

# Agility @ Scale - Enabling Collaboration across Silos

*Erich Gamma*  
*IBM distinguished engineer*  
*IBM rational zurich research lab*

# First Assignment: Eclipse

- A tools integration platform
  - Scalable
  - Easy to extend
  - Enable a tools ecosystem
- Goal: Built to last

# Inspiration: how buildings last

- **Stewart Brand: how buildings learn  
– what happens after they're built**

stuff: furniture

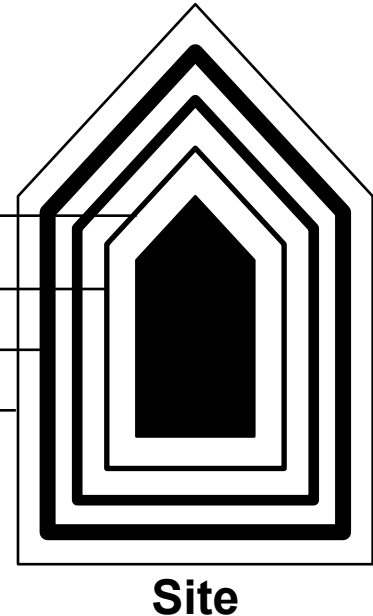
services: electrical, plumbing (7-15y)

structure: foundation, load bearing walls (30-300y)

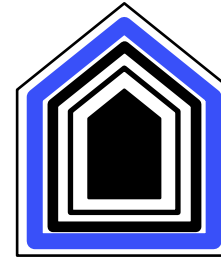
site: geographical setting (forever)

- **layers:**

- evolve at different rates during the life of a building
- shear against each other as they change at different rates
- an adaptive building must allow **slippage**
- a building that lasts is adaptive and can change over time
- lasts for generations without total rebuilding



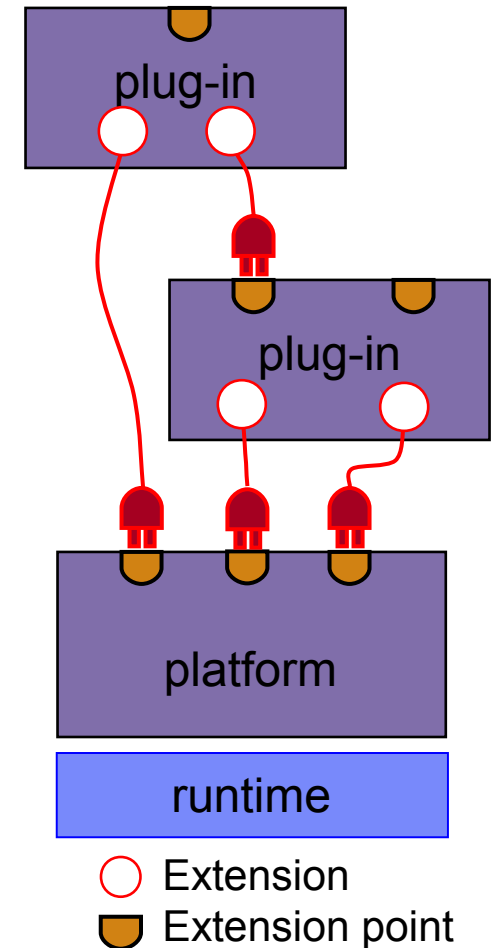
# structure foundation



- the eclipse plug-in architecture
- everything is a plug-in
  - simple and consistent

# eclipse plug-in architecture

- **plug-in == component**
  - set of contributions
  - smallest unit of Eclipse function
  - details spelled out in plug-in manifest
- **extension point** – named entity for collecting contributions
- **extension** – a contribution
  - Example: a specific spam filter tool
- **runtime** – controls and manages contributions



# scalability

## user visible appearance

<action

toolbarPath="search"

icon="icons/opentype.gif"

toolTip="Open Type"

class="org.eclipse.jdt.OpenTypeAction"/>

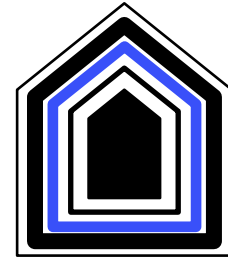
Declarative  
Definition  
(manifest)

lazily instantiated using  
reflection

Procedural  
Implementation  
(Java JAR)

org.eclipse.jdt.OpenTypeAction.class

contribution  
implementation



## services plumbing: APIs

- Plug-in dependencies through APIs
- define APIs for stability
  - binary compatibility is highest priority

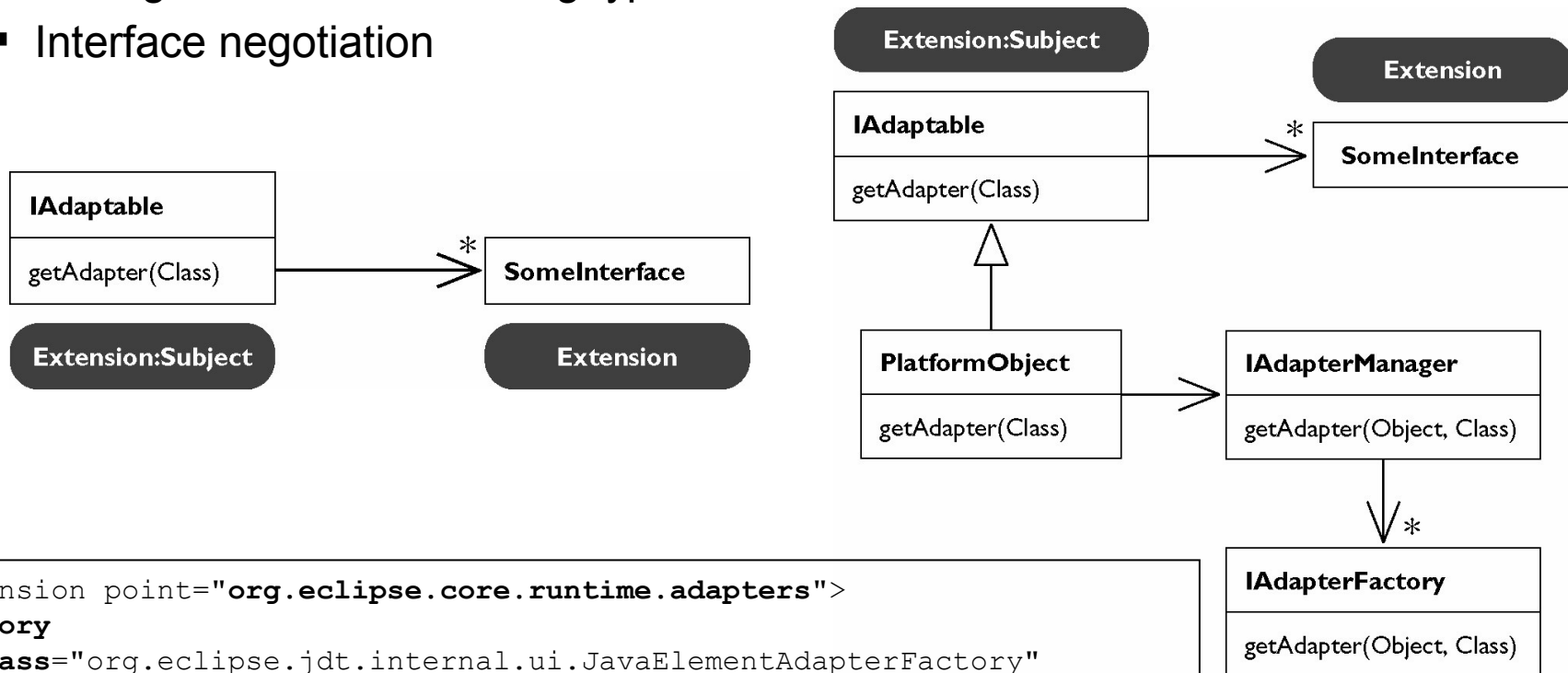
# APIs first

- APIs don't just happen; we need to design them
- specifications with precisely defined behavior
  - what you can assume (and what you cannot)
  - it works  $\neq$  API compliant
  - documented classes  $\neq$  API
- must have at least one client involved, preferably more



