Improving the Integration of the Software Supply Chain via the Semantic Web

Roy Oberhauser, Rainer Schmidt
Computer Science Department
Aalen University, Beethovenstr. 1, 73430 Aalen, Germany
Roy.Oberhauser@htw-aalen.de, Rainer.Schmidt@htw-aalen.de

Abstract

Supply chains are a significant concept for improving the cooperation of vendors and customers for material products. While the assembly of increasingly complex components in modern software development resembles a supply chain, classical supply chain management concepts are rarely applied. An interruption in the flow of information exists between the development/maintenance phases and operational phases, resulting in ignorance, inefficiencies, and suboptimal quality and support levels. One reason for this is the effort necessary to integrate the software lifecycle processes, coupled with the dispersion of information needed for execution of (semi-)automated software lifecycle workflows across semantically heterogeneous sources. Furthermore, the manual derivation of abstract IT (e.g., ITIL) and software processes into concrete processes and workflows results in low quality and inefficiencies. This paper describes an approach for improving the integration of software supply chains via the (semi-)automated realization of abstract software lifecycle processes and workflows in conjunction with Semantic Web technologies.

1. Introduction

In classical industries, the integrated consideration of vendors and customers in so-called supply chains is a very important concept to improve productivity and responsiveness of production. To improve supply chains, the classical industries developed a large spectrum of reference models, tools, etc. to streamline the information flow along the supply chain and integrate the stages of the supply chain. In a software supply chain, components [4] are similarly assembled into greater and more complex software systems until the final solution for the customer is provided. However, the classical supply chain management concepts are rarely applied to software supply chains, and currently the supply chain stages in the software industry are isolated. This isolation is a significant obstacle for the industrialization of software development and operation as shown in Figure 1. A particular problem is the multiplicity of the lower supply chain stages. From stage to stage, there are more users of a component. Thus the information flow is tree-shaped.

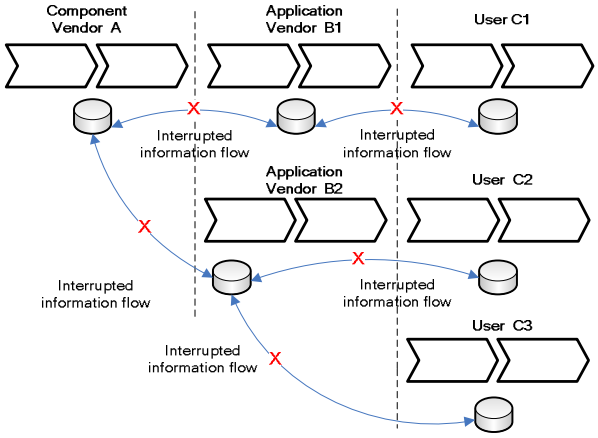

Figure 1: Interrupted information flows in software supply chains

On the other hand, there is a growing interest in a integrated view of the entire software supply chain. In the enterprise area, software lifecycle processes such as the ITIL [7] application management process provide an abstract view that includes phases of the software lifecycle. Software lifecycle processes have a broader view than most software development processes, which typically include only the phases from requirements analysis to roll-out, but ignore the operation and the retirement phases of software. Application development consists of the requirements, design, and build phases. Service management consists of deploy, operate, optimize and retirement phases.

However, software lifecycle processes have not been able to realize the vision of a software supply chain. Therefore, this paper will analyze the obstacles for an integrated software supply chain and provide an initial approach towards overcoming these obstacles.

2. Problem

Dual reasons for the lack of integration of the software supply chain can be identified: the information needed for execution of (semi-)automated software lifecycle workflows is not readily accessible and is typically scattered across semantically heterogeneous sources; and second, the extensive effort necessary to implement software supply chain processes.

2.1. Interrupted software supply chains

A significant problem for the implementation of software supply chains are breaks in the information flow between the component or application vendor's software development and the user's software operation of the component or application. This break is magnified because of the many potential component and application users. Thus, the integration is not a 1:1 but a 1:n problem.

