Open Services for Lifecycle Collaboration

open community. open interfaces. open possibilities.

An Introduction to OSLC and Linked Data

Source: http://open-services.net/resources/presentations/introduction-to-oslc-slideshow/,
by Steve Speicher

Modifications: Jad El-khoury (jad@kth.se)

Workshop Objectives

After completing this session, you

- Understand the structure and content of the OSLC standard
  - OSLC Core specification
  - OSLC domain specification(s)
    - Change Management – an example

- Understand the basics of Linked Data
  - And its supporting technologies (RDF, RDF Schema, …)

- Gained hands-on experience in developing OSLC-based adapters

Pre-requisites

You understand

- Basics of web technologies
  - URI, HTTP, web services, web servers, …
- The REST architectural style

For the hands-on tutorial

- You are familiar with
  - Java development
  - Eclipse environment
  - Web development
    - web services, HTML, jsp-files, etc.

Who we are …

Jad El-khoury, PhD

- @KTH
- Researcher
- Teacher, Master program director, …

- Research focus
  - Tool interoperability
  - Model-based development
  - Eclipse Committer
    - the OSLC Lyo project

Frederic Loiret, PhD

- @KTH and @OFFIS
- Researcher
- European Project(s) Manager

- Research focus:
  - Tool Interoperability
  - OSLC (pre-)standardization activities

Who are you?

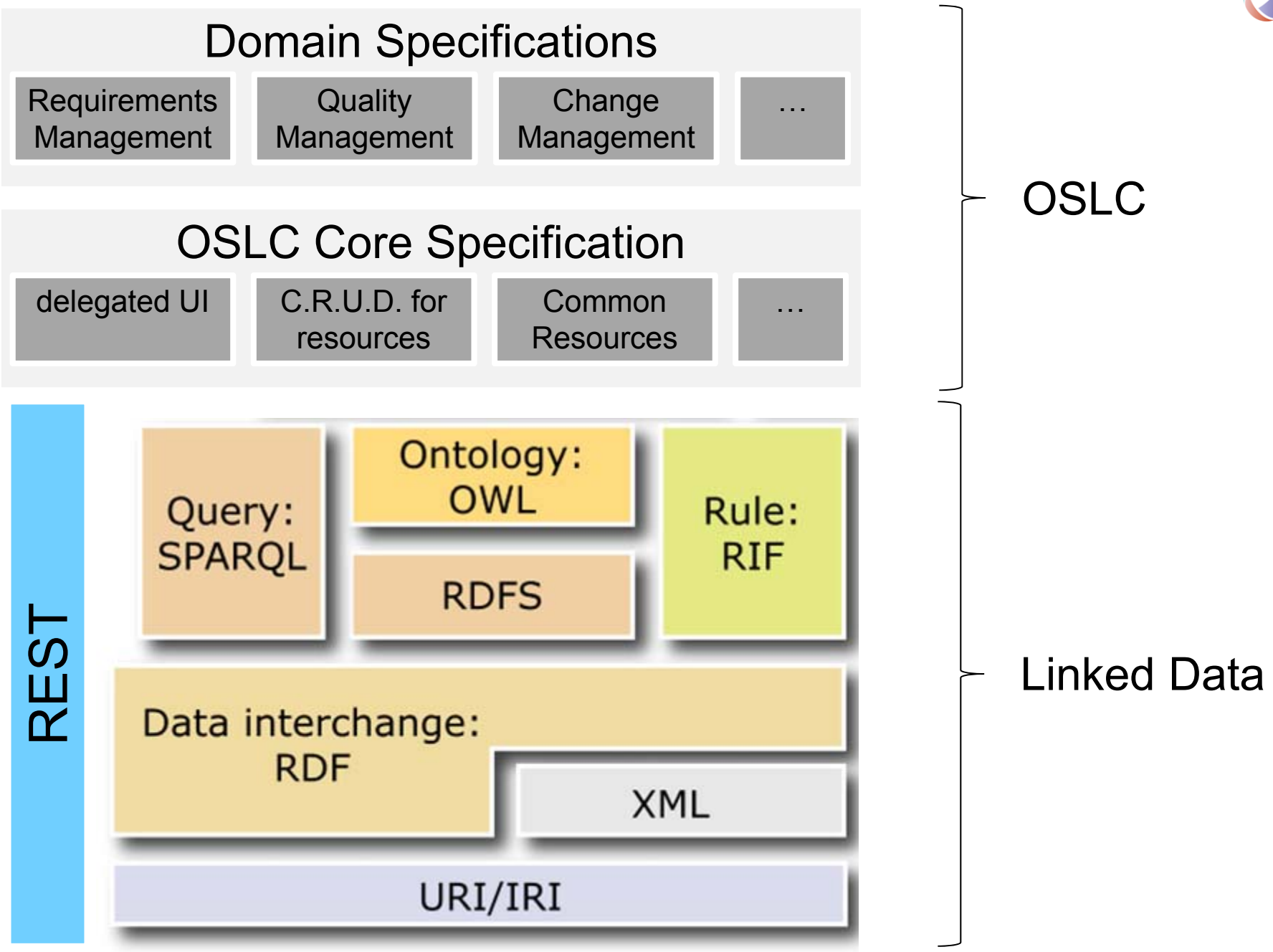
Short round-table presentations

- Who are you? What do you do?
- What do you expect from this workshop?
- Any [basic] technologies you want us to cover?

Today's Schedule

When	What
09:15-10:15	Introduction to Linked Data
10:15-11:00	*Coffee*
11:00-12:00	Introduction to OSLC
12:00-13:20	*Lunch and Exhibition*
13:20-13:50	*Keynote Presentation*
14:00-15:00	OSLC Hands-on Tutorial
15:00-15:30	*Coffee*
15:30-17:00	OSLC Hands-on Tutorial

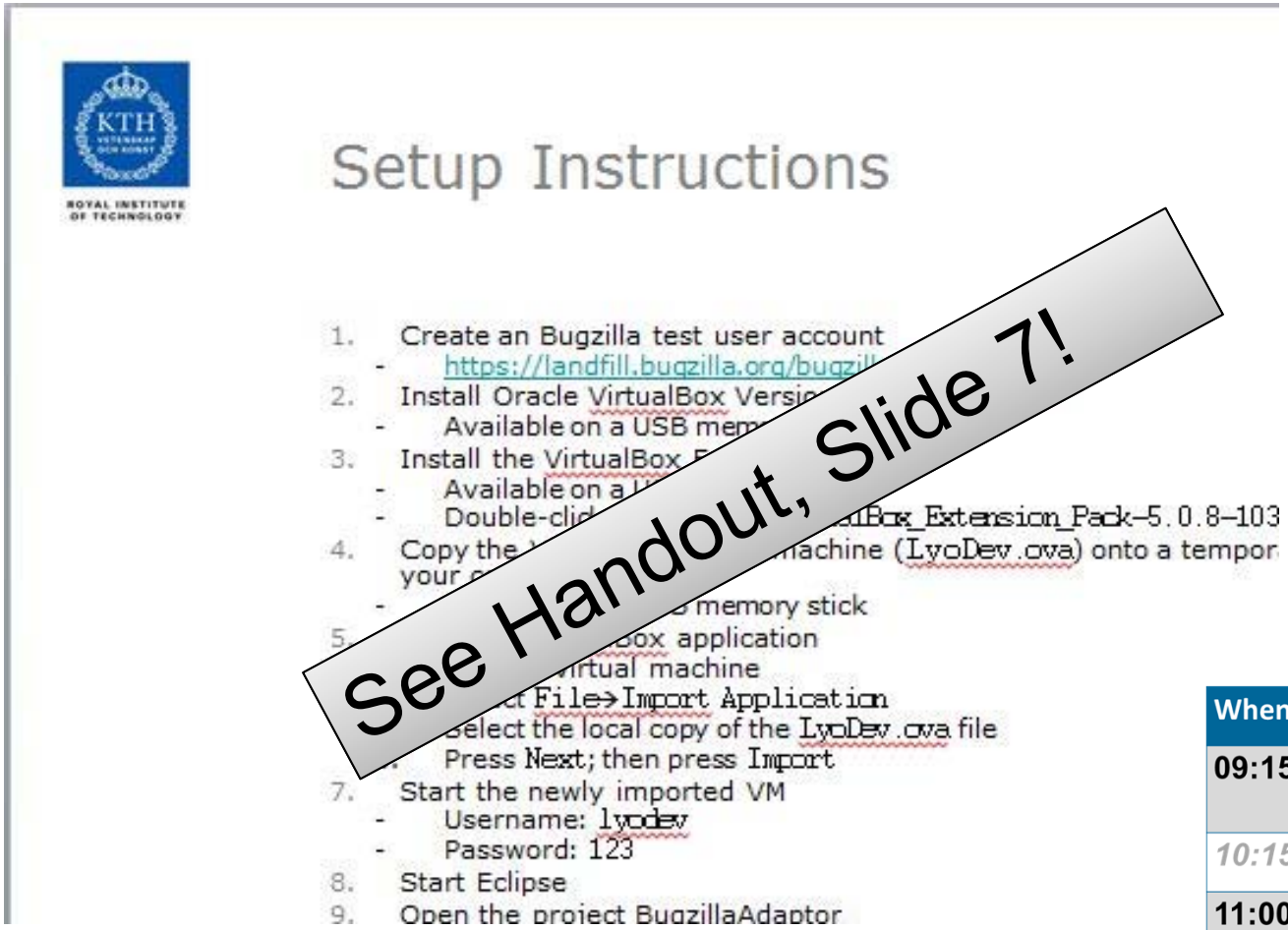
The OSLC Technology Stack



Disclaimer: This is not strictly correct

Source: http://www.w3.org/2007/03/layerCake.png

Preparation for the Afternoon Tutorial …



→ Work in pairs?

When	What
09:15-10:15	Introduction to Linked Data
10:15-11:00	*Coffee*
11:00-12:00	Introduction to OSLC
12:00-13:20	*Lunch and Exhibition*
13:20-13:50	*Keynote Presentation*
14:00-15:00	OSLC Hands-on workshop
15:00-15:30	*Coffee*
15:30-17:00	OSLC Hands-on workshop

Agenda

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - domain specification(s)
    - Requirement Management – an example

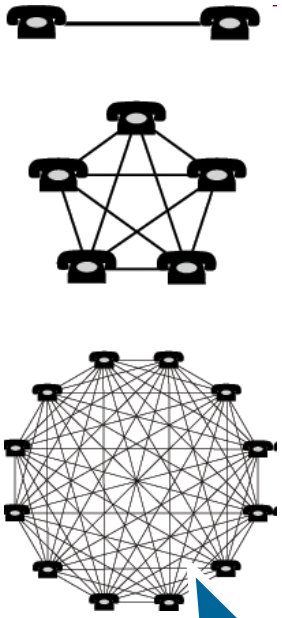
… Followed by the OSLC Hands-on Tutorial

What's next

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - Domain specification(s)
    - Requirement Management – an example

The Integration Problem

Point-to-point Integrations don't scale



Creating new integrations is unpredictable

Monocultures lock you in

Past choices restrict present action and future vision

Maintenance, management, and change costs go up over time

Ongoing and unexpected costs drain resources

End-user productivity suffers: Either stuck with the wrong tool, stuck doing manual integration; often stuck doing both

Integrations consume more of the IT budget: integration failures are the top 2 causes of software project delays*

More limited ability to respond to change
Constrained by exhausted IT budget and lower productivity

* Commissioned study conducted by Forrester Consulting on behalf of IBM.