# extension interfaces: IAdaptable

- adding interfaces to existing types
- Interface negotiation



```

<extension point="org.eclipse.core.runtime.adapters">
<factory
  class="org.eclipse.jdt.internal.ui.JavaElementAdapterFactory"
  adaptableType="org.eclipse.jdt.core.IJavaElement">
  <adapter type="org.eclipse.ui.IPersistableElement"/>
  ...
</factory>

```

## I\*2 extension interfaces

- add new methods in extending API interface with extension interfaces
  - avoids breaking existing implementors of an interface

```
public interface IActionDelegate { ... } // original interface
```

```
public interface IActionDelegate2 extends IActionDelegate {  
    void dispose();  
}
```

```
if (d instanceof IActionDelegate2) {  
    IActionDelegate2 d2 = (IActionDelegate2) d;  
    d2.dispose(); // call new method  
}
```

# Key Lessons

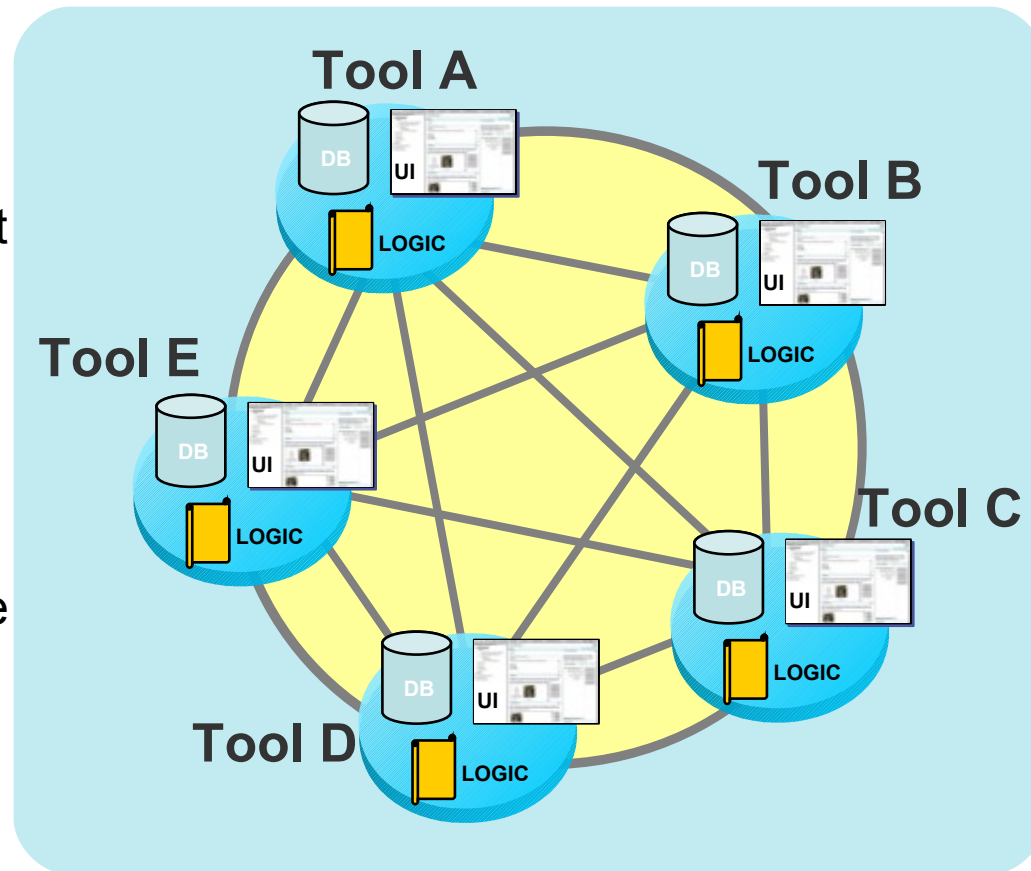
- **Modularity matters**
  - Everything is a plug-in
  - “no exceptions”

## APIs are a huge commitment

- we would rather provide less API than desired (and augment) than provide the wrong (or unnecessary) API and need to support it indefinitely
- the tyranny of stable APIs
  - API layers...
- the challenge of product developers
  - which API level does our product require and support
    - $n-1$ ,  $n-2$

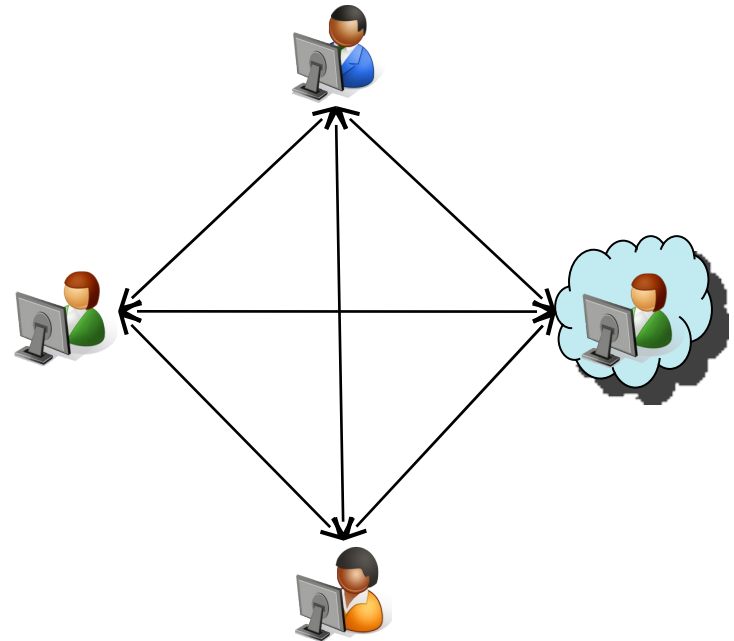
# Next assignment: A Tool Integration Platform

- Integrate many tools
  - Heterogeneous environments** that are flexible for partners and suppliers
  - Acquisitions raise expectations for product **integrations**
- Global Connectedness
  - Distributed** development, cross site product development
- Lifecycle / Agile Methods
  - Flexible** tools and process

