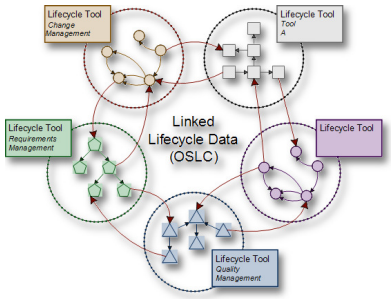
requisis

Auf dem Weg zum Online-Austausch von Anforderungen mit OSLC: Chancen und Hürden



REConf 2017

Nikolai Stein-Cieslak
Dr. Tim Meyer
REQUISIS GmbH

requisis

Übersicht

- Unser Interesse an OSLC
- Was ist OSLC?
- Unser Fazit:
  - Stärken und Schwächen / Ist OSLC heute im RM einsetzbar?
- Beispiele für Einsatzszenarien
- Integration von Anforderungen und Modellen
- OSLC und ReqIF:
  - Vergleich / Transformation / Zusammenspiel
- OSLC 3.0
  - Blick in die Zukunft

2

requisis

Nikolai Stein

Requirements Manager & Geschäftsführer

- Seit 2002 im Bereich des RE&M mit DOORS im Automobilsektor tätig
- Seit 2007 Senior Consultant & Geschäftsführer der REQUISIS GmbH
- RE&M-Methoden und –Konzepte, Daten-Austausch mit ReqIF und OSLC

Dr. Tim Meyer

Software Project Engineer

- Vorbereitung und Begleitung zur ReqIF-Einführung
- Konzeptentwicklungen zur Toolintegration
- Analyse des OSLC-Standards und Mitwirkung an Implementierung

3

requisis

REQUISIS GmbH

- Outstanding in Engineering
- Bundesweit agierendes Unternehmen mit Standorten in Berlin und Stuttgart
- Spezialist für toolunterstütztes Anforderungsmanagement
- Viele Projektkunden, z.B. im Automobilsektor
- Softwareprodukte (für DOORS):
  - DoX – Document eXport
  - ReX – Requirements eXchange
  - MoRE – Mobile Requirements Editor
  - SproX – Specification Proposal eXtractor
  - Insight – Requirements Project Monitor

4

requisis

Unser Interesse an OSLC

- **Themen unserer täglichen Arbeit beim Kunden:**
  - Austausch von Anforderungen (Abstimmung nach HIS, Tool-Migration, etc.)
  - Umgang mit Austauschformaten: ReqIF, Rif, etc.
  - Problemlösungen finden
- **Eigene Produkte zum Austausch von Anforderungen (ReqIF, Word, HTML)**
- **Aktive Mitarbeit im ReqIF-Implementor-Forum**
- **Zukünftig immer wichtiger:**
  - Standards & Interoperabilität zwischen Werkzeugen
  - Zusammenspiel RM und anderen Disziplinen, z.B. MBSE

→ OSLC: Nicht nur Verlinkung, sondern auch für Austausch/Datentransfer

5

requisis

Was ist OSLC?

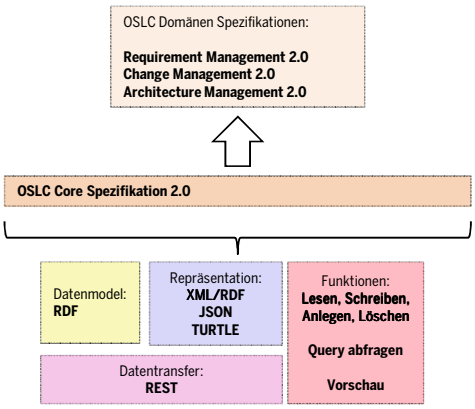
- **OSLC: "Open Services for Lifecycle Collaboration"**
- **Spezifikationen unter: open-services.net**
- **Seit 2013 ist die OSLC-Initiative Mitglied von OASIS.**
- **Derzeit sind die meisten Spezifikation in Version 2.0 verfügbar.**
- **OSLC Core 3.0 ist in Arbeit: Ein Entwurf wird derzeit verabschiedet.**

