M. Koch, A. Butz & J. Schlichter (Hrsg.): Mensch und Computer 2014 Workshopband,
München: Oldenbourg Wissenschaftsverlag, 2014, S. 313-320.

Open Services for Lifecycle Collaboration: Ein Ansatz zur Unterstützung der Zusammenarbeit in der Produktentwicklung

Stefan Paschke, Selver Softic

Information & Process Management, VIRTUAL VEHICLE Research Center[1]

Zusammenfassung

Komplexe Produkte zu entwickeln und in entsprechender Qualität zu produzieren erfordert ein hohes Maß an Kooperation auf menschlicher und technischer Ebene. Die technische Ebene zeichnet sich heute durch eine Vielzahl an technischen Hilfsmittel (sog. Tools) aus, mit deren Hilfe Informationsartefakte (z.B. Anforderungen, Testergebnisse, …) verwaltet werden. Die Herausforderung besteht auf technischer Ebene nach wie vor darin, diese Tools bestmöglich zu integrieren. Mit „Open Service for Lifecycle Collaboration" wird im folgenden Beitrag ein Community-Ansatz vorgestellt, wie Tools möglichst pragmatisch integriert werden können, ohne proprietäre Schnittstellen entwickeln zu müssen. OSLC versucht die Effizienz des Kooperationsprozess auf technischer Ebene zu verbessern und dadurch allen beteiligten Stakeholdern Kosten zu ersparen.

1 Einleitung und Motivation

Die Entwicklung eines komplexen Produktes (sog. „cyber-physischer Systeme" im Kontext Industrie 4.0) ist eine große Herausforderung. Es gibt heute noch keine vollständige IKT-Lösung, um die Komplexität im Produktentwicklungsprozess zufriedenstellend zu bewältigen (Herbsleb, 2007). Deshalb existieren viele unterschiedliche Tools für einzelne Aufgaben im Produktentwicklungsprozess, wie beispielsweise die Definition von Produktanforderungen, Zuweisung von Entwicklungsaufgaben oder Verwaltung von Systemdefinitionen. Auf Grund der Schwierigkeit einer Integration (Wassermann, 1989 weiterentwickelt durch Thomas & Nejmeh, 1992) wurden diese Tools, in den meisten Fällen, als Silos konzipiert, und es besteht nur minimale oder überhaupt keine Integration mit anderen Tools. Das erschwert die für die Bewältigung der Komplexität nötige kombinierte Wertschöpfung aus den einzelnen Entwicklungstools.

Erfolgreiche Produktentwicklung erfordert optimale Zusammenarbeit von Menschen in einem global verteilten Team. Doch die derzeitige Toollandschaft und die mangelnde Toolintegration gestalten Zusammenarbeit als schwierig (Müller et al, 2012). Es ist nur schwer möglich die Ergebnisse der einzelnen Entwicklungsschritte so zu verknüpfen, dass die Komplexität zufriedenstellend beherrscht werden kann. Daher können beispielsweise Verantwortliche für den Gesamtprozess nur sehr aufwendig nachverfolgen, welche Anforderungen es gibt, ob sie implementiert wurden und ob sie bereits erfolgreich getestet worden sind. Die Entwicklung einer diese Antworten vermittelnde Integrationsschicht ist eine aktuelle Frage der Forschung.

Bisher wurden in der Praxis, mit mehr oder weniger Erfolg, unterschiedliche Ansätze gewählt, um Tools zu integrieren:

- Die Entwicklung proprietärer Schnittstellen zwischen einzelnen Tools ist eine aufwendige und kostenintensive Strategie. Bei jeder Toolversion muss auch die Schnittstelle aktualisiert werden. Die Anzahl der Schnittstellen wird schnell unüberschaubar. Jedes neue Tool muss bei n beteiligten Tools n-1 Schnittstellen implementieren.
- Der Plattform-Ansatz möchte alle Tools auf derselben Plattform integrieren und wurde von Firmen wie bspw. Siemens (Teamcenter), Microsoft (Visual Studio) oder Eclipse gewählt. Tools verwenden gemeinsame zentrale Komponenten. In der Praxis beschränkt sich die Entwicklung nicht auf Tools einer Plattform daher kommt es zu ähnlichen Problemen zwischen Plattformen wie beim ersten Ansatz. Tools und Plattformen sind so zahlreich, dass dieser Ansatz vermutlich nicht erfolgreich sein wird.

