A. Weisbecker, M. Burmester & A. Schmidt (Hrsg.): Mensch und Computer 2015 Workshopband, Stuttgart: Oldenbourg Wissenschaftsverlag, 2015, S. 269-274.

Smart Engineering for Smart Factories: How OSLC Could Enable Plug & Play Tool Integration

Christian Kaiser[1], Beate Herbst[2]

Information & Process Management, Virtual Vehicle Research Center[1]
Electrics/Electronics & Software, Virtual Vehicle Research Center[2]

Abstract

In this paper OSLC (Open Services for Lifecycle Collaboration) is proposed as key element in an approach for tool integration. OSLC enables data exchange and resource linking across tools within smart engineering, assisting thereby factories in behaving smart. According to OSLC's description, a core specification and several domain specifications assist software developers in implementing standardized interfaces for information exchange, which allow low-cost exchangeability of tools providing the same type of data. Due to the resource linking approach of OSLC (instead of data synchronization), the typical integration challenges of traceability, data consistency and data interoperability across the whole lifecycle process are appropriately managed and therefore assist collaboration, reuse and integration. Traceability across engineering tools can be a key enabler for smart factories. However, applied to the whole engineering process of a smart production lifecycle, which is in this approach based on the V-model, advantages and challenges yet to be solved of this approach arise, which is why this paper focuses engineering to be able to assist smart factories in future.

1 Introduction

In the production lifecycle process several tools are included, whereas in many cases software tools need to integrate with other tools in order to support a productive workflow (Wagner et al. 2014; Paschke & Softic 2014). A company might already use tools in their tool chain, which support interoperability to relevant other tools of the tool chain. Tool integration challenges can have its origin in merging two companies, as companies might use different tool vendors for the same tasks, employees are used to their tools and long-term contracts might exist. Another tool integration cause is the decision to buy a component from a supplier. A supplier receives requirements and modelling information, has to integrate it

into their tools and finally provides software and hardware components. Again, the supplier might use different tools in its tool chain. Tool integration is a noticeable topic in research since the 1990's (Brown et al. 1992). Tools available on the market employ their own underlying metamodel. Those metamodels tend to vary greatly in size and complexity, therefore confronting production companies with the challenge of tool integration (Wolvers & Seceleanu 2013; Zhang et al. 2012). When tools are embedded in a tool chain of an engineering lifecycle, then data input has to be made available in the tool, whereas data output has to be provided in a standardized or proper format, e.g. using an export functionality. The current shift from production to smart production and therefore from factories to smart factories further pressures the development of smart tools and environments. In smart factories ideally every object turns into a cyber-physical system and is able to communicate with others and act in an intelligent way, e.g. the intelligent or so called smart products are able to exchange information regarding their further processing with robotic machines within a smart factory. Production factories can only be smart if the engineering lifecycle provides the possibility for it. Two reasons are explained in the following.

On the one hand, smart products in a smart factory environment enable for example collaboration between humans and machines such as robotic arms. Therefore decisions by smart products and the smart factory environment will have to be made in real time. This causes plenty of new requirements regarding smart production and smart factory to be considered and fulfilled by smart engineering, by facility planning, by designers, etc. This new requirements have to be adopted by the product requirements management in the early engineering phase, modelling, design and production have to consider them. As a result, the engineering lifecycle is capable of the success of smart production by providing its basis. For example in the automotive domain, the number of variants and derivates has multiplied. Hence, due to configuration possibilities, the lot size in production converges towards one. A lot size of one means that each product is probably just produced once in its specific configuration, a realistic example in the automotive industry. Production lines producing only one specific vehicle for a longer time range will no longer be needed. Thus, a technology enabling a flexible production line is needed. Graph-based reasoning assists the production to act in an intelligent way. This could be achieved by a resource linking approach, creating a graph of linked data.

On the other hand, cyber physical systems, smart products being able to communicate with other products and infrastructure, need access to relevant data to enable intelligent behavior, such as data of the engineering lifecycle process. To be able to understand context and dependencies of retrieved data, traceability across tools used in this engineering lifecycle process is necessary, e.g. which requirements lead to a specific configuration and which requirements are covered by a test scenario. Traceability can be achieved by linking data artifacts, representing real artifacts like model elements that are maintained by tools (Zhang & Moller-Pedersen 2013). Thus, today a convenient technology is needed to enable data exchange and linking between tools with little effort for the user.

