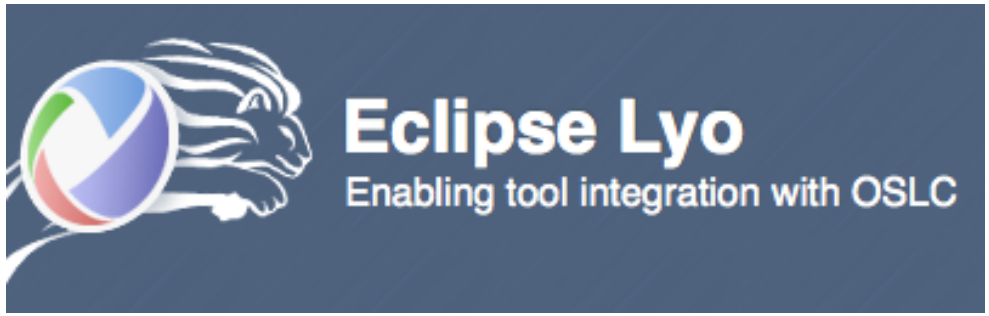
Open Services for Lifecycle Collaboration

open community. open interfaces. open possibilities.

Eclipse Lyo: Re-thinking tool integrations

http://eclipse.org/lyo

Michael Fiedler (@mffiedler, @oslcnews)
Steve Speicher (@sspeiche)



Agenda

- **Tool integrations are hard**
- **Linked Data approaches and OSLC**
- **Eclipse Lyo**
- **Lyo Content and Plans**
- **Participating**

The Core Problem of Tools

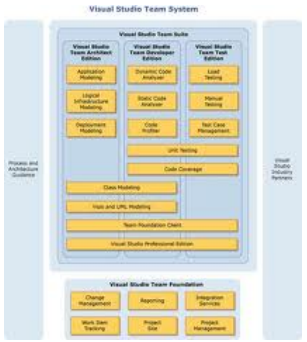
- Producing a good tool is relatively easy
- Producing a single tool to support all aspects of ALM is impossible
  - Organizations cannot or will not deploy a single tool solution
- But, integrating multiple tools has been unsatisfactory

Past approaches to lifecycle integration have fallen short

∅ Limited choice and coverage ∅

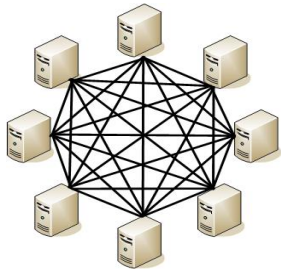
Single repository

"Can I really expect one vendor to provide all the functionality I need? And what about my existing tools?"



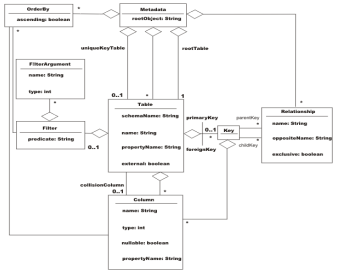
Point-to-point integrations

"How can I ever upgrade one tool without breaking everything else?"



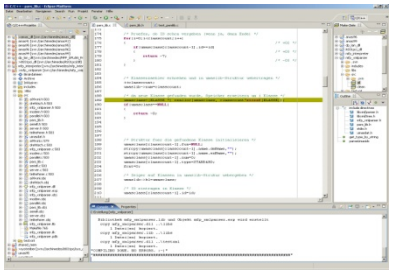
Universal metadata standard

"How did I ever think all those vendors would be able to agree?"



Standard implementations

"Did I really believe that every vendor would rewrite their tools on a single framework?"

