OSLC-Based Integration Between Jama and MagicDraw

Axel Reichwein
April 13, 2018
Koneksys

Objectives

Demonstration of an integration between Jama and No Magic using OSLC

Through the OSLC adaptors of Jama and MagicDraw, two requirements and their relationship will be added to MagicDraw, and then exported to Jama

Demo Scenario

Show what needs to be exchanged between Jama and MagicDraw

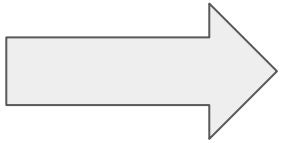
Show that Jama and MagicDraw have different APIs, data formats, and data identifiers

Show traditional approach

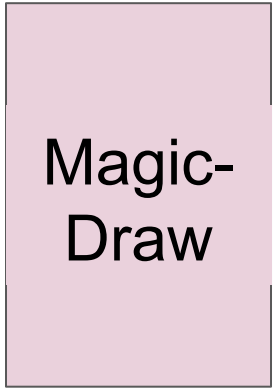
Show new approach using a common OSLC API for both Jama and MagicDraw

Data Exchange Scenario between Jama and MagicDraw

Step1: Client using OSLC API of MagicDraw



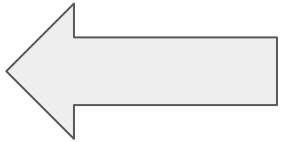
1. Add Requirement #1 to MagicDraw (e.g. with ID = 60)
2. Add Requirement #2 to MagicDraw (e.g. with ID = 61) with relationship to Requirement #1 (e.g. with ID = 60)



Data Exchange Scenario between Jama and MagicDraw

Step2: Client using OSLC API
of MagicDraw
and OSLC API of Jama

Jama



Magic-
Draw

3. Read Requirement #2 from MagicDraw (e.g. with ID = 61), which has a relationship to Requirement #1, and send it to Jama

Data Exchange Scenario between Jama and MagicDraw

Step3: Expected Result

Jama

- Requirement #2 from MagicDraw (e.g. with ID = 61) is converted into a Jama requirement.
- As Requirement #2 is related to Requirement #1, Requirement #1 is also converted into a Jama requirement
- As Requirement #2 is related to Requirement #1, the relationship is converted into a Jama relationship

Magic-Draw

Different APIs, data formats, and data identifiers

	Jama	**MagicDraw**
API	REST API	Java API
Data format	JSON	XMI
Data identifier scheme	Internal IDs (e.g. 6739)	Internal IDs (e.g. UUID)

Traditional approach to achieve interoperability: use 2 different APIs, manage 2 different data formats, and use 2 different data identification schemes -> time-consuming and expensive!

Common API + data format + data identifiers

	Jama	**MagicDraw**	**OSLC**
API	REST API	Java API	Hypermedia REST API
Data format	JSON	XMI	RDF (e.g. XML or JSON-LD)
Data identifier scheme	Internal IDs (e.g. 6739)	Internal IDs (e.g. UUID)	URL

Approach to achieve interoperability with OSLC: use 1 common API, 1 common data format, and 1 common data identification scheme -> fast and cheap!

Prerequisites for running Jama-MagicDraw Integration

Run MagicDraw OSLC adapter on port 8181 (different port than used for Jama OSLC adapter if adapters are both deployed locally on same machine for testing)

Instructions at https://github.com/ld4mbse/oslc-adapter-magicdraw-sysml

Run Jama OSLC adapter for example on port 8080 (default port)

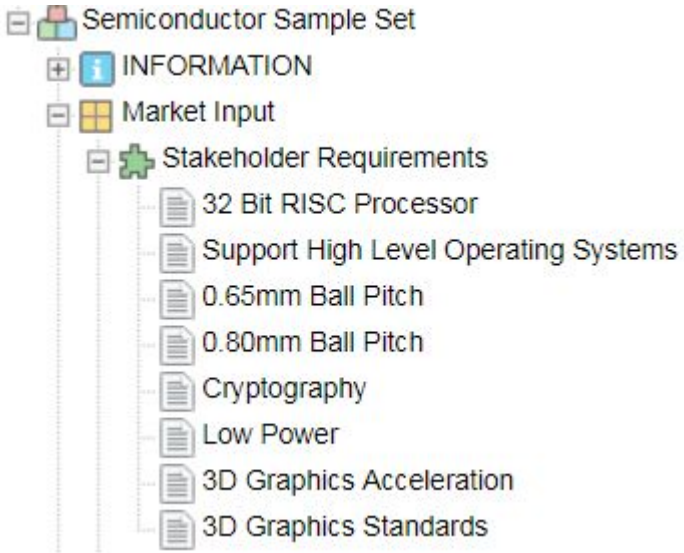
Instructions at https://github.com/OSLC/oslc-adapter-jama