6

requisis

OSLC: Technologie

> Basiert auf Web Technologien des W3C und „Linked Data“ Prinzipien
> Definiert, welche Technologien wie eingesetzt werden können/müssen



→ Anpassung an ein Themengebiet

→ Allgemeines

Die OSLC-Spezifikationen enthalten:
wenig: „**MUST**“
viele: „**SHOULD**“ / „**MAY**“

(nicht vollständig)

7

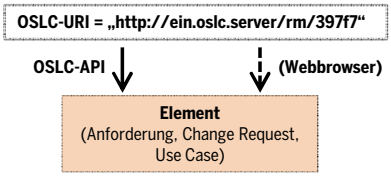
requisis

Grundgedanken von OSLC

> Jedes Element (Ressource) wird durch eine URI identifiziert, die meist eine Netzwerkadresse ist.

> OSLC definiert eine API, um auf die Daten über eine URI zuzugreifen

> OSLC definiert Dienste, die ein Server anbieten kann



8

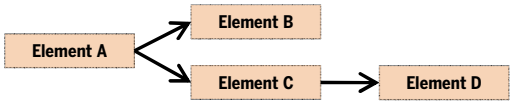
requisis

Grundgedanken von OSLC (im Vergleich zu ReqIF)

Identifizierung einzelner Elemente:

ReqIF-ID = „_03cnhf5235weh“

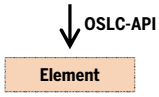
OSLC-URI = „http://ein.oslc.server/rm/397f7“



URIs werden verwendet für:
- Abbildung von Datenstrukturen
- Abbildung von Beziehungen (Links)

Möglichkeiten der OSLC-URI:

- Eindeutige Identifizierung
- Adresse zum Abrufen, falls das Element noch unbekannt ist



- Referenzieren, ohne das Zielelement zu besitzen
- Verteilte Speicherung

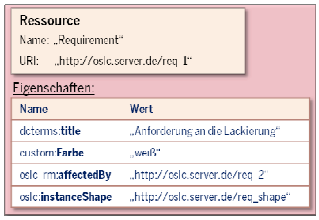
9

requisis

OSLC: Repräsentation von Informationen

> **Alle Informationen stecken in Ressourcen**

> **Ressourcen:**
> > Haben eine **URI** (→ ID)
> > Haben einen **Namen**
> > Haben eine beliebige Anzahl von **Eigenschaften** (engl. „properties“)



> **Eigenschaften (Attribute)**
> > Namespace:**Name**
> > Wert
> > Datendefinition:
> > > Datentyp
> > > Häufigkeit
> > > > „zero-or-one“
> > > > „zero-or-many“
> > > > etc.

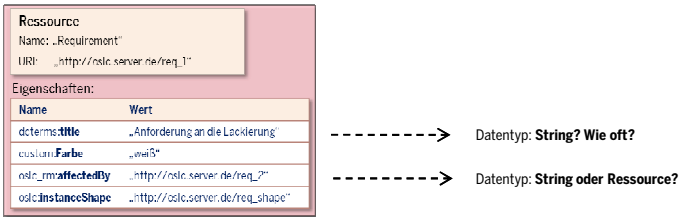
> **Datentypen für Werte:**
> > Drei Kategorien von Typen möglich:
> > 1. **Literal** (z.B. Integer-, Float-, String-Wert)
> > 2. **URI** zu einer anderen **Ressource**
> > 3. **Inline Ressource (eingebettet/verschachtelt)**

10

requisis

OSLC: Datenstruktur

> **Keine Angabe zur Eigenschafts-Definition (Typ) vorgeschrieben**



> **Lösung:**
  > In Spezifikation nachschauen (falls Standard-Eigenschaft)
  > „instanceShape"-Ressource verwenden (falls angeboten)

„instanceShape" ist optional!