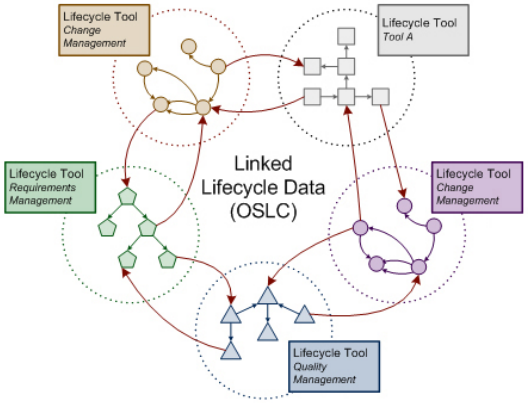


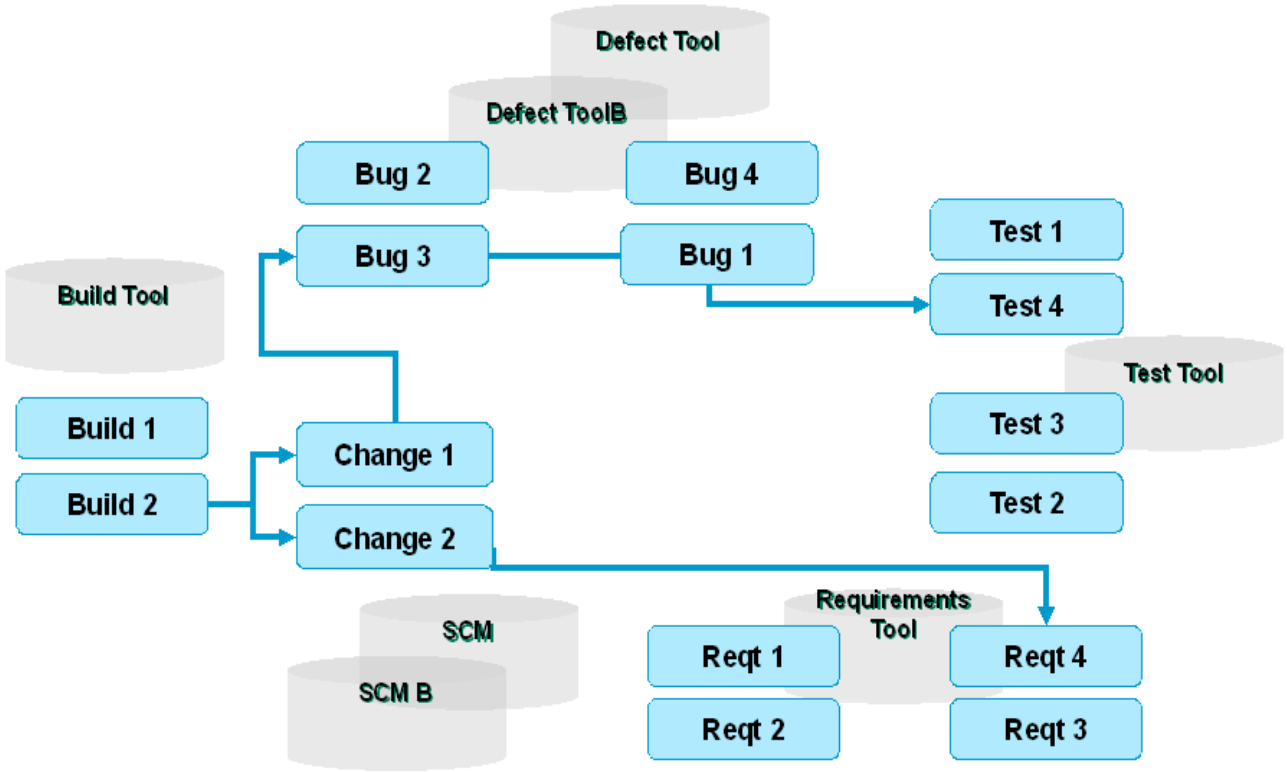
∅ Slow to emerge and disruptive to adopt ∅

Agenda

- **Tool integrations are hard**
- **Linked Data approaches and OSLC**
- **Eclipse Lyo**
- **Lyo Content and Plans**
- **Participating**

Linked Lifecycle Data





- **The data is the thing**
  - Resources and relationships
  - Tools operate on the data
  - Tools execute the process
  - Tools expose their data in a common way (REST)
- **Lifecycle integration = tracing, indexing, analyzing the web of lifecycle data where it lives**
- **Utilizes architecture of the internet**
  - All data are resources with HTTP URIs
  - Open standards
  - Loosely coupled
  - Technology neutral
  - Scalable, extensible

OSLC Community and Specifications

http://open-services.net/members/

- Range of interests, expertise, involvement
  - 400+ registered community members
  - Individuals from 34+ different companies have participated in OSLC workgroups

Accenture	**Lender Processing Services**
APG	**Northrop Grumman**
Black Duck	**Oracle**
Boeing	**QSM**
BSD Group	**Rally Software**
Citigroup	**Ravenflow**
EADS	**Shell**
Emphasys Group	**Siemens**
Empulsys	**Sogeti**
Fokus Fraunhofer	**SourceGear/Teamprise**
Galorath	**State Street**
General Motors	**Tasktop (Eclipse Mylyn)**
Health Care Services Corp	**Thales**
IBM	**Tieto**
Institut TELECOM	**TOPIC Embedded Systems**
Integrate Systems	**UrbanCode**
	WebLayers