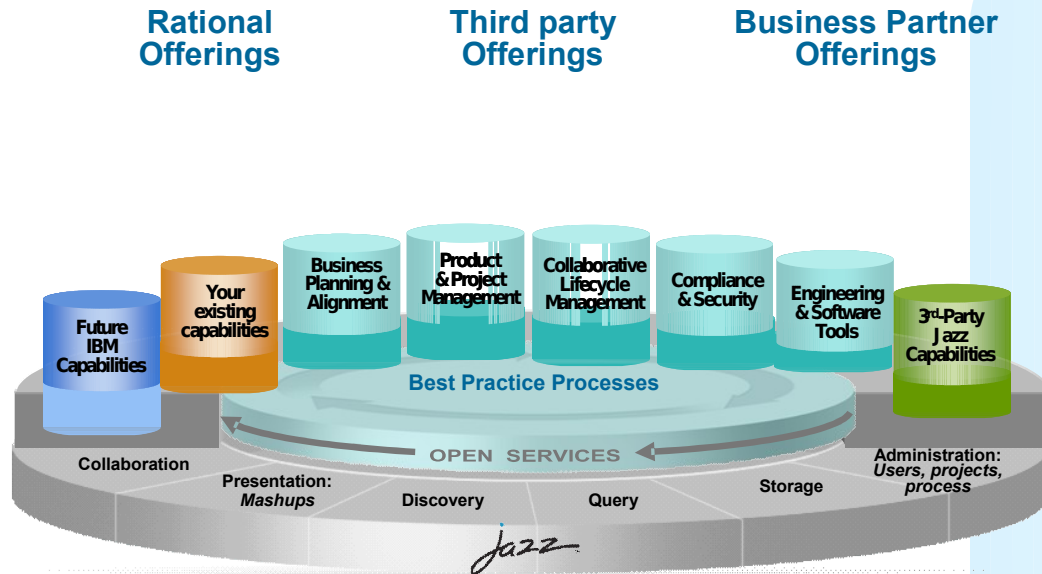


# Traditional Tool Integration. Ouch.

- N2 possible point-to-point connections
  - Limited coverage
- Closed APIs
  - Vendor lock-in
- Tight Coupling
  - Dependence on internal structures
- Lockstep upgrades
  - Version incompatibilities
- Need something better...



# Jazz is a platform for transforming software delivery



## ***Jazz is...***

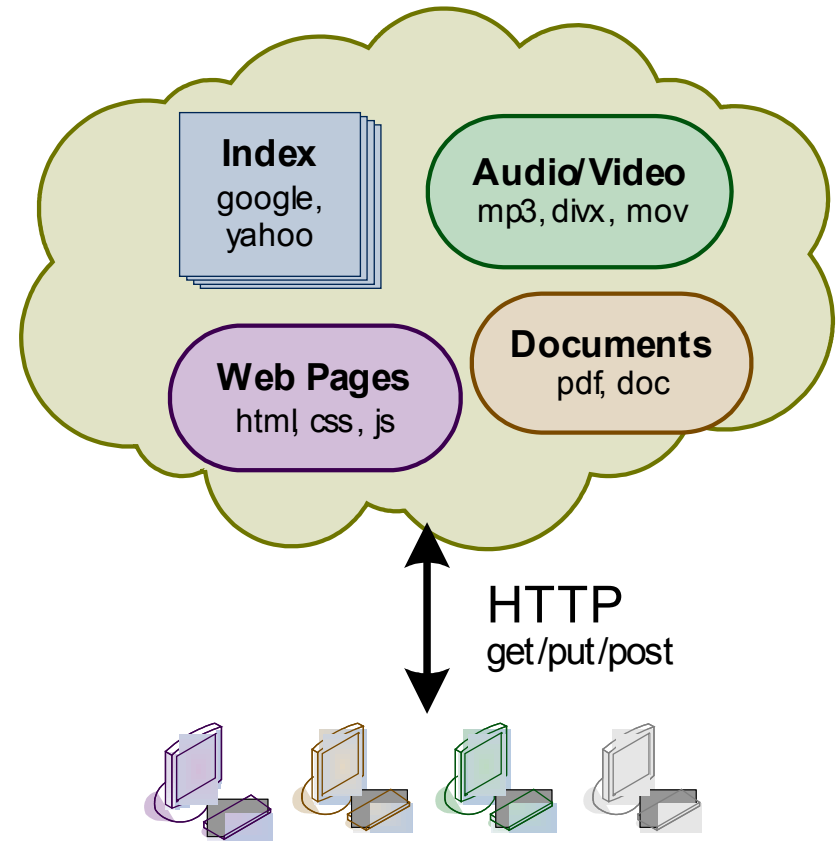
- Our vision of the future of systems and software delivery
- A scalable, extensible team collaboration platform
- An integration architecture enabling mashups and non-Jazz products to participate
- A community at Jazz.net where Jazz products are built

**Jazz is a platform for *transforming how people work together* to deliver greater value and performance from their software investments.**



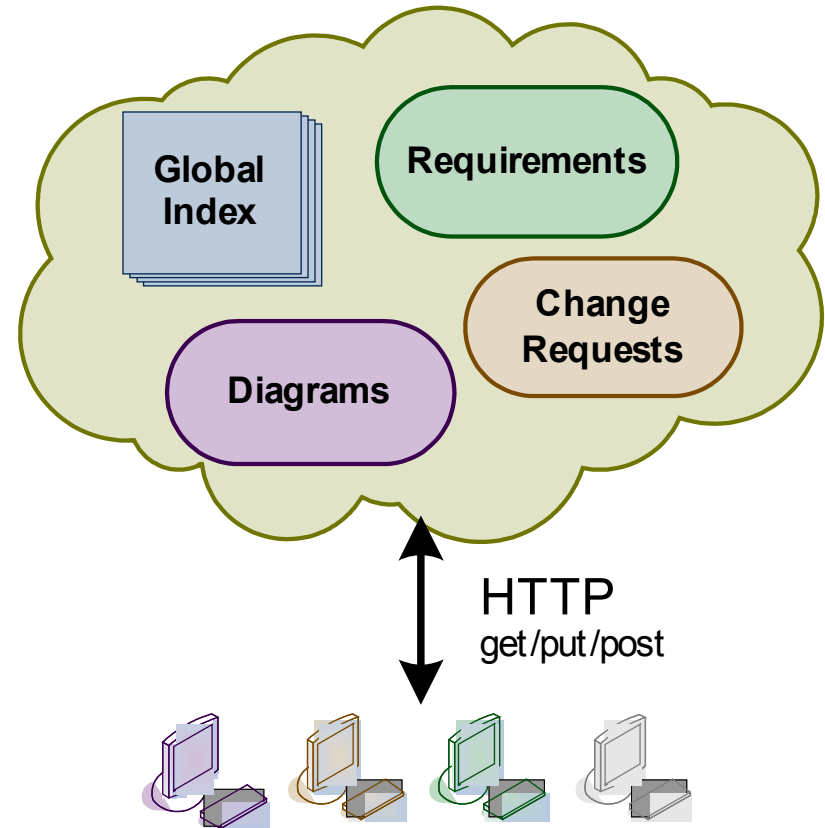
# Inspiration: the Internet

- Amazingly scalable
- Integrates information on a massive scale
- Infinitely extensible
- Collaboration on unprecedented scale
- World-wide information visibility



## How does this work?

- All data are resources with URLs
- Resources have representations
- Representations are specified independently of tools
- Links are embedded URLs
- Tools (multiple) access data through HTTP get/put/post/delete