A promising technology used for such purposes is OSLC (OSLC 2015a), introduced by an open community that provides a collection of web-service based specifications to ease representation, access, and linking of resources between tools. All offered specifications are based on the W3C Linked Data (W3C 2015) standard, which builds upon common web

technologies such as HTTP (Hypertext Transfer Protocol), the Representational State Transfer (REST) technology, RDF (Resource Description Format), and URI (Uniform Resource Identifier) (Elaasar & Conallen 2013; Paschke & Softic 2014). While several integration approaches and frameworks are available on the market, ModelBus (Fraunhofer FOKUS 2015) and OSLC are appropriate integration technologies which are not proprietary, as they have an open specification and implementation. (Wagner et al. 2014)

Figure 1 depicts data integration with OSLC, which mainly consists of two players:

- A Provider, which uses a web service to store and provide data by implementing CRUD (Create, Read, Update, Delete) functionality.
- A Consumer, which is able to request and manipulate provided data via HTTP requests (HTTP GET, POST, PUT, and DELETE methods).

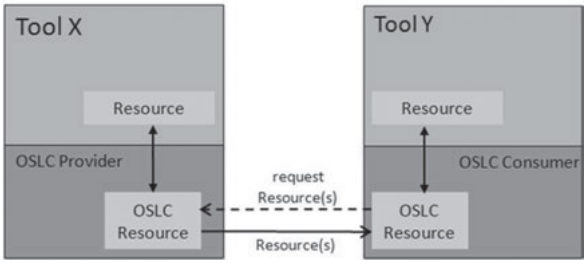

Figure 1: Overview of an OSLC Provider and Consumer concept.

Furthermore, this web-based technology supports the representation of data in form of RDF graphs, consisting of so called triples (subject, predicate, and object). Each triple represents a relationship between two specific artifacts, whereas a subject is linked to an object and the predicate describes the link type. If plenty of artifacts are linked with each other, then one or even more graphs emerge, because not all resulting graph parts necessarily need to be connected. The resulting graphs of, e.g. an engineering lifecycle using OSLC, can then be used for graph-based reasoning. Reasoning on graphs is scalable and most operations are suitable for bigger amounts of data (Urbani et al. 2009). In this way, OSLC enables analysis of multi tool workflow resources and therefore eases lifecycle traceability.

In the presented approach for the production lifecycle process, data artifacts of different tools and therefore different domains, such as requirements management (RM) and architecture management (AM), are linked. As mentioned above, for this connection, an OSLC Provider and Consumer are needed. A mandatory requirement for their data exchange is that both interfaces need to implement the OSLC core specification as well as a domain specification, which defines permitted properties.

2 Smart Production Lifecycle Integration using OSLC

Figure 2 represents a production lifecycle process in form of a V-model workflow management. The V-model usually starts the project definition with requirements

management, e.g. integrating requirements provided from customers, legal norms and regulations, the own (smart) production (facility, etc.) and requirements based on know-how into a requirements management tool. In the second step, in a model based engineering process, models are designed based on specific requirements. In the third step, design and prototype manufacturing is done based on the created models. In the fourth step, all necessary (single) prototype components are tested against the requirements (the model was based on) in the ramp-up phase. Therefore a back link to the model is needed. In the last step, all components are combined into the final system and tested against all requirements to ensure requirement fulfillment. In the following subsections, an example using OSLC is provided.

