Integration & Interoperability: Lessons Learned from ALM and OSLC

Sky Matthews
CTO, IBM Rational Systems Products
The application lifecycle requires cross-discipline collaboration and cooperation

*The challenge in enabling it*

- Traditionally, each tool came with its own
  - user interface
  - logic and workflow, process
  - storage

- Resulting in...
  - brittle integrations
    - proprietary API’s
    - version incompatibility
  - administrative complexity
  - teams working in silos

- Point to point solutions have a higher business cost for users, consultants, vendors
Count the arrowheads

The arrowheads represent the number of custom product adapters that would be required to support all of the possible point-to-point integrations.

<table>
<thead>
<tr>
<th>Tools</th>
<th>Adapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>90</td>
</tr>
<tr>
<td>N</td>
<td>N (N-1)</td>
</tr>
<tr>
<td></td>
<td>Approx N^2</td>
</tr>
</tbody>
</table>
What have we tried to do in the past ……………?

- “We’ll keep the tools updated”
- “One WILL be master” (at least here !)
- We will copy the data from Tool A to Tool B
- We will add new function to tool A….and then Tool B
- Add another tool ………
- We will turn off Tool C (at least try…….)
- Start again, and again and…………
- Each unit, group, product line, team, site decides on their own
- Do it in PLM ? SAP ? Or Sharepoint ? Or Bugzilla !
- Enterprise back bone and integration middleware
- ?
- All of the above ……………………did it succeed …?
- ........................................................................and for how long ?
Benefits and Issues with Standardization

- Standards are GOOD
  - Ensure market forces balance selfish motives of vendors

- Almost all tool standards are focused on data exchange
  - Standards help ensure that customers aren’t locked in to one tool

- Standards are not typically concerned with workflows across tools in a lifecycle
  - Standards are focused on one particular tool domain

- Standardization is a cumbersome, lengthy and contentious process
  - The result is not always the most

- Standards typically take many years to complete
  - How many years from inception to UML interchange standard? (hint: >10 years)
What can we learn from software architecture principles?

- A scalable and robust software design adheres to two core principles which allow an architecture to be
  - Flexible: able to address new requirements over time without breakage
  - Adaptable: and reusability

- Those principles are:
  - Loose Coupling – well-defined interfaces that hide unnecessary implementation details
  - High Cohesion – module should encapsulate all information related to a particular domain

From work by David Parnas and others in 1970s/80s
The software development process has also evolved some key principles

- Software development (ALM) has evolved from waterfall process to lean/agile
  - Be more flexible
  - Adapt to changing requirements/priorities

- Agile development focused on break down of system project in to set of sprints that work of prioritized backlog of features/defects

- At the end of each sprint, do a retrospective to learn and adjust the process as needed
Towards a modular integration architecture

- Our experience is that a tool environment is an integrated system of modular components.

- So, how to design an integration architecture that meets common software design principles and best practices?
  - Must be modular
  - Must be flexible
  - Must be adaptable over time
The Internet – an inspiration for an architecture

- Amazingly scalable
- Integrates information on a massive scale
- Highly and rapidly extensible
- Collaboration on unprecedented scale
- World-wide information visibility
So what if…

- All data are resources with URLs
  - Requirement, test case, piece of UML model (like a use case), defect
- Resources have representations
- Representations specified independently of tools as standards
- Links are embedded URLs
- Multiple tools access data
- REST (Representational State Transfer)
  - REST provides a services oriented framework for applications/tools
  - Tools that conform to the REST architecture are referred to as being RESTful
OSLC: Standardizing how tools share lifecycle data

Scenario-driven & Solution-oriented
Leading choice for strategic integration technology
Generally applicable: specs available for many domains covering ALM, DevOps, ISM, and PLM

Open Services for Lifecycle Collaboration
Lifecycle integration inspired by the web

Linked Data Platform Working Group
Based on and Shaping the Future of Internet Architecture

OSLC Member Section
Open and Independent Governance and Leadership

Eclipse Lyo
Enabling tool integration with OSLC
The Resource for OSLC Implementers

OSLC: Inspired by the web
Proven
Free to use and share
Open
Innovative

Changing the industry
OSLC promotes open loosely coupled integration using a web style architecture based upon Linked Data

- Provides
  - Exploits the increasingly popular RESTful services
  - Link and query style, information centric approach
  - Wrapping of legacy data using Resource Description Framework (RDF)
  - Path to semantic web via Web Ontology Language (OWL)
  - Open Services for Lifecycle Collaboration (OSLC) provides the domain vocabulary

- Benefits
  - Minimum and sufficient approach > Information mostly stays in its place
  - Easier to mashup information to provide process support
  - Faster to respond to changing circumstance and needs

Tim Berners-Lee’s no mug!
http://www.w3.org/DesignIssues/LinkedData
What about the process?