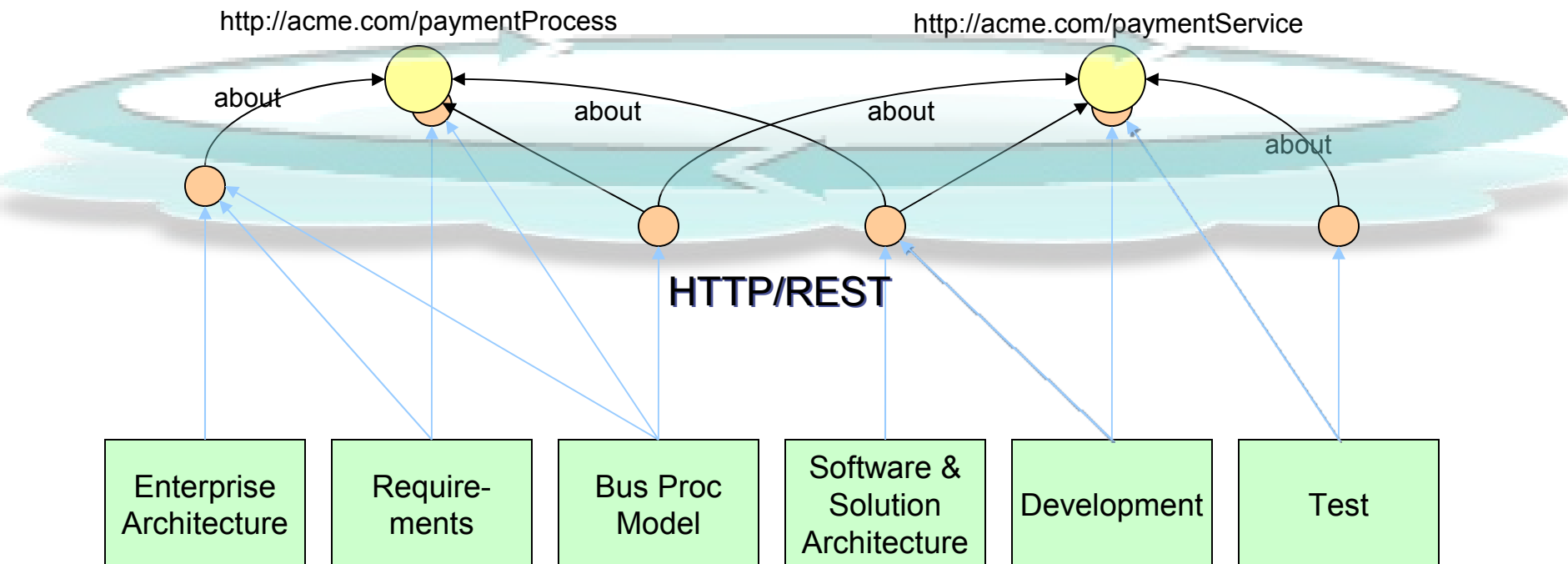




## Jazz architectural principles

- Jazz separates the implementation of tools from the definition of and access to the data
  - Data semantics do not rely on "secret knowledge" embedded in product code.
- Jazz can access and integrate data where it resides
  - Jazz does not need to import and export data between tools or repositories
- Jazz assumes an open, flexible, distributed data model.
  - Jazz does not assume that there is a single data model that is centrally managed, nor that each tool needs to understand the entire data model in order to participate.
- Jazz allows tools to be implemented in any Internet-aware programming language or platform.
  - Jazz does not impose an implementation framework tied to a particular language or technology platform
  - Provide optional toolkits to aid in tool implementation

# Data Integration – the new way – “www linked data”



# Architectural Rules

- R1: Independent upgrade
- R2: Rich Integration
- R3: Limited application coupling
- R4: Open world

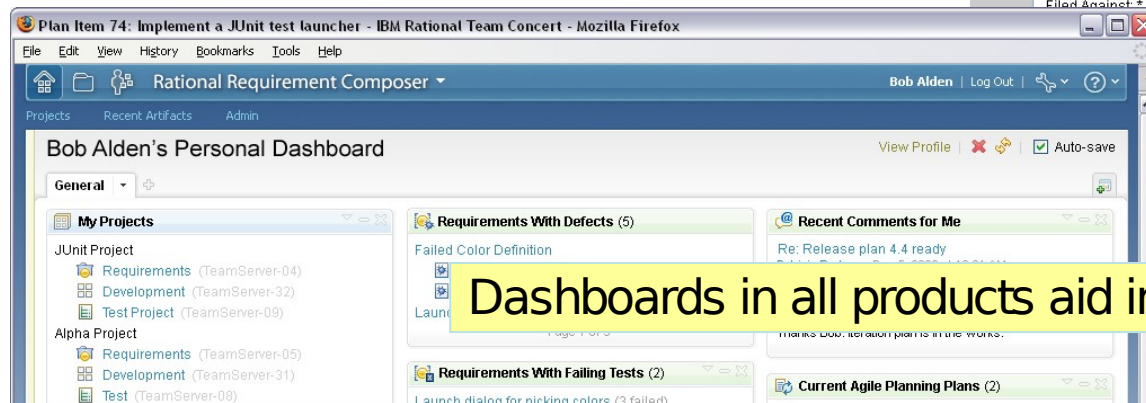
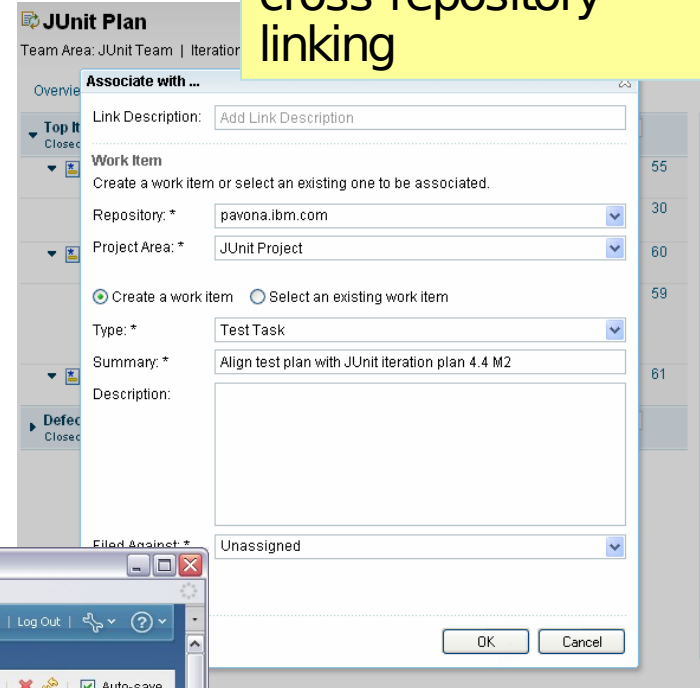
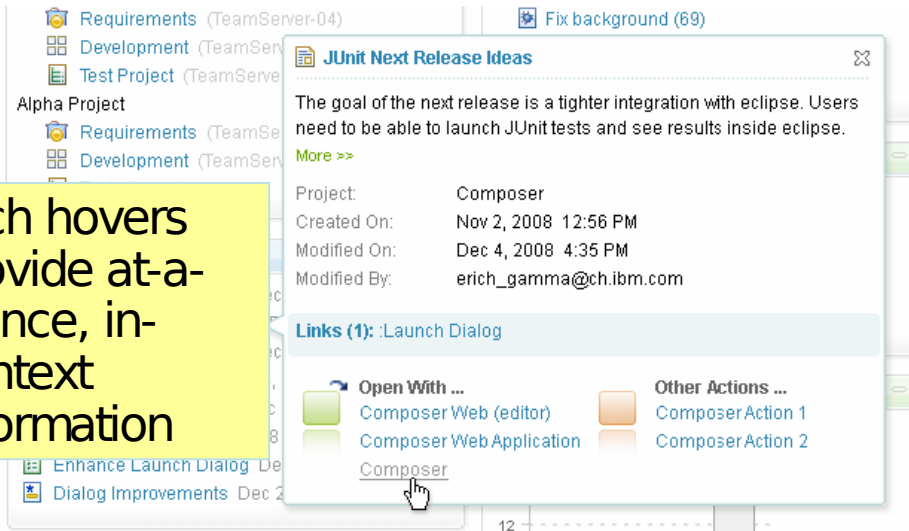
# R1: Independent upgrade

- Customers must be able to upgrade their products **one at a time** in the order of their choice
  - product teams must commit to managing their dependencies so that this will always be the case
- Easy to say; easy to understand; highly motivational
- Smooth upgrading is a corollary
  - customers must not feel that they are losing/breaking their applications (or application data) as a side effect of upgrading any of their products.
  - Client - server compatibility issues are included here.