http://open-services.net/software/

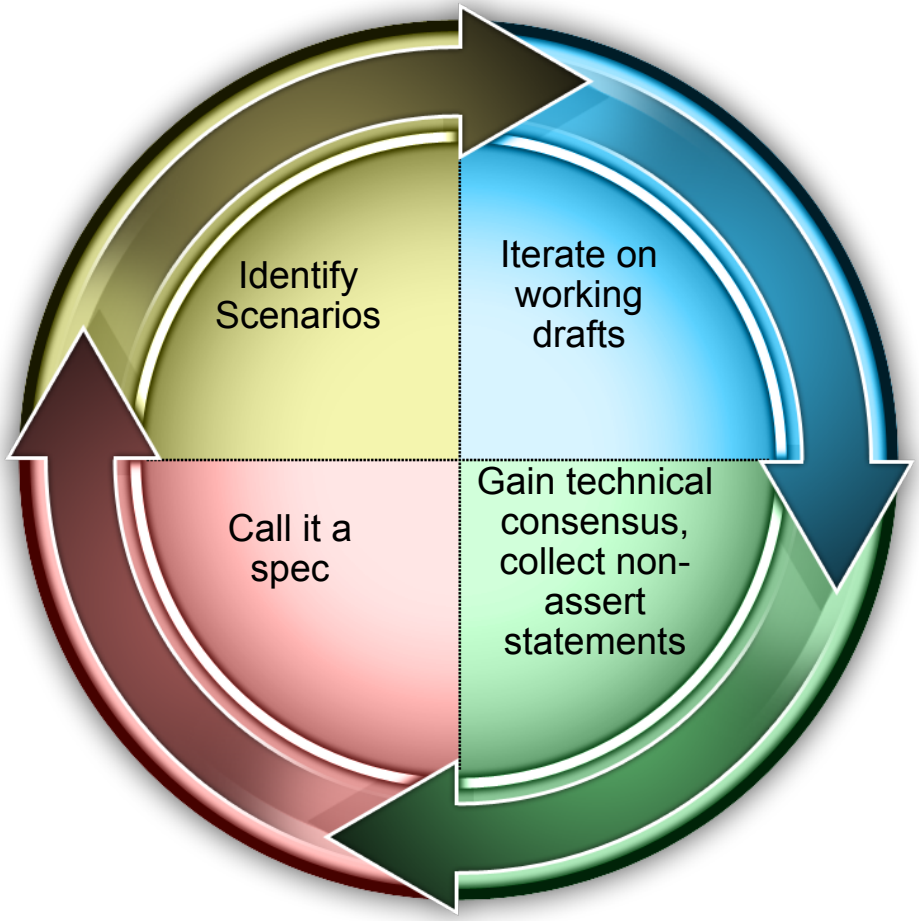
- Growing list of IBM and 3rd party software
  - Atlassian JIRA adapter for OSLC
  - Jenkins Plugin for OSLC
  - Kovair
  - Oracle Team Productivity Center
  - Tasktop

http://open-services.net/specifications/

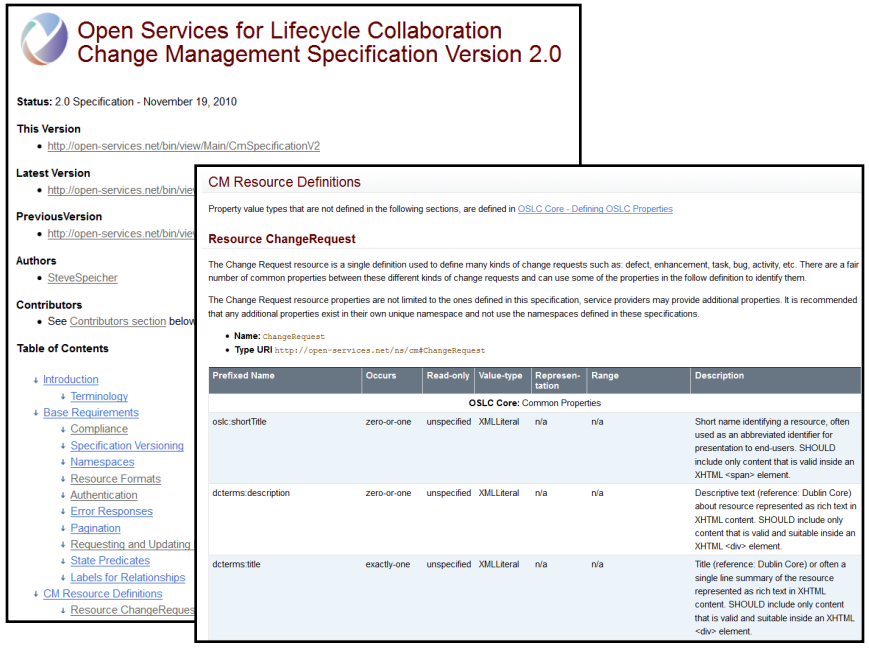
- Domain and Solution specifications
  - Change Management
  - Quality Management
  - Requirements Management
  - Product Lifecycle Management
  - More…