In der Praxis zeigte sich also, dass das Integrationsproblem noch nicht gelöst werden konnte. Der vorliegende Beitrag beschreibt mit Open Services for Lifecycle Collaboration (OSLC) eine neue Herangehensweise, welche von IBM initiiert und mittlerweile durch andere große industrielle und wissenschaftliche Player in der IT-Industrie voran getrieben wird (IBM, Siemens, ORACLE, GM, KTH …). Dabei bedient sich OSLC technologischer Elemente aus dem World Wide Web, wie etwa semantischer Technologien oder spezieller Abfragesprachen.

Der vorliegende Beitrag fasst wesentliche technologischen Entwicklungen rund um das Web zusammen und beschreibt RESTful Webservices und Linked Data als technologisches Fundament für OSLC in Abschnitt 2. Abschnitt 3 beschreibt OSLC als innovativen Community – Ansatz, um das Integrationsproblem zwischen den Toolwelten zu lösen. Abschnitt 4 schließt mit einer Zusammenfassung der wesentlichen Aussagen des Beitrags sowie einem Ausblick.

2 Technologische Grundlagen für OSLC

Eine wesentliche Grundlage für OSLC liefern die Entwicklungen rund um das Semantic Web. Schon 1998 skizzierte der Erfinder des Word Wide Web, Sir Tim Berners Lee, eine

Roadmap für ein auf dem klassischen Web aufsetzendes Semantic Web (Berners-Lee, 1998). Das Ziel des Semantic Web lag in der Verbesserung der Nutzbarkeit der Daten durch Maschinen. Um dies zu gewährleisten, muss die Struktur des Webs und der darin enthaltenen Daten für Maschinen interpretierbar gemacht werden. Erst dann wird es intelligenten Software-Agenten ermöglicht, menschlichen Agenten mehr Nutzen zu stiften. 2001 haben Tim Berners-Lee, James Hendler und Ora Lassila (Berners-Lee et al., 2001) im „Scientific American" vorgestellt, wie Web Inhalte, welche von Computern verstanden werden können, eine Vielzahl neuer Anwendungen für Menschen ermöglichen. In den dort beschriebenen und zukunftsweisenden Anwendungsbeispielen erledigen intelligente Semantic Web Agenten für Menschen automatisiert komplexe Aufgaben.

Als Grundlage des Semantic Web, wurden vom World Wide Web Consortium eine Reihe von Standards, Technologien und Projekten verabschiedet, welche dazu dienen, Informationen im Web in einer für Maschinen interpretierbaren Form bereitzustellen. Alle Informationen im Semantic Web werden maschinell interpretierbar in der Form von Triples, also Statements bestehend aus Subjekt, Prädikat und Objekt abgebildet. Alle Ressourcen werden einheitlich über Uniform Resource Identifier (URI) identifiziert. Das Resource Description Framework (RDF) ist ein einheitliches Modell zur formalen Beschreibung von Informationen über Objekte. Das Ressource Description Framework Schema (RDFS) und die Web Ontology Language (OWL) stellen formale Sprachen dar, um die Bedeutung der verwendeten Vokabeln einheitlich zu spezifizieren. Die zwei bedeutendsten Standards wie die OWL oder RDF sind in der Zwischenzeit gereift, weshalb die Verfügbarkeit maschineninterpretierbarer Daten am Web sowie deren Vernetzung zunehmen wird.