After transferring the component from the component vendor to the component user, only some or no information is passed back to the software developer. However, two sets of valuable information could be sent to the component or application vendor as shown in Figure 2.

First, there is run-time information which is machine-observable. That means, this information can be gathered automatically. Such machine-observable run-time information is, for example, the application log, the server log, and the system log. Furthermore, data in the application database may be important for debugging, etc.

Second, there is run-time information which cannot be directly gathered by a machine, for example functional defects in the software: the software continues its operations but delivers false results or exhibits inappropriate behavior. This information is gathered indirectly by the user or administrator and is typically communicated to a single point of contact, called a help or service desk. All information gathered by the help desk is processed by incident and problem management processes (e.g., as defined in ITIL) and stored in an incident database. The incident and problem analysis is supported by the configuration management database which contains information about the planned configuration of the run-time environment, e.g., details such as software versions used on which systems, hardware, networking, etc.

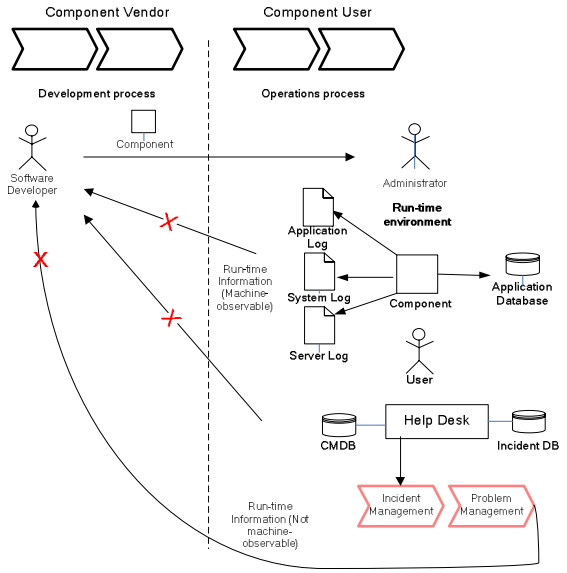

Figure 2: Broken information flows in software supply chains

Both machine-readable and non-readable run-time information could be advantageous to the software developer, however, differences in syntax and semantics impede the usage of such information. Furthermore, important information cannot be discovered because it is dispersed across different sources using incompatible syntaxes and semantics. Therefore a landscape of disconnected tools, systems, and databases emerges as shown in Figure 3.

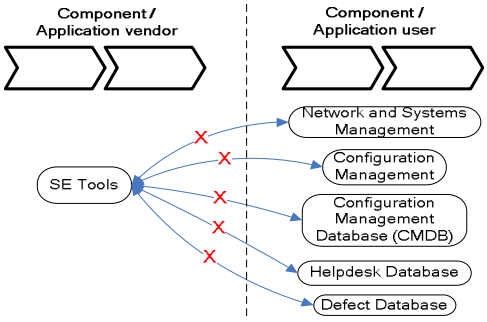

Figure 3: Disconnected Tool Landscape.

In practice, an important problem is deciding which information can be used to diagnose which kind of problem. In Figure 4, three hypothetical examples are shown. Performance problems can be analyzed, e.g., using the server log and the application database. Reliability problems can be analyzed using the server, system and application log in conjunction with the configuration management database (CMDB). Finally

problems in the application logic affecting correctness can be analyzed using the application database and incident database. There, observations of incorrect behavior of the application may be stored.