OSLC's Simple Solution

Users can work seamlessly across their tools

*** Architecture of the Web**

*** Linked Data**

Increased reuse

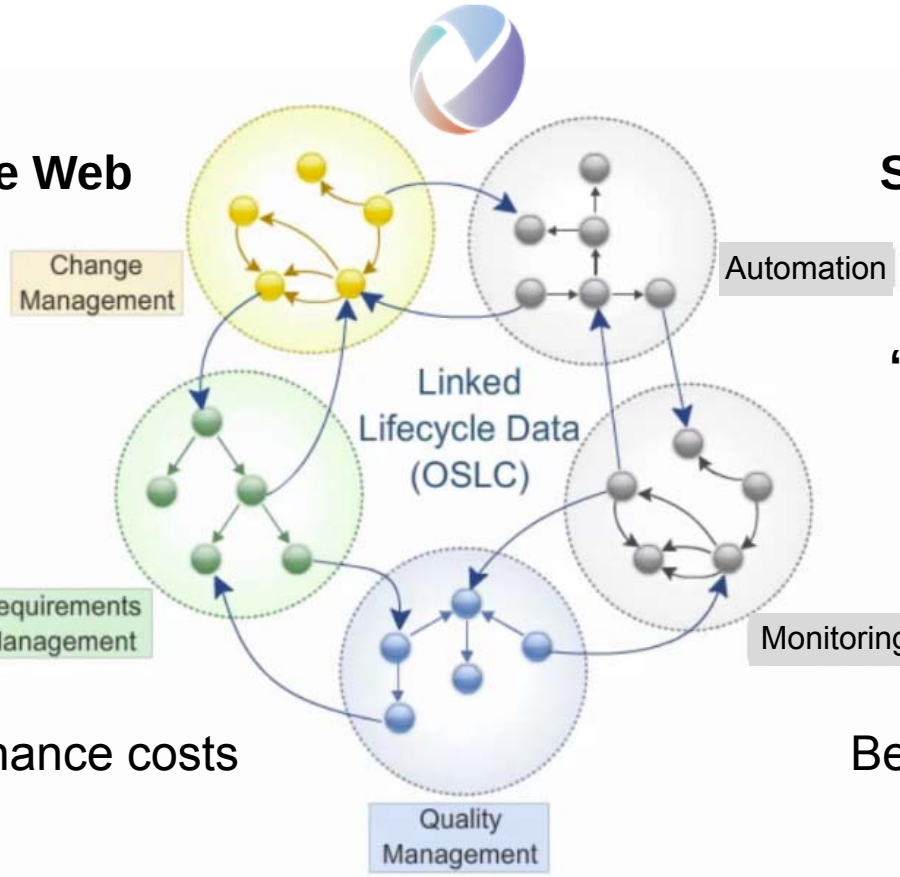
Decreased maintenance costs

Standard Interfaces *

"Just Enough" integration *

Increased traceability

Better visibility

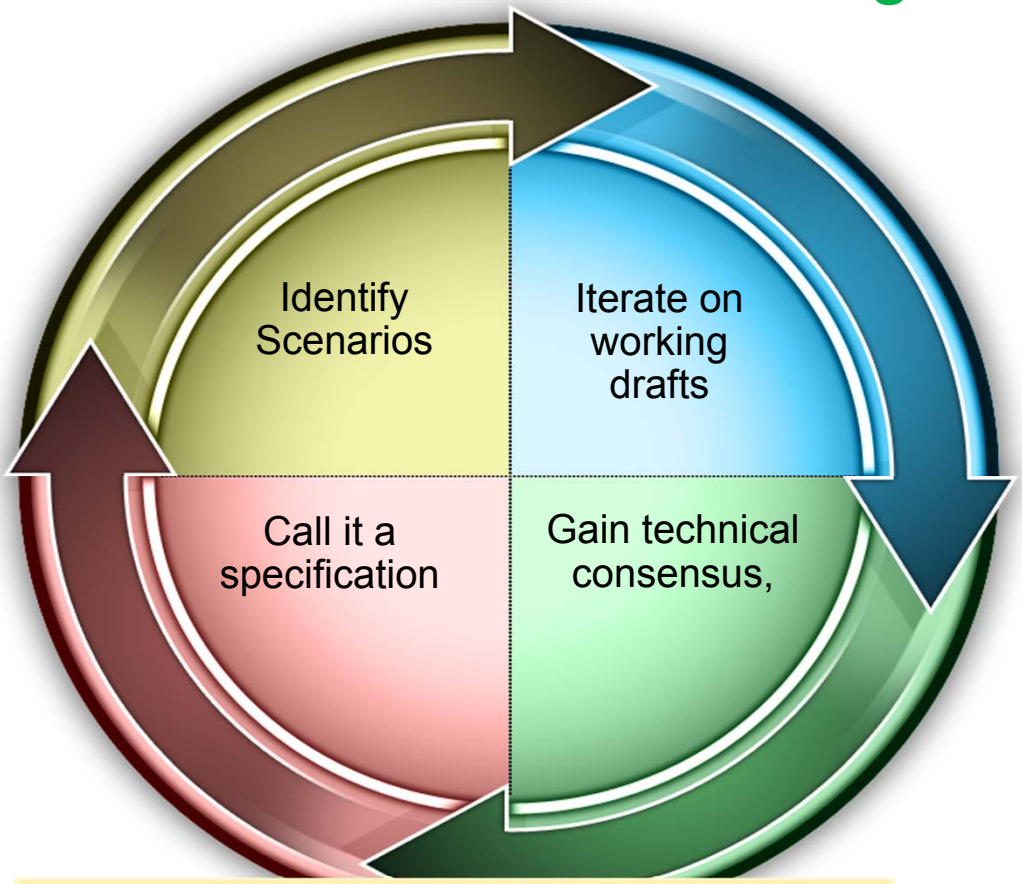


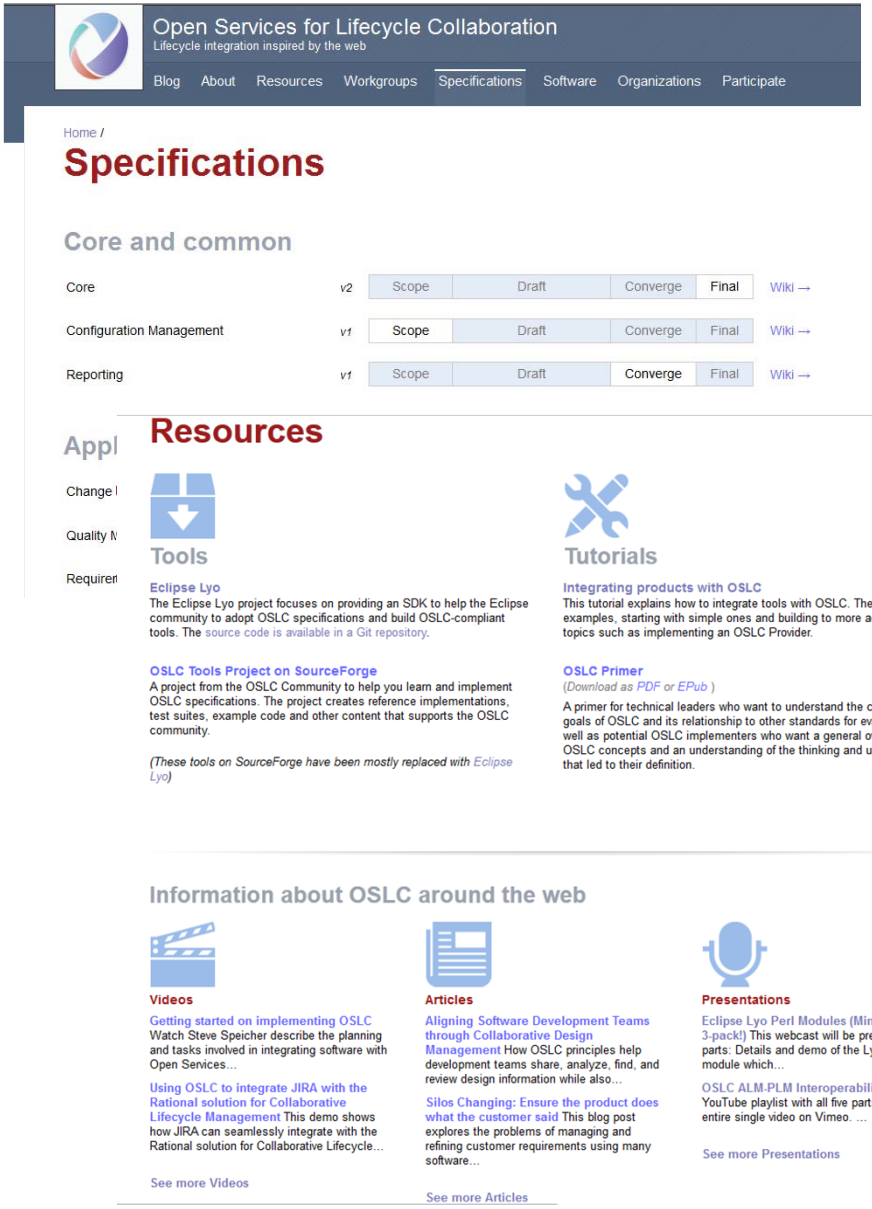
OSLC is an open and scalable approach to lifecycle integration. It simplifies key integration scenarios across heterogeneous tools

* Jad's highlights

The OSLC Approach

An open community building practical specifications for integrating software



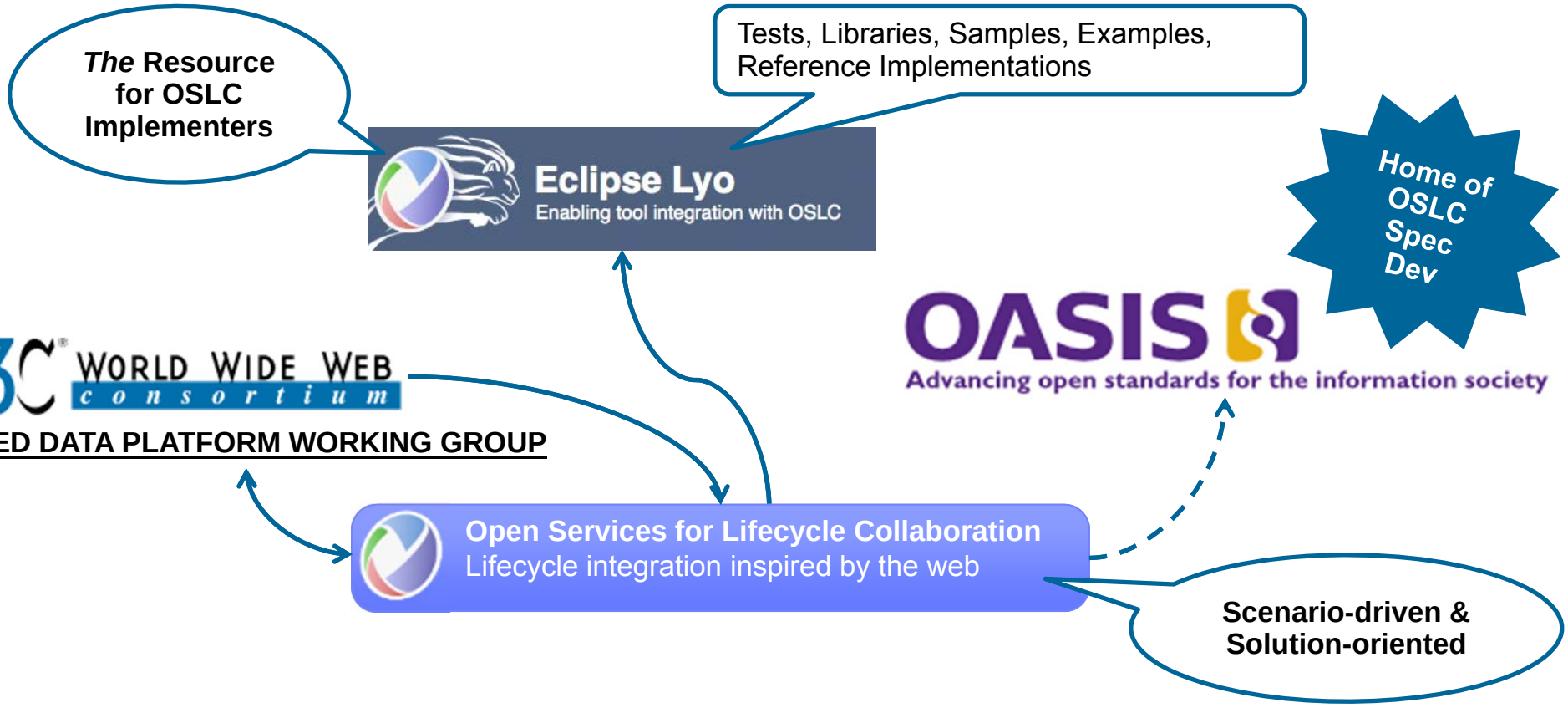


http://open-services.net

Now also an OASIS standard

http://www.oasis-oslc.org/

OSLC's Big Picture

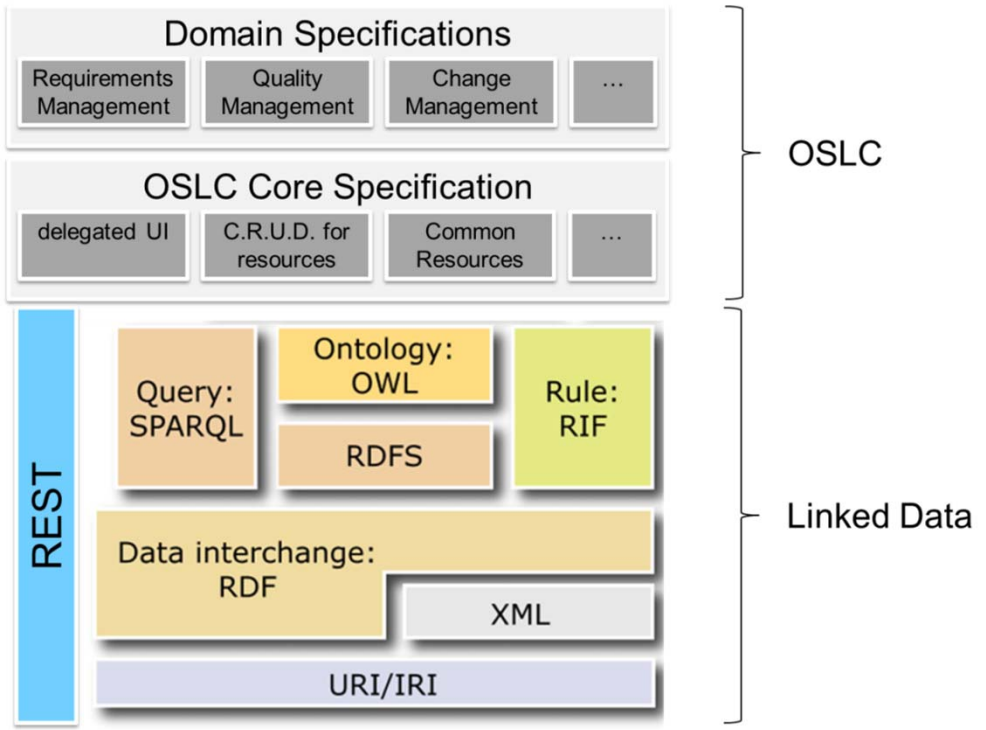


OSLC:

- Inspired by the web — **Proven**
- Free to use and share — **Open**
- Changing the industry — **Innovative**

What's next

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - Domain specification(s)
    - Requirement Management – an example