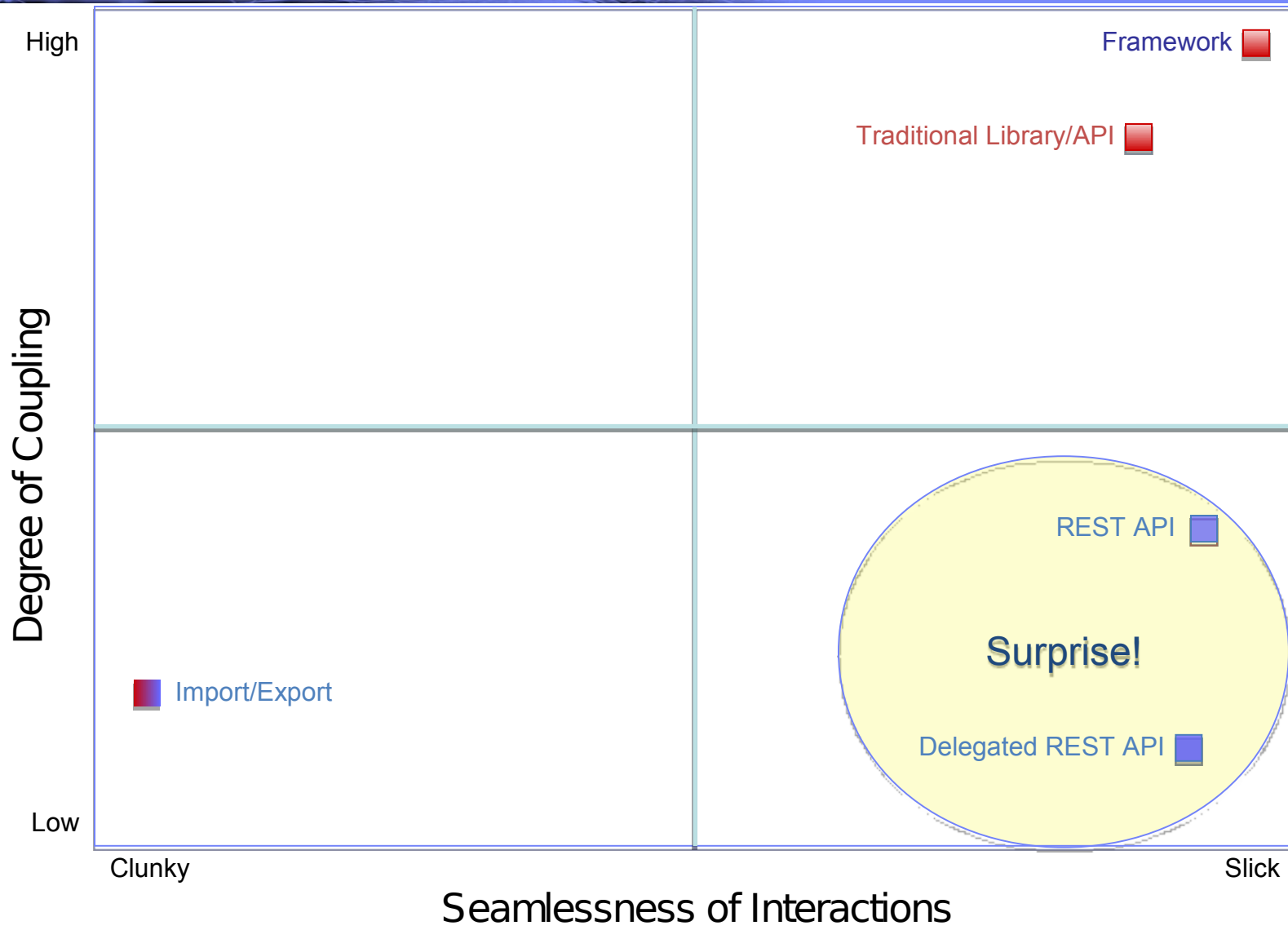
# R2: Rich integration (with loose coupling!)

Link Dialogs enable cross-repository linking

Rich hovers provide at-a-glance, in-context information



Dashboards in all products aid in transparency



## R3: Limited application coupling

- Applications will depend on few other applications.
- If we're not careful, we get caught in the dependency web
- Yet, applications need to interact

## R4: open world

- New products can be integrated after the fact, and their capabilities are reflected in the user and programmatic interfaces
- Don't assume you know everything up front





# Open Services for Lifecycle Collaboration

*An initiative aimed at simplifying tool integration across the software delivery lifecycle*

## Open Services for Lifecycle Collaboration

### Barriers to sharing resources and assets across the software lifecycle

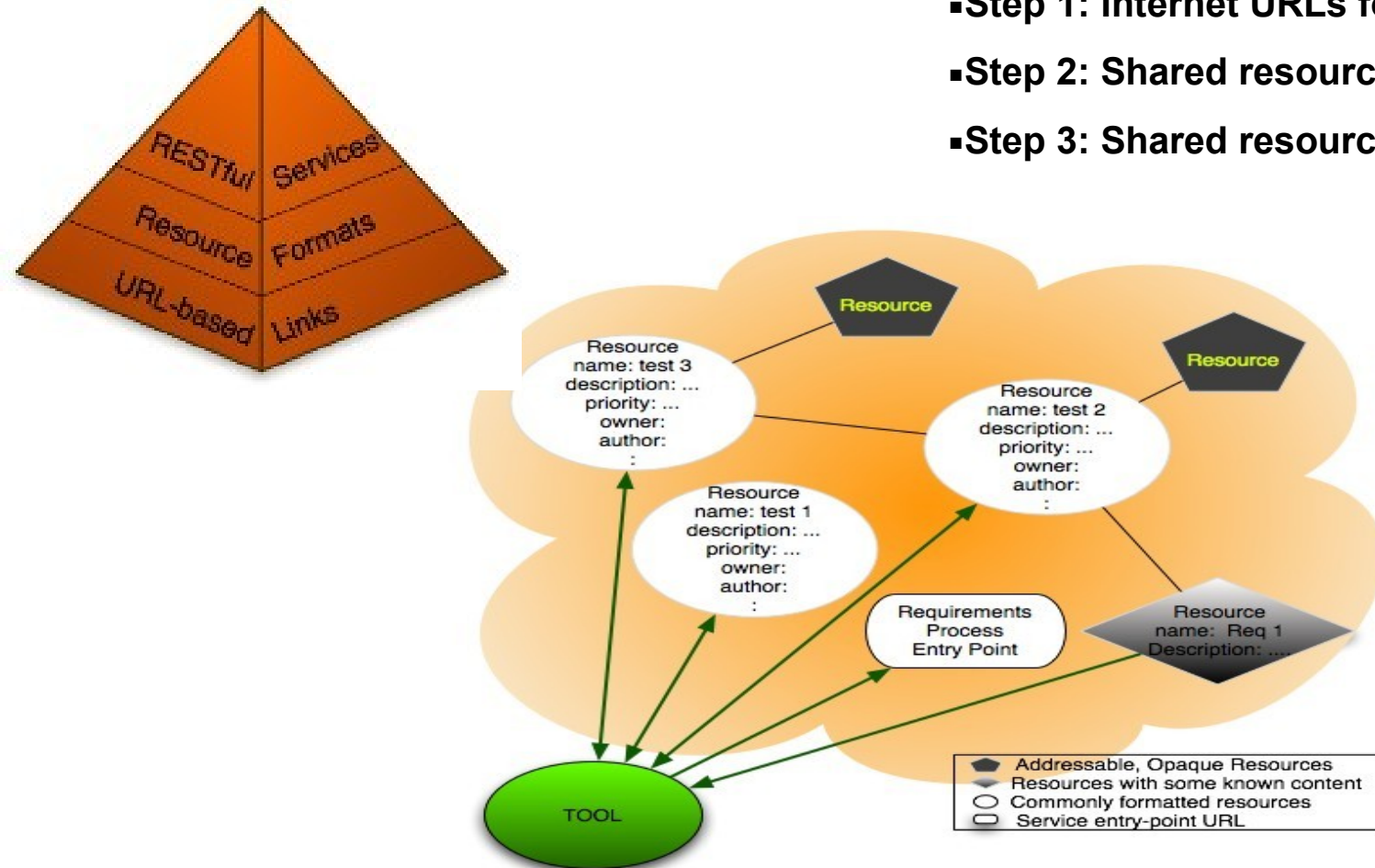
- ▶ Multiple vendors, open source projects, and in-house tools
- ▶ Private vocabularies, formats and stores
- ▶ Inextricable entanglement of tools with their data