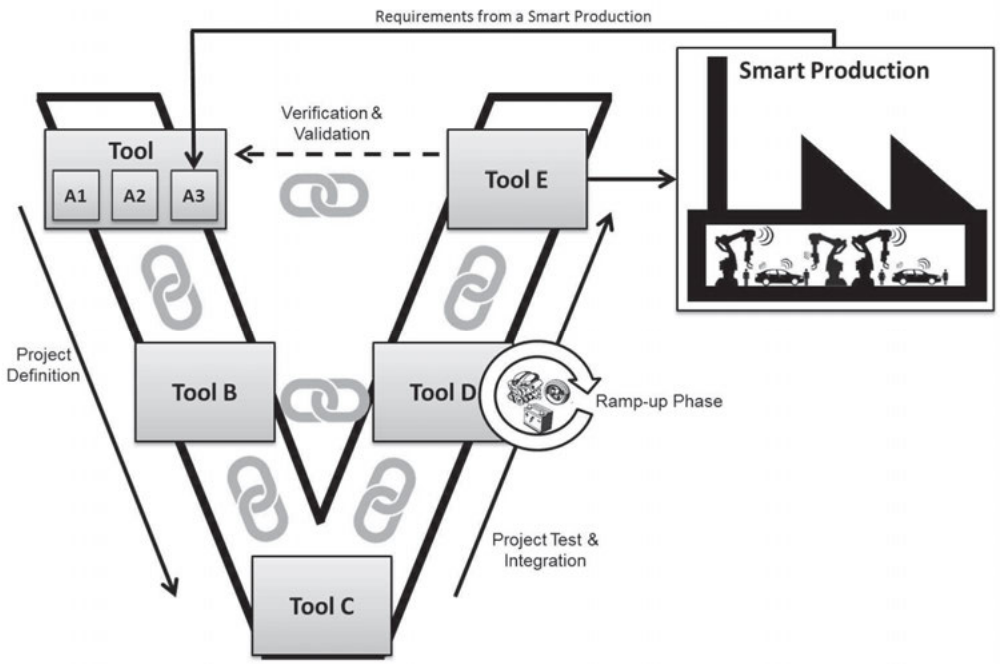

Figure 2: A smart production process represented as V-model.

In the presented example the need for software integration in several consecutive steps is shown. As OSLC still bears some unsolved challenges, the following realistic example is chosen to show advantages as well as challenges of this technology simultaneously.

2.1 System models based on requirements

As shown in Figure 2, the presented approach uses different requirement sources to describe the new product. These requirements are defined by different stakeholders and therefore are managed in different tools in this fictive example:

- The costumer uses IBM Doors to manage its requirements (Tool A1). These requirements are provided via the built-in IBM Doors OSLC RM provider.
- There are known legal norms and regulations, managed in HP Quality Center (Tool A2). These requirements are provided via the built-in HP Quality Center OSLC RM provider.
- The quality management department has formulated requirements out of production know-how. However, they use PTC Integrity to manage them (Tool A3). These requirements are provided as a ReqIF (Requirements Interchange Format) file (OMG 2015), represented as asset management resource in OSLC.

To create system models of a product, the product is divided into several subcomponents. Those subcomponents will be put together at a higher level and build a system model of the smart product. These system models as well as its subcomponents are designed in a modeling tool, whereas each component is based on one or more requirements. To reference these requirements, OSLC linking can be used. To overcome consistency problems when synchronizing requirements (possibly changing over time) between the tools, OSLC provides the opportunity to retrieve requirements from all above mentioned requirements sources with the same Consumer implementation as long as all OSLC Providers stick to the OSLC RM specification (OSLC 2015b).

As OSLC tries to enable plug & play of tools of the same domain, e.g. tools providing requirements. The specification is kept to a minimum, what bears a contradiction to enabling plug & play at a closer look, as in some cases it is necessary to extend the provided OSLC domain properties by using so called *extended properties*. For example, the OSLC RM specification only defines the minimum set of properties needed (e.g. *oslc_rm:validatedBy,* to describe a resource, such as a test case, which validates this requirement). However, most requirement management sources use further properties in their tools, e.g. parameter values to be able to provide the parameter values machine readable (Marko et al. 2015), therefore information would be lost if those are not exposed. Thus, an OSLC Consumer of OSLC RM sources has to map retrieved *extended properties* of a source to its internal data storage. As a result, if different OSLC RM sources add different extended properties, plug & play is not possible any more without an initial mapping.

Furthermore, even if no extended properties are in use, it might happen that properties of the loosely defined domain specifications are misused or interpreted variously and therefore lead to different OSLC resource representations, e.g. important information is provided in the *dcterms:description* property. Hence, the development of plug & play OSLC adapters seems to be hard to realize, as long as more precise specifications are missing.