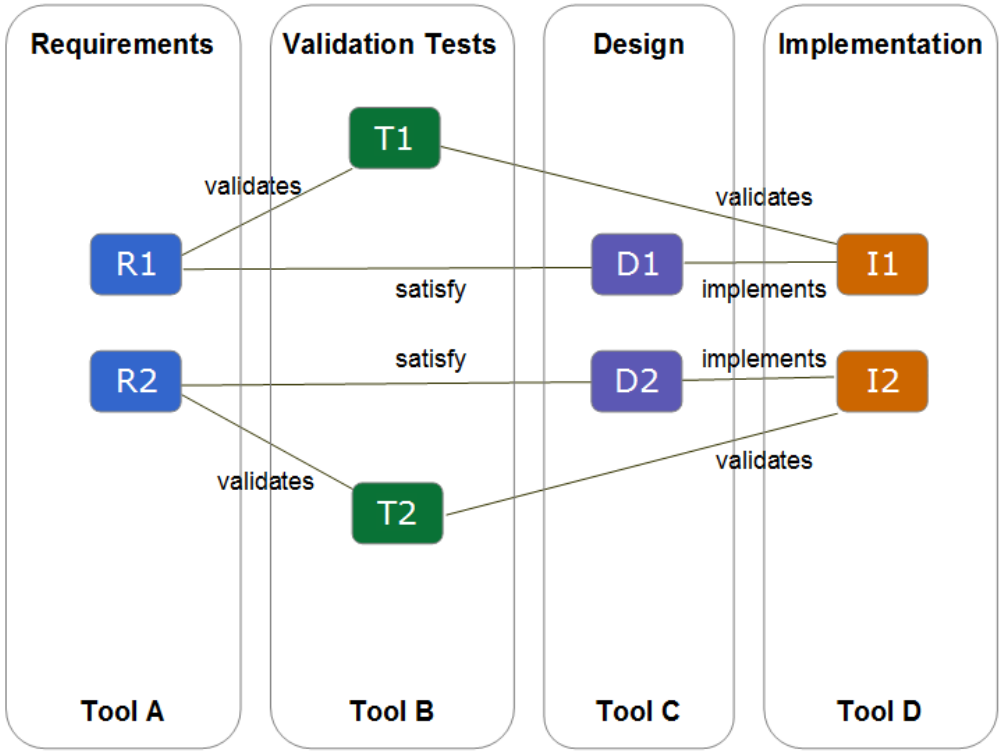


Linked Data

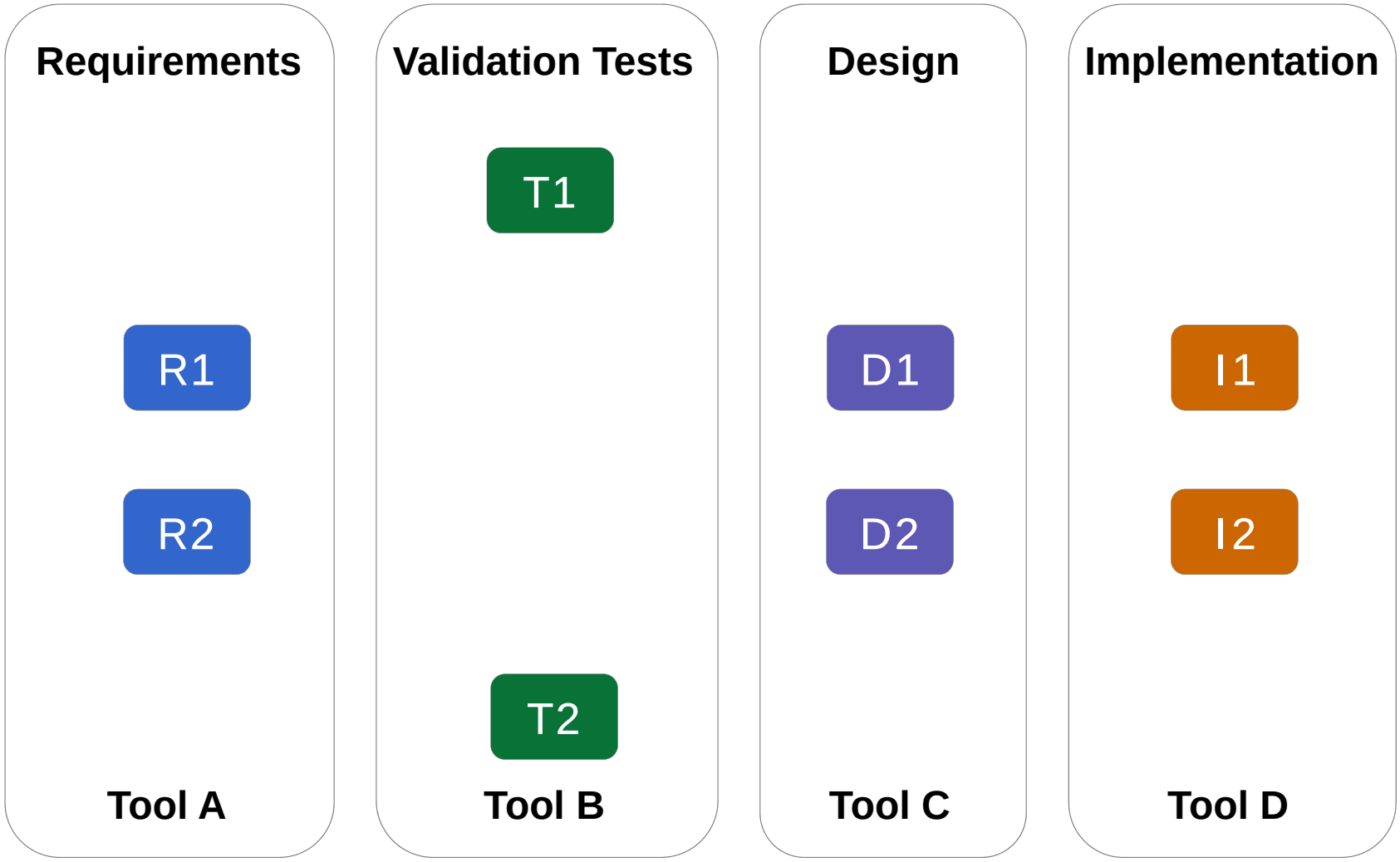
- An approach of publishing structured data, such that

1. Data from different sources can be connected
   → Data gets more meaning

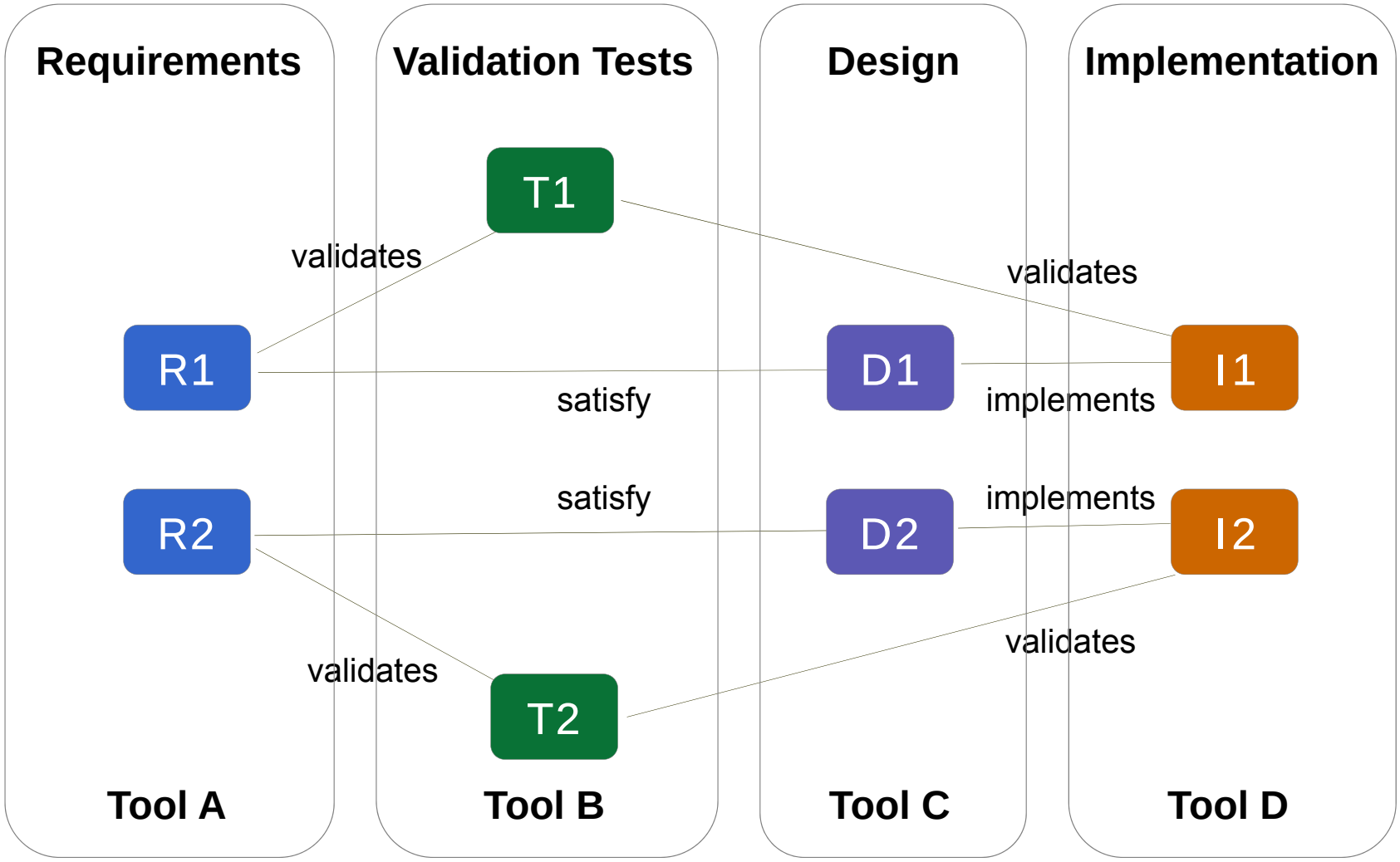
2. Data from different sources can be queried
   → Data becomes more useful



Linked Data turns data into...



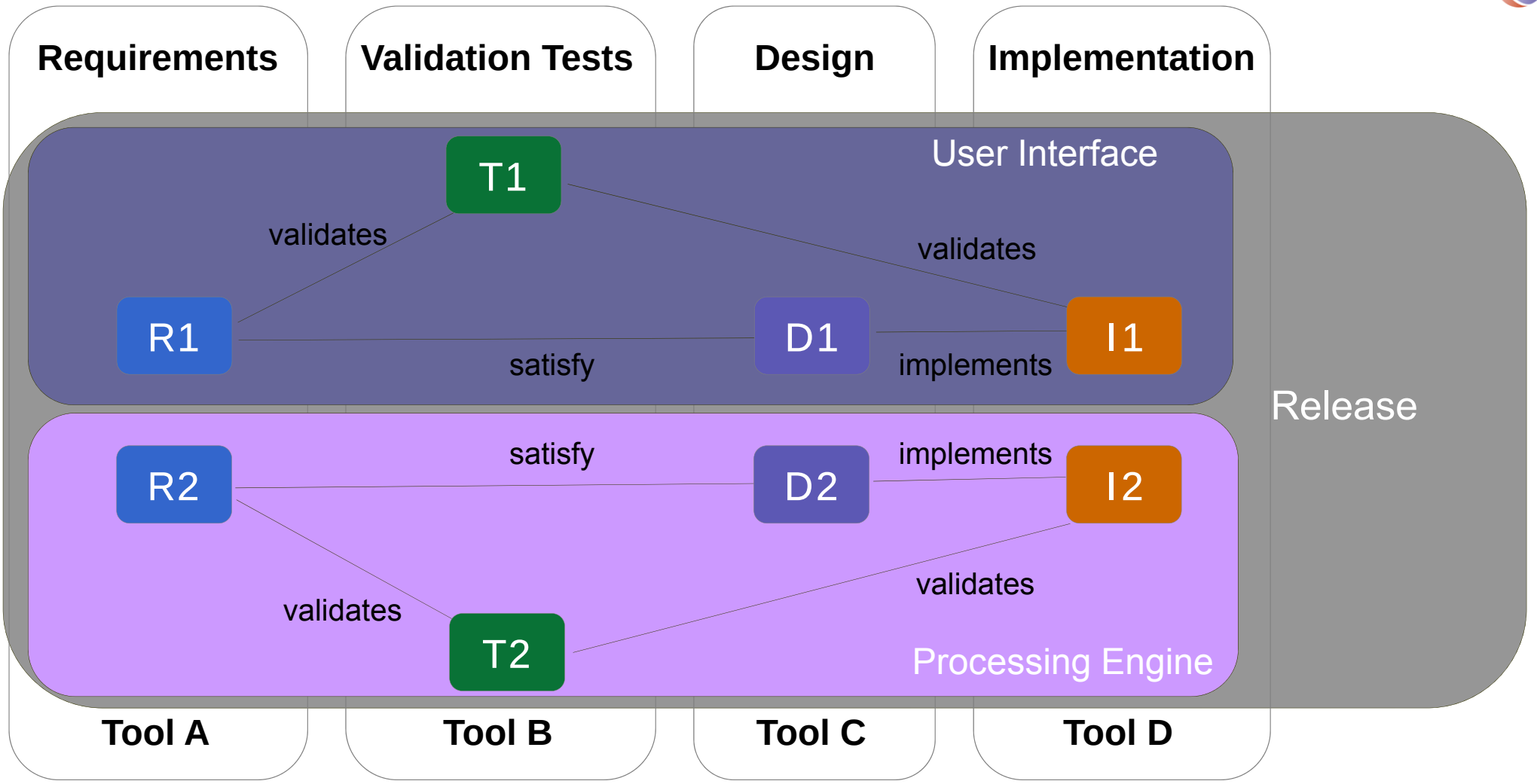
...connected information...



Does every requirement have a test to validate it?

Which requirements for the UI are related to test cases that failed on their last run?

...that can facilitate applied knowledge



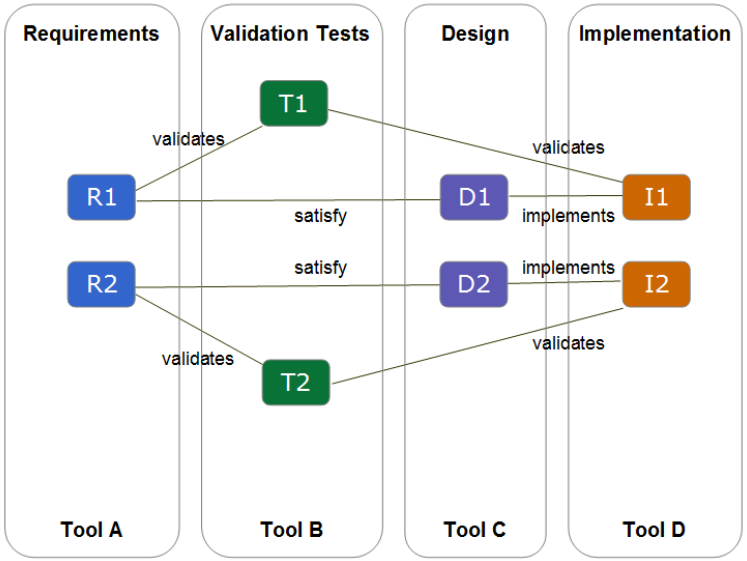
Why is the number of failed test cases for the UI increasing in each iteration?

How much faster is work progressing on the UI versus the Processing Engine?