- ▶ Specifications for sharing lifecycle resources
- ▶ Inspired by Internet architecture
  - Loosely coupled integration with “just enough” standardization
  - Common resource formats and services
- ▶ A different approach to industry-wide proliferation

# Open Services for Lifecycle Collaboration

## *Putting the approach into practice*

- Step 1: Internet URLs for resources
- Step 2: Shared resource formats
- Step 3: Shared resource services



# Community: open-services.net

- Started in 2008
- Open community contribution
- Scenario driven...a minimalist approach
- Divided into focus areas
  - Change Management
  - Quality Management
  - Estimation & Measurement,
  - Requirements Management, ...
- Solving integration in the open

The screenshot shows a web browser window titled "WebHome - Main - TWiki - Mozilla Firefox: IBM Edition". The address bar shows the URL "http://open-services.net/bin/view/Main/WebHome". The page content includes a sidebar with navigation links like "Main", "Log In or Register", "Main Web", and "Webs". The main content area is titled "Open Services for Lifecycle Collaboration (OSLC) Wiki" and contains text about the integration of software delivery tools. It also features a table titled "Scenarios and Topic Areas" with columns for "Scenarios" and "Topics".

Scenarios	Topics
<a href="#">Collaborative Application Lifecycle Management</a>	<a href="#">Change Management</a> - Integrations with software work item and change management repositories. <a href="#">Quality Management</a> - integrations in quality management and testing.
<a href="#">Software Project Management</a>	<a href="#">Requirements Management</a> - integrations in requirements management and requirements definition tools. <a href="#">Estimation and Measurement</a> - integrations with estimation tools and performance, project, and portfolio management.

Recent Updates:

- [Change management 1.0 specification drafted](#)
- [Estimation & Measurement topic started](#)
- [Quality management and test execution tools topic started](#)

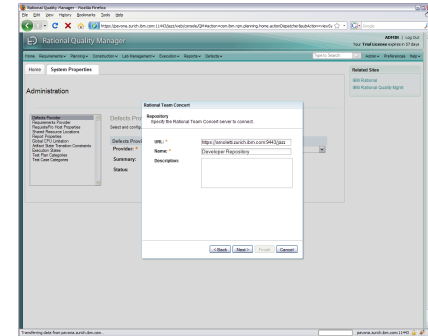
Community:

- [Mailing list](#)
- [Resource guidelines](#)
- [Common vocabulary](#)

Links:

- [Terms of use](#)
- [About this site](#)

The diagram illustrates the interaction between two systems via an OSLC interface. On the left is the **Change Mgmt System**, represented by a purple rounded rectangle containing a blue cylinder icon with a globe. On the right is the **Test Management** system, represented by a purple rounded rectangle containing a blue cylinder icon with a hexagon. Between them is a light blue vertical rectangle labeled **OSLC interface**. A horizontal line with arrows at both ends connects the two systems, with the text **POST, Query, etc change requests** centered below it.



- Spec dictates the bare minimal aspects of defect
- QM system posts “seed data”
- QM system gets URL of form; delegates back to CM system

QM system can interface with *any* OSLC-compliant change management system

# Styles of Integration

- HTTP REST API – “Rich” style
  - Web technologies – pervasive support across languages and Operating Systems
  - Resource-oriented – requires agreement on the resource representations
  - Careful resource design can avoid “closed world” assumptions
  - Exposes details of the data in resource representations
  - Can leverage client libraries, but does they are outside of the API boundary
- HTTP REST API “Delegated”/Widget Style
  - Relies on discoverable URLs for services
  - Minimizes dependencies: delegates back to application
  - Introduces out-of-bands communication between delegated form and host application

# OSLC Specification <http://open-services.net/bin/view/Main/CmSpecificationV1>

## Document

[CM RESTful Services](#)

[CM Change Request Resource Definition](#)

[CM Simple Query Syntax](#)

[CM JSON Format](#)

[CM Delegated Resource Selection and Creation](#)

[CM Service Description](#)

## Resource URIs and Methods

Resource	URI	GET	POST	PUT	DELETE	Description
Collection of Change Requests	{CR Collection URI}	Y	*	N	N	A collection of change requests
Change Request	{CR URI}	Y	N	Y	Y	An identifiable change request, by a permanent URI

\* - the collection MAY support creation on its URI, see [Create a new Change Request](#)

N - in the HTTP verb column indicates that a Service Provider MUST return a 405 Not Supported response

For a complete list of [HTTP Response Codes](#)

## URIs for working with Change Requests

The following table outlines the key items that are exposed in the Change Management Service Discovery Document. Details of each of these capabilities will follow in subsequent sections.

Purpose	Discovery Element	URL*	Section	Support
Resource Creation	<factory>	{Resource Creation URL}	<a href="#">Create a new Change Request</a>	REQUIRED
Resource Query	<simpleQuery>	{Simple Query URL}	<a href="#">Get a Collection of Change Requests</a>	REQUIRED
Resource Selection UI	<selectionDialog>	{Selection Dialog URL}	<a href="#">Resource Selection</a>	REQUIRED
Resource Creation UI	<creationDialog>	{Creation Dialog URL}	<a href="#">Resource Creation</a>	REQUIRED