11

requisis

OSLC: „Requirement Management 2.0" Domäne

> **Übernimmt OSLC-Core 2.0**
> **Definiert zwei Arten von Ressourcen: „Requirement" + „Requirement Collection"**

  > „**Requirement**" Ressource:
    > Repräsentiert eine Anforderung
    > 12 definierte „Attribut"-Eigenschaften
    > 14 definierte „Link"-Eigenschaften

  > „**Requirement Collection**" Ressource:
    > Repräsentiert eine ungeordnete Liste von Anforderungen (ein Modul ohne Hierarchie)
    > 12+1 definierte „Attribut"-Eigenschaften
    > 14 definierte „Link"-Eigenschaften
    > „uses"-Eigenschaft: Liste mit „Requirement"-Ressourcen

12

requisis

Zusammenfassung

- **OSLC**
  - definiert eine dezentrale Datenstruktur
  - definiert eine API für den Zugriff: Lesen, Schreiben, Suchen, etc. über REST
- **Stärken:**
  - Offener Standard
  - Leicht zu implementieren, Daten anzubieten
- **Schwächen:**
  - Definition für Eigenschaften („instanceShape“) ist optional
  - Definition für Eigenschaften ist teilweise vage:
    - Für Aufzählungen gibt es keine Richtlinie
    - Formatierter Text ist uneingeschränktes XHTML
  - Viel „Nachladen“ von Informationen notwendig

13

requisis

Hürden im RE-Umfeld

Essenzielle Funktionen, die OSLC fehlen:

- Strukturierte Anforderungen (in Modulen) darstellen
  - Defizit in RM 2.0
  - Könnte in RM 2.1/3.0 gelöst werden
  - Kann heute implementationsspezifisch gelöst werden („Child“/„Parent“-Eigenschaften)
- Umgang mit Dateianhängen
  - Defizit in Core 2.0
  - Wird in Core 3.0 gelöst

➔ **Reines OSLC 2.0 ist nur eingeschränkt geeignet, um es im Anforderungsmanagement einzusetzen!**

14

requisis

OSLC bereits heute mit Mehrwert nutzen

15

requisis

Kopieren von Daten statt nur Referenzieren (Link-Setzen)

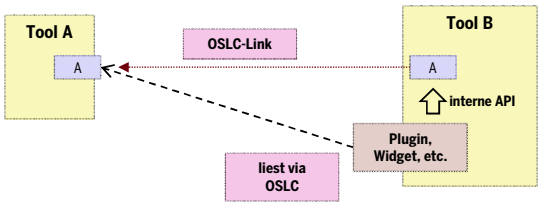
- **Grundsätzlich beruht das Prinzip von Linked Data auf einem Single-Source-Prinzip**
  - Informationen werden bei Bedarf nachgeladen
- **Lokale Kopie oder Cache der Daten sinnvoll:**
  - Performance bei Abfragen/Filterungen auf verlinkte Elemente
  - Referenzstand zum Feststellen von Änderungen

16

requisis

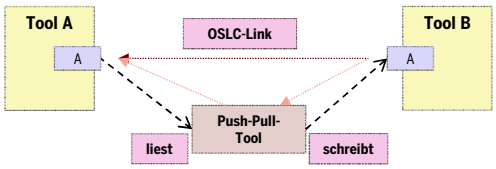
Umsetzung

> Variante 1)



> Vorteil: keine zusätzliche Infrastruktur, eigene interne API-Funktionen nutzbar

> Variante 2)



> Anlegen von Links auf das Original oder auf eine „Proxy-Adresse“ des Push-Pull-Tools
> Vorteil: Zentrales Mapping / Transformation von Daten möglich (im Proxybetrieb)
> Nachteil: Für das initiale Importieren wird die interne API oder ReqIF benötigt (zum Anlegen der Attribute im Tool).