Prerequisites for running Jama-MagicDraw Integration

Check beforehand content in MagicDraw and Jama projects

Example in Jama

Requirements in Jama project called Semiconductor Sample Set



Prerequisites for running Jama-MagicDraw Integration

Check beforehand content in exposed by OSLC adapter

Example in Jama

Requirements in Jama project called Semiconductor Sample Set exposed by OSLC adapter in HTML at http://localhost:8080/jama-oslc-adapter/services/Semiconductor_Sample_Set/requirement

Jama Requirements Ser

Requirements

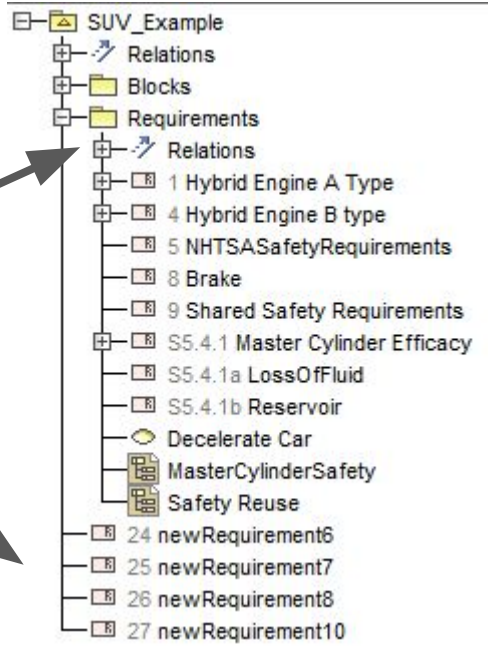
ID	Title
3920	Support High Level Operating Systems
3921	0.65mm Ball Pitch
3922	0.80mm Ball Pitch
3923	Cryptography
3924	Low Power
3925	3D Graphics Acceleration
3926	3D Graphics Standards
3919	32 Bit RISC Processor

Prerequisites for running Jama-MagicDraw Integration

Check beforehand content in MagicDraw and Jama projects

Example in MagicDraw

Requirements in MagicDraw project called SUV_Example located under Requirements package and under root project



Prerequisites for running Jama-MagicDraw Integration

Check beforehand content in exposed by OSLC adapter

Example in Jama

Requirements in MagicDraw project called SUV_Example located under Requirements package and under root project exposed by OSLC adapter in HTML at http://localhost:8181/oslc4jmagicdraw/services/SUV_Example/requirements

MagicDraw Requirements SUV_Example

Requirements

Safety_Requirements_for_type_A
Shared_Safety_Requirements
newRequirement10
NHTSASafetyRequirements
newRequirement8
Reservoir

Shared_Safety_Requirements
Brake
Shared_Safety_Requirements
newRequirement6
Hybrid_Engine_A_Type

Safety_Requirements_for_type_B
LossOfFluid
Master_Cylinder_Efficacy
newRequirement7
Hybrid_Engine_B_type

Data Exchange Scenario Steps

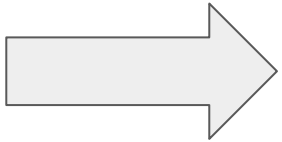
Step 1	Client using OSLC API of MagicDraw	1. Add Requirement #1 to MagicDraw (e.g. with ID = 60) 2. Add Requirement #2 to MagicDraw (e.g. with ID = 61) with relationship to Requirement #1 (e.g. with ID = 60)
Step 2	Client using OSLC API of MagicDraw and OSLC API of Jama	3. Read Requirement #2 from MagicDraw (e.g. with ID = 61), which has a relationship to Requirement #1, and send it to Jama
Step 3		Verification

Data Exchange Scenario Step 1

Step1: Client using OSLC API of MagicDraw

Run Java applications acting as REST Clients

1. Run OSLC_POST_req_to_MagicDraw
2. Run OSLC_POST_req_and_rel_to_MagicDraw



1. Add Requirement #1 to MagicDraw (e.g. with ID = 60)
2. Add Requirement #2 to MagicDraw (e.g. with ID = 61) with relationship to Requirement #1 (e.g. with ID = 60)