# Retrieving a Defect

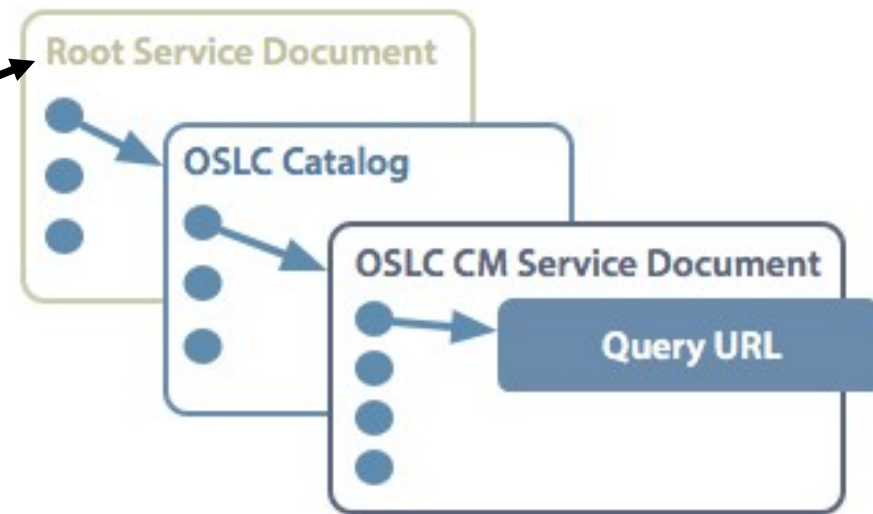
```
GET https://rtc.com:9443/jazz/resource/itemOid/com.ibm.team.workitem.WorkItem/_0J39QJu-Ed6cerS9lb5AWw
Accept: application/x-oslc-cm-change-request+xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<oslc_cm:ChangeRequest
  xmlns:rtc_cm="http://jazz.net/xmlns/prod/jazz/rtc/cm/1.0/" xmlns:oslc_disc="http://open-services.net/xmlns/disc
  xmlns:dc="http://purl.org/dc/terms/" xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:jp="http://jazz.net/xmlns/prod/jazz/presentation/1.0/" xmlns:jd="http://jazz.net/xmlns/prod/jazz/discover
  xmlns:oslc_cm="http://open-services.net/xmlns/cm/1.0/" xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:calm="http://jazz.net/xmlns/prod/jazz/calm/1.0/">
  <dc:type rdf:resource="https://rtc:9443/jazz/oslc/types/_gasc4Ju-Ed6cerS9lb5AWw/defect"/>
  <dc:identifier>9</dc:identifier>
  <dc:created>2009-09-07T14:59:06.333Z</dc:created>
  <dc:creator rdf:resource="https://rtc:9443/jazz/oslc/users/_6I8ZMJu9Ed6cerS9lb5AWw"/>
  <dc:title>My First Bug</dc:title>
  <dc:description>This is my first bug</dc:description>
  <dc:subject/>
  <dc:modified>2009-09-07T14:59:06.348Z</dc:modified>
  <oslc_cm:priority rdf:resource="https://rtc:9443/jazz/oslc/enumerations/_gasc4Ju-Ed6cerS9lb5AWw/priority/priori
  <oslc_cm:severity rdf:resource="https://rtc:9443/jazz/oslc/enumerations/_gasc4Ju-Ed6cerS9lb5AWw/severity/severi
```



# Service Discovery

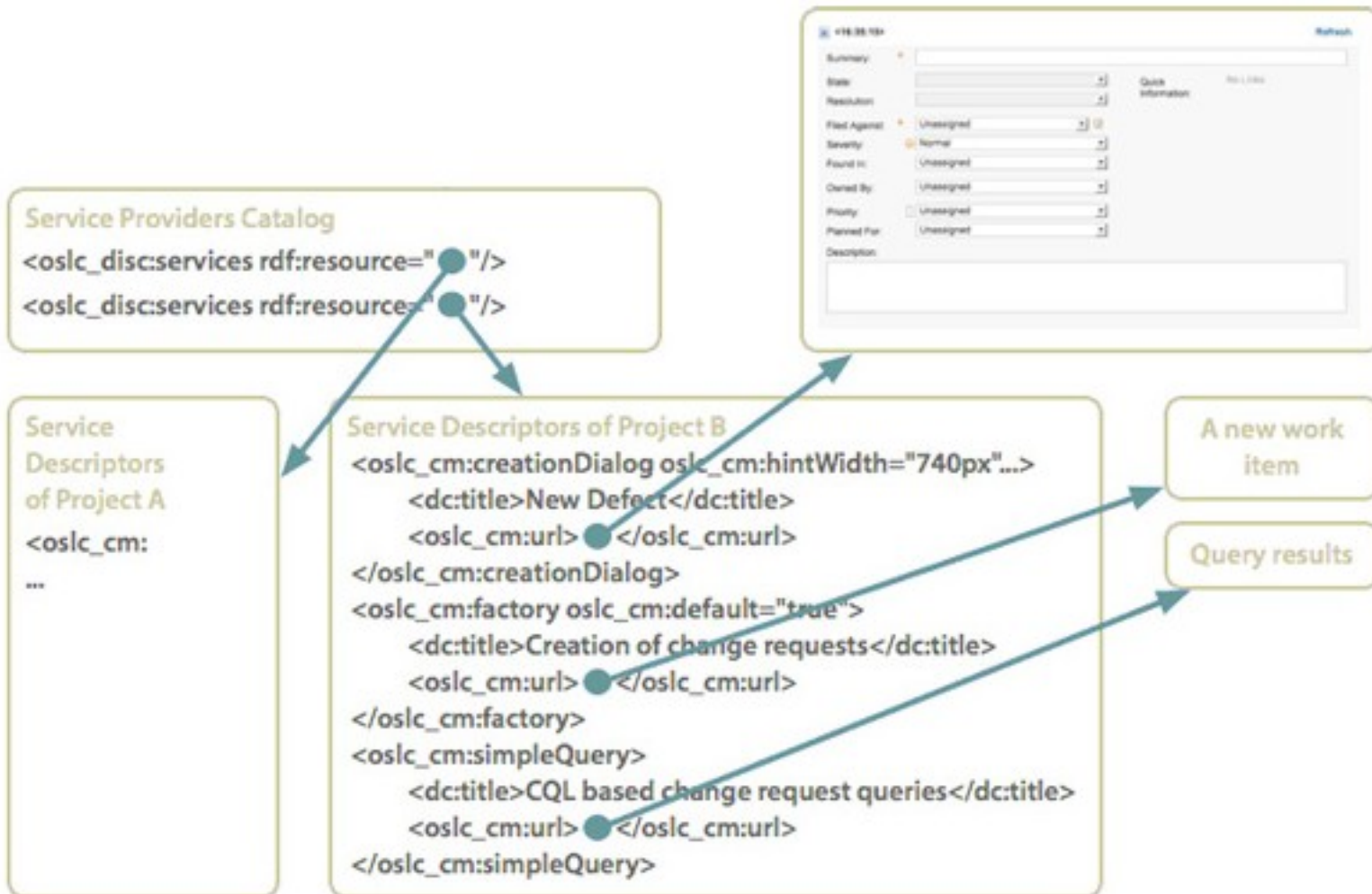
1. Discover the existence of the Change Management system itself, known URL
  - E.g. <https://rtc:9443/rtc/rootservices>
1. Discover the contexts (e.g. projects) in which change requests may exist, e.g project
2. Discover the services that are provided within that context



```
<rdf:Description rdf:about="https://rtc:9443/jazz/rootservices">
  ...
  <oslc_cm:cmServiceProviders rdf:resource="https://rtc:9443/jazz/oslc/workitems/catalog"/>
  ...
</rdf:Description>
```



# Discovering the Creation Dialog



# OSLC example: What are you testing?

Rational Quality Manager

Home Requirements Planning Construction Lab Management Execution Reports Defects Builds Type to Search Admin Preferences SmarterLiving

Home Smarter-Living Web S... Verify Mortgage Calc... \* Verify Mortgage Calc... View Test Execution Records

Verify Mortgage Calc... Verify mortgage calc...

**Manage Sections**

**Table Of Contents**

- Summary
- Test Case Design
- Formal Review
- Requirements
- Plan Items**
- Risk Assessment
- Pre-Condition
- Post-Condition
- Expected Results
- Test Scripts
- Test Execution Records
- Attachments
- Show All Sections

**Plan Items**

Test Case Overview

Originator: ADMIN Action

Change management item

Show All Items per

ID

No items found.