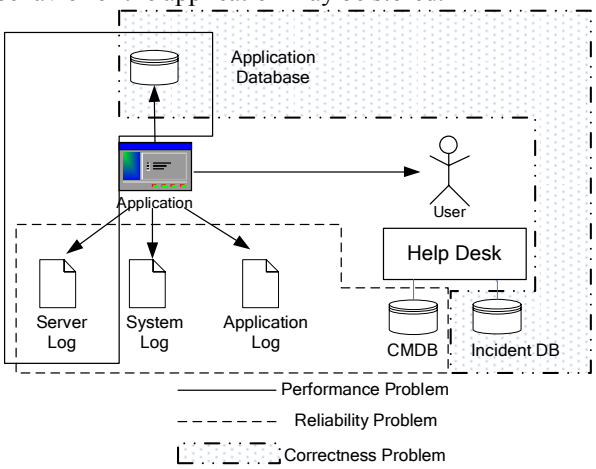

Figure 4: Best practices for problem diagnosis

It is evident that the selection of the appropriate information sources is non-trivial task. However, in practice there is no means to appropriately store the information as to which kind of problem requires which information source. This knowledge is not theory-based, but comes from observation and experience. Thus it can be seen as collection of best practices.

2.2. High implementation effort for software lifecycle processes

A further problem in implementing software supply chains is the high implementation effort created by a large semantic gap between the abstract process description and the executable process, as shown in Figure 5. There is no method to deduce executable processes from the abstract process description. Furthermore, there is no appropriate representation of best practices to improve the supply chain, as found in classical supply chain management, e.g., the Supply Chain Reference Model [17]. Workflows which already have been proven beneficial cannot be easily reused because they are not easily accessible. Thus, the derivation of executable processes from the abstract process description has to be done manually. However, the manual derivation causes a number of subsequent problems: it is an error-prone task resulting in a higher error rate. Second, there is only a low degree of reuse; the executable software lifecycle processes reuse only few elements, leading to a higher error rate because tested workflows are rarely reused. Third, there is only a low degree of standardization. This is of particular significance if the software lifecycle process spans multiple organizations which have to interact.

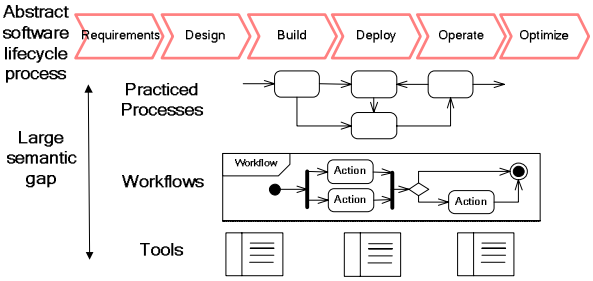

Figure 5: Semantic gap

3. Solution approach

Any feasible and concomitantly practical approach towards a holistic solution needs to address the previously mentioned problems of section 2 within realistic constraints. E.g., to further adoption, it should be platform-independent, vendor-neutral, utilize standards, and support a high degree of flexibility and loose-coupling. A possible approach that this paper presents is SWLIFE: Semantic Web-based Lifecycle Integration Framework for Enterprises. Before describing the framework elements, the underlying principles of the framework's architecture will be elucidated.

3.1. Solution Principles

I) *Holistic need-based integration of human and machine processes, services, and ontologically structured data for the operational lifecycle.* The lack of integration between software engineering processes and customer (e.g., enterprise or IT) software operational processes (e.g., ITIL) cause extensive inefficiencies especially in the maintenance phase of a software product, both in the processes and in regard to accessing necessary data. Yet this is typically the longest, most expensive, and least predictable phase.

II) *Round-trip process engineering: Prescriptive while concurrently descriptive abstract processes.* Too often abstract processes are not transformed or mapped to concrete processes correctly, losing its prescriptive nature. Abstract processes are mapped (semi-)automatically to concrete processes, and customized current concrete processes are abstracted to determine if they still conform to the abstract processes. If not, either they or the abstract processes are adjusted so that the abstract processes are also descriptive.

III) *Semantic Web accessibility and utilization of meaningful data.* Instead of hidden, obscure (operational) data in applications and tools such as log files, this data is discoverable, accessible, and its meaning described via Semantic Web Services.