OSLC and Open Community
Iterative Specification Authoring

Open Services for Lifecycle Collaboration
open community. open interfaces. open possibilities



- Minimalist/additive approach
  - "Just enough" definition for a given domain
- Scenario driven scope
- Co-evolve spec and implementations
- Open participation around active groups


Open Services for Lifecycle Collaboration
Change Management Specification Version 2.0

Status: 2.0 Specification - November 19, 2010

This Version
- http://open-services.net/bin/view/Main/CmSpecificationV2

Latest Version
- http://open-services.net/bin/vie

PreviousVersion
- http://open-services.net/bin/vie

Authors
- SteveSpeicher

Contributors
- See Contributors section below

Table of Contents
- Introduction
  - Terminology
- Base Requirements
  - Compliance
  - Specification Versioning
  - Namespaces
  - Resource Formats
  - Authentication
  - Error Responses
  - Pagination
  - Requesting and Updating
  - State Predicates
  - Labels for Relationships
- CM Resource Definitions
  - Resource ChangeReques

CM Resource Definitions

Property value types that are not defined in the following sections, are defined in OSLC Core - Defining OSLC Properties

Resource ChangeRequest

The Change Request resource is a single definition used to define many kinds of change requests such as: defect, enhancement, task, bug, activity, etc. There are a fair number of common properties between these different kinds of change requests and can use some of the properties in the follow definition to identify them.

The Change Request resource properties are not limited to the ones defined in this specification, service providers may provide additional properties. It is recommended that any additional properties exist in their own unique namespace and not use the namespaces defined in these specifications.

- **Name:** `ChangeRequest`
- **Type URI** `http://open-services.net/ns/cm#ChangeRequest`

Prefixed Name	Occurs	Read-only	Value-type	Represen-tation	Range	Description
OSLC Core: Common Properties						
oslc:shortTitle	zero-or-one	unspecified	XMLLiteral	n/a	n/a	Short name identifying a resource, often used as an abbreviated identifier for presentation to end-users. SHOULD include only content that is valid inside an XHTML element.
dcterms:description	zero-or-one	unspecified	XMLLiteral	n/a	n/a	Descriptive text (reference: Dublin Core) about resource represented as rich text in XHTML content. SHOULD include only content that is valid and suitable inside an XHTML element.
dcterms:title	exactly-one	unspecified	XMLLiteral	n/a	n/a	Title (reference: Dublin Core) or often a single line summary of the resource represented as rich text in XHTML content. SHOULD include only content that is valid and suitable inside an XHTML element.

OSLC Specifications

- OSLC Core spec defines
  - HOW to use HTTP and RDF, how to define resources and services
  - Defines some common resource types and properties

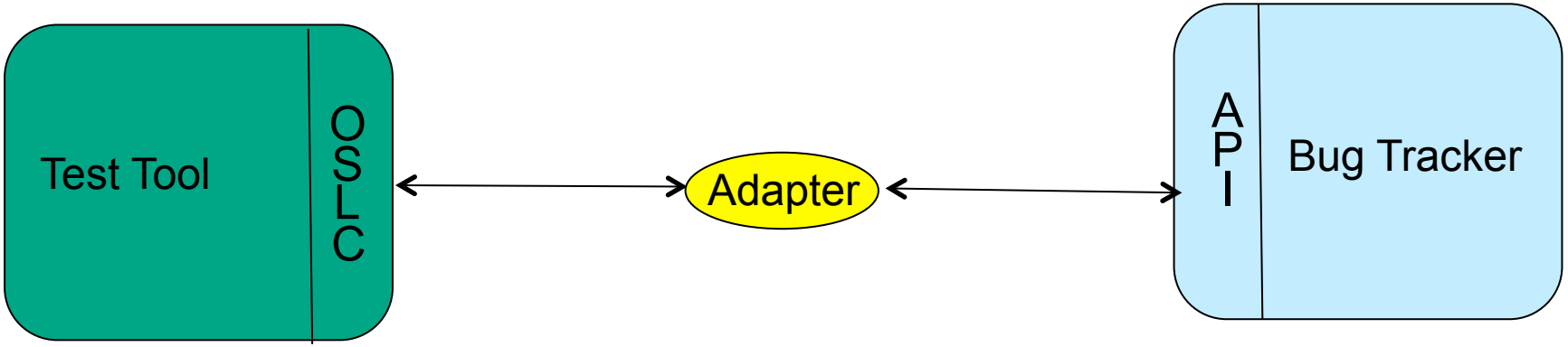
- OSLC domain specs (Change Management, Requirements, etc.)
  - Define WHAT resources and services required in the domain
  - Resource types, properties and relationships
  - Service providers, creation factories, query capabilities, operations