Magic-Draw

Data Exchange Scenario Step 1.1

Run OSLC_POST_req_to_MagicDraw

https://github.com/OSLC/oslc-adapter-jama/blob/clean-ver/src/main/java/com/jama/oslc/client/OSLC_POST_req_to_MagicDraw.java

Pseudo Code of OSLC_POST_req_to_MagicDraw

1. Create POJO describing new MD requirement
2. Convert POJO into RDF
3. Send POST request with RDF as body

Data Exchange Scenario Step 1.1

Creating POJO describing new MD requirement

https://github.com/OSLC/oslc-adapter-jama/blob/clean-ver/src/main/java/com/jama/oslc/client/OSLC_POST_req_to_MagicDraw.java#L56

URL used as requirement identifier (important OSLC concept!)

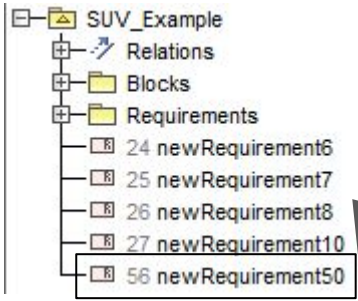
```
SysMLRequirement newRequirementToAdd = new SysMLRequirement();
String requirementIdentifier = "newRequirement50";
newRequirementToAdd.setAbout(URI.create("http://localhost:8181/oslc4jmagicdraw/services/SUV_Example/requirements/"
newRequirementToAdd.setTitle("New MagicDraw Requirement X");
newRequirementToAdd.setDescription("description of " + requirementIdentifier);
newRequirementToAdd.setIdentifier(requirementIdentifier);
```

Data Exchange Scenario Step 1.1 Verification

After running OSLC_POST_req_to_MagicDraw -> new requirement in MagicDraw

New requirement exposed by OSLC adapter

New requirement in MagicDraw



MagicDraw Requirements SUV_Example

Requirements

Safety_Requirements_for_type_A	Shared_Safety_Requirements	Safety_Requirements_for_type_B
Shared_Safety_Requirements	Brake	LossOfFluid
newRequirement10	Shared_Safety_Requirements	Master_Cylinder_Efficacy
NHTSASafetyRequirements	newRequirement6	newRequirement7
newRequirement8	newRequirement50	Hybrid_Engine_A_Type
Hybrid_Engine_B_type	Reservoir	

Data Exchange Scenario Step 1.2

Run
OSLC_POST_req_and_rel_to_MagicDraw

https://github.com/OSLC/oslc-adapter-jama/blob/clean-ver/src/main/java/com/jama/oslc/client/OSLC_POST_req_and_rel_to_MagicDraw.java

Pseudo Code of
OSLC_POST_req_to_MagicDraw

1. Creating POJO describing new MD requirement **(including link describing deriveFrom relationship to other requirement)**
2. Converting POJO into RDF
3. Sending POST request with RDF as body

Data Exchange Scenario Step 1.2

Creating POJO describing new MD requirement with relationship to other requirement

https://github.com/OSLC/oslc-adapter-jama/blob/clean-ver/src/main/java/com/jama/oslc/client/OSLC_POST_req_and_rel_to_MagicDraw.java#L64

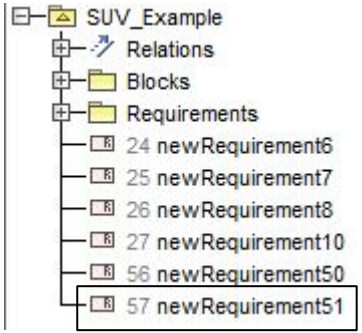
Link defined as POJO attribute and link type and link target defined by URLs u(important OSLC concept!)

```
String requirementIdentifier2 = "newRequirement50";
Link[] derivedFromLinksArray = new Link[1];
derivedFromLinksArray[0] = new Link(URI.create("http://localhost:81[illegible]/oslc4jmagicdraw/services/SUV_Example/requirem
newRequirementToAdd.setDerivedFromElements(derivedFromLinksArray);
```