**Plan Item** Select the plan item

Project Area: SmarterLiving

Type: Story

Use Work Item ID or Words Contained in the Text: 2 result(s)

mortgage

Matching Work Items:

- 31: Manage mortgage provider list
- 33: Implement mortgage calculator**

OK Cancel

Team Concert (delegated UI)

Single URL (OSLC) calls RTC

Creates link on Test Case & Team Concert work-item

# OSLC example: Creating Test Cases from Requirements

The screenshot displays the IBM Rational Quality Manager (RQM) web interface. The top navigation bar includes the RQM logo, the text 'Rational Quality Manager', and a trial license expiration notice. The main content area is titled 'CD Collection Test P...' and shows a 'CD Collection Test Plan' overview. A sidebar on the left lists various project sections, with 'Requirement Collection Links' highlighted in a red box. The main panel shows a table of linked requirement collections, with one entry 'CD Collection First Version' selected. A tooltip 'Create Test Case(s) from Requirement(s)' is visible over the table. The right sidebar contains links for 'Test Plan Workitems', 'Related Test Suite(s)', and 'Related Sites'.

**Rational Quality Manager**  
Your Server Trial License expires in 58 days | **rqm** | Log Out | Type to Search

Admin ▾ Preferences RQM CD Collection Project ▾

**Dashboards** **CD Collection Test P... \***

**Manage Sections**

**Table Of Contents**

- Summary
- Business Objectives
- Test Objectives
- Formal Review
- Requirements
- Requirement Collection Links**
- Risk Assessment
- Test Schedules
- Test Estimation
- Test Environments
- Test Team
- Quality Objectives
- Entry Criteria
- Exit Criteria
- Test Cases
- Resources
- Attachments
- ☐ Show All Sections

**CD Collection Test Plan** ?  
Test Plan Overview | [View Snapshots](#)

Originator: **rqm** Action: **Select Action** ▾ ⇌ State: Draft

Description: < Click here to enter a description >

**Requirement Collection Links** ?  
Linked requirement collections that are being validated

Type Filter Text

Show All ▾ Items per page Previous | 1 - 1 of 1 | Next

	Summary
<input type="checkbox"/>	
<input checked="" type="checkbox"/>	CD Collection First Version

Previous | 1 - 1 of 1 | Next

**Discard Changes** **Save**  
Contains Unsaved Changes

**Test Plan Workitems**

**Related Test Suite(s)**

**Related Sites**

- IBM Rational
- IBM Rational Quality Mgmt

Create Test Case(s) from Requirement(s)

# OSLC example: Resource Links in Requirements Tool

The screenshot displays the Rational Requirements Composer interface. The main window shows a requirement named 'Localization (L10n)' with a description '<Enter a Description>'. The requirement type is 'Supplemental'. The 'Links' section on the right shows three links: 'Outgoing Links (0)', 'Incoming Links (2)', and 'Implemented By (1)'. The 'Implemented By' link is highlighted, showing a back link to '30: Localization (L10n)'. The details for '30: Localization (L10n)' are shown below, including status, summary, type, and creation date. The 'Quick Information' section at the bottom right highlights 'Implements Requirement (1)'.

**Implemented By**

**Validated By**

**Back Link**

**Information**

- Overview
- Comments (0)
- Requirements (0)
- Links (3)
  - Outgoing Links (0)
  - Incoming Links (2)
    - Corporate Supplementary Requirements
    - First Release
  - Implemented By (1)
    - [30: Localization \(L10n\)](#)
  - Validated By (0)

**30: Localization (L10n)**

Status	Summary
New	Localization (L10n)

**Details**

Type:	Story	Created By:	rtc
Filed Against:	RTC CD Collection Project	Tags:	
Story Points:	0 pts	Owned By:	rtc
Progress:		Priority:	Unassigned
Project Area:	RTC CD Collection Project	Planned For:	Sprint 1 (1.0)
Creation Date:	November 6, 2009 10:32 AM		

**Quick Information**

Subscribers (1): **Implements Requirement (1)**



# What Makes the OSLC Approach Better?

## Traditional Approach

- Brittle integrations, version-specific APIs
- Monolithic repository or import/export
- “Boil the ocean” meta-model design
- Forced migration to a common code base
- Premature architectural decisions
- A vendor-led “partners” program

## OSLC Approach

- Loosely-coupled
- URLs
- Minimalist
- Technology-neutral
- Incremental
- Open

# See it live at Jazz.net

- Transparent development
  - Jazz architecture
  - Jazz products
- Self-hosting
  - Using Jazz products...
  - ... to develop Jazz products
- Learn about Jazz at Jazz.net
  - Participate in the evolution
- Try it
  - Sandbox available

The screenshot displays the Jazz.net Community Site interface. At the top, the 'jazz' logo is followed by the text 'COMMUNITY SITE' and a banner stating: 'We're building a new generation of products to help make software development more collaborative, productive and enjoyable.' Below this, three main navigation buttons are visible: 'Learn about Jazz' (with a book icon), 'Explore projects' (with a folder icon), and 'Download products' (with a green arrow icon).

The main content area is divided into several sections:

- Jazz Team Blog:** Features a post by 'Lemieux' titled 'Enterprise performance and scalability testing' dated 'Fri, 03 Apr 2009 @ 09:41:40'. The post discusses performance testing using metronome, selfhosting sizing numbers, repository workspace scalability, etc. A 'More >' link is provided.
- Upcoming Events:** Announces the 'Rational Software Conference 2009' from '31 May - 4 June 2009 in Orlando, FL'. It includes a quote: 'It's no secret that in today's economy, we are facing more challenges than ever. And...' and a 'More >' link.
- Rational Quality Manager and Rational Test Lab Manager now available!**: A large announcement banner with the text: 'We host the following development projects right here on Jazz.net. Take a look around and get involved!'
- Projects List:**
  - Rational Quality Manager and Rational Test Lab Manager:** Described as a centralized test management environment that helps increase the efficiency and quality of software delivery through test planning, workflow control, tracking and traceability, and metrics reporting. Rational Test Lab Manager, an extended component of Rational Quality Manager, helps to improve the efficiency of the test lab environment and optimize its utilization, cutting workload and saving on test infrastructure.
  - Rational Requirements Composer:** Described as a platform for collaborative requirements definition that enables business analysts, client stakeholders and software development teams to elicit, capture, elaborate, discuss, review, and validate requirements using a variety of requirements definition techniques and collaboration capabilities.
  - Rational Team Concert:** Described as a team-aware software development platform that integrates work item tracking, builds, source control, and agile planning. Rational Team Concert interoperates with other products by providing Visual Studio integration and connectors for ClearCase and ClearQuest.
  - Collaborative ALM:** Described as integrations that leverage and build upon the Jazz Foundation to increase the productivity of software delivery teams by unifying the disciplines in software lifecycle using in-context collaboration on software artifacts, web-like artifact navigation, and status tracking across...
- Community Resources:** A sidebar section with links to 'Join the conversation and interact with developers and community members:', 'Jazz.net Forums', 'Jazz Team Blog', 'Jazz Team Wiki', and 'Jazz Mailing Lists'.
- Related Downloads:** A sidebar section listing 'Rational Quality Manager and Rational Test Lab Manager 1.0.1.1', 'Rational Requirements Composer 1.0.0.1', and 'Rational Team Concert 1.0.1.1', with an 'All Downloads >' link.