2.2 Prototype development based on system models

Once system models in all granularities exist, development has to take place to realize a prototype as next step in the smart production lifecycle. Considering the success of companies like Apple, product design aspects should be in a leading part, comparable to software systems where “design is more important than ever” (Elaasar & Conallen 2013, 165) However, the required functionality of the desired product, described in the above

mentioned system models, still has to be fulfilled. Hence an intensive negotiation or calibration process between system model developers (using Tool B in Figure 2) and prototype designers (Tool C in Figure 2) takes place. At the end of the design and development process, a design and prototype description as well as a production process description exists, which fulfils the system models and therefore is based on the requirements indirectly. The design or prototype may consist of one single part or multiple components representing the whole system if connected. For the purpose of testing, a prototype is manufactured.

As OSLC is currently embossed by the automotive, aerospace and rail industries' engineering communities, certain aspects, e.g. to assist smart series production, are not covered yet. OSLC specifications exist for many domains, such as architecture management, asset management and requirements management. However, as shown in subsection 2.1, tools provided by different vendors may use different extended properties and therefore plug & play seems hard to realize. In case of asset management, the intention of extended properties even may be harder to guess without a context, as every data artifact can be an asset. From an overview perspective, the challenge is to define, how precise a specification should be. Should a new specification be created for the domain of design? Is one design domain enough or is the gap between the design of, for instance, a car and a USB stick too big, so that individual domains will be needed? Will every industry have to define its own set of extended properties in each domain?

2.3 Integration and testing of components

A prototype is ready for testing and it is assumed that it consists of multiple components, whereas some components are bought from a supplier. Each component is based on the system models and therefore is based on a specific set of requirements. Aichernig et al. explain in their paper how test cases can be generated based on formalized requirements using OSLC (Aichernig et al. 2014). Tests are executed at component level. Tests may reveal issues of system models or the design, which triggers redesign of models or redevelopment of components. If tests have been successful, then components are connected and tested again.

Typically multiple test cases are needed for each single component, for each combination of components and for the whole system. Therefore every test case and its result will be linked with components, design, requirements and so on. OSLC allows developers to use unidirectional or bidirectional links. Bidirectional links may simplify graph-based reasoning algorithms. However, it can lead to inconsistencies as shown in the following example. It is assumed that resource X is a component and resource Y is a test case.

- The link between X and Y is bidirectional, as shown in Figure 3a.
- A component was redesigned and therefore the link should be changed to link the new component X1 bidirectional with test case Y, as shown in Figure 3b.
- X deletes its link to Y and creates a new link from X1 to Y. In the next step Y is triggered to change its link from X to X1 as well, however, Y is not available or a problem occurs, as shown in Figure 3c.

- As a result, the back link from Y to X is still there and now points to the wrong resource, as shown in Figure 3d. Now it is not defined which link is the right one and where the back link is missing. As unidirectional links are suitable as well, the problem might not even be recognized and may lead to inconsistency.

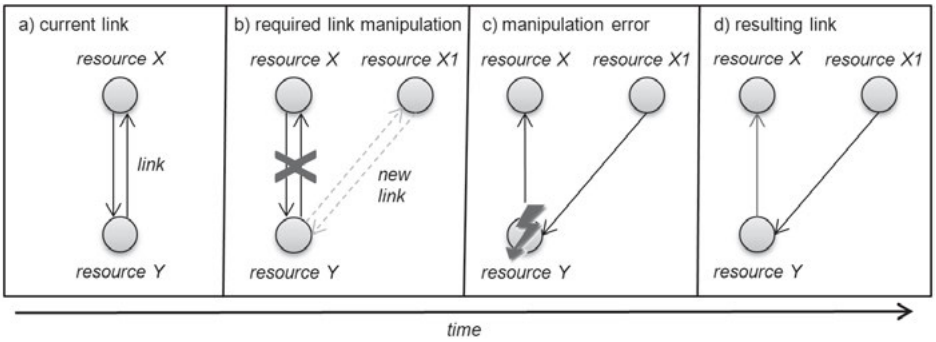

Figure 3: The occurrence of a back link problem causing inconsistency.