Um als Unternehmen Daten am Semantic Web anzubieten und sich zu beteiligen existieren bereits Standards und Prinzipien wie Linked Data (Berners-Lee, 2006): eine Menge an Prinzipien, um strukturierte Daten am Web of Data zu publizieren und zu verlinken. Tim Berners Lee hat zu diesem Zweck vier einfache Regeln definiert, welche Datenprovider anwenden sollen, um den globalen Datenraum für maschineninterpretierbare Daten, das Web of Data, zu erschließen:

- Zur Adressierung von allen Entitäten in einem Datenbestand sollen URIs verwendet werden.
- Es sollen nach Möglichkeit HTTP-URIs verwendet werden, um so die Inhalte der Datenbestände Web-konform abrufen zu können.
- Werden URIs dereferenziert (und damit RDF Eigenschaften als Hyperlinks interpretiert) soll eine Web of Data konforme Information geliefert werden.
- Von einem Datenbestand sollen URIs auf andere (externe) Datenbestände gesetzt werden, um so auf weiterführende Daten zu gelangen.

Eine weitere wesentliche Grundlage ist Representational State Transfer (REST). REST wurde ursprünglich als ein Programmierparadigma entwickelt um groß angelegte verteilte hypermedia systeme implementieren zu können. Das Programmierparadigma an sich ist recht abstrakt definiert. In seine Prinzipien sind auch Definitionen des Hypertext Transfer Protocol (HTTP) eingeflossen, vor allem die gute Skalierbarkeit beruht darauf. Daher wird REST sehr oft mit HTTP in Zusammenhang gebracht. Im Folgenden werden die Prinzipien von REST dargestellt mit dem Fokus auf eine weit verbreitete Interpretation des Programmierparadigma

der „RESTful“ Web Services. Weiter führende Informationen finden sich in (Pautasso et al. 2008, 807). Das REST Programmierparadigma beruht auf 4 Prinzipien:

- *Identifizierung einer Ressource über ein URI*: Ein RESTful Web Service veröffentlicht eine Sammlung von Ressourcen, um sich, als Ziel einer Interaktion, bei einem Client eindeutig zu identifizieren. Ressourcen werden über URIs identifiziert, einem weltweiten Adressraum für Ressourcen und Service Discovery.
- *Eine einheitliche Schnittstelle*: Ressourcen werden über 4 Operationen erzeugt, gelesen, verändert und gelöscht: PUT, GET, POST, DELETE.
- *Sich selbst beschreibende Informationen*: Ressourcen und Darstellung werden getrennt gehalten, daher kann der Inhalt in unterschiedlichen Formaten abgerufen werden (z.B. HTML, XML, JSON, Klartext). Meta-Informationen über eine Ressource ist vorhanden und wird beispielweise verwendet für: Caching, Feststellung von Übertragungsfehlern oder die Zugriffskontrolle.
- *Zustandsbehaftung durch Hyperlinks*: Jede Interaktion mit einer Ressource ist zustandslos z.B. eine Abfragenachricht ist in sich abgeschlossen. Zustandsbehaftung wird möglich indem Zustandsinformationen explizit übertragen werden. Es gibt verschiedene Technik um Zustandsinformationen auszutauschen z.B. URI Rewriting, Cookies oder versteckte Formularfelder. Ein Zustand kann in einer Antwort eingebettet seinen und auf gültige Folgezustände der Interaktion hinweisen.

Wie weiter unten beschrieben greift OSLC die Prinzipien des Semantic Web, Linked Data und „RESTful“ Web Services auf, um sie auf Unternehmensinterne Wissensstrukturen anzuwenden.

3 OSLC als Community-Ansatz zur Lösung des Integrationsproblems

OSLC besteht aus einer Reihe von Spezifikationen, die einen Ansatz für die Integration von Produktentwicklungstools beschreiben[1]. Diese Spezifikationen definieren, wie ihnen entsprechende Tools ihre Daten und Workflows integrieren können, um Durchgängigkeit über den gesamten Entwicklungsprozess zu erreichen. OSLC standardisiert dabei nicht die Funktionsweise eines Tools oder einer Klasse von Tools, sondern spezifiziert ein leichtgewichtiges Protokoll, dass es erlaubt Tools vergleichsweise nahtlos zu verwenden. OSLC versucht auch sich an möglichst viele Tool-Implementierungstechnologien anzupassen.