- But, what do I do with an OSLC spec? How do I use it for real integrations?
  - Commercial tools available with OSLC APIs
  - Many organizations developing OSLC integrations in-house
  - Educational and research institutions developing tool integrations based on OSLC
  - Would be good to have open source sample code, SDKs, tests, etc to help get started.

Integration approaches

- Native OSLC support tools
  - Works if you have control over the source
  - Implement the service provider and resources directly in the tool
  - Consider natively representing resources with RDF if it makes sense.

- Adapter approaches to enable OSLC in tools
  - Rely on the APIs provided by the tool
  - Plugin or standalone adapters depending on the tool architecture
  - Know your tool and what data you want to enable for linking



Getting Started

- Before writing any code
  - Scenarios to support?
  - What does integration mean?
  - What APIs do I have to work with?

- Example: Want to link data from my bug tracker to a test tool that already supports OSLC

  - From the test tool:
    - Open new bug while running a test
    - Search existing bugs and link a bug to a test case
    - Show a simple “view” of a bug’s details in my test tools web UI

  - From the bug tracker:
    - See (link or simple view) the test case that found this bug
    - See the test plan this bug is affecting

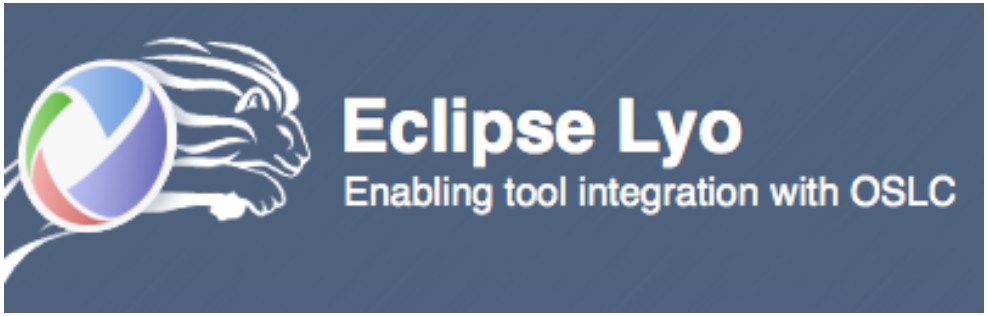
Technology Considerations

- Decide what libraries or SDKs to use
  - OSLC friendly technologies and open source
    - RDF – http://w3.org/TR/rdf-primer
      - Java libraries include: Apache Jena, OpenRDF Sesame
      - See http://www.w3.org/2001/sw/wiki/Category:Programming_Environment for a list of libraries for other languages
    - RESTful web services
      - HTTP client libraries (Apache, .NET)
      - JAX-RS (Apache Wink or Oracle Jersey) libraries useful, but not required
      - JSON for web clients
    - Eclipse Lyo

Agenda

- **Tool integrations are hard**
- **Linked Data approaches and OSLC**
- **Eclipse Lyo**
- **Lyo Content and Plans**
- **Participating**

Eclipse Lyo



- Evolution of OSLC tool repositories on SourceForge and some private to OSLC community participants
- Project approved by Eclipse PMC in July 2011 with the goal of providing an SDK to enable adoption of OSLC specifications, including
  - Code libraries
  - Reference implementations of specifications
  - Test suites and test reporting
  - Samples, tutorials and documentation
  - **NOT** a plugin for the Eclipse IDE, **NOT** related to OSGI – although Lyo artifacts could be used in Eclipse plugins.
- Eclipse chosen as the home for the project for its mature governance model and IP policies.
- Eclipse community includes tool vendors and tool interop projects. Other participants in the OSLC community are also active in Eclipse projects related to OSLC.
- Project content is dual-licensed under the Eclipse Public License and the Eclipse Distribution License

Agenda

- Tool integrations are hard
- Linked Data approaches and OSLC
- Eclipse Lyo
- Lyo Content and Plans
- Participating