IV) *(Semi-)Automated exchangeability and reuse of machine-readable best practices in the form of workflows*. Rather than just textual exchange via documents, which are difficult and arduous to automate, both abstract and concrete processes and workflows are described in standardized markup languages, e.g., including XMI (e.g., from UML Activity Diagrams), BPEL4WS, various Grid Workflow languages, etc. These are stored and exchanged in enterprise-wide and trusted internet repositories. Apposite choices or standards for Markup Languages are outside the scope of this paper.

V) *Service-Oriented Architecture (SOA).* In order to achieve the level of integration necessary, SOA principles including loose-coupling, encapsulation, reusability, composability, discoverability, etc. are applied, generally in conjunction with Web Services standards.

VI) *Feasible and reasonable automation.* Common operational or maintenance routines are automated to the degree economically reasonable in workflows, (semi)-automated composition of services.

VII) *Web-based human process documentation and integrated infrastructure binding.* Web-based process documentation of software engineering processes (e.g., VM-XT, RUP) and IT processes (e.g., ITIL) are tailored to the context and bound to the actual human and machine workflows used, with the services hyperlinked within the documentation. Thus a human can determine the status of a workflow, start an (automated) workflow, etc., all in the context of the organization's processes.

3.2. Semantic Supply Chain Integration with SWLIFE

This section describes the solution approach of SWLIFE as depicted in the high-level illustration of Figure 6. The layers used are for grouping purposes and are not intended to show strict abstraction or dependency relations. The white rectangles show examples of possible current processes and tools. The black rectangles show new areas that SWLIFE provides or would enable. It is not intended to be exhaustive but rather illustrative of the approach, additional layers and items are conceivable.

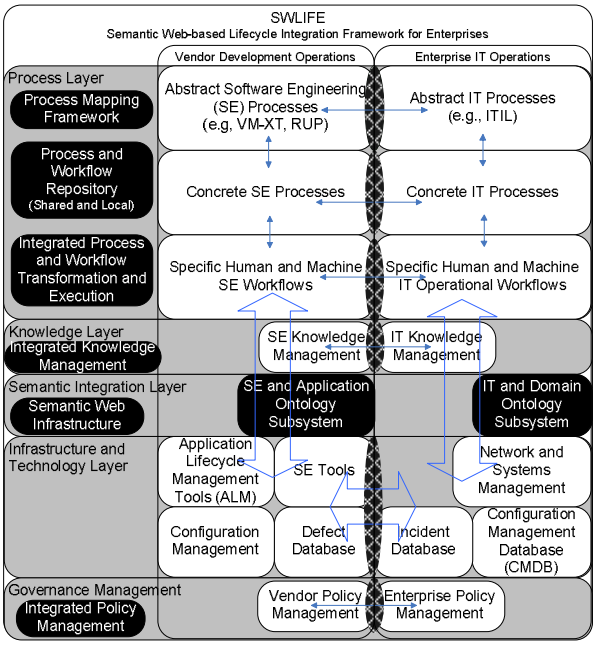

Figure 6: SWLIFE

Vendor Development Operations includes the processes and infrastructure of a product developer, while the vertical column Enterprise IT Operations includes those of the product customer who operates the software product. The black hashed ellipses between these columns show a new permeation between these two usually distinct organizational entities at a given layer.

The Process Layer can include processes defined at an organizational level, and depicts abstract process models (e.g., RUP or ITIL) that are tailored via a Process Mapping Framework to concrete processes. These processes typically include workflows performed at various points by various human roles defined by theses processes, and can include machine-based workflows to automate certain recurring tasks. The Process Mapping Framework includes techniques and tooling to support the tailoring of abstract processes and workflows to concrete processes, and for analyzing concrete processes and workflows, abstracting them, and comparing them to intended or previous abstract processes. The Process and Workflow Repository provides a retrieval, update, and exchange mechanism for processes and workflows, such as software engineering related, domain-specific (e.g., ecommerce, banks), enterprise-specific, platform-specific, and vendor- and application- specific areas. Such a mechanism could improve quality and reduce investment costs by enhancing the distribution and

interchange of best practices in these areas. The Integrated Process and Workflow Transformation and Execution Subsystem ensures that integrated human and machine processes and workflows are transformed, including (semi-)automatic composition, as necessary at runtime and ensures and monitors their execution.