OSLC definiert zwei primäre Techniken der Toolintegration (OSLC Core Specification Workgroup, 2013): Die erste besteht darin, Daten skalierbar und plattformunabhängig durch den Einsatz von Webtechnologien zu vernetzen. Dabei verwendet OSLC Linked Data, um Informationen über Toolartefakte, die eindeutig mittels HTTP URIs adressierbar sind, in RDF Ressourcen abzubilden (Linked Lifecycle Data). Für die Manipulation der Ressourcen

[1] http://open-services.net/

bietet OSLC ein einheitliches Protokoll, dass, in Anlehnung an RESTful Web Services, HTTP CRUD (Create, Read, Update, Delete) Operationen ermöglicht. Ressourcen werden verknüpft, indem die URI einer Ressource in der Repräsentation einer anderen abgelegt wird. Die zweite Technik beschreibt die Verknüpfung von Daten über eine HTML Benutzeroberfläche. Hier spezifiziert OSLC ein Protokoll, das es einem Tool erlaubt, Fragmente der Benutzeroberfläche eines anderen Tools (z.B. einen Dialog zur Selektion eines Elements) aufzurufen. Dies erlaubt einem Tool, das User Interface und die Geschäftslogik aus einem anderen Tool für die Integration der Daten und Prozesse zu nutzen.

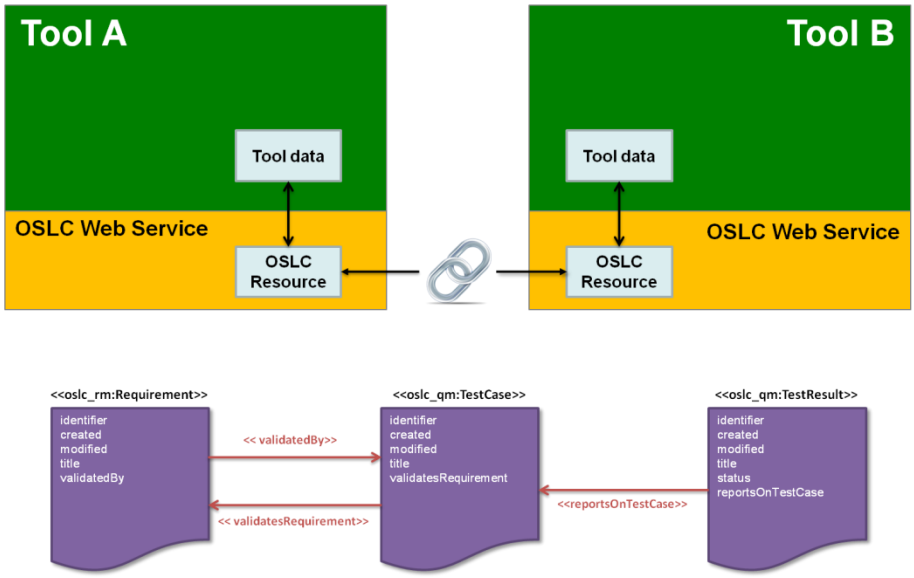

Abbildung 1: Beispiel für Implementierungsansatz von OSLC Services und erreichte Verknüpfung von Ressourcen

OSLC-Spezifikationen unterteilen sich in eine Core und, darauf aufbauen, domänenspezifische Definitionen (z.B. Change Management, Requirements Management und Quality Management). Der Core beschreibt den primären Integrationsansatz und besteht hauptsächlich aus Regeln und Paradigmen für den Einsatz von HTTP und RDF. Er ist nicht für den alleinstehenden Einsatz gedacht, sondern für die Anwendung in Verbindung mit einer oder mehrerer Domänen Spezifikationen. Daher muss für ein spezielles Tool ein Verbund aus Spezifikationen implementiert werden. Beispielsweise würde ein Requirements Management Tool sowohl den Core, als auch die Requirements Management (RM) Spezifikation implementieren.