Lyo Content

- Initial code contributions 4Q 2011
  - Reference Implementations for OSLC (RIOs) for the Change, Requirement and Architecture Management specifications.
    - Provides samples of implementations
    - Enable prototyping and experimentation during spec development
  - OSLC Test Suite and Reports
    - Measure implementation compliance against Core and domain specifications
    - Improve implementation quality by finding bugs
  - Samples
    - Change Management adapter for Bugzilla
    - Change Management adapter for Microsoft Excel

Plans for new content in 2012

- Code libraries
  - OSLC4J SDK for Java with example implementations
  - Other technologies (.NET, PHP, JavaScript, Python, Perl)
    - Code contributions welcome
- Test Suites
  - Increase domain coverage
  - Increase specification coverage within domains
  - Move the tests towards a true compliance suite
- Samples
  - OAuth consumer and provider samples
  - Sample integrations with lifecycle tools (community contributions)
  - OSLC Workshop/Tutorial code

Where is Lyo at today?

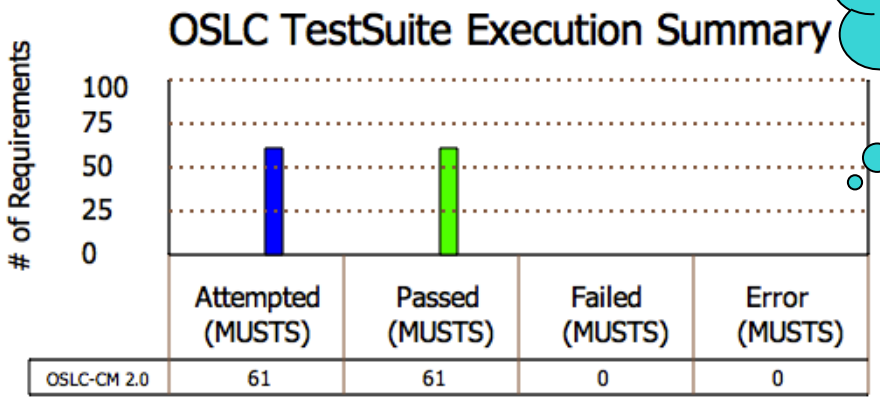
- See project plans at: http://wiki.eclipse.org/Lyo/ProjectPlans
- M1/M2 Milestones
  - ✔ M1 (4Q2011)
    - Focus was on test suite enhancements and reporting
  - ✔ M2 (1Q2012)
    - OSLC4J SDK for Java – initial contribution of source
    - Example implementations based on OSLC4J
- Working towards a release later this year
- What's not there yet
  - As of today, need to build the OSLC4J SDK and test suite from source
  - OSLC4J consumable JARs are close – working on publishing to Eclipse's Maven repo.
  - Getting started: http://wiki.eclipse.org/Lyo

OSLC Test Suites

- OSLC Core and Domain coverage
  - Initial focus testable MUST requirements in the specifications
  - Compliance assessor and adoption accelerator
  - Code contributions welcome
- Test suite as an OSLC consumer
  - Example of how to interact with an OSLC provider
  - Does not yet include OAuth interactions
  - Patterns for GET/POST/PUT
  - Patterns for validating and handling RDF using Jena
- Getting started with the tests and reports: http://wiki.eclipse.org/Lyo/BuildTestSuite

OSLC Test Suite
As an OSLC Compliance Assessor

Coverage Metrics



- Attempted = Pass + Fail + Error. # of Tests Executed for a Specification Type
- Pass: # of Test(s) achieving the respective test design's expected result
- Fail: # of Test(s) deviating from the respective test design's expected result.
- Error: # of Inconclusive Results. Test executions encountering error due to poor test design, faulty environment or invalid configuration. Test results could not be assessed.

Distinct Compliance Levels

OSLC compliance

Report Date	OSLC Domain	Version	OSLC Service Provider	Compliance Level	Test Coverage Statement	Test Development Statement
Thu Nov 17 00:04:44 EST 2011	OSLC-CM	2.0	JIRA	Level 1 Compliance	**54.2%** (58/107) 58 of the 107 OSLC-CM 2.0 MUST requirements are currently testable via the Lyo OSLC testsuite.	**87.9%** (51/58) 51 of the 58 testable MUST requirements currently have JUnit test case coverage within the Lyo OSLC testsuite

Non-Compliant: One or more attempted tests covering a MUST requirement has encountered a failure or error
Level 1 Compliance: All Attempted Tests covering a MUST requirement are Passing and free of failure or error
Level 2 Compliance: All Attempted Tests covering a MUST and SHOULD requirement are Passing and free of failure or error
Level 3 Compliance: All Attempted Tests covering a MUST, SHOULD and MAY requirement are Passing and free of failure or error
Note: This testsuite will continue to evolve and expand. Requirements may have one or more associated test(s) for coverage to address positive and negative input behaviors.