The transformations and mappings in the Process Layer can utilize, where available, a Knowledge Layer, where Knowledge Management repositories of the applicable organizations may contain knowledge in the form of rules for defining process and workflow transformation.

The Semantic Integration Layer includes the Semantic Web interfaces for tools and services, including adapters when not provided, and any necessary infrastructure. This includes existing subsystems on the vendor and enterprise side for ontology loading and reasoning, as well as an integrated capability for SWLIFE scenarios.

The Infrastructure and Technology Layer includes the tools, applications, and technology utilized in the various organizations.

Governance Management includes the aspects necessary to govern the integration between organizations, including the integration of policy management.

An example of the interaction of the layers is shown in Figure 7. The Semantic Integration Layer encapsulates Semantic Web Services (SWS) interfaces of information sources such as CMDB, Incident DB, and Application Logs. In the Semantic Integration Layer, ontologies both from the vendor and enterprise are used to classify the SWS and to clarify their relationships. Aggregation and filtering is done via a Reasoner to support the workflow transformation process. Rules provided in the Knowledge Management Layer assist the Workflow Transformation Service in order to transform integrated workflows from the vendor and enterprise into concrete executable workflows which are provided and executed as SWS.

3.3. Reducing the effort to deduce concrete process

To perform an efficient, correct, and reliable transformation of abstract processes to and from concrete processes, non-trivial workflows and their (semi-)automatic composition should be supported on tooling suitable for IT and engineering settings. Although there is much research in the automatic composition of OWL-S based services, various HTN planners mentioned in research were considered unsuitable, e.g., due to the lack of access, instability, or lack of usable results in practice. E.g., OWLS-XPlan was considered, but lacked branching capabilities, sufficient documentation, and an execution engine for its PDDXML workflow format. E.g., Mindswap Composer also lacked branching and loop capabilities.

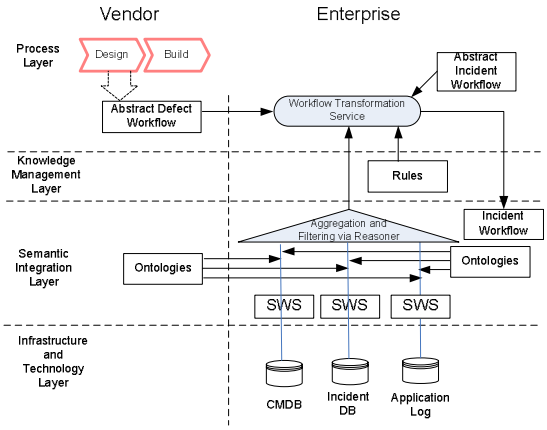

Figure 7: Layer interaction.

Thus, given the current lack of usable planners for workflow composition, a separate and practical solution for the Workflow Transformation Service was developed that combines an Ontology Subsystem with a Rule Engine, as shown in Figure 8. For the Ontology Subsystem, the Protégé OWL API 3.2 Beta, was used to read in the ontologies, and the integrated Protégé Reasoner was used (external reasoners are an option) to select a set of possible service matches. For the Rule Engine, JBoss Rules 3.0.5 was used. As the abstract workflow is read in and parsed, any abstract concepts are detected and the Reasoner returns a set of service instances that are a possible match. Because the choices may be highly dependent on a number of factors, this set is passed to the rule engine as a fact model and inserted into its working memory. The rule engine then selects the appropriate service based on the set of rules in the Rule Base. The use of a rule engine allows a greater degree of flexibility and change for enterprises without the encumbrances inherent in lower-level programming languages, although script languages are an alternative. If no matching service was found, an error is issued. If more than one service matched, then the first one was used, replacing the abstract concept in the workflow with a concrete service.