Linked Data Principles

Tim Berners-Lee's four principles for Linking Data:

1. Use URIs as names (*identity*) for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information using the standards (RDF, SPARQL)
4. Include links to other URIs so that they can discover more things



Linked Data Example
- The Linking Open Data project

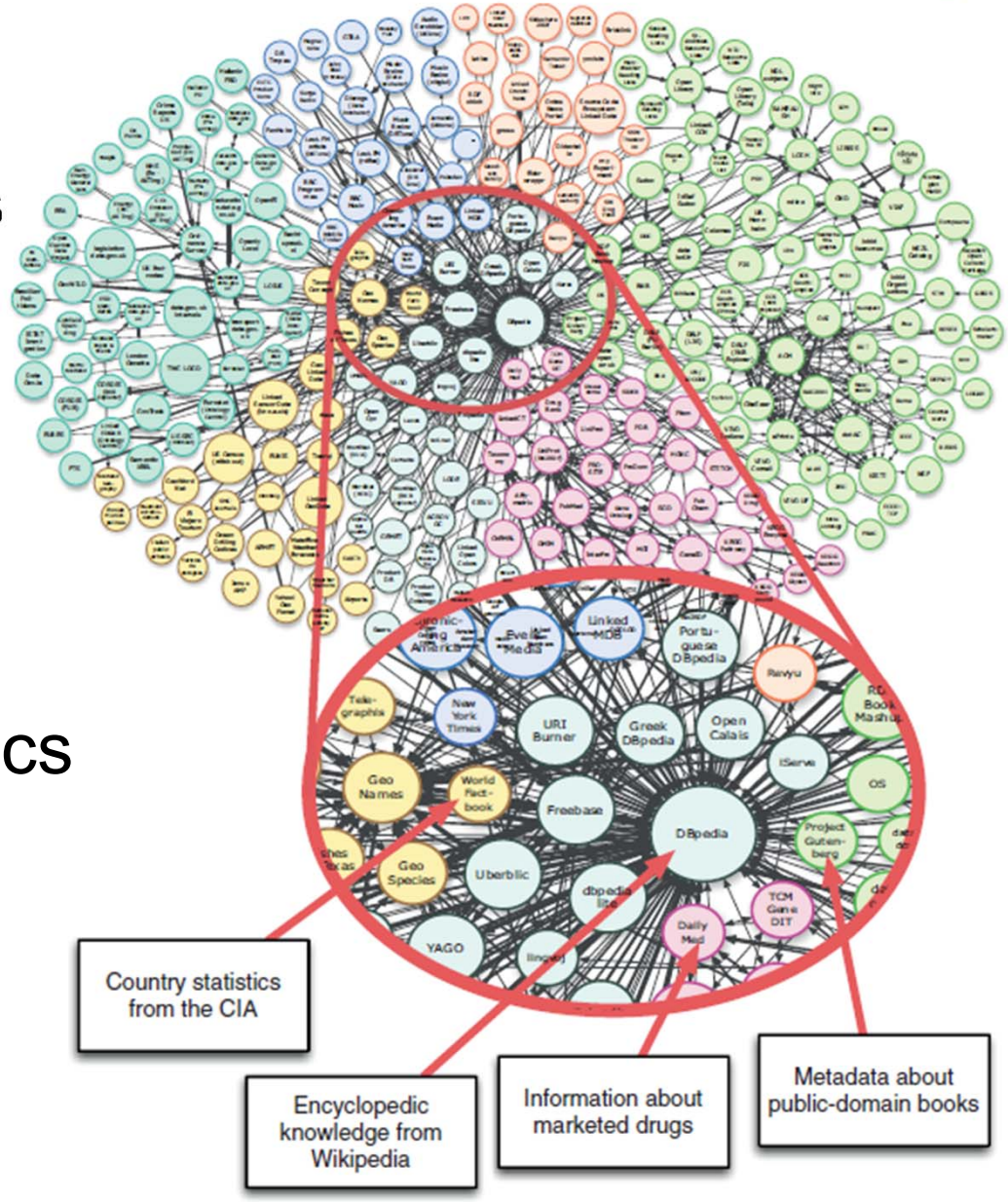
- Links data from open-content projects such as
  - encyclopaedias and dictionaries
  - government statistics
  - bibliographic data
  - music
  - research papers
  - …

→ Access to data & its semantics

→ No longer Data Silos

→ Discoverability

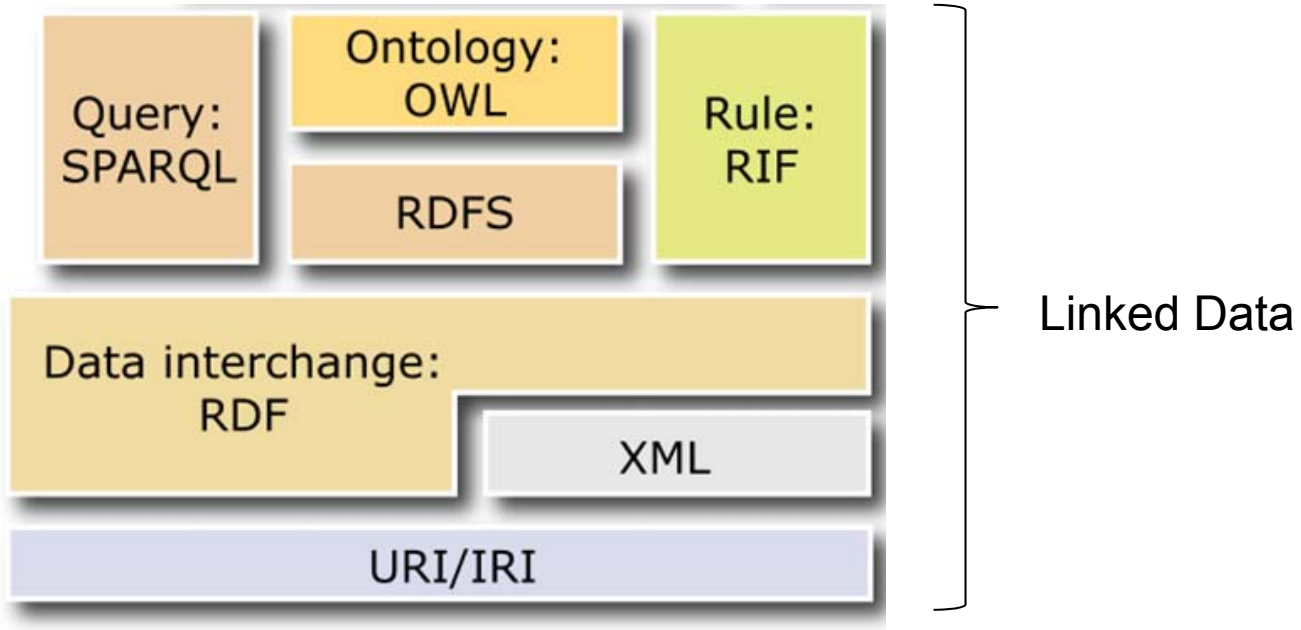
→ Data discovered and used in unpredictable ways



Linked Data - Structured Data on the Web; by David Wood, Marsha Zaidman, and Luke Ruth; Fig 1.5

Linked Data Technologies

- Builds upon standard Web technologies
  - RDF standard(s)
  - HTTP
  - URIs



Source: http://www.w3.org/2007/03/layerCake.png

RDF Standard(s)

RDF (Resource Description Framework)

- a standard to describe structured data on the web.
- designed to be understood by computers (xml) - not to be displayed to people (html)

Examples of Use

- Describing time schedules for web events
- Describing information about web pages (content, author, created and modified date)

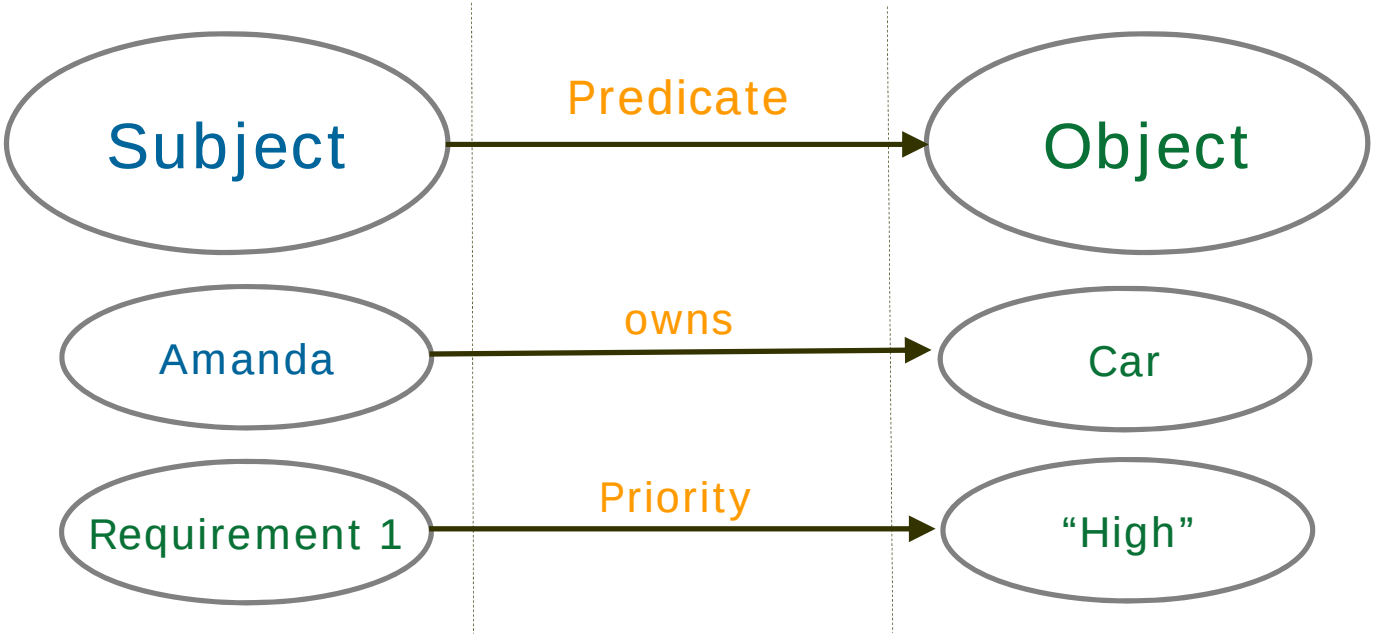
RDF key concepts:

1. Graph data model
2. URI-based vocabulary
3. Serialization syntaxes
4. Vocabularies

→ *We'll briefly look at some of them*.

Basic structure - The Triple

- consisting of a subject, a predicate and an object.

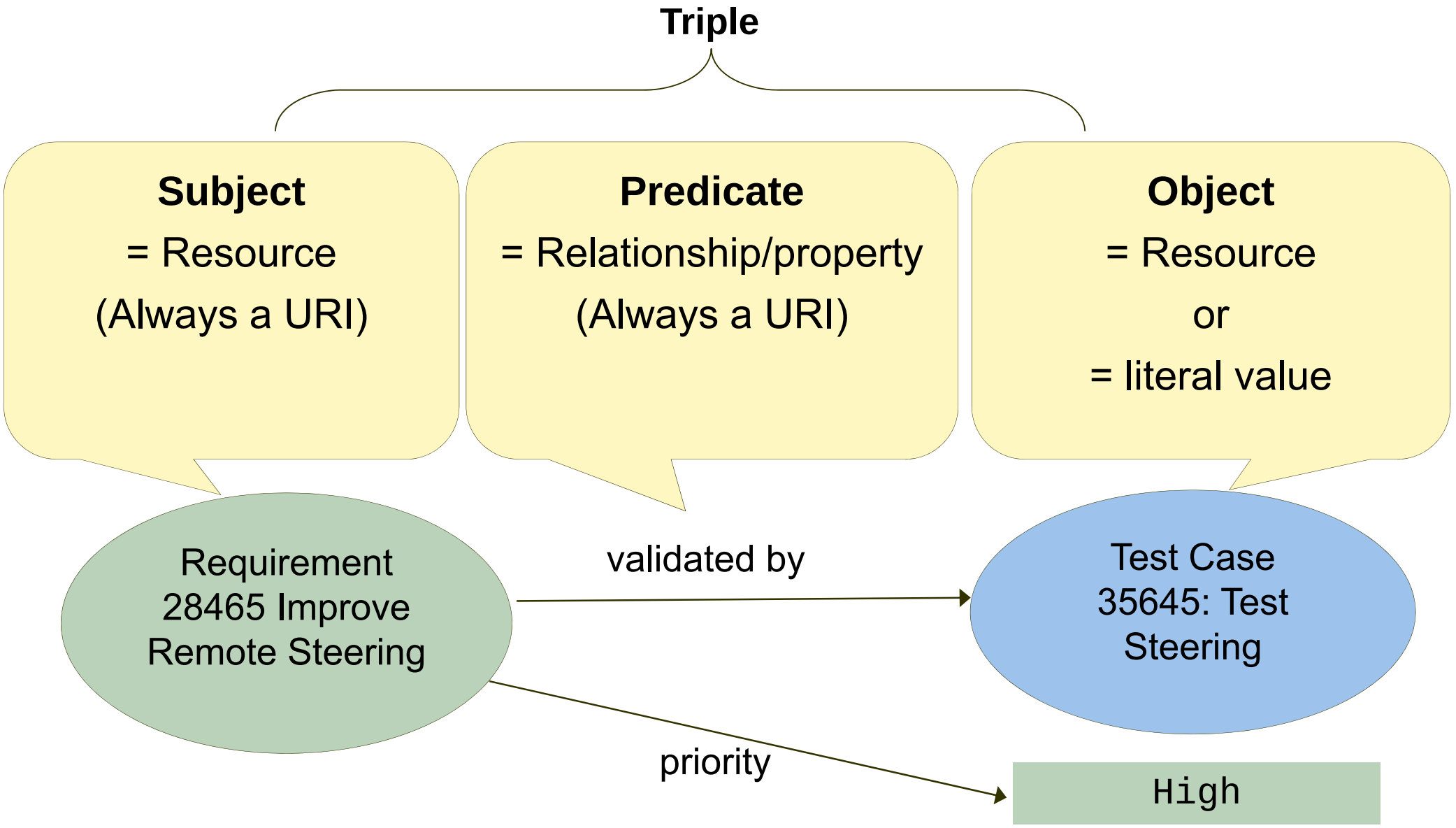


The predicate (also called a property) denotes a relationship between the subject and object.