In terms of linking between different artifacts, OSLC provides so called delegated user interfaces (UIs). Thereby OSLC enable web applications to provide a user interface for creating and selecting resources and therefore enable traceability. However, another challenge of these delegated UIs is the context-free access caused by the small amount of information provided. For example, if a test case should be linked to a component, then the delegated UI may provide plenty of available component versions to choose from. Without additional context, it may be hard to identify the correct version. Packing a delegated UI with lots of information about the component may not be the ideal solution as well, as then the overview is lost. The remaining advantage is the simplified linking which enables traceability throughout the whole production lifecycle. However, it will not replace collaboration between developers.

OSLC advantages and challenges have been shown so far, instead of continuing the lifecycle process describing verification and validation, traceability in the resulting graph of linked data is the topic in the following subchapter 2.4.

2.4 Traceability

Traceability gains more and more importance in recent years, e.g. the automotive “safety standard ISO 26262 (ISO 2015) requires traceability” (Baumgart & Ellen 2014, 300), as “correctness evidence for designed systems must be presented to regulatory bodies” (Aichernig et al. 2014, 117). Previous subchapters described the interconnection of data artifacts (data artifacts are e.g. a requirement, a system component model, the design of a component or the test case for a component) and how OSLC assists interlinking those artifacts, while unsolved challenges are yet to be solved.

However, if OSLC is used to link artifacts in the whole lifecycle, then the advantage of traceability is offered. Graph-based reasoning is able to calculate the degree of fulfilled requirements (Tool A in the lifecycle in Figure 2) by positive test results (Tool E in the lifecycle), as the trace of engineering is made explicit by links, e.g. it is clear which component is tested, which system model the component is based on and again which requirements the system model is based on. A significant reduction of test cases to fulfill all requirements will be the outcome.

Visualizing all interconnections usually looks like a network of nodes and links. History showed that network analysis may reveal hidden secrets and bear potential for development shortcuts and innovations (Gloor et al. 2009; Mayer-Schönberger & Cukier 2013). Visualizing links between artifacts across tools provides flexible data insights, created by semantically driven customized views on the data. Hence, this assists data analysis, decision taking and collaboration between participants (Softic et al. 2013).

3 Outlook and Conclusion

In this paper OSLC is presented as key technology to enable tool integration, applied to the whole engineering process of a smart production lifecycle. Since there currently is a shift from factories to smart factories, the development of smart tools and environments is pressured. As a result, a new technology to even enable a smart engineering lifecycle, which provides the possibility of graph-based reasoning in a scalable approach, is needed. In the demonstrated example, OSLC is applied to a smart production lifecycle based on the V-model. It is shown that OSLC enables smart engineering, by easily retrieving and linking different lifecycle artifacts. Thereby traceability is made possible, to track the relation of different artifacts e.g. requirements, system component models, or test cases. This easy tool exchangeability and flexibility has an impact on tool market shares, as any innovation by any vendors can be tested and adopted easily. Even though OSLC is a promising technology to create a smart production lifecycle, this paper showed that this approach also involves challenges. As an example, the hard to realize plug & play integration of OSLC adapters was depicted, since the loosely defined specification might not be enough and the provided domains will not fit perfectly in many cases. However, since OSLC is introduced by an open community, adaptation and creation of domain specifications might easily be possible.

Acknowledgement

The research work of Christian Kaiser & Beate Herbst was partially funded by the European Commission within the CRYSTAL project under the ARTEMIS Joint Undertaking funding scheme, by the COMET K2 - Competence Centres for Excellent Technologies Programme of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Science, Research and Economy (BMWFW), the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG).

Literature

Aichernig, B.K., Hormaier, K., Lorber, F., Nickovic, D., Schlick, R., Simoneau, D. & Tiran, S. (2014). Integration of Requirements Engineering and Test-Case Generation via OSLC. In *Proceedings of the 14th International Conference on Quality Software* (QSIC), 117-126.

Baumgart, A. & Ellen, C. (2014). A recipe for tool interoperability. In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD), 300-308.

Brown, A.W., Feiler, P.H. & Wallnau, K.C. (1992). Past and future models of CASE integration. In *the Proceedings of the Fifth International Workshop on Computer-Aided Software Engineering,* 36-45.

