OSLC ALM-PLM Interoperability
Proof of Concept

Mike Loeffler
Systems Engineering IT Specialist
General Motors Company
michael.loeffler@gm.com
Disclaimers and Fine Print

• No information contained herein represents implied or expressed product direction of any of the below mentioned third parties

• Trademarks used herein are property of their respective owners:
  – Teamcenter and TcUA are trademarks property of Siemens PLM
  – Rational Team Concert is trademark property of IBM
  – Java is trademark property of Oracle
  – OSLC is trademark property of Open Services for Lifecycle Collaboration standards group
  – Eclipse is trademark property of Eclipse Foundation
  – SysML is trademark property of the Object Management Group
  – Other names and trademarks may be claimed by others

• Example data mentioned herein is modeled after the “HSUV Example” from OMG SysML Specification Version 1.2, Appendix B, at: [http://www.omg.org/spec/SysML/1.2/PDF/](http://www.omg.org/spec/SysML/1.2/PDF/)
What is ALM?

• Application Lifecycle Management
• PLM for Software Products (Applications)
• Domains Include
  – Requirements
  – Change and Configuration Management
  – Architecture Resources (Models)
  – Source Code
  – “Assets” (Binary Objects)
  – Quality Management (Testing)
  – Automation of Software Build Processes
• New and Evolving Concept
What Is OSLC?

• **Open Services for Lifecycle Collaboration**
• Emerging Standard for Tool Integrations in ALM Domains
• Loosely Coupled
• Semantic Web Linked Data
• Based on Web Architecture – RDF, HTTP
• RESTful Services
• Details at [http://www.open-services.net](http://www.open-services.net)
Why ALM-PLM?

• GM Products Require Increasing Amounts of Embedded Software (Mechatronics)
• Development Requires a Systems Engineering Approach to Assure Best Customer Value
• Systems Engineering of These Products Forces Collaboration Around Cross-Cutting Concerns
  – Requirements (Traceability)
  – Behavioral Definition and Models
  – Physical Implementation Allocation Decisions (Hardware vs. Firmware vs. Software)
OSLC PLM Work Group

- Chartered in 2010
- Identifying OSLC Core Spec Extensions and/or Specific PLM Domain Specification
- Details at http://openservices.net/bin/view/Main/PlmHome
- Initial Scenario: “A Systems Engineer responds to a change in requirements for an existing product release”
ALM-PLM Scenario

• High Level Steps
  – Identify the change and assign it to be worked
  – Find impacted objects (requirements, designs, etc.)
  – Satisfy change with reused, modified or new versions of objects
  – Approve revised design and close the change request

• System being changed is composed of hardware, firmware and software elements

• Changes required to all aspects of the system

• Details at [http://open-services.net/bin/view/Main/PlmSystemsEngineeringScenarioSystemsEngineerReactstoChangedRequirements](http://open-services.net/bin/view/Main/PlmSystemsEngineeringScenarioSystemsEngineerReactstoChangedRequirements)
POC Approach

• Goals of the POC
  – Prove that certain Objects in Teamcenter can be meaningfully “exposed” as OSLC Resources
  – Identify holes in current OSLC standards and propose solutions
  – Guidance for implementation

• Not Intended to Represent Any Production Implementation Approach
POC Architecture

[Diagram showing a detailed architecture of a POC system, including various devices, environments, and connectivity.]
Based on RIO Project

• “Reference Implementation of OSLC”
• Open source project originally hosted at: [http://sourceforge.net/projects/oslc-tools/](http://sourceforge.net/projects/oslc-tools/)
• Simple Standalone OSLC Web Services with RDF Store
• Provided Patterns and Code Base for POC
• Project Has Morphed into Eclipse Lyo at: [http://www.eclipse.org/projects/project.php?id=technology.lyo](http://www.eclipse.org/projects/project.php?id=technology.lyo)
Data Model Mapping