Adapted from:
http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-data-model

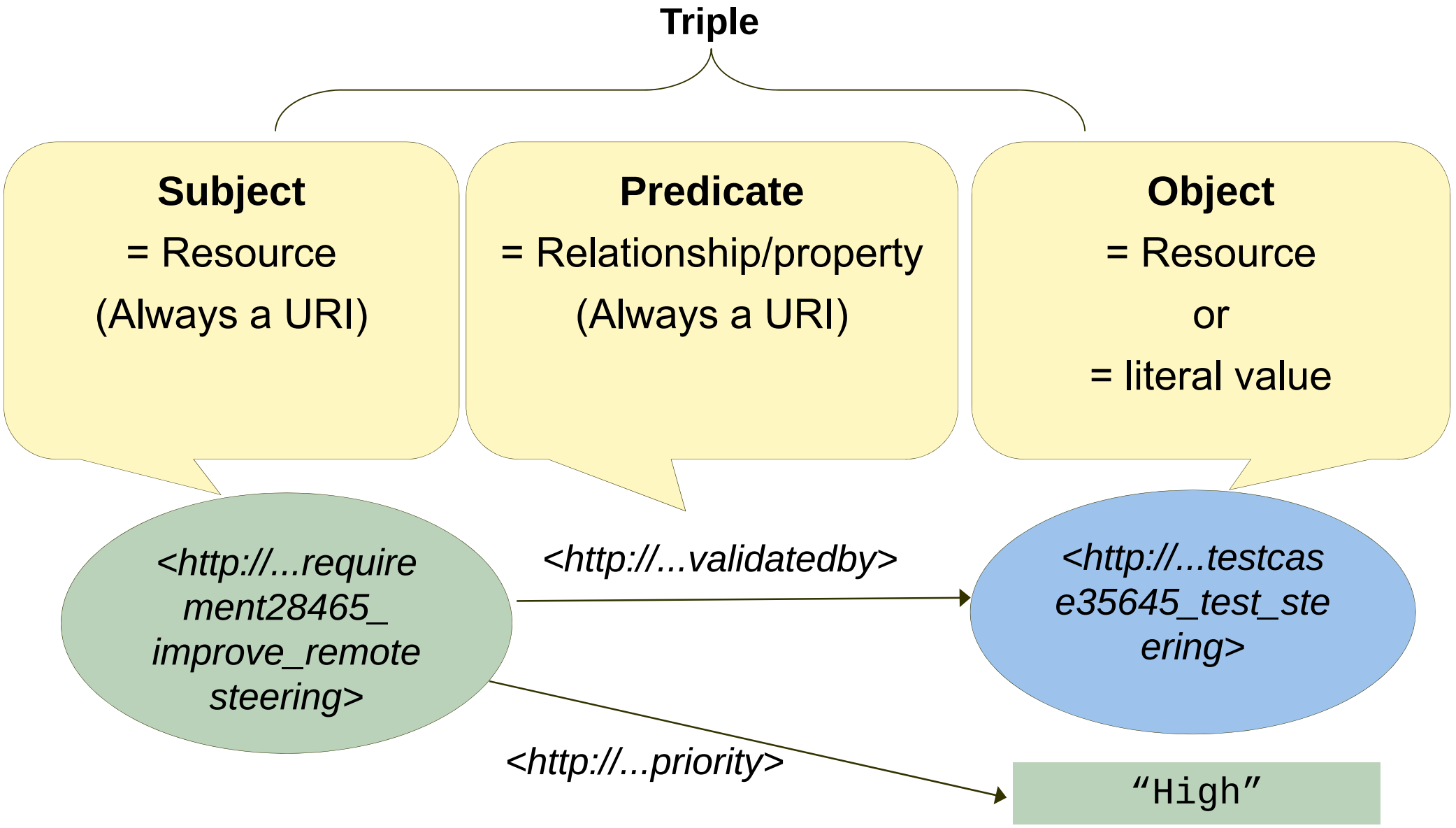
1. The RDF graph data model

RDF triple (subject-predicate-object)



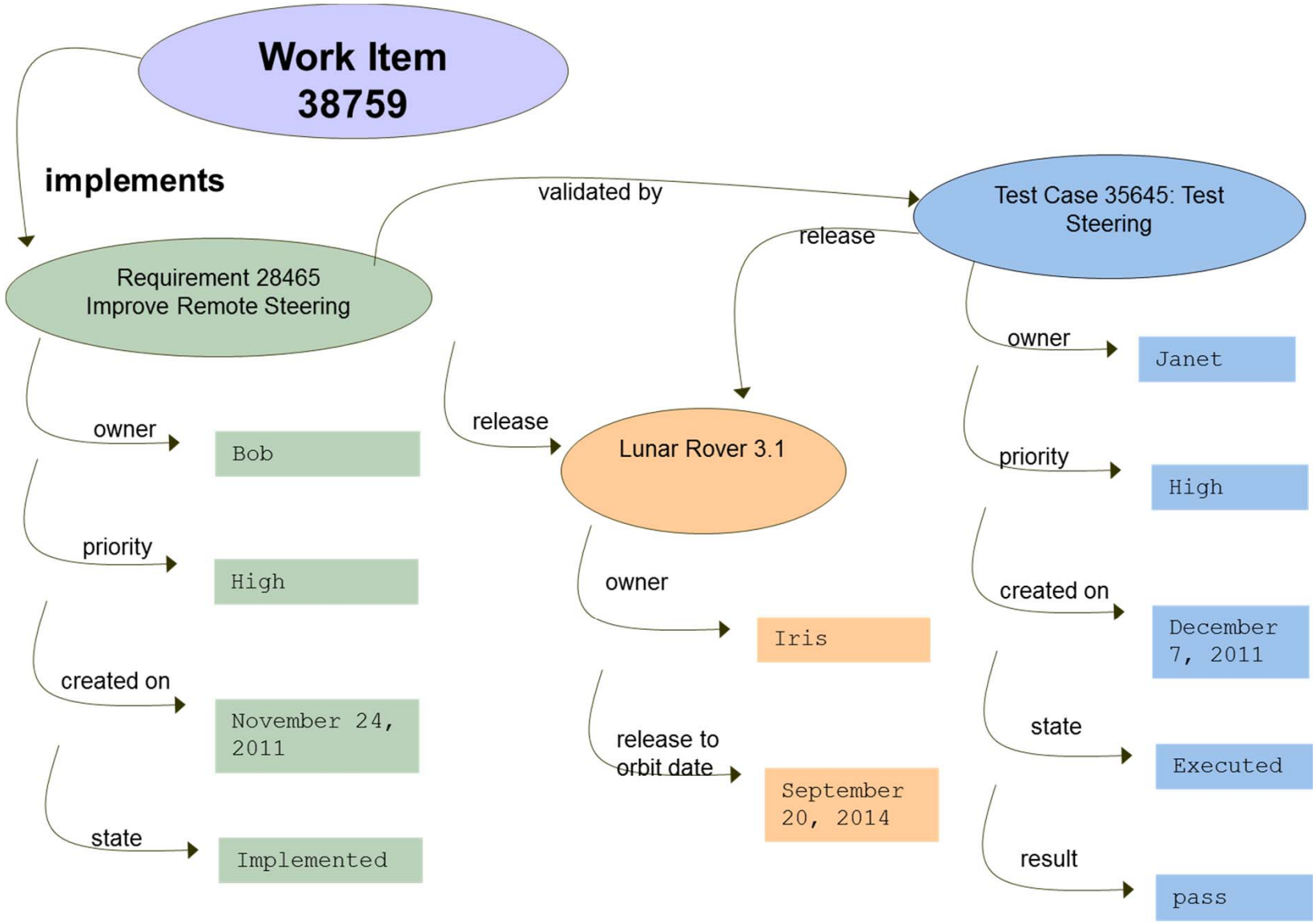
1. The RDF graph data model

RDF triple (subject-predicate-object)



1. The RDF graph data model

- Set of triples – leads to an RDF graph
  - No hierarchical relationships



There is a web of URIs around a development effort

1. The RDF graph data model
- Compare to other data models

Compare to other data models

- *Relational model*
- *object-oriented model*

1. Closed-world assumption vs Open-world assumption
   - Relational model & object-oriented model
     - If you are of type X, you must have these properties.
   - RDF (& the natural world)
     - If you have these properties, you must be of type X.

   → Implications?

2. A property in the RDF model is the “first-class citizen”
   - In the OO model, it’s defined in the context of a class.

3. Unlike in the OO model
   - The RDF model does not have methods
   - All parts of the RDF graph are public.

1. The RDF graph data model
- Compare to other data models

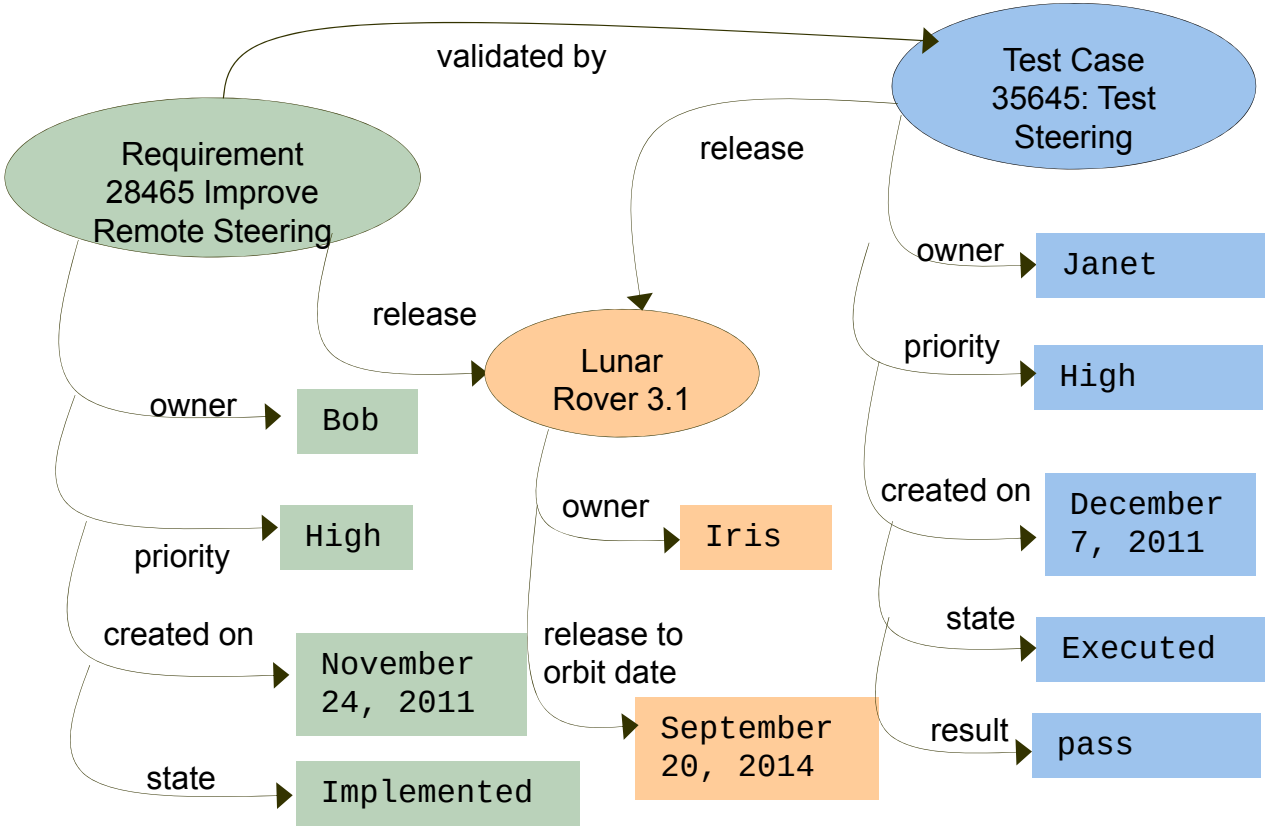
Requirement	Owner	Priority	…	Release	Validated by
R28464 …	…	…	…	…	
R28465 Improve Remote Steering	Bob	High	…	LR3.1	TC35645
R28466 …	…	…	…	…	…

Rover Release	Owner	Release to orbit date
Lunar Rover 3.0	…	…
Lunar Rover 3.1	Iris	Sept 14, 2014

Test Case	Owner	Priority	...
Test Case 35645 Test Steering	Janet	High	...
Lunar Rover 3.1	…		...

Closed-world assumption vs Open-world assumption

→ Implications?



2. URI-based vocabulary

When there is a need to identify anything, use a URI (there are a few exceptions).

- Using URIs allows everything to be linked together.
- It also allows common agreed-upon meaning for relationships and for resource types

<http://...Test Case 1> <http://...validates> <http://...Requirement 1>

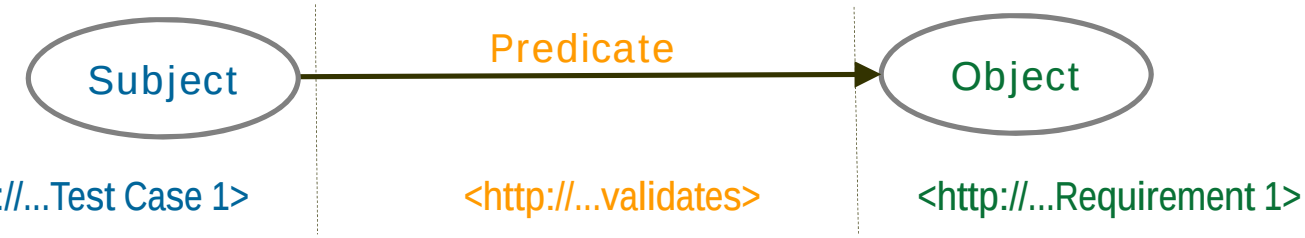
3. Serialization syntaxes

The RDF model provides for describing RDF triples.

Support for different serialization formats:

- Turtle - specialized for RDF
- RDF/XML – derived from standard XML
- JSON

3. Serialization syntaxes - Example




```
<http://example.com/TestCases/1> a oslc_qm:TestCase ;
        oslc_qm:validatesRequirement <http://example.com/Requirements/1>
```

Turtle

```
{
        "rdf:about": "http:\/\/example.com\/TestCases\/1",
        "rdf:type": [ {
                "rdf:resource": "http:\/\/open-services.net\/ns\/qm#TestPlan"
        } ],
        "oslc_qm:validatesRequirement": {
                "rdf:resource": "http:\/\/example.com\/Requirements\/1"
        }
}
```

JSON

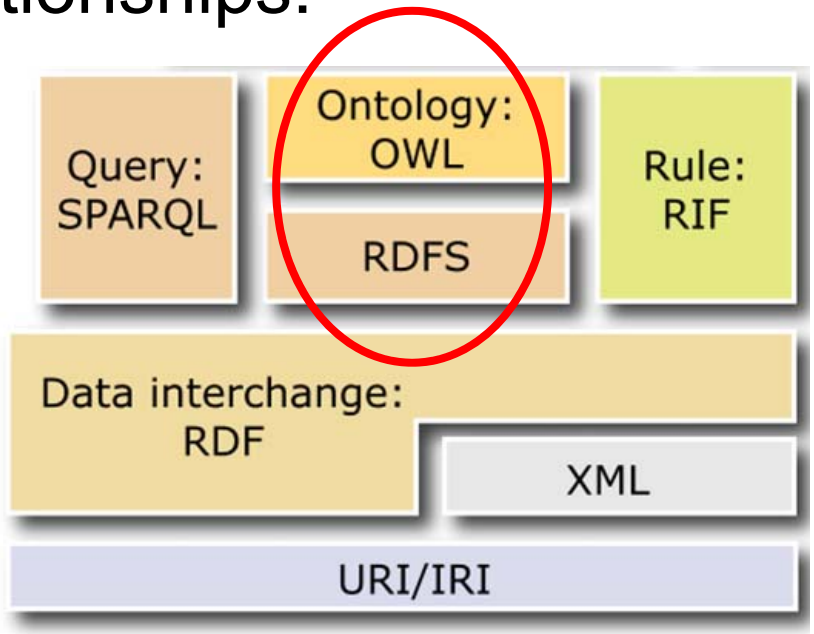
```
<oslc_qm:TestCase rdf:about="http://example.com/TestCases/1">
        <oslc_qm:validatesRequirement rdf:resource="http://example.com/Requirements/1"/>
</oslc_qm:TestCase>
```

RDF/XML

4. Vocabularies

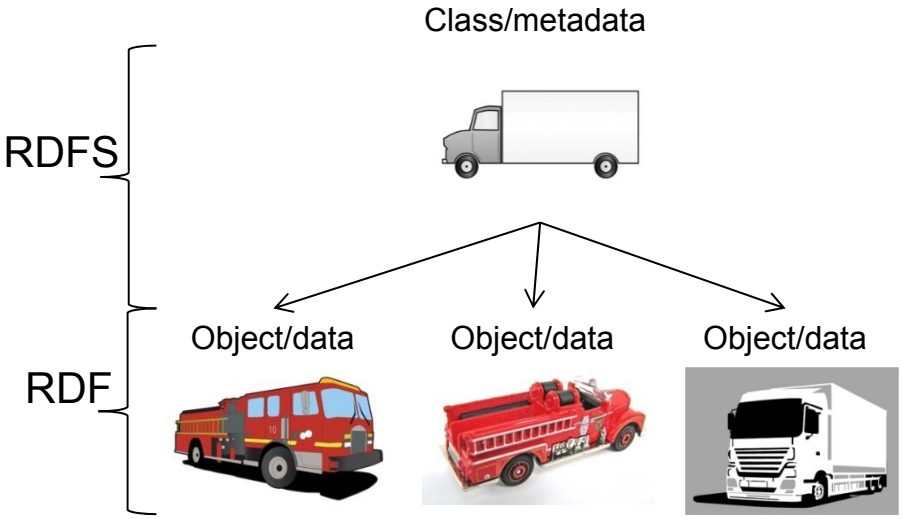
- RDF describes resources

→ We need a **vocabulary** to define the kind of resources (Classes) that can exist and their relationships!