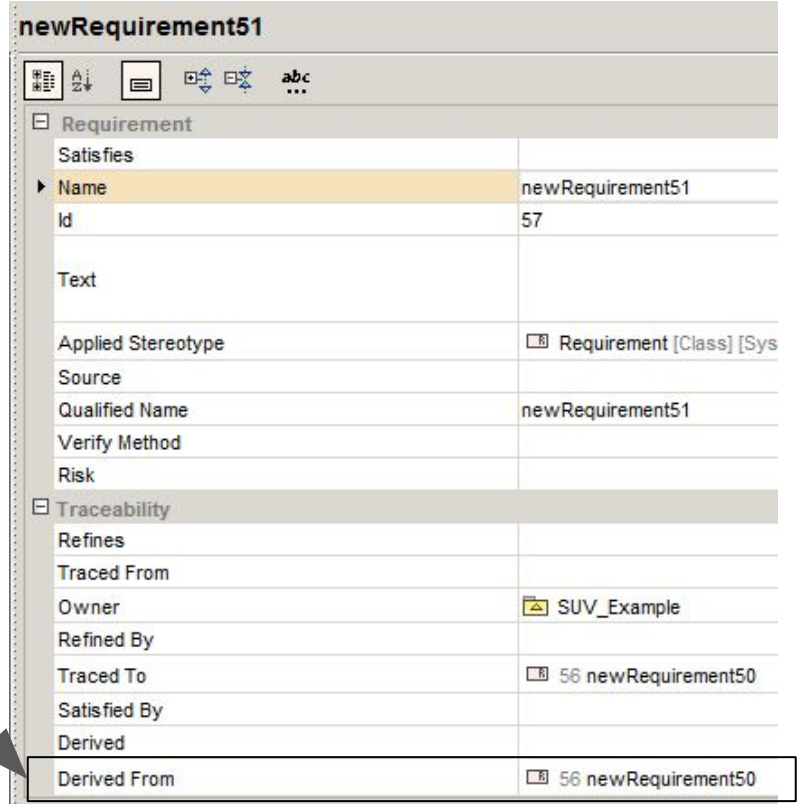
Data Exchange Scenario Step 1.2 Verification

After running OSLC_POST_req_and_rel_to_MagicDraw -> new requirement and relationship in MagicDraw

New requirement in MagicDraw



New requirement relationship in MagicDraw



Data Exchange Scenario Step 1.2 Verification

After running OSLC_POST_req_and_rel_to_MagicDraw -> new requirement and relationship in MagicDraw

New requirement in MagicDraw exposed by OSLC adapter

MagicDraw Requirements SUV_Example

Requirements

Safety_Requirements_for_type_A	Shared_Safety_Requirements	Safety_Requirements_for_type_B
Shared_Safety_Requirements	Brake	LossOfFluid
newRequirement10	Shared_Safety_Requirements	Master_Cylinder_Efficacy
NHTSASafetyRequirements	newRequirement6	newRequirement51
newRequirement7	newRequirement8	newRequirement50
Hybrid_Engine_A_Type	Hybrid_Engine_B_type	Reservoir

MagicDraw Requirement newRequirement51

Description: description of newRequirement51

Derived From

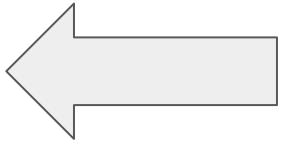
newRequirement50

New requirement relationship in MagicDraw exposed by OSLC adapter http://localhost:8181/oslc4jmagicdraw/services/SUV_Example/requirements/newRequirement51

Data Exchange Scenario Step 2

Step2: Client using OSLC API
of MagicDraw
and OSLC API of Jama

Jama



Magic-
Draw

3. Read Requirement #2 from MagicDraw (e.g. with ID = 61), which has a relationship to Requirement #1, and send it to Jama

Run Java application acting as REST Clients
Run OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama

Data Exchange Scenario Step 1.1

Run OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama

https://github.com/OSLC/oslc-adapter-jama/blob/clean-ver/src/main/java/com/jama/oslc/client/OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama.java

Pseudo Code of OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama

1. Perfom GET request on MagicDraw requirement containing relationship to other requirement
2. Convert RDF representation of requirement into POJO
3. Check relationships of requirement
4. For every relationship to another requirement, create requirement and relationship in Jama by using POST requests
5. Send POST request to create original requirement in Jama

Data Exchange Scenario Step 2 Verification

After running OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama
-> new requirement and relationship in Jama corresponding to new requirement and relationship in MagicDraw