Die OSLC-Initiative ist eine offene Gemeinschaft (d.h. eine Community) in der alle Beteiligten etwas beitragen können[2]. Es wird vorausgesetzt, dass die eingebrachten Neuer-

[2] http://open-services.net/participate/

ungen nicht bereits industriell verwertet werden und im Besitzt eines Unternehmens sind[3]. Die OSLC Initiative wurde ursprünglich von IBM Rational ins Leben gerufen um ihre Erfahrungen aus der JAZZ Plattform an die Öffentlichkeit weiter zu geben und eine Basis für die weitere Verbreitung des Integrationsansatzes ihrer Plattform zu schaffen[4].

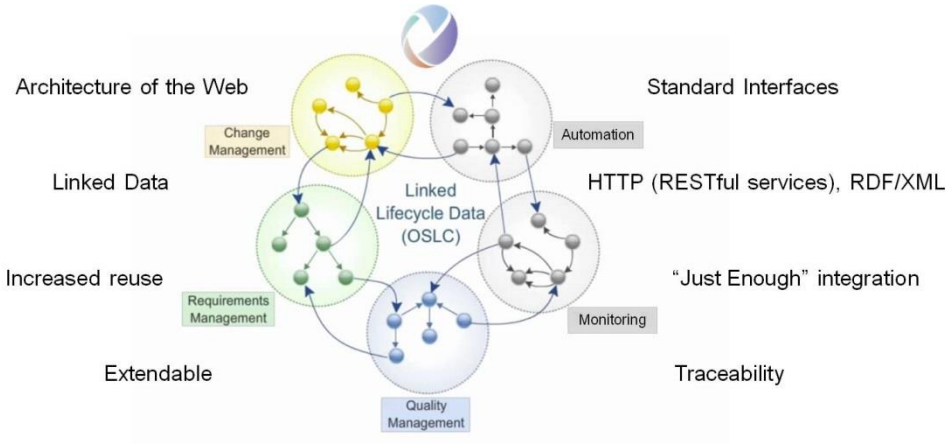

Abbildung 2: Open Services for Lifecycle Collaboration

OSLC bedient sich der *Architekturkonzepte aus dem Web*, die sich durch die Dimension und Heterogenität des Internet bereits als skalierbar und plattformunabhängig bewiesen haben. *Linked Data* wird eingesetzt um Meta-Informationen über Tool-spezifische Daten zu veröffentlichen. Zur Kommunikation setzt es etablierte und *standardisierte Schnittstellen* ein, die das Management der Meta-Informationen auf Basis von *RESTful Services* ermöglichen. Die Core und Domänen Spezifikationen beschreiben gemeinsam ein Wissensmodell das sie mit Hilfe von *RDF* abbilden.

Aus Sicht des Endanwenders wird *bessere Wiederverwendbarkeit* der Implementierungen erreicht, da alle Tools, die OSLC implementieren, miteinander kommunizieren und Daten verlinken können. Sobald diese eine Domänen Spezifikationen implementieren können sie mit allen Tools in der gleichen Domäne ihre Daten austauschen. Das spezifizierte Protokoll ist *leichtgewichtig*, daher ohne zu großen Aufwand zu implementieren und ist bemüht sich auf das Wesentliche zu beschränken. *Rückverfolgbarkeit* und Analyse der Zusammenhänge von Ressourcen ist möglich durch die existierenden Verknüpfungen des in RDF abgebildeten Wissensmodells. Die klare Trennung des Ansatzes in einen Core und die Domänen Spezifikationen gewährleisten *Erweiterbarkeit* für weitere Bereiche.