- Approaches:
  - RDF Schema (RDFS)
    - A basic language framework
    - Adds classes, subclasses and properties to resources
  - Web Ontology Language (OWL)
    - More complex formalised language
    - uses logic to process information and make deductions.

- RDF Schema – an extension of RDF
  - Provides the framework to describe application-specific classes of resources.
  - Does **not** provide actual application-specific classes and properties.
  - Resources are defined as instances of classes, and subclasses of classes.




```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.animals.fake/animals#">

<rdf:Description rdf:ID="animal">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="horse">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#animal"/>
</rdf:Description>

</rdf:RDF>
```

- Dublin Core Metadata Initiative (DCMI)
  - Defines a set of properties for describing documents.

Property	Definition
Creator	An entity primarily responsible for making the content of the resource
Title	A name given to the resource
Format	The physical or digital manifestation of the resource
Date	A date of an event in the lifecycle of the resource
Publisher	An entity responsible for making the resource available
Subject	A topic of the content of the resource
…	

Linked Data Example
- The Linking Open Data project

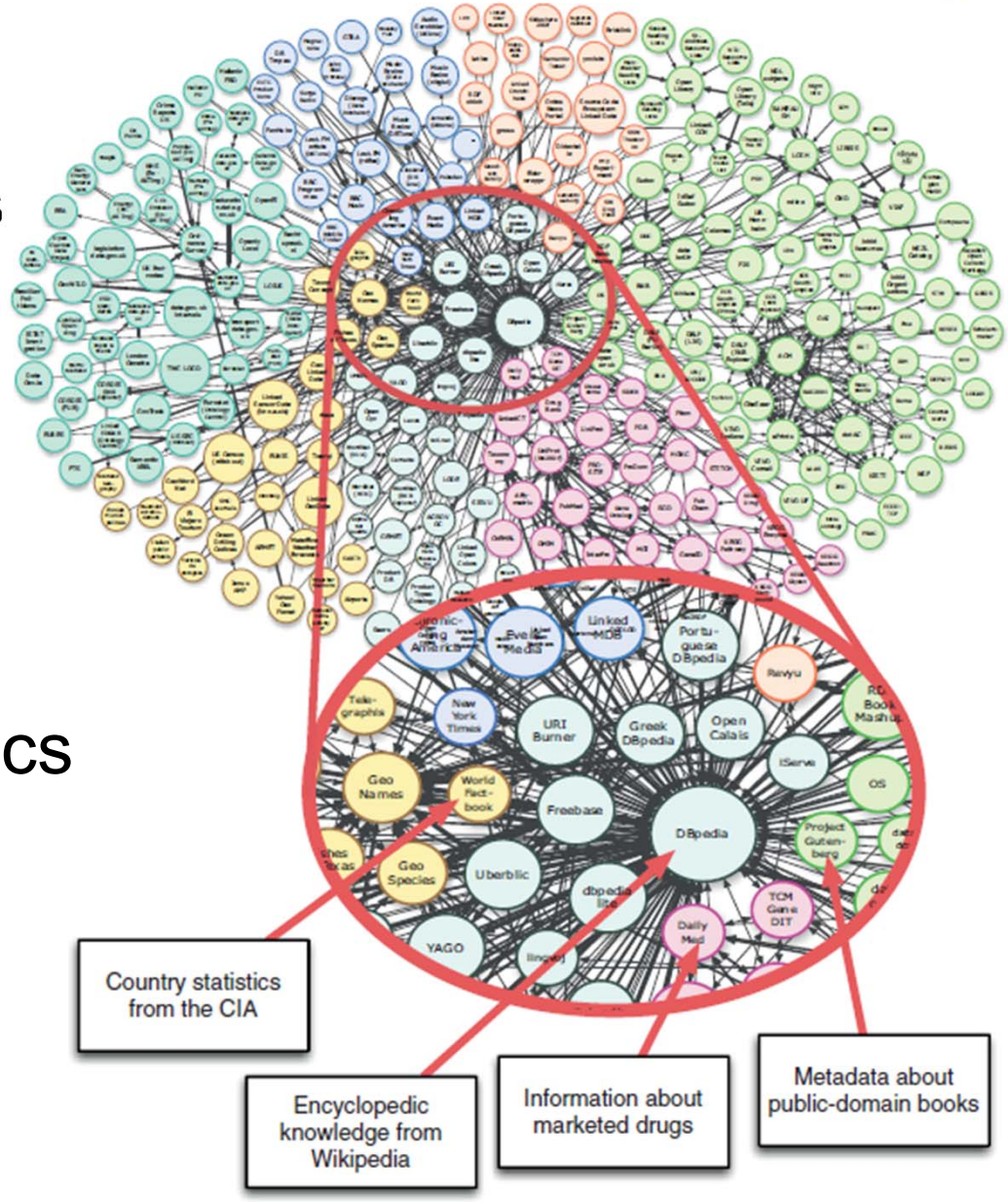
- Links data from open-content projects such as
  - encyclopaedias and dictionaries
  - government statistics
  - bibliographic data
  - music
  - research papers
  - …

→ Access to data & its semantics

→ No longer Data Silos

→ Discoverability

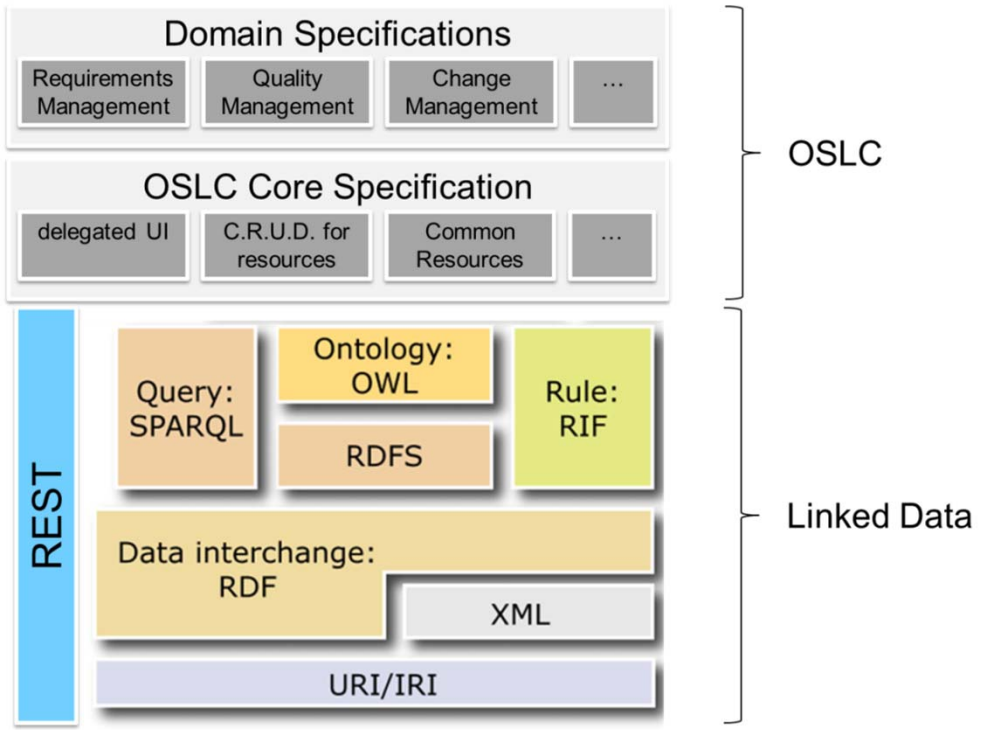
→ Data discovered and used in unpredictable ways



Linked Data - Structured Data on the Web; by David Wood, Marsha Zaidman, and Luke Ruth; Fig 1.5

What's next

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - Domain specification(s)
    - Requirement Management – an example



OSLC's Simple Solution

Users can work seamlessly across their tools

*** Architecture of the Web**

*** Linked Data**

Increased reuse

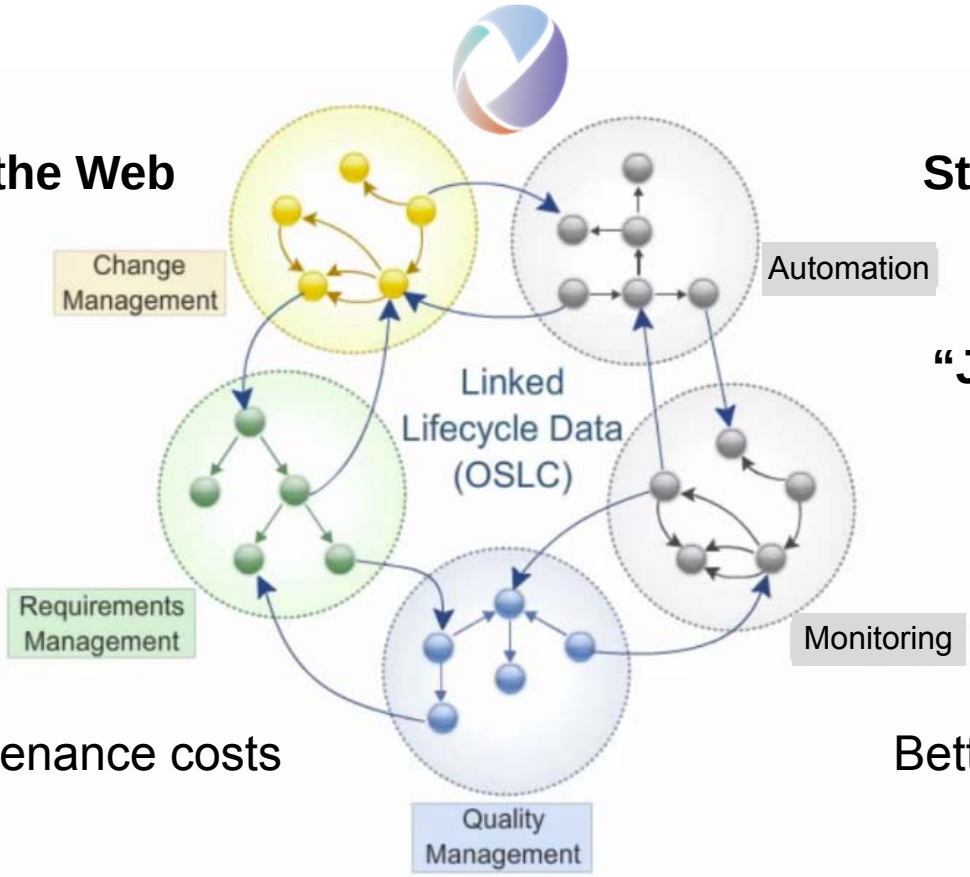
Decreased maintenance costs

Standard Interfaces *

"Just Enough" integration *

Increased traceability

Better visibility

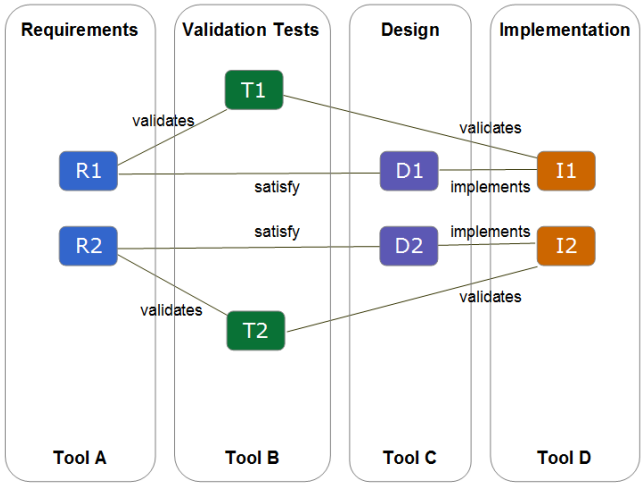


OSLC is an open and scalable approach to lifecycle integration. It simplifies key integration scenarios across heterogeneous tools

* Jad's highlights

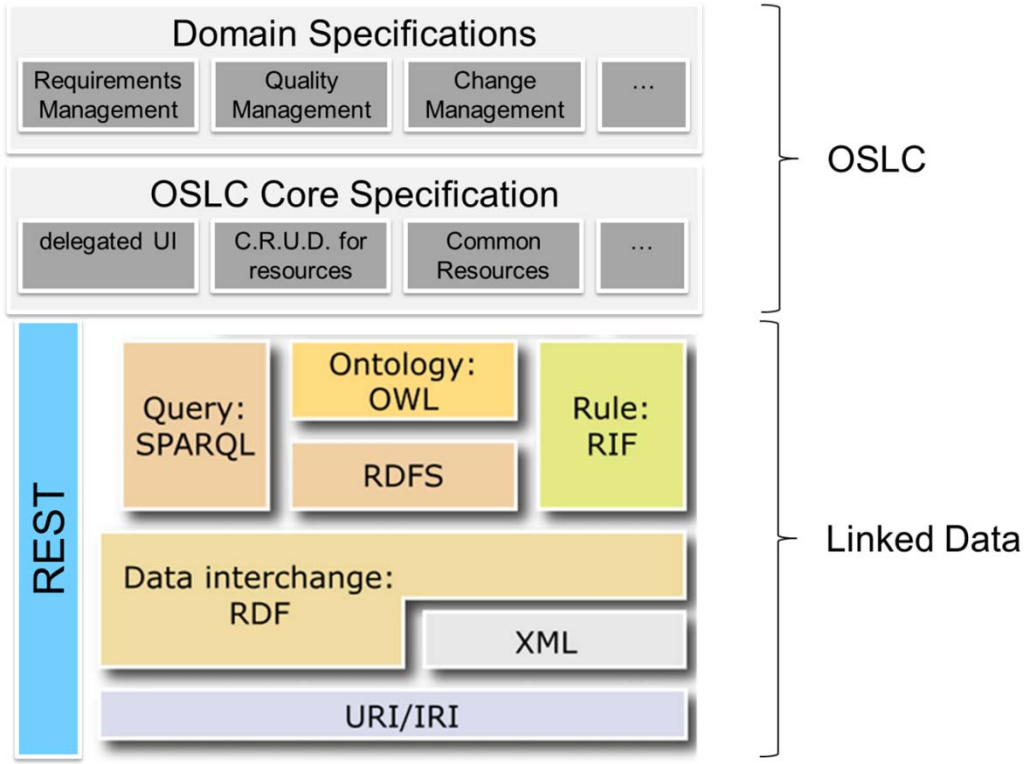
OSLC – relation to Linked Data?