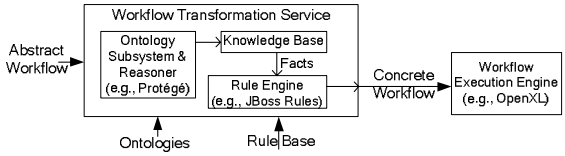

Figure 8: Abstract to Concrete Workflow Transformation

Various workflow languages would be suitable for SWLIFE, however, the domain-specific language Service Language Layer (SLL) [5], [9], was chosen for the implementation due to its simplicity as a workflow language at the programming language level and because of its service-centric nature. SLL can also be viewed in its XML form and be transformed into BPEL4WS or another workflow language.

For feasibility test purposes for such a workflow scenario, Web Service adapters and OWL-S descriptions for typically non-SOA software lifecycle applications (e.g., Subversion version control system, compiler, etc.) were created, along with an abstract workflow specified in SLL that specified various abstract tools to automatically generate a report for an engineer. This abstract SLL workflow was passed to the Workflow Transformation Service which then converted it into a concrete SLL workflow referencing concrete tool service instances, and the workflow was then deployed and executed as a service on the OpenXL platform.

For performance measurements, a Windows XP SP2 PC with an AMD 2600+ CPU, 1GB RAM, a 1Mb/s Internet connection, Java 5, Apache Tomcat Version 5.5.20, Apache Axis 1.4, and OpenXL 1.0 was used.

In applying a SWLIFE scenario, it was determined that a major contributing factor to the performance of the workflow transformation was the ontology loading time. Because of the difficulty of creating local copies, adapting import URLs, and maintaining these against new versions, Internet retrieval of OWL-S related ontologies is a highly likely use case. The average across 100 attempts for inserting an OWL-S-based service ontology to the knowledge base was 5.66 seconds. This would indicate that it is best to load and initialize the Workflow Transformation Service once as a long-running service in an enterprise setting.

The actual abstract to concrete workflow transformation time given a simple rule file was measured, averaged across 500 attempts as shown in Figure 9, based on the number of available services, and utilizing 4 or 100 abstract concepts. This shows that the performance of workflow transformation from abstract to concrete workflows utilizing a rule engine is practicable.

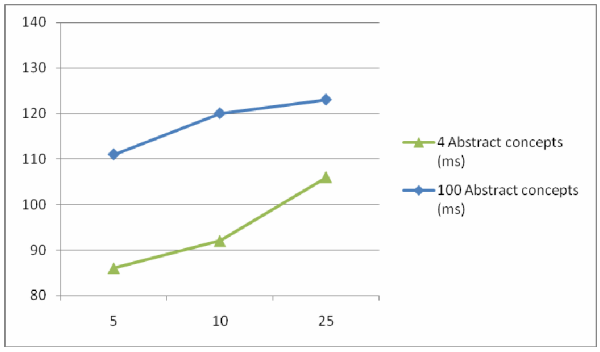

Figure 9: Average workflow transformation time.

4. Related work

Approaches for a holistic access and integration of information include collaboration software, ALM (Application Lifecycle Management), and ECM (Enterprise Content Management) which allows the management of an organization's unstructured information, wherever it exists as the AIIM (Association for Information and Image Management) intends. Yet most of the current applications in these areas have only partial solutions, non-standard interfaces, and little to no integration in the operational IT processes of customers. Grid technologies, platforms (e.g., the Globus Toolkit), and related research (e.g., the Adaptive Services Grid) could well provide an infrastructural basis for some aspects of SWLIFE, so while this approach does not preclude it, it does not require the Grid since the primary focus of SWLIFE is not the sharing computational resources or services. The concept of virtual organizations as used in the Grid could be leveraged for the ad-hoc integration and access by a vendor to the provided customer software and/or data.

The SWLIFE approach can leverage and integrate work on software engineering ontologies, e.g., [3] includes work on SWEBOK, software maintenance, software measurement, and other related ontologies. Work on generic IT and domain-specific ontologies could be used in SWLIFE, an example is the Health Information Technology Ontology Project (HITOP).