[3] http://open-services.net/legal-agreements/members-agreement/

[4] http://open-services.net/wiki/communications/File:OSLC_History_Lesson_v201310.ppt

4 Zusammenfassung und Ausblick

Dieser Beitrag hat OSLC als Community-Ansatz zur Lösung der Toolintegrations-Problematik in der Industrie vorgestellt. Mit OSLC wird versucht einen Standard zu etablieren, um die Zusammenarbeit auf der technischen Ebene zu verbessern. IBM setzt als Vorreiter schon das auf OSLC aufbauende eigene Tool Rational Team Concert unternehmensweit ein, um die Zusammenarbeit der Entwickler im Entwicklungsprozess zu verbessern. Während die Software-Industrie, für die OSLC ursprünglich definiert wurde, heute schon als Front-Runner auftritt, ist es in der klassischen Industrie wie beispielsweise Automotive noch nicht weit verbreitet.

Die OSLC Community umfasst heute schon einige der großen Softwarehersteller und Endanwender wie IBM, ORACLE, Mentor Graphics, PTC, Siemens, Boeing, EADS, GM und Ericsson. Es existieren bereits zahlreiche OSLC Implementierungen für unterschiedlichste Tools[5]. Um diesen Spezifikationen noch mehr Gewicht zu verleihen wurde 2013 begonnen die Spezifikationsarbeit im Rahmen des OASIS Konsortium zu erstellen[6]. Mit dem Eclipse Foundation Projekt Lyo[7] existiert für JAVA ein Source Development Kit (SDK), Testumgebung und Referenzimplementierung für OSLC. Das Codeplex Projekt OSLC4Net[8] bietet eine SDK und Referenzimplementierung für .NET.

Mit Forschungsprojekten wie CESAR, SPRINT, iFEST wurde gemeinsam mit zahlreichen Partnern versucht die Spezifikationen der einzelnen Domänen voran zu treiben und damit OSLC weiter auszurollen.

Danksagung

Part of the work described here has received funding from the EU ARTEMIS Joint Undertaking under grant agreement n° 269335 (MBAT) and the partners the Austrian Research Promotion Agency (FFG).

Literaturverzeichnis

Pautasso, C., Zimmermann, O. & Leymann, F. (2008). Restful web services vs. "big" web services: making the right architectural decision. *Proceedings of the 17th international conference on World Wide Web*, S 805-814.

Berners-Lee, T. (1998). Semantic Web Road map. www.w3.org/DesignIssues/Semantic.html (Zugriff am 23.05.2014).

Berners-Lee, T.; Hendler, J., Lassila, O. (2001): The semantic web. *Scientific american*, *284*(5), S 28-37.

[5] http://open-services.net/software/

[6] http://www.oasis-oslc.org/node/7

[7] http://eclipse.org/lyo/

[8] http://oslc4net.codeplex.com/

Berners-Lee, T. (2006): Linked Data. www.w3.org/DesignIssues/LinkedData.html (Zugriff am 23.05.2014).

OSLC Core Specification Workgroup (2013): Open Services for Lifecycle Collaboration Core Specification Version 2.0. http://open-services.net/bin/view/Main/OslcCoreSpecification (Zugriff am 23.05.2014)

Herbsleb, J. D. (2007). Global software engineering: The future of socio-technical coordination. *Future of Software Engineering*. IEEE Computer Society, S 188-198.

Wassermann, A (1989): Tool Integration in software engineering environments. *The International Workshop on Environments Software Engineering Environments, volume 647 of Lecture Notes in Computer Sciences*. Berlin: Springer, S 137-149.

Thomas, I., Nejmeh, B. (1992): Definitions of Tool Integration for Environments. *IEEE Software, 9(2)*. S 29-35.

Müller, P., Pasch, F., Drewinski, R., Bedenbender, H., Hayka, H., & Stark, R. (2012). Study on collaborative product development and digital engineering tools. *Product Lifecycle Management. Towards Knowledge-Rich Enterprises*. Berlin: Springer, S 389-399.

Kontaktinformationen

DI (FH) Stefan Paschke

Information & Process Management (Area A)
VIRTUAL VEHICLE Research Center
Kompetenzzentrum - Das Virtuelle Fahrzeug Forschungsgesellschaft mbH

Inffeldgasse 21a
A-8010 Graz/Österreich

Tel: +43/316/873-9049
Fax: +43/316/873-9002

E-Mail: stefan.paschke@v2c2.at