open Group Architecture Framework, Version-9*, The Open Group.

Open Group, 2010. Service-Oriented Architecture Ontology, Open Group, ISBN 1931624887, https://www2.opengroup.org/ogsys/catalog/C104.

Patinge, P. D., Kolhare, N. R., 2012. Smart Onboard Public Information System using GPS and GSM Integration for Public Transport, *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 1, Issue V, July 2012, pp. 308- 312.

Semantic Web, 2014. http://semanticweb.org/wiki/Main_Page.

Wilde, N., Coffey, J., Reichherzer, T., White, L., 2012. Open SOALab: Case Study Artifacts for SOA Research and Education, *Principles of Engineering Service-Oriented Systems, PESOS 2012, Zurich, Switzerland*, pp. 59-60, June 4, 2012, doi: 10.1109/PESOS.2012.6225941.

Witte, R., Zhang, Y., Rilling, J., 2007. Empowering Software Maintainers with Semantic Web Technologies, *Lecture Notes in Computer Science* 2007, 4519, pp. 37-52.

Zimmermann, A., Pretz, M., Zimmermann, G., Firesmith, D. G., Petrov, I., El-Sheikh, E., 2013. Towards Service-oriented Enterprise Architectures for Big Data Applications in the Cloud, *17th IEEE International EDOC Conference (EDOCW 2013): The Enterprise Computing Conference with SoEA4EE*, 9-13 September 2013, *Vancouver, BC, Canada*, pp. 130-135.

Zimmermann, A., Zimmermann, G., 2011. ESARC - Enterprise Services Architecture Reference Cube for Capability Assessments of Service-oriented Systems, *SERVICE COMPUTATION* 2011, pp. 63-68, September 25 - 30, 2011, Rome, Italy.

Zimmermann, A., Gonen, B., Schmidt, R., El-Sheikh, E., Bagui, S., Wilde, N., 2014. Adaptable Enterprise Architectures for Software Evolution of SmartLife Ecosystems, to appear, *SoEA4EE 2014*, Ulm, September 2014.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS