JEK1

33rd Annual INCOSE international symposium

hybrid event

Honolulu, HI, USA
July 15 - 20, 2023

Tutorial: Digital threads with OSLC

Erik Herzog – Technical Fellow, SAAB Aeronautics
Eran Gery – Global Systems Engineering Solutions Lead, IBM

Contributors: Jad El-khoury, LynxWorks
Ian Green, IBM

Slide 1

JEK0 1. Can we list (a) Speakers (b) contributors/authors to make it clear.
1. My name is Jad El-khoury 😊
2. Should we put our affiliations?
Jad El-Khoury, 2023-06-12T11:19:20.767

JEK1 Overall comment:

The slides seem to have different formats, layout, fonts. This can be confusing/irritating to the reader. Can we make more consistent?
Jad El-Khoury, 2023-06-12T11:49:49.499

JEK0

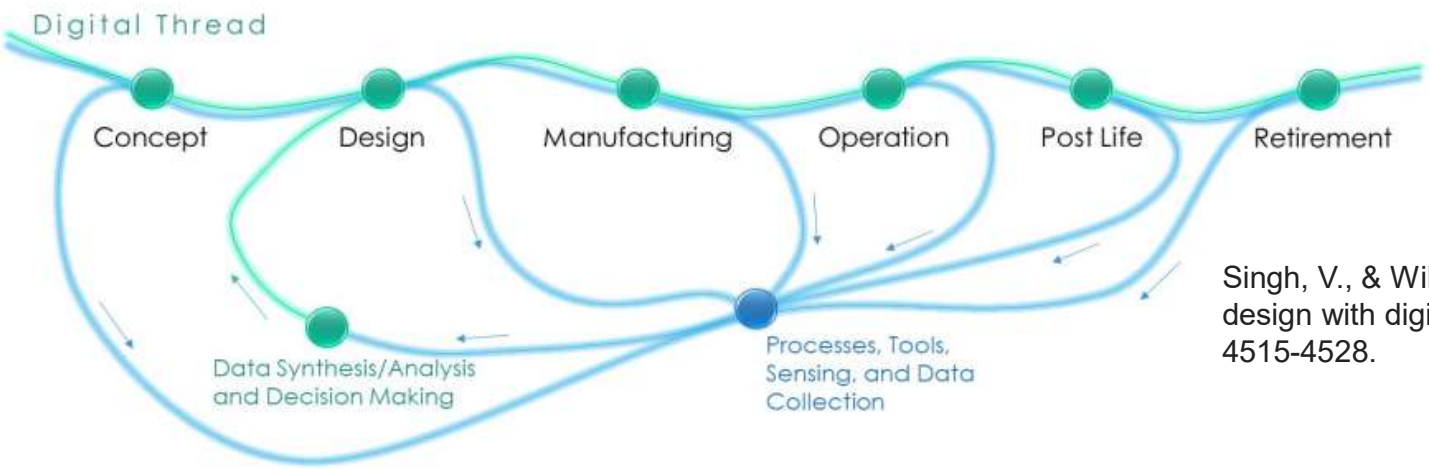
Agenda

1. **OSLC goals and digital threads use cases**
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

Slide 2

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

Digital thread



Singh, V., & Willcox, K. E. (2018). Engineering design with digital thread. *AIAA Journal*, *56*(11), 4515-4528.

The use of digital tools and representations for design, evaluation, and life cycle management

- Digital
- Multi-disciplinary
- Fine-granular
- Linked
- Analytics
- Authoritative

Saab Aeronautics as an example of

End-user needs

Saab Aeronautics – the old game

1950 1970 1990

- One customer
- One operations approach – national defence
- One project at a time
- Long development times
- Predictability: Sweden and Saab

The new Game

2000 2020

- Multiple parallel projects
- International operations and interoperability
- Exports
- International collaboration
  - Multi-site Development & Production
- More stringent international regulations
- Speed!
  - Product development
  - Enabling systems
- Unpredictable future

System characteristics

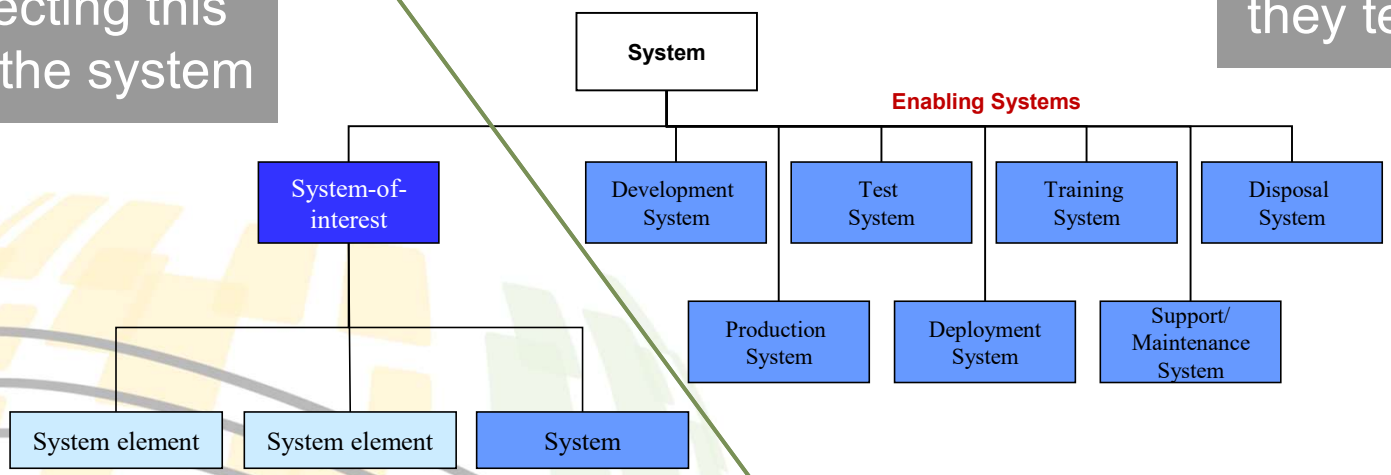
- Long lifecycles – yes, indeed!
- Safety critical systems
- Continuous development
- Development system life is **shorter** than System life
- Historical observation
  - Need to replace development system **trice** over the life of the system

Early concept (5 years) → Full-scale development (basic platform) (10 years) → Fully operational capabilities (10 years) → Upgraded capabilities (20 years) → Maintenance and support (10 years) → Disposal (5 years)

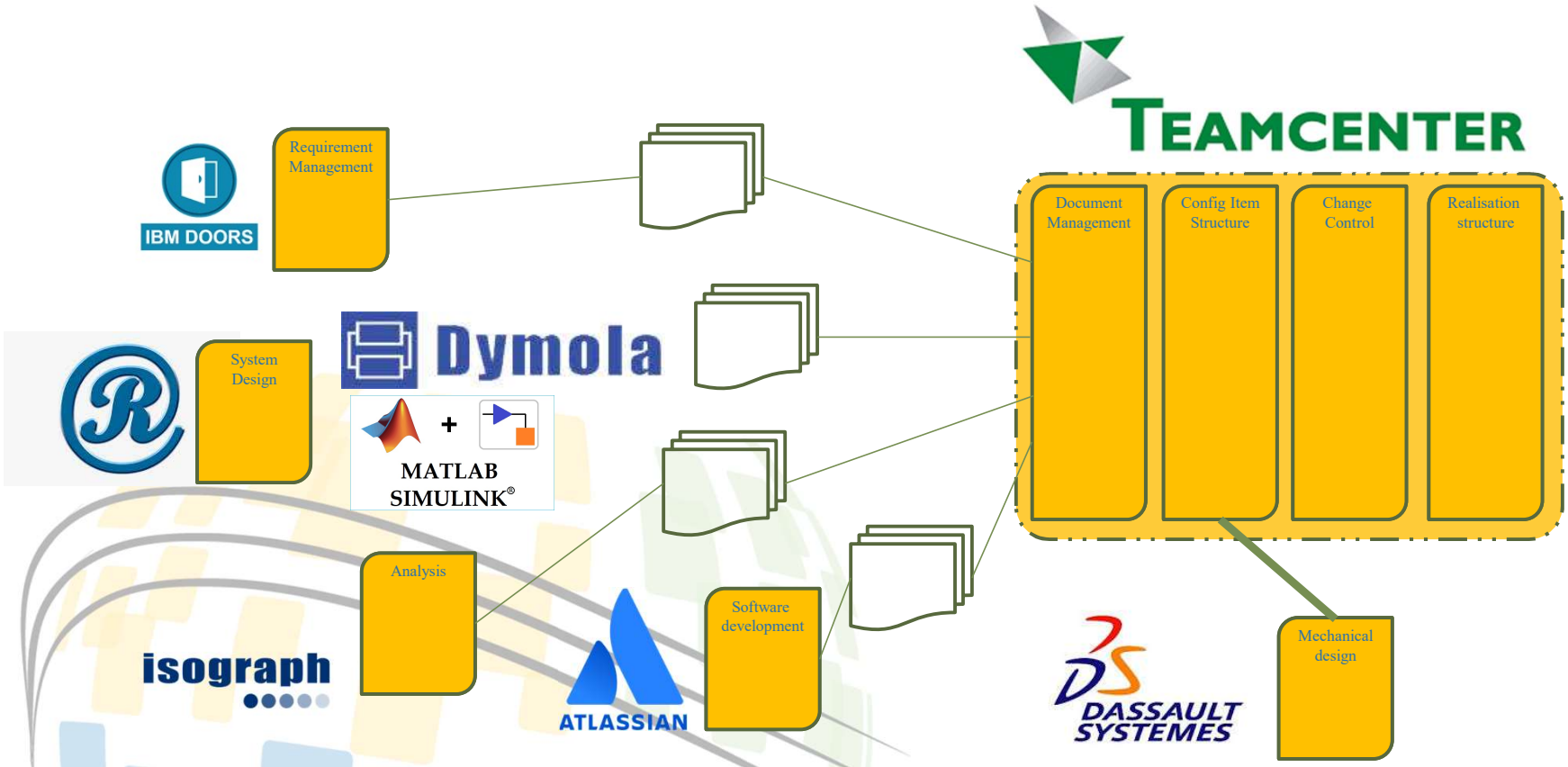
A look at the enabling systems

We are good at architecting this part of the system

Less attention on Enabling systems – they tend to emerge



SAAB as-is development system landscape



Architecting the development system

New strategic directions for thriving in the new unpredictable world:

- Alignment with best international practise
- Need to architect organisation and development environment for **Flexibility**
  - Optimise overall capability
  - Ability to adapt the latest processes, methodology and tools
- Capability to maintain **multiple versions** and **variants** of product data
- Keep the development system **relevant** over time
- Quick adaptation to new scenarios
  - At low cost

Criteria for a development system

- Digital
  - Fine granular objects and relationships
- Integrated
  - Integrated support for multiple disciplines and roles
- Heterogeneous
  - Provided by multiple vendors
- Modular
  - Each component provide self-sufficient services
- User-friendly
  - Easy to use across integration boundaries
- Affordable
  - Integration of new components off the shelf or att very low investment cost
- Maintainable
  - Ability to stand the test of time as individual components evolve

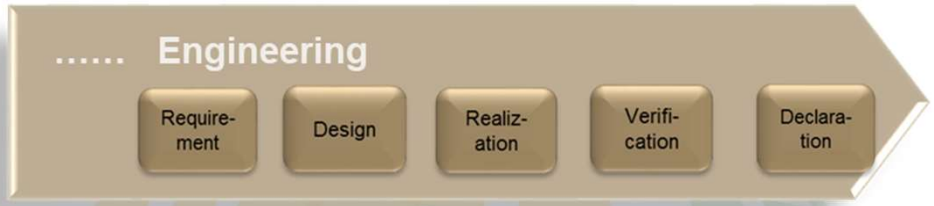
Additional criteria?

An Example

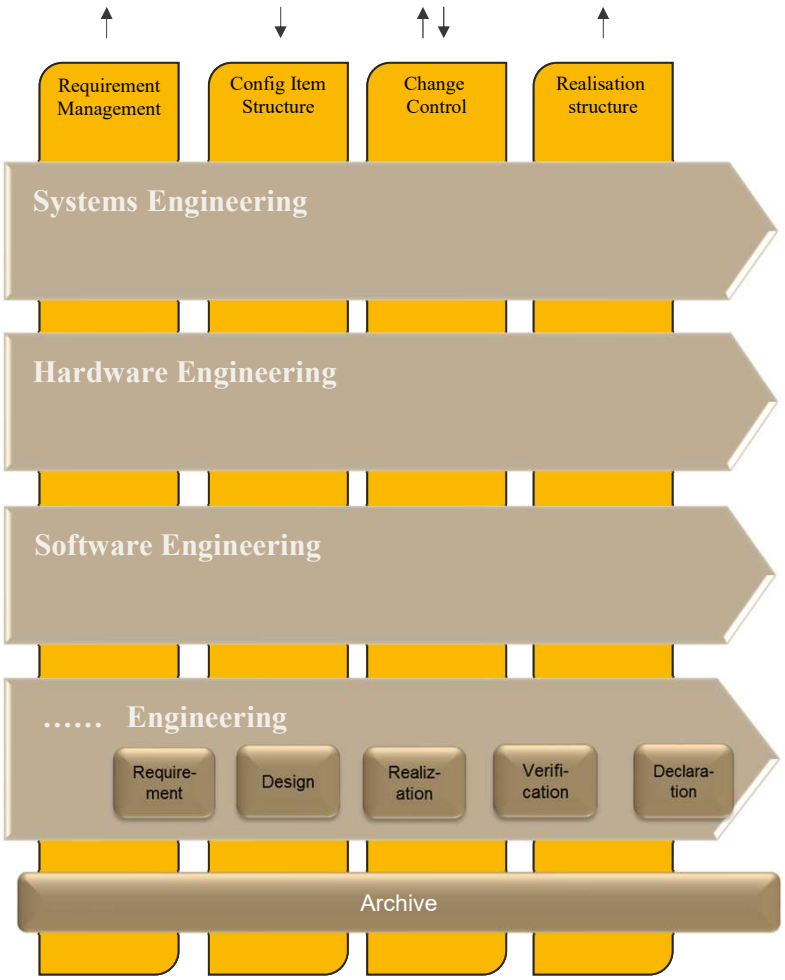
The Saab Genesis architecture

Genesis PLM Model

- Engineering Disciplines
- Fine granularity product data



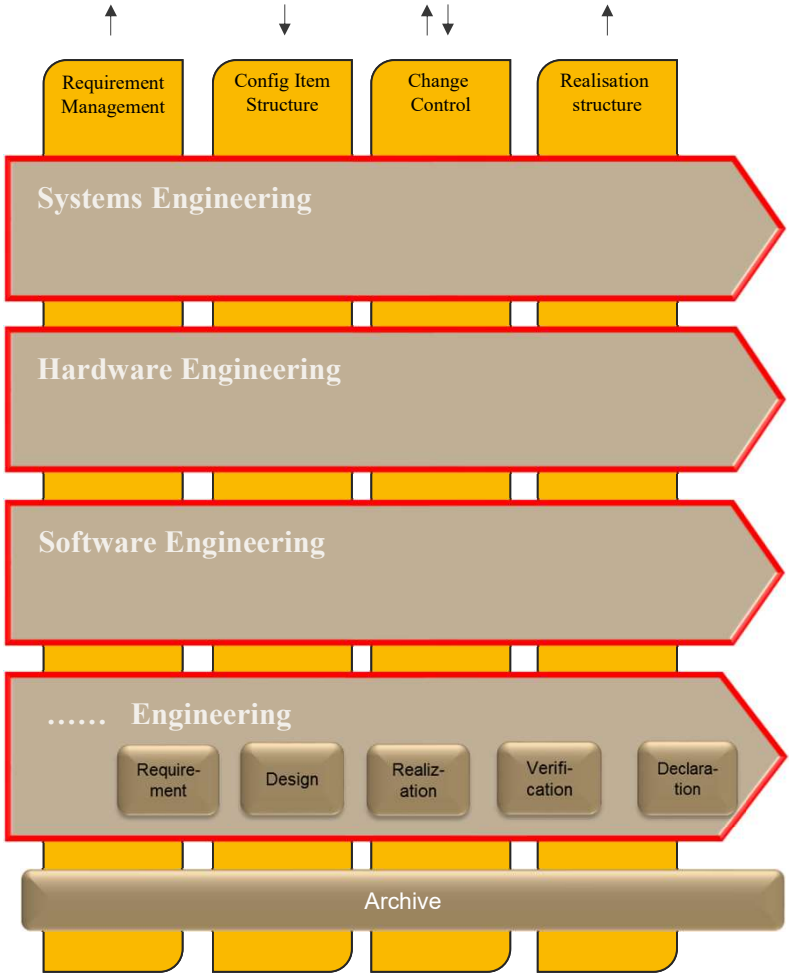
- Design Traceability Dimensions
  - We believe there are four of them only
- Archiving



SAAB

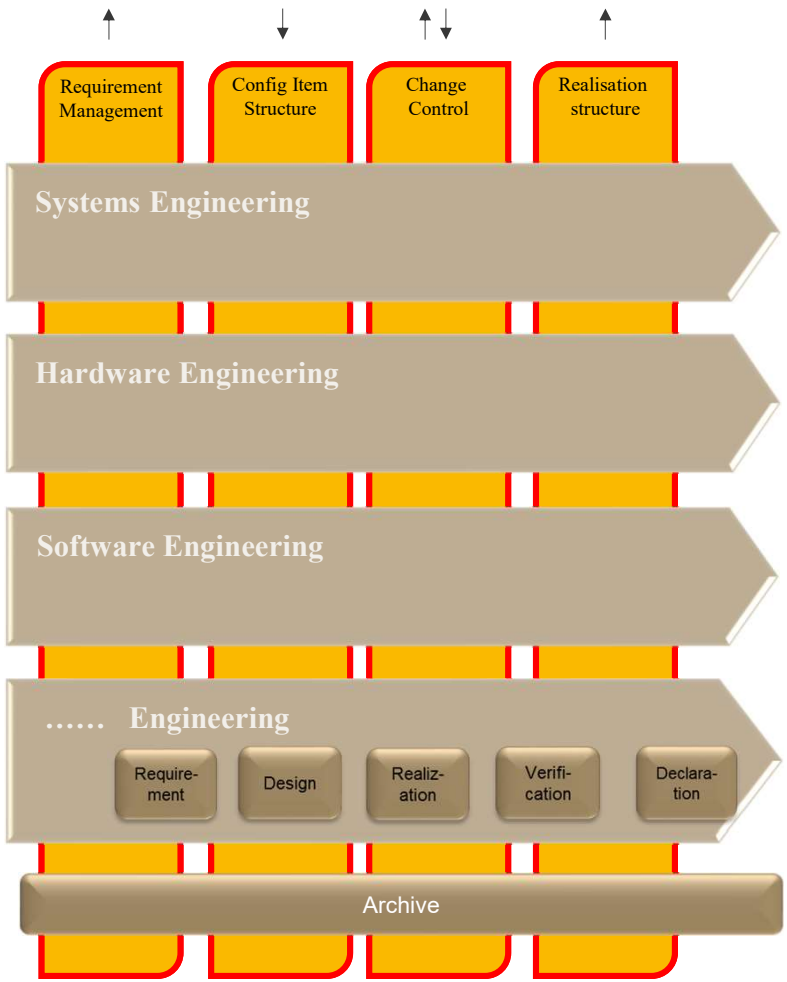
Modularity

- Optimise support for each engineering discipline
  - Maximise automation, as provided by the supplier
  - Minimise application family switching
- Bring together management and engineers in a single environment
  - E.g., Change management and Status reporting
- Ability to upgrade individual capabilities independent of others
- Redundant capabilities accepted
- Ability to replace environment without upsetting the complete PLM landscape



Traceability

- Need capability to ensure traceability and integrity of product data
- Traceability dimensions between engineering discipline environments
  - Requirements
  - Configuration item structure
  - Change management
  - Realization
- Configuration Management capability required for Requirements Traceability, Configuration item structure and Realization structure
  - Versions and baseline capability
- The OSLC standard offers the desired capabilities
  - Exploit for low cost and high quality integrations



The Tutorial question:

Does OSLC meet your needs?

Prevailing situation and solution alternatives

The challenge – sustainining multidisciplinary complex engineering environments

- Engineering data is siloed across teams and applications (ASOTs)
- How to ensure
  - Continuity and consistency of data across related artifacts?
  - Proper assessment of impact and manage changes across all ASOTs?
  - Gaining the right insights to conduct engineering assessments by enabling joint digital viewpoints?
  - Management of configurations baselines and branches across all datasets?
  - effective collaboration across all stakeholders to foster agile engineering?

Few key engineering questions…

How do I make sure that my design satisfies all the sustainability requirements?

What is the impact of modifying this requirement across all disciplines?

How can I reuse design components from another product?

Does the operational data align with my design assumptions?

Is my electrical design baseline consistent with my software baseline?