• Used Following Mappings for POC:
  – OSLC Requirement
    • TcUA Requirement, RequirementSpec or Paragraph
  – OSLC ChangeRequest
    • TcUA EngChange
  – OSLC Resource
    • TcUA CORP_Schematic, CORP_Software or CORP_Part
  – OSLC Product (New Proposed OSLC Resource for PLM)
    • TcUA CORP_Product, CORP_Proc_Plan, CORP_Vehicle or CORP_Install

• CORP_* Object Types are GM Overlay Specific, Similar OOTB Types Exist for Most

• Mappings Are Configurable
Semantic Assumptions

• OSLC Uses Semantic Web “Open World” Assumption
  – Every Resource has a URI (ideally persistent)
  – Non-existence is not implied by empty query result
  – Client must be tolerant of and responsible to preserve any unknown data elements
  – See: http://www.w3.org/standards/semanticweb/

• TcUA Uses Relational/Object-Oriented “Closed World” Assumption
  – Empty query result implies the queried object does not exist (as far as we are concerned)
  – Client has full data model knowledge and will likely fail if sent unknown data
Semantic Level Mismatch

• RIO-Lyo Design Assumed RDF Triple Store Backend
  – Triple store similar to relational table attribute level (e.g. one triple = 
    <subject> <predicate> <object>)

• TcUA SOA API is at Object Level
  – Higher level concepts (Item/Item Revision/Item View)
  – Many referential constraints built in

• TcUA-OSLC Connector Maps Resources to Objects (approximately)
  – RIO-Lyo code was forked and heavily modified to connect at a higher level 
    than the original RDF triple store
  – Made some simplifying assumptions on referential constraints

• POC Identified and Implemented Proposed OSLC PLM Extensions
  – Versions
  – View Definitions
  – Variants
Interaction Examples Lyo Web UI

- Login
- Interactive Change Request Query
- Interactive Navigation Using Linked Data
- Interactive Trace Link Navigation
- Interactive Trace to External Resource (Web Page)
- Create a Requirement
- Update a Requirement
- Add a Trace Link
Teamcenter Learnings

• Teamcenter SOA API is Very Complicated to Develop With
  – Many overlapping/ambiguous classes and methods, not clear which to use
• SOA Sample Code Very Helpful
  – Should be expanded
  – PLM Users Community Open Source effort???
• SOA Covers The Core Teamcenter Concepts Well
• Policy Tuning is Important
OSLC Learnings

- OSLC Needs PLM Concepts
  - Version/Revision Handling
  - Variants, Options and Effectivity
- OSLC RIO-Lyo Code Very Helpful
  - Open Source Eclipse-Lyo starting up
- OSLC Makes Loose Integrations Simple
- OSLC May Simplify Distributed/Federated Repository Implementation
Further Work Planned

• Demonstration of OSLC Configuration Management and PLM Specs
• Porting to New Eclipse Lyo SDK
• Add Lyo UI Support for Link Creation
• Security support for Oauth and Teamcenter Security Service (SSO)
• Study Integration into Teamcenter Web Application Server
• SPARQL Query Support
• Investigate Teamcenter RAC and Web Client Support (Rich Hover, Links)
• Demonstrate Specialized UI’s Based on OSLC (RSS Feeds for Change Notification, Mobile Apps, etc.)
• POC SysML Modeling Tool Integration
Call for Participation in OSLC

• ALM-PLM Users Unite!
• Open Call for Vendors to Support OSLC
  – Requirements Management
  – Change Management
  – Source Code Management
  – Architecture Management
  – Quality Management
  – Product Management
Credits

Special thanks to the following for their participation, help and encouragement in this POC:

- **The OSLC Connectors POC Team**
  - Mohamed Egal
  - Raheel Syed
  - James Rozum (advisor)
  - Thomas Tecco (advisor)

- **Members of the OSLC ALM-PLM Workgroup**
  - Rainer Ersch
  - Gray Bachelor

- **Members of other various OSLC Workgroups and the RIO-Lyo Implementers**
  - Jim Conallen
  - Steve Speicher
  - Mike Fiedler
  - Hiroaki Nakamura
  - Arvind Rengarajan