OSLC4J Overview

- Getting started: http://wiki.eclipse.org/Lyo/BuildingOSLC4J
- Java SDK for OSLC provider and consumer implementations
  - Based on OSLC related Java annotations and JAX-RS for REST services
  - Includes a Change Management reference implementation and other samples
  - Jena and Apache Wink provide RDF, JSON and JAX-RS capabilities out the box
  - Implementers can choose to use OSLC4J with other “providers” such as OpenRDF and Jersey
- What OSLC4J and JAX-RS handle for you
  - Resource shapes and service provider documents
  - Marshaling of Java objects to RDF (XML or JSON)
  - Un-marshaling of RDF (XML or JSON) to Java objects
  - Mapping REST service calls (GET, POST, PUT, DELETE) to Java methods
- What OSLC4J and JAX-RS do not handle for you
  - Persistence of your OSLC resources
  - Business logic for mapping Java objects to native resources
  - Automatic support for OSLC query syntax (working on some helpers)

Implementing providers with OSLC4J

- Some basics
  - Set up your JAX-RS Application (extend OslcWink if using the Apache Wink provider)
  - Create a Java class for your OSLC resource.
    - Annotate the class to indicate the resource type the class represents
    - Annotate the getters with OSLC resource shape info
  - Create a Java class for your JAX-RS services
    - Indicate GET/POST/PUT/DELETE methods with JAX-RS annotations
    - Annotate POST methods with OSLC creation factory info
    - Implement the “business logic” for CRUD operations on resources
      - Querying/retrieving bug requests from our bug tracker
      - Creating/updating bug requests in our bug tracker
  - Implement OSLC dialogs (or enhance existing dialogs)
    - Creation, selection, compact representation dialogs.

OSLC consumers using OSLC4J

- OSLC REST Client based on Apache Wink
  - Included in the OSLC4JWink project
  - Methods to Get/Create/Update/Delete OSLC Resources
  - Handles Java/RDF marshaling
  - Requires same POJO resource definitions used by the service provider
  - See the OSLC4J Junit Tests for examples of using the client

OSLC4J Samples

- Change Management provider
- Stock Quote provider
  - example of OSLC app not based on a current domain specification
- OSLC4J Registry application
  - Implementation of a standalone OSLC catalog registry server
  - REST API to allow service providers to register/de-register themselves
  - Used by the OSLC4J CM and StockQuote samples

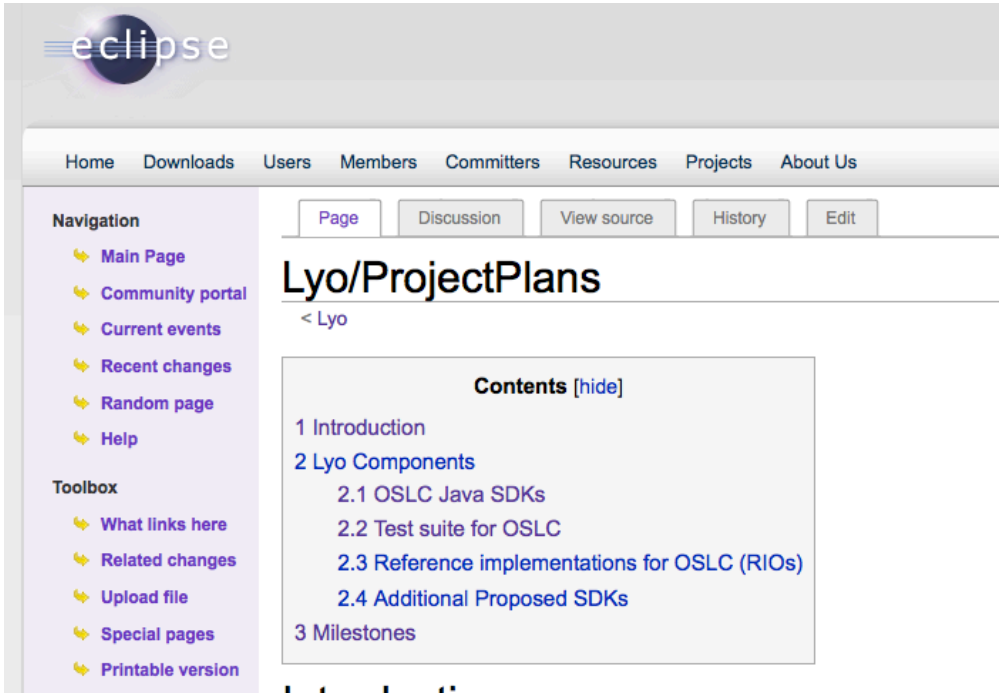
Other samples and reference implementation