The approach presented here also has relationships to the semantic-based composition of Web Services as described in [11], [16], and [10]. These approaches show how Web Services can be integrated using Semantic Web technologies. This also applies to the ontology-based description of business processes as defined in [8] and in [13] and [14]. Work utilizing the Semantic Web for automated software engineering purposes includes [6]. A tighter relationship exists with approaches for the ontology-based representation of service processes such as the incident and problem

management processes as defined in [15]. The semantic alignment of business processes using ontologies is described in [2]. Ontologies can also be used for supporting the composition of web services as described in [1]. A mixed initiative framework for semantic web service discovery and composition is presented in [12]. It interleaves human decision making and automated functionality. Thus it can also be applied even if annotations are incomplete and inconsistent. This scenario is rather similar to the scenario of SWLIFE, and therefore this approach will be further investigated.

5. Conclusion

The concepts of supply chain management have not been applied systematically and consistently to improve the cooperation between vendors and customers in the software industry. Instead, in the overall software lifecycle process, a trichotomy continues to subsist between software development, enterprise IT processes, and the enterprise software runtime environment. This results in an interruption of the flow of information between the development/maintenance phases and operational phases in the software lifecycle. There is extensive effort necessary when attempting to integrate the software lifecycle processes across organizations, due to the dispersion of information needed for execution of (semi-)automated software lifecycle workflows across semantically heterogeneous sources. The manual derivation of concrete processes and workflows from today's abstract IT (e.g., ITIL) and software development processes causes errors, extensive effort, and limits widespread adoption of best practices.

This lack of integration in software supply chains manifests itself in perplexed administrators, in extensive, expensive and difficult defect remediation, long defect duration times, long service response and patch turn-around times, etc. As software continues to increase in its complexity and degree of integration, this situation will deteriorate without attention.

To address this situation, the SWLIFE approach bridges these process and information flow breaks by providing a semantic integration of the software supply chain, shared human and machine (semi-)automated workflows, and enabling better quality and more efficient semantic integration of infrastructure elements and artifacts. SWLIFE is influenced by background trends such as a high degree of networked systems, increasing integration via Service-Oriented Architecture (SOA) and Web Services, Semantic Web, outsourcing, and cross-organizational value creation and service providing in the IT area.

SWLIFE allows enterprises to improve the software supply chain processes with their vendors, which can yield positive results in incident response and quality. An important advantage of the SWLIFE approach is that each vendor of a product invests in the Semantic Web interface to its product once, and all customers can benefit from faster and higher quality maintenance response, providing an incentive to customers. Customers can utilize the Semantic Web interfaces to automate workflows, integrate workflows into higher-level processes, and more efficiently gather and analyze quality metrics. Via a standardized exchange format and repositories for processes and workflows, the investments can be incremental and shared. The implementation work showed that a feasible and practical solution for the (semi-)automatic composition of workflows using Semantic Web Service interfaces for the tools and services involved in SWLIFE scenarios exists.

A future market for packaged processes and workflows (analogous to the IBM Rational Unified Process) which are trusted, supported, and maintained, is thinkable. In addition, Semantic Web Service adapters can be provided to integrate legacy or non-conforming products.

Even a partial application of the SWLIFE approach can yield advantages via increased and systematic application of best practices in the form of human and machine-readable processes and workflows, the holistic needs-based integration of vendor and customer infrastructure, and the utilization of machine-processable semantic infrastructure. This a viable approach to addressing the skyrocketing complexity and maintenance costs associated with the use of software products.

Some challenges for the SWLIFE approach include: the need for highly-trained personnel for semantic, ontology, process, and workflow creation and usage; for cross-organizational access trust, security, authorization, and policy issues need to be addressed; the lack of (standardized) ontologies in the various SE and IT areas; and the adoption and usage of abstract processes and their description in machine-processable form.

Further work will include exploring additional application scenarios, for instance, enhanced response workflows for security intrusions, where rapid response and high quality incident data play important roles.