- Can we learn from agile community to build a better standards process?
- Focus on prioritized set of objectives, not on “completeness” of standard
- Work on time-bounded milestones
- Use end-user scenarios to drive the technical requirements
- Adopt “open source”-like community
Open Services for Lifecycle Collaboration (OSLC)

Working to standardize the way software lifecycle tools share data

- Community Driven – @open-services.net
- Open specifications for numerous disciplines
  - Including ALM, PLM, and DevOps
  - Defined by scenarios – solution oriented
- Inspired by Internet architecture
- A different approach to industry-wide proliferation
- Based on W3C® Linked Data

Inspired by the web
Free to use and share
Changing the industry

GET INVOLVED AND CONTRIBUTE!
Open Services for Lifecycle Collaboration (also known as OSLC or Open Services) is a community effort to help software delivery teams by making it easier to use lifecycle tools in combination. The OSLC community is creating open, public descriptions of resources and interfaces for sharing the things that software delivery teams rely on, like change requests, test cases, defects, requirements and user stories.

By agreeing on common specifications for lifecycle resources and the services to access them, we can eliminate traditional barriers between tools and open the door to new forms of collaboration. OSLC can bring value to software delivery teams and tool providers alike, from the most Agile to the most ceremonial of projects, and for commercially-licensed, open source, and internally developed tools.

See a snapshot of what’s happening with OSLC today.

With OSLC's open and scenario-based approach, businesses benefit from the ability to tie disparate tools together. This collaborative approach gives our consultants the flexibility to make lifecycle tool choices based on specific client project demands.

Randy Vogel, Accenture
**OSLC Specification Highlights**

### New Workgroups
- OSLC Performance Monitoring
- OSLC Reconciliation
- OSLC Configuration Management

### New Contributions
- **OSLC Tracked Resource Set** (defines what a service must provide to enable indexing)
- **OSLC Partial Update** (guidance on using HTTP PATCH)

### Finalized Specifications
- OSLC Asset Management V2
- OSLC Automation V2
- OSLC Reconciliation V2
- OSLC Performance Monitoring V2

### Ongoing Specification Work
- OSLC Core and many domain specs working towards V3
- W3C Linked Data Platform 2nd Editor’s Draft published
Rich hovers – Encapsulation
Delegation of operation details

Test Management tool

Defect tracking tool – Delegated UI

Related defects

Creates link from test result to defect
Real world example using the OSLC Change Management specification – Change Request flow highlighted

1. Control system Integration Test Failure- opens Problem Report
2. Program Management Triages PR and Creates Change Request for analysis
3. Program Management Creates Change Notice Assigns To Engineering Lead to build software fix
4. Integration & Verification re-tests the Control System and the release is ready

Trouble report system
Enterprise PLM
Rational Team Concert

OSLC CM Spec is used to associate and pre-populate the Change Request from a Trouble Report
OSLC CM Spec is used to create the SW discipline Change Notice in RTC from the Product Change Request in PLM
OSLC Change Management example
Logical System Context outline

PLM Tool

Existing Service end point
Extended Service end point

Proprietary PLM Web Service Boundary

API Boundary

Existing Service end point
Extended Service end point

Software Change Management Tool
The Vision for OSLC

OASIS OSLC Member Section

Steering Committee
Subcommittees
Technical Committees
Approved
Approved
Approved Standard

Use what you need, share what you can.

Eclipse Lyo

Libraries
Test Suites
Reference Implementations
Samples and Examples

open-services.net

News
Implementer’s Expo
User Groups
Forums
Resources and Tutorials

W3C LDP WG

Innovation for a smarter planet
<table>
<thead>
<tr>
<th>Accenture</th>
<th>Bank of America</th>
<th>Boeing</th>
<th>Creative Intellect</th>
<th>EADS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse Foundation</td>
<td>Ericsson</td>
<td>Fujitsu</td>
<td>IBM</td>
<td>JPMorgan Chase</td>
</tr>
<tr>
<td>Kovair Software</td>
<td>KTH Royal Institute of Technology</td>
<td>Mentor Graphics</td>
<td>National Instruments</td>
<td>Persistent</td>
</tr>
<tr>
<td>Red Hat</td>
<td>Siemens</td>
<td>Sodius</td>
<td>Software AG</td>
<td>Tasktop</td>
</tr>
</tbody>
</table>

**Universidad Politécnica de Madrid**

**WSO2**
Lessons Learned

- More agile approach is good but must be balanced by adoption considerations
- Adoption in embedded systems has been very good – definitely satisfies a need
- Linked Data approach is very positive but technology still maturing
- Need to evolve ways to address overlap / intersection with other non-linked data standards, and for standards organizations to better cooperate
- Linked data is a **fundamentally different** approach to “System” architecture
  - Need to deeply understand and see the possibilities
    - A graph of distributed objects
    - An easily extensible graph – new properties can be “asserted” on existing artifacts
    - Self-discoverable and extensible type system