- OSLC adopts the Linked Data principles
  - OSLC links lifecycle data



- OSLC adopts the RDF standards and its key concepts
  1. Graph data model
  2. URI-based vocabulary
  3. Serialization syntaxes
  4. …

- OSLC Contributes with
  - The standard rules and patterns for integrating lifecycle tools.
  - Common approach to perform resource creation, queries, …
  - Common resource properties
  - Domain specifications (vocabularies)
    - resource definitions for Lifecycle tools

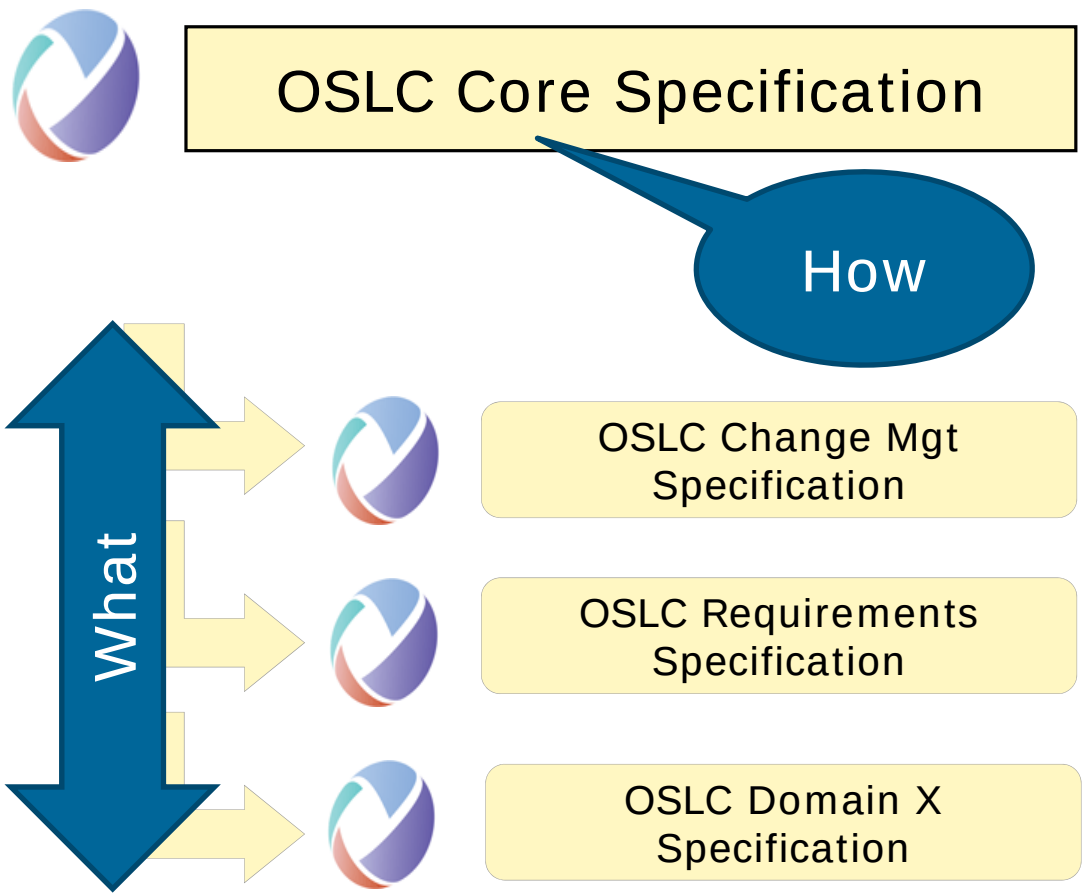


OSLC – relation to Linked Data?

Tim Berners-Lee's four principles applied to OSLC:

- Use URIs as names for things
  - In OSLC, each artifact in the lifecycle (for example, requirements, change requests, test cases...) is identified by a URI.
- Use HTTP URIs so that people can look up those names.
  - In OSLC, each artifact in the lifecycle is an HTTP resource. Standard HTTP methods (GET, PUT, POST, DELETE) are used to interact with them.
- When someone looks up a URI, provide useful information using the standards (RDF*, SPARQL)
  - Each OSLC resource has an RDF representation. OSLC resources can be queried using SPARQL.
- Include links to other URIs so that they can discover more things.
  - OSLC lifecycle artifacts are linked by relationships (for example, validatesRequirement or testedByTestCase) which are defined by URIs.

Anatomy of OSLC



Core: Specifies the primary integration techniques for integrating lifecycle tools – the standard rules and patterns for using HTTP and RDF that all the domain workgroups must adopt in their specifications

Domain:

1. Defines integration **scenarios** for a given lifecycle topic
2. Specifies a **common vocabulary** for the lifecycle artifacts needed to support the scenarios.

Example:

- The Core specification describes Delegated UIs and Creation Factories and states that OSLC service providers MAY provide them.
- The Change Management specification states that CM service providers MUST provide them.

What's next

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - Domain specification(s)
    - Requirement Management – an example

First, What is a tool? (from an integration perspective)

OSLC Service Provider catalog

provides

OSLC Service Provider

provides an implementation of

OSLC Service

manages

OSLC Resource

- Allows for the discovery of the service provider set(s).
- They help to simplify the configuration of tools (ex. OAuthConfiguration).

example: IBM Rational Team Concert

The central organizing concept of OSLC.

- Reflects the tool's containers or partitions
- Enables tools to expose resources
- Provides access to services (enabling consumers to navigate resources, and create new ones)

example: project, module, …

Set of capabilities that enable a web client to create, retrieve, update and delete resources

example: Change Management capability

Managed by an OSLC Service, may have properties and may link to other resources including those provided by other OSLC Services.

example: work item (bug, defect, enhancement request)

OSLC defines the following technical areas:

1. Discovery of capabilities

2. HTTP C.R.U.D. for resources

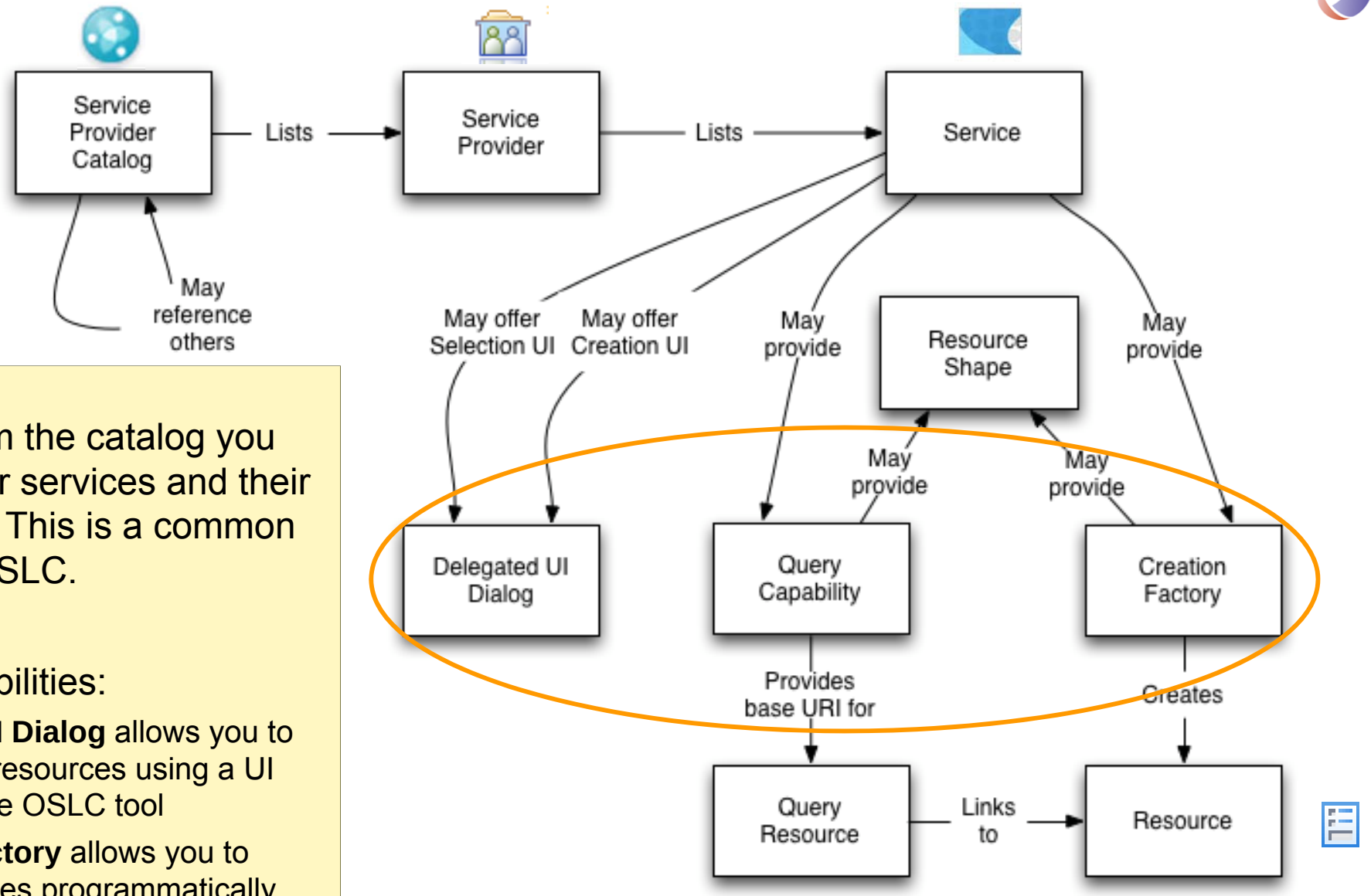
3. Standard resource representations

4. Querying for resources

5. Delegated UI for Create and Select

6. UI Previews for Resource Links

1. Discovery of capabilities



Starting from the catalog you can discover services and their capabilities. This is a common pattern in OSLC.

OSLC capabilities:

- **Delegated UI Dialog** allows you to create or find resources using a UI provided by the OSLC tool
- **Creation Factory** allows you to create resources programmatically
- **Query Capability** allows you to query for resources

2. HTTP C.R.U.D

OSLC allows manipulation of resources using standard HTTP C.R.U.D

Create = POST

Request = GET

Update = PUT

Delete = DELETE

→ OSLC follows the REST architectural pattern.

The REST Architectural Pattern

- Is a software architecture style for web services.
  - a simpler alternative to SOAP and WSDL-based Web services
- The primary purpose of a RESTful service is to manipulate representations of Web resources using a uniform set of stateless operations.
- The design pattern for REST interfaces
  - Interface with external systems using resources identified by URIs, for example '/person/paul'
  - A resource can be operated upon using standard HTTP verbs (GET, POST, PUT, DELETE).

RESTful API HTTP methods

Resource	GET	PUT	POST	DELETE
Element URI, such as `http://api.example.com/v1/resources/item17`	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it does not exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own right and **create** a new entry in it.[10]	**Delete** the addressed member of the collection.

- Architectural constraints
  - Client-server
    - Servers and clients may be replaced/developed independently.
  - Stateless
    - no client context being stored on the server between requests.
    - session state is held in the client
  - …

Source: http://en.wikipedia.org/wiki/Representational_state_transfer

2. HTTP C.R.U.D
- Resource Retrieval (Request)

- Use HTTP GET and standard HTTP content negotiation
  - Client uses HTTP Accept request header to specify desired resource formats

```
Accept: application/json, application/xml
```

- Use standard content(MIME) types

- Partial representations can be requested via HTTP URL key=value pair as `?oslc.properties=`
  - Allows for minimal retrieval of properties
  - Get Defect 123 (all properties)

```
GET http://bugs/123
```

  - Get Defect 123 (just title and status)

```
GET http://bugs/123?oslc.properties=dcterms:title,oslc_cm:status
```

2. HTTP C.R.U.D
- Resource Creation (Create)

Create a resource using HTTP POST, with the resource body in format of choice

- URI for doing the POST is defined in the `oslc:ServiceProvider` in the `oslc:creationFactory` service

Response is a `201-Created` with `Location` HTTP header indicating URI for resource

Request may be rejected for any number of reasons

- Insufficient permissions
- Missing required values
- Invalid data choices
- ...and … and ...