New requirements in Jama exposed by OSLC adapter

Jama Requireme

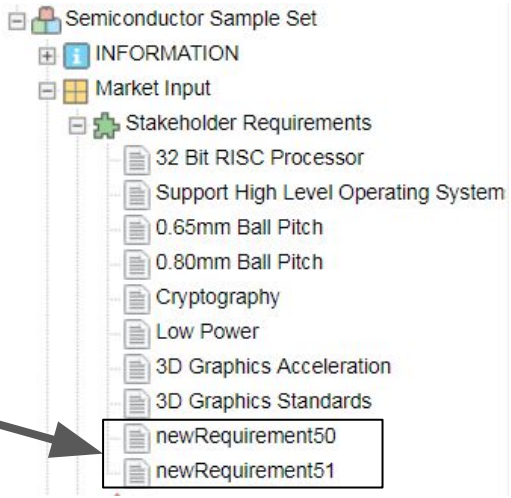
Requirements

ID	Title
3920	Support High Level Operating Systems
3921	0.65mm Ball Pitch
3922	0.80mm Ball Pitch
3923	Cryptography
6916	newRequirement50
3924	Low Power
3925	3D Graphics Acceleration
3926	3D Graphics Standards
6917	newRequirement51
3919	32 Bit RISC Processor

Data Exchange Scenario Step 2 Verification

After running OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama
-> new requirement and relationship in Jama corresponding to new requirement and relationship in MagicDraw

New requirements in Jama



Data Exchange Scenario Step 2 Verification

After running OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama
-> new requirement and relationship in Jama corresponding to new requirement and relationship in MagicDraw

New requirement relationship in Jama exposed by OSLC adapter

Jama Requirement newRequirement51

Identifier: 6917

Description: description of newRequirement51

Document Key: null

Global ID: null

Project: Semiconductor_Sample_Set

Created: null

Modified: null

Parent ID: 569

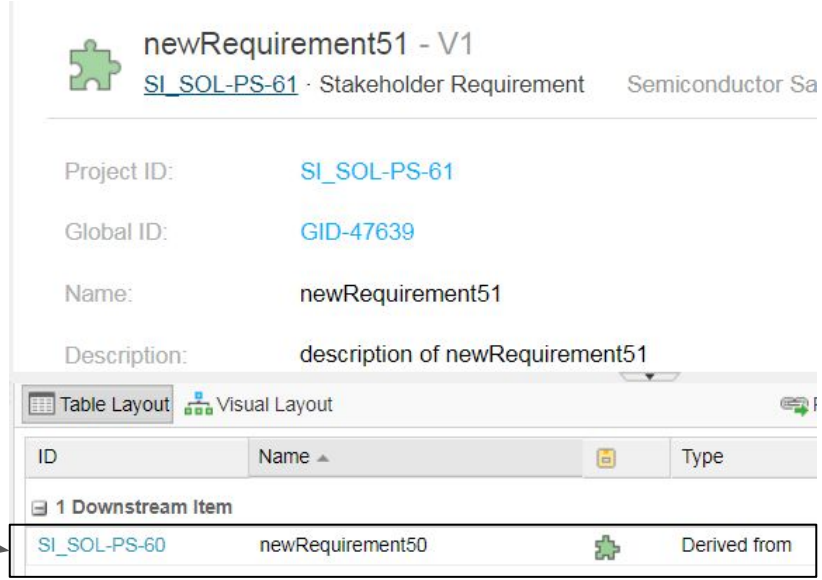
DerivedFrom

6916

Data Exchange Scenario Step 2 Verification

After running OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama
-> new requirement and relationship in Jama corresponding to new requirement and relationship in MagicDraw

New requirement relationship in Jama



More REST clients available for testing

On GitHub at
https://github.com/OSLC/oslc-adapter-jama/tree/clean-ver/src/main/java/com/jama/oslc/client

- GET_relationships_from_Jama.java
- OSLC_GET_req_and_rel_from_MagicDraw_and_POST_to_Jama.java
- OSLC_GET_req_from_MagicDraw.java
- OSLC_GET_req_from_MagicDraw_and_POST_to_Jama.java
- OSLC_JamaAdapterDiscoveryClient.java
- OSLC_POST_req_and_rel_to_Jama.java
- OSLC_POST_req_and_rel_to_MagicDraw.java
- OSLC_POST_req_to_Jama.java
- OSLC_POST_req_to_MagicDraw.java
- POST_relationship_to_Jama.java
- POST_req_and_rel_to_Jama.java
- POST_requirement_to_Jama.java

Thanks and get in touch!

axel.reichwein@koneksys.com