Elaasar, M. & Conallen, J. (2013). Design management: a collaborative design solution. In *Proceedings of the 9th European conference on Modelling Foundations and Applications* (ECMFA'13), Springer-Verlag, Berlin, Heidelberg, 165-178.

Fraunhofer FOKUS (The Fraunhofer Institute for Open Communication Systems FOKUS) (2015). *ModelBus - Welcome To ModelBus.org*. http://www.modelbus.org. Retrieved 2015-06-01.

OMG (Object Management Group) (2015). *ReqIF*. http://www.omg.org/spec/ReqIF/. Retrieved 2015-06-01.

Gloor, P.A., Krauss, J., Nann, S., Fischbach, K. & Schoder, D. (2009). Web Science 2.0: Identifying Trends through Semantic Social Network Analysis. In *Proceedings of the International Conference on Computational Science and Engineering* (CSE '09), vol.4, 215-222.

ISO (International Organization for Standardization) (2015). *ISO 26262-1:2011 - Road vehicles -- Functional safety -- Part 1: Vocabulary*. http://www.iso.org/iso/catalogue_detail?csnumber=43464. Retrieved 2015-06-01.

Marko, N., Leitner, A., Herbst, B. & Wallner, A. (2015). Combining Xtext and OLSC for integrated model-based requirements engineering. In *Proceedings of the 41th Euromicro Conference on Software Engineering and Advanced Applications* (SEAA 2015), page unknown as not published yet.

Mayer-Schönberger, V. & Cukier, K. (2013). *Big data: a revolution that will transform how we live, work, and think*. Boston: Houghton Mifflin Harcourt.

OSLC (2015a). *Open Services for Lifecycle Collaboration*. http://www.open-services.net. Retrieved 2015-06-01.

OSLC (2015b). *RmSpecificationV2*. http://open-services.net/bin/view/Main/RmSpecificationV2. Retrieved 2015-06-01.

Urbani, J., Kotoulas, S., Oren, E. & Harmelen, F. (2009). Scalable Distributed Reasoning Using MapReduce. In *Proceedings of the 8th International Semantic Web Conference* (ISWC '09), Springer-Verlag, Berlin, Heidelberg, 634-649.

Paschke, S. & Softic, S. (2014). Open Services for Lifecycle Collaboration: Ein Ansatz zur Unterstützung der Zusammenarbeit in der Produktentwicklung. In: Butz, A., Koch, M. & Schlichter, J. (Hrsg.), Mensch & Computer 2014 - Workshopband. Berlin: De Gruyter Oldenbourg, 313-320.

Softic, S., Rosenberger, M., Denger, A., Fritz, J. & Stocker, A. (2013). Semantically based visual tracking of engineering tasks in automotive product lifecycle. In *Proceedings of the 13th*

International Conference on Knowledge Management and Knowledge Technologies (i-Know '13), ACM, New York, NY, USA, Article 36 , 4 pages.

Wagner, M., Dudeck, G., Hein, C., Tcholtchev, N., Gebhardt, C. & Korff, A. (2014). VARIES framework to support tool integration in product line engineering. In *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2* (SPLC '14), Vol. 2. ACM, New York, NY, USA, 117-120.

Wolvers, R. & Seceleanu, T. (2013). Embedded Systems Design Flows: Integrating Requirements Authoring and Design Tools. In *Proceedings of the 39th EUROMICRO Conference on Software Engineering and Advanced Applications* (SEAA), 244-251.

World Wide Web Consortium (W3C) (2015). *Data - W3C*. http://www.w3.org/standards/semanticweb/data. Retrieved 2015-06-01.

Zhang, W., Leilde, V., Moller-Pedersen, B., Champeau, J. & Guychard, C. (2012). Towards Tool Integration through Artifacts and Roles. In Proceedings of the *19th Asia-Pacific Software Engineering Conference* (APSEC), 603-613.

Zhang, W. & Moller-Pedersen, B. (2013). Establishing Tool Chains Above the Service Cloud with Integration Models. In *Proceedings of the IEEE 20th International Conference on Web Services* (ICWS), 372-379.

Kontaktinformationen

Christian Kaiser
Virtual Vehicle Research Center
Department information- & process management
Inffeldgasse 21a
8010 Graz
christian.kaiser@v2c2.at
http://www.v2c2.at