6. Acknowledgement

The authors would like to thank and acknowledge Thomas Ilzhöfer for his thesis work.

7. References

[1] Agarwal, V.; Koustuv, D.; Karnik, N.; Kumar, A.; Kundu, A.; Mittal, S,; Srivastava, B.; 2005: A service creation environment based on end to end composition of Web services. Proceedings of the 14th International Conference on World Wide Web, Chiba, Japan p 128 – 137, 2005.

[2] Brockmans, S.; Ehrig, M.; Koschmider, K.; Oberweis, A.; Studer, R., 2006.: Semantic Alignment of Business Processes. In: Proceedings of the Eighth International Conference on Enterprise Information Systems (ICEIS 2006), pp. 191-196. INSTICC Press, Cyprus, 2006.

[3] Calero, C., Ruiz, F., Piattini, M., 2006. Ontologies for Software Engineering and Software Technology. Springer-Verlag.

[4] Ciupke, O., Schmidt, R.: Components As Context-Independent Units of Software. WCOP 96, Linz 1996. Special Issues in Object-Oriented Programming. Workshop Reader of the 10th European Conference on Object-Oriented Programming ECOOP96. Dpunkt.verlag, Verlag 1996.

[5] Dinger, U., Oberhauser, R., Reichel, C., 2006. SWS-ASE: Leveraging Web Service-based Software Engineering. In Proceedings of the International Conference on Software Engineering Applications, 2006.

[6] Dinger, U., Reichel, C., 2006. Service Language Layer (SLL) Specification Version 1.1.3, Working Draft. http://www.dbis.ethz.ch/people/reichech/sll_service_language_layer_specification_version_1.1.3.pdf

[7] ITSMF, Service Management Forum, 2004.: IT Service Management – An Introduction. Van Haren Publishing, Amsterdam, 2004.

[8] Koschmider, A.; Oberweis, A., 2005.: Ontology based Business Process Description. In: Proceedings of the CAiSE´05 WORKSHOPS, no. 2, pp. 321-333, Portugal, 2005.

[9] Kossmann, D., Reichel, C., 2005. SLL: Running My Web Services on Your WS Platforms. Proceedings of ICWS 2005

[10] McIlraith, S.A. Son, T.C. Honglei Zeng, 2000. Semantic Web sServices 2000. IEEE Intelligent Systems and Their Applications Volume: 16, Issue: 2, pages 46- 53

[11] Medjahed, B.; Bouguettaya, A.; and Elmagarmid, A. K. 2003.: Composing Web services on the Semantic Web. The VLDB Journal The International Journal on Very Large Data Bases, Volume 12.,

[12] Rao, J.,; Dimitrov, D.;, Hofmann, P.;, Sadeh, N., 2006. ; "A Mixed Initiative Approach to Semantic Web Service Discovery and Composition: SAP's Guided Procedures Framework," icws , pp. 401-410, 2006.

[13] Rosemann, M.; Green, P., 2002.: Developing a meta model for the Bunge-Wand-Weber ontological constructs. In: Information Systems (27), pp. 75-91, 2002.

[14] Rosemann, M.; Indulska, M.; Green, P., 2004.: A Reference Methodology for Conducting Ontological Analyses. In: Proceedings of the 23rd International Conference on Conceptual Modelling (ER 2004). pp. 110-121, Shanghai, 2004.

[15] Schmidt, R.; Bartsch C.:, 2007. Ontology-based Modelling of Service Processes and Services. In: Proceedings of the IADIS International Conference, Applied Computing 2007, pp. 67-74, Salamanca, Spain, 2007.

[16] Sivashanmugam, K.; Miller, J. A.; Sheth, A. P.; Verma K. 2004:. Framework for Semantic Web Process Composition International Journal of Electronic Commerce Volume 9, Number 2 / Winter 2004-5, pages 71 - 106

[17] Supply Chain Council. Supply Chain Operations Reference (SCOR), www.score.org: 2007