- Bugzilla Adapter
  - Full CM service provider adapter for Bugzilla
  - Includes OAuth and Rational Jazz rootservices support
  - Good example for connecting to Rational Jazz OSLC providers
- OAuth sample web app
  - Sample code for handling OAuth tokens and authentication
- Reference implementations for OSLC (RIOs)
  - Example OSLC providers built using OpenRDF and a traditional servlet approach
  - Change, Architecture and Requirement Managemt providers
  - Include simple delegated UIs
  - Includes OSLC query support (oslc.where)
- Microsoft Excel change management adapter
  - Example of exposing rows of an Microsoft Excel sheet as change requests
  - Map columns to OSLC attributes

Agenda

- **Tool integrations are hard**
- **Linked Data approaches and OSLC**
- **Eclipse Lyo**
- **Content and Plans**
- **Participating**

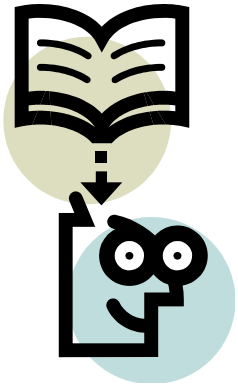
Participating in Lyo



- Participating in Lyo
  - Lyo is currently source only – goal to have builds of consumable jars starting in February
  - Looking for developers interested in promoting OSLC adoption by developing SDKs, reference implementations, compliance tests and examples
  - Visit http://wiki.eclipse.org/Lyo to get more info, see milestone plans, etc
  - Open Bugzilla requests at: https://bugs.eclipse.org/bugs/enter_bug.cgi?product=Lyo
  - Subscribe to the lyo-dev@eclipse.org mailing list and introduce yourself.

Resources

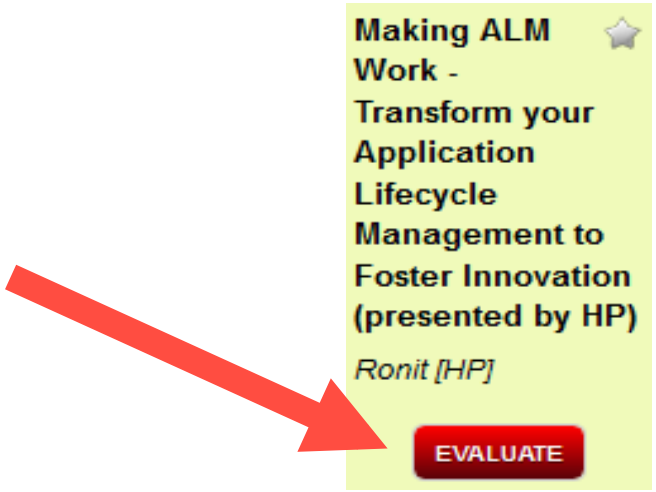
- OSLC Web Site
  - http://open-services.net
- OSLC Primer
  - http://open-services.net/primer
- OSLC Tutorial
  - http://open-services.net/tutorial
- Open source - Eclipse Lyo Project
  - http://eclipse.org/lyo
  - http://wiki.eclipse.org/Lyo



Give Feedback on the Sessions

1. **Sign In:** www.eclipsecon.org

2. **Select Session Evaluate**



3. **Vote** +1 0 -1

Questions?

- Questions?

Backup

- OSLC4J project mapping - repo is git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.core.git
  - OSLC4J – Core project
  - OSLC4JJenaProvider – RDF and RDF/XML provider based on Jena
  - OSLC4JJSON4JProvider – Apache Wink JSON4J provider
  - OSLC4JWink – Apache Wink JAX-RS provider + OSLC client
  - OSLC4JRegistry – Sample catalog registry application.
  - OSLC4JTest – Test provider
  - OSLC4JTestTest – Junit Tests for Test provider
  - OSLC4JStockQuote – Stock Quote provider
  - OSLC4JStockQuoteCommon – Stock Quote common classes
  - OSLC4JStockQuoteTest – Junit Tests for StockQuote provider
  - OSLC4JCoreRelEng- release engineering files (master pom.xml for Maven)

- OSLC4J Change Mgmt sample – repo is : git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.rio.git
  - OSLC4JChangeManagement – CM provider
  - OSLC4JChangeManagementCommon – CM common classes
  - OSLC4JChangeManagementTest – Junit tests for CM provider