Valid resource formats for creation are defined by:

- domain specifications
- service provider may define its own resources and formats
- optionally, by resource shape associated with creation factory

1. Use HTTP GET to get resource properties to be updated
   - You'll get an ETag back
2. Change only the property values you need to change
   - Clients must preserve unknown content
3. Use HTTP PUT to send updated resource
   - Use If-Match HTTP request header with ETag, services may reject your request without it
   - HTTP PUT will completely replace the resource representation
   - We are moving towards PATCH – new HTTP verb http://tools.ietf.org/html/rfc5789

- It is possible to update only selected properties

Use HTTP DELETE on the resource identifier

May not be allowed

Response usually:

- `200-OK`
- `204-No-Content`
- `400-Bad-Request`
- `403-Forbidden`

3. Resource representations

OSLC services should handle any type of resource

- Not just those defined by OSLC

Resources defined by OSLC use RDF data model

- therefore are simply defined by their set of properties

OSLC services MUST produce and consume RDF/XML representations

- Clients and services MUST NOT assume any subset of RDF/XML

Other representations are allowed such as:

- XML: OSLC defined format that allows for consistent formats and is RDF/XML valid
- JSON: Rules for representing namespaces and QName properties
- Turtle: No constraints, use as is (may be preferred by future specs)
- Atom Syndication Format: <atom:content> SHOULD be RDF/XML

Links are properties where the property values are URIs

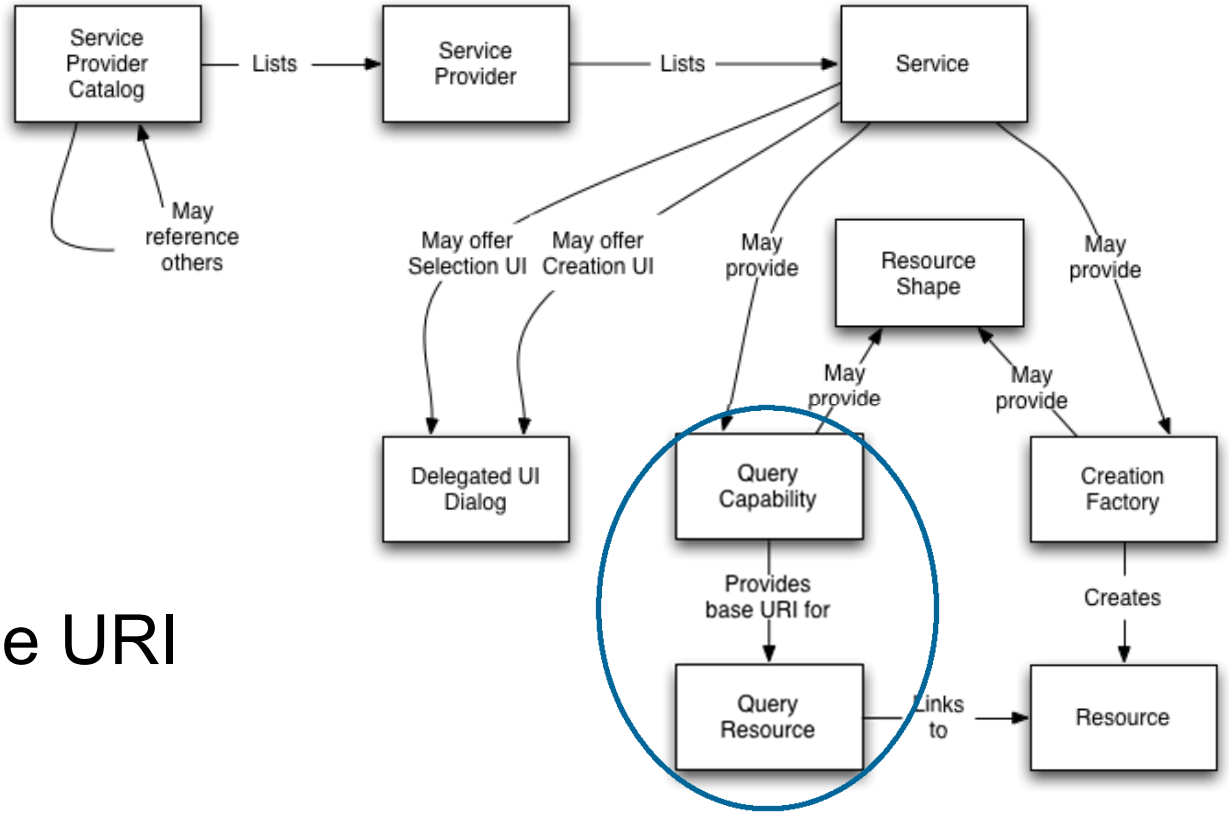
Turtle format for a bug resource (abbreviated)

```
<http://example.com/bugs/2314>
    a oslc_cm:ChangeRequest ;
    dcterms:relation
<http://server/app/bugs/1235> ;
```

Don't make assumptions about the target of links

- OSLC supports an open model
- Needed to achieve goal of “loosely coupled” integrations
- Clients need to be flexible and expect anything

4. Querying for resources



Query capability has base URI

Clients form query URI and HTTP GET the results

OSLC services MAY support OSLC Query Syntax

- http://open-services.net/bin/view/Main/OSLCCoreSpecQuery

Query syntax overview

- Filter results by appending "`oslc.where=`" with query clause to query base URI

- Only boolean operation allowed is "`and`" which represents conjunction
  - "`or`" for disjunction is not defined in the interests of keeping the syntax simple.

- Retrieve just what you want with "`oslc.select=`"
- Defined ordering using "`oslc.orderBy=`"
- Full-text search via "`oslc.searchTerms=`"

'`in`' operator:

Test for equality to any of the values in a list. The list is a comma-separated sequence of values, enclosed in square brackets: in ["high","critical"]

Comparison Operators

=	test for equality
!=	test for inequality
<	test less-than
>	test greater-than
<=	test less-than or equal
>=	test greater-than or equal

Query syntax example

Find high severity bugs created after April fools day

```
http://example.com/bugs?oslc.where=
  cm:severity="high" and dcterms:created>"2010-04-01"
```

Find bugs related to test case 31459

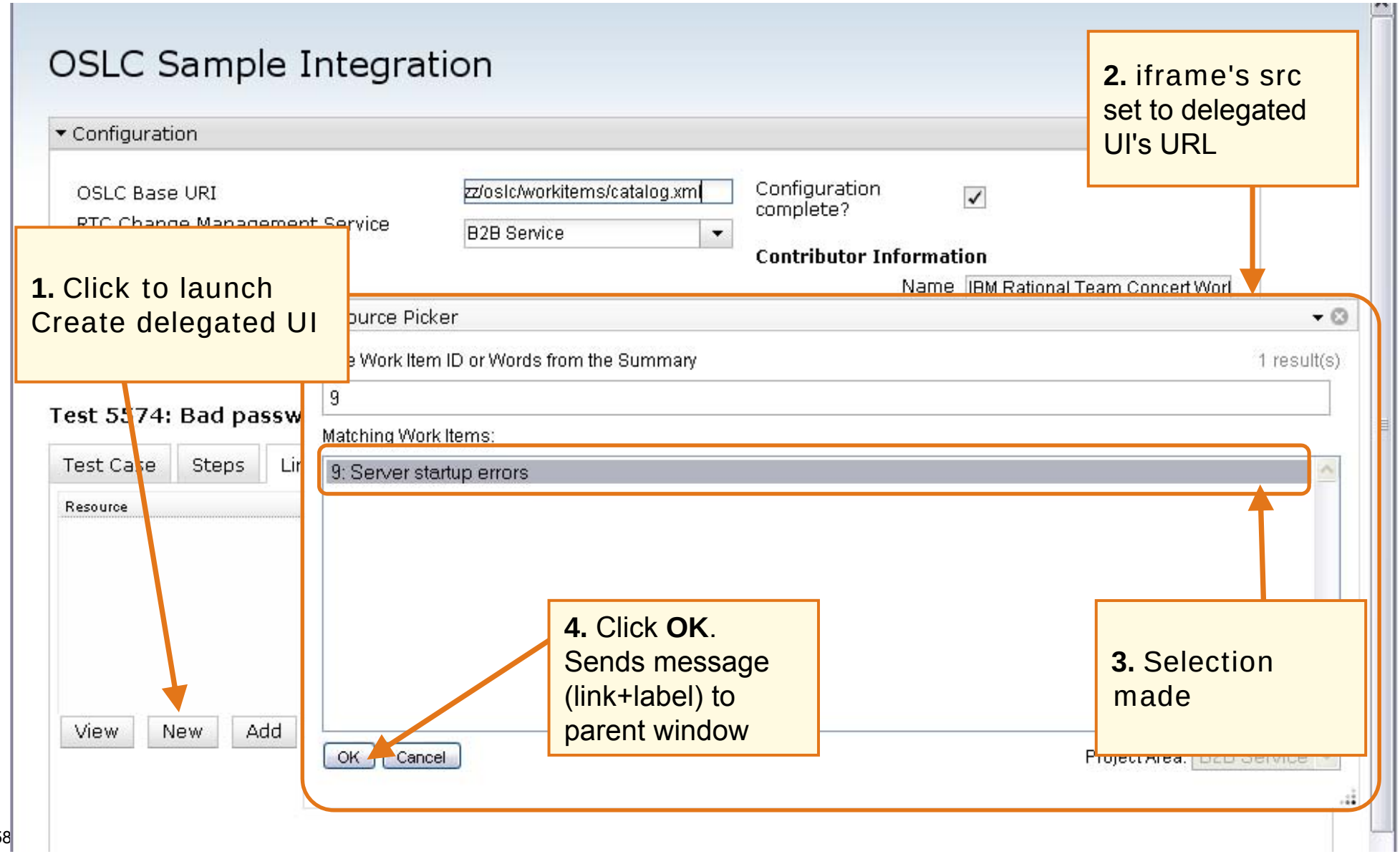
```
http://example.com/bugs?oslc.prefix=qm=
  <http://qm.example.com/ns>&
  oslc.where=qm:testcase=<http://example.com/tests/31459>
```

Find all bugs created by John Smith

```
http://example.com/bugs?oslc.where=
  dcterms:creator{
  foaf:givenName="John" and foaf:familyName="Smith"}
```

5. Delegated UI for Create and Select

Delegated UI - renders the source application UI in the target application.



Delegated UI key points

Delegated UIs support both creation and selection of resources

Two communication protocols are supported for iframes:

- HTML5 postMessage() ← preferred method
  - Supported in most modern browers
- Window object's window.name
  - Supported in older browsers and Eclipse embedded web widget
- Consumer selects which protocol to use, informs provider via fragment identifier

Tremendous value for resource creation

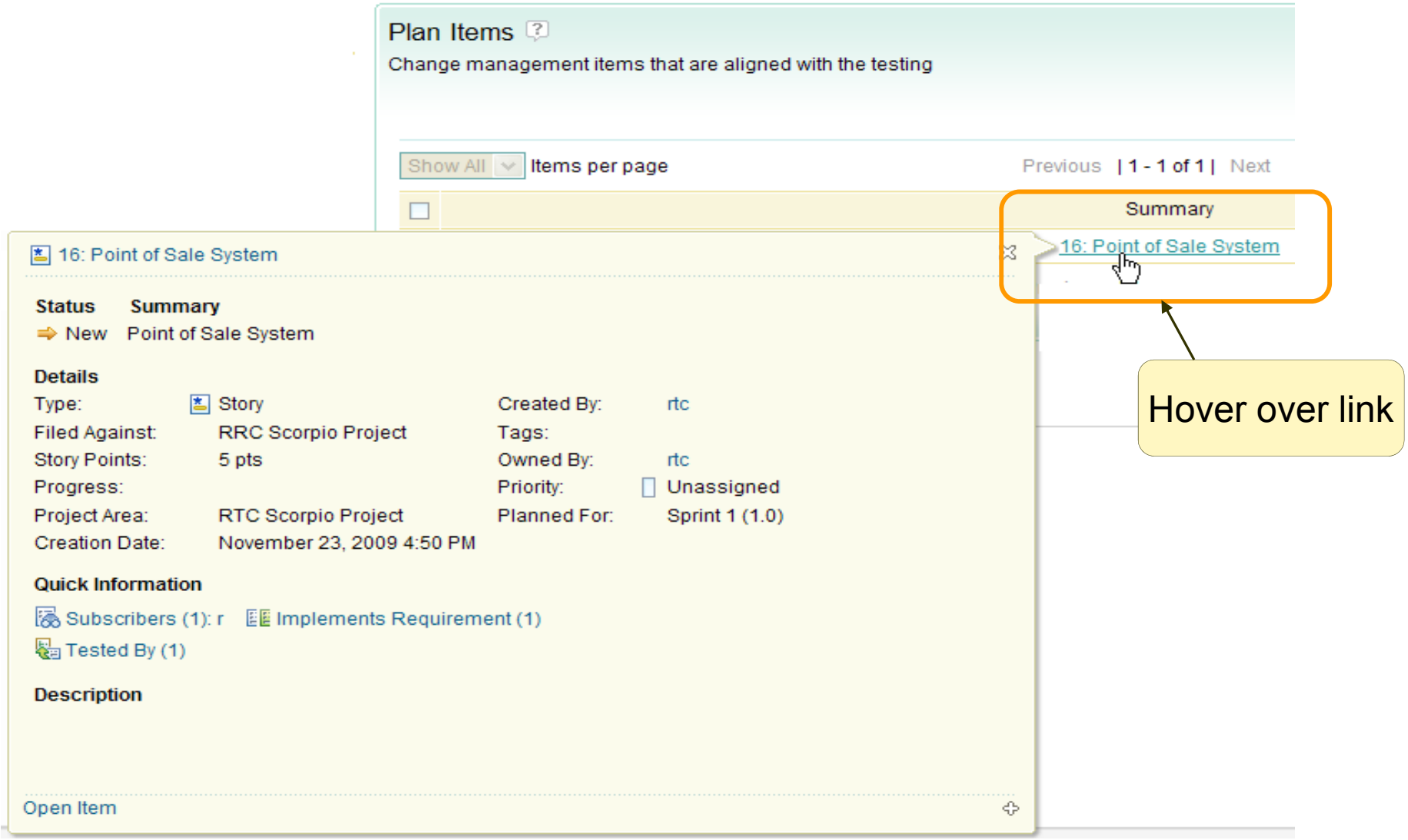
- Traditionally most service logic was communicated to client and new dialog built
- Now the rules for creation and dialog change as needed

Prefilling of creation dialog done by “creating” a dialog resource

- HTTP POST of resource format to creation dialog URL, response is URL of dialog prefilled

6. UI Preview

Scenario supported: hover over link to get in context preview of resource



What's next

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - Domain specification(s)
    - Requirement Management – an example

Domain specification(s)

OSLC Specifications Cover Many Domains

- Architecture Management
- Asset Management
- Automation
- Change Management
- Configuration Management
- Quality Management
- Requirements Management
- …

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

Requirements Management

http://open-services.net/bin/view/Main/RmSpecificationV2

Defining a Resource & its Properties

Resource Requirement

- **Name:** `Requirement`
- **Type URI** `http://open-services.net/ns/rm#Requirement`

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
OSLC Core: Common Properties						
dcterms:title	exactly-one	unspecified	XMLLiteral	n/a	n/a	Title (reference: Dublin Core) of the resource include only content that is valid inside an XHT
dcterms:description	zero-or-one	unspecified	XMLLiteral	n/a	n/a	Descriptive text (reference: Dublin Core) abou **SHOULD** include only content that is valid and
dcterms:subject	zero-or-many	False	String	n/a	n/a	Tag or keyword for a resource. Each occurre tag for the resource.
dcterms:creator	zero-or-many	unspecified	Resource or Local Resource	Either Reference or Inline	any	Creator(s) of resource (reference: Dublin Cor `foaf:Person` but that is not necessarily the c
dcterms:contributor	zero-or-many	unspecified	Resource or Local Resource	Either Reference or Inline	any	Contributor(s) to resource (reference: Dublin `foaf:Person` but that is not necessarily the c
dcterms:created	zero-or-one	True	DateTime	n/a	n/a	Timestamp of resource creation (reference: D
Prefixed Name	**Occurs**	**Read-only**	**Value-type**	**Represen-tation**	**Range**	**Description**
Relationship properties: This grouping of properties are used to identify relationships between resources managed by other OSLC						
oslc_rm:elaboratedBy	zero-or-many	False	Resource	Reference	any	The subject is elaborated by the object. For ex
oslc_rm:elaborates	zero-or-many	False	Resource	Reference	any	The object is elaborated by the subject.
oslc_rm:specifiedBy	zero-or-many	False	Resource	Reference	any	The subject is specified by the object. For exa

What's next

- The OSLC approach
- Linked Data and RDF
- The OSLC standard
  - Core specification
  - domain specification(s)
    - Requirement Management – an example

… the OSLC Hands-on Tutorial