17

requisis

Beispiele für Szenarien (aus technischer Sicht)

Die nachfolgenden Szenarien betrachten mögliche Anwendungen von OSLC.

> Umsetzung aktuell mit Toolerweiterungen möglich.
> Ohne Berücksichtigung rechtlicher / organisationspolitscher Randaspekte!

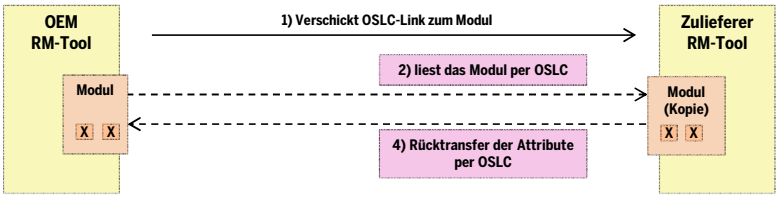
Aufgrund der entdeckten Einschränkungen von OSLC erfordern diese:

> Einen Workaround mit OSLC-Mitteln (z.B. Hierarchie über Eigenschaften)
> Oder einen Verzicht auf Funktionen (z.B. Dateianhänge)
> Oder die Nutzung zusätzlicher proprietärer APIs

18

requisis

HIS-Abstimmung



3) Zulieferer füllt Status/Kommentar-Attribut aus

4) Rückweg:
- > Variante 1: Zulieferer schreibt seine Werte zurück.
- > Variante 2: Zulieferer schickt OSLC-Link. → OEM liest Werte.

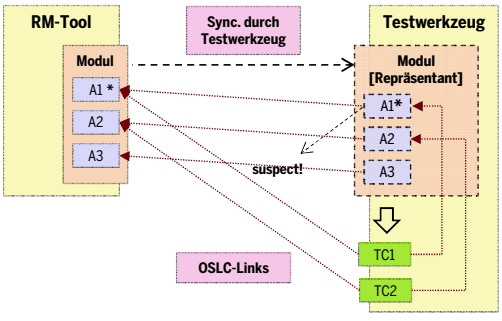
> OEM und Zulieferer können jeweils einen bestimmten Stand der Daten freigeben
- > Jeder Stand kann eine eigene OSLC-URI haben.

> Über Firmengrenzen hinweg: vermittelnder Proxy-Server

19

requisis

Toolintegration: Testfälle



1) Kopie und Sync. des Moduls per OSLC
2) Testfälle mit OSLC-Links zur Original-Anforderung
3) Änderung an Anforderung
4) OSLC-Link wird als suspect markiert

Vorteil gegenüber einfachen OSLC-Links (ohne Repräsentant):
> Das repräsentative Modul kann als interne Grundlage zum Erstellen der Testfälle dienen.
> Änderungen können automatisch detektiert werden (wegen vorhandener Vergleichsbasis).

20

requisis

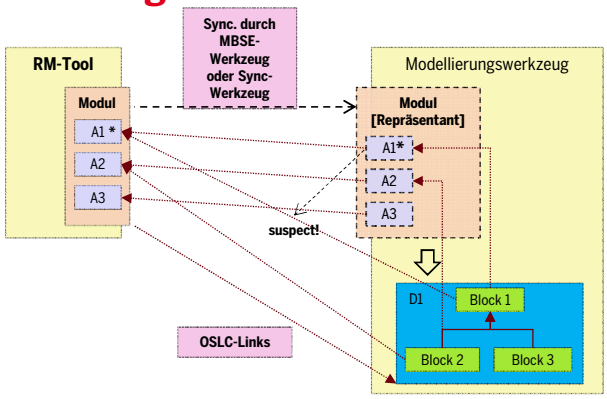
OSLC und Modelle:

- **Domäne „Architecture Management 2.0“ ist sehr einfach gehalten:**
  - Nur zwei Ressourcentypen
  - Ein Objekt für alle Elemente + eines für Verbindungstypen
  - Kein Konzept für grafische Darstellung von Diagrammen
  - Kein Konzept zum Umgang mit Stereotypes
  - ➔ Für Übertragung von kompletten Modellen vermutlich wenig hilfreich
- **Verlinkung zu Anforderungen analog zu den diskutierten Szenarien**

21

requisis

Toolintegration: Modellbasierte Entwicklung



1) Kopie und Sync. des Moduls per OSLC
2) Modellelemente mit OSLC-Links zur Original-Anforderung
3) Änderung an Anforderung
4) OSLC-Link wird als suspect markiert
5) Diagramme können mittels OSLC aus dem RE-Tool referenziert werden

Vorteil gegenüber einfachen OSLC-Links (ohne Repräsentant):

- **Das repräsentative Modul kann als interne Grundlage zum Erstellen der Elemente dienen.**
- **Änderungen können automatisch detektiert werden (wegen vorhandener Vergleichsbasis).**

22

requisis

Vergleich: ReqIF vs. OSLC

OSLC + RM 2.0 Domäne	ReqIF + Implementor Guide
Keine verpflichtende Definition der Datentypen	Fest definiertes Format zur Beschreibung der Datenstruktur
Namenskonvention in Spec	Namenskonvention im Implementor Guide
Online (Zugriff auf „Live"-Stand möglich)	Offline (Dateibasiert)
Wird abgerufen: Elemente können einzeln gelesen (und ggf. geschrieben) werden	Wird verschickt: Wird nur als „Gesamtpaket" ausgetauscht
Dezentrale Speicherung	Zentrale Speicherung

23

requisis

Abbildung: ReqIF ↔ OSLC

- **Ähnliche Datentypen**
- **Problematisch:**
  - Hierarchien in OSLC
  - Eigenschaften mit Häufigkeit „*–or-many" in OSLC (außer bei Links)
  - Vollständiges XHTML in OSLC
  - Links in ReqIF sind eigene Elemente (Attribute, ID, Quelle, Ziel)
  - Definition von Aufzählungstypen in OSLC nicht eindeutig definiert

24

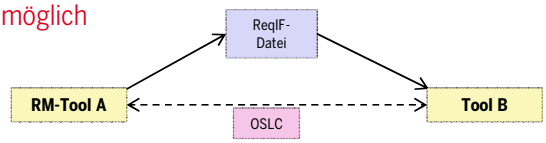
requisis

Zusammenspiel: ReqIF + OSLC

- **Zunächst: Bekanntmachen eines Datensatzes mit ReqIF**
  - Enthält OSLC-Links auf die Quelle
  - Enthält die Definition der Datenstruktur
  - Enthält Hierarchieinformationen

Für die Darstellung von OSLC-Links in ReqIF gibt es einen Vorschlag des ReqIF-Implementor Forums

- **Danach: Synchronisation per OSLC**
  - z.B. für:
    - Rückspielen von Änderungen
    - Prüfen auf Aktualität
  - Schneller Abgleich möglich



25

requisis

Ausblick: OSLC Core 3.0

- **Status**
  - 2. öffentliche Reviewphase kürzlich beendet
  - Weitgehend abwärtskompatibel zu Core 2.0
- **Verbesserungen**
  - Definition der Datenstruktur („Resource Shape“) ist Teil des Standards (Immer noch optional)
  - Ressourcen können Dateianhänge haben
  - Empfehlung, wie Aufzählungen dargestellt werden sollen

➔ **Wichtige Kritikpunkte sind bereits erkannt.**

26

requisis

Vielen Dank für Ihre Aufmerksamkeit.

> Besuchen Sie auch unseren Messestand
> Lassen Sie sich von unseren Dienstleistungen und Produkten überzeugen!

> Für Fragen und Anregungen stehen wir gerne zur Verfügung!

> Nikolai Stein-Cieslak
nikolai.stein@requisis.com
+49 (30) 536506-711

Dr. Tim Meyer
tim.meyer@requisis.com
+49 (30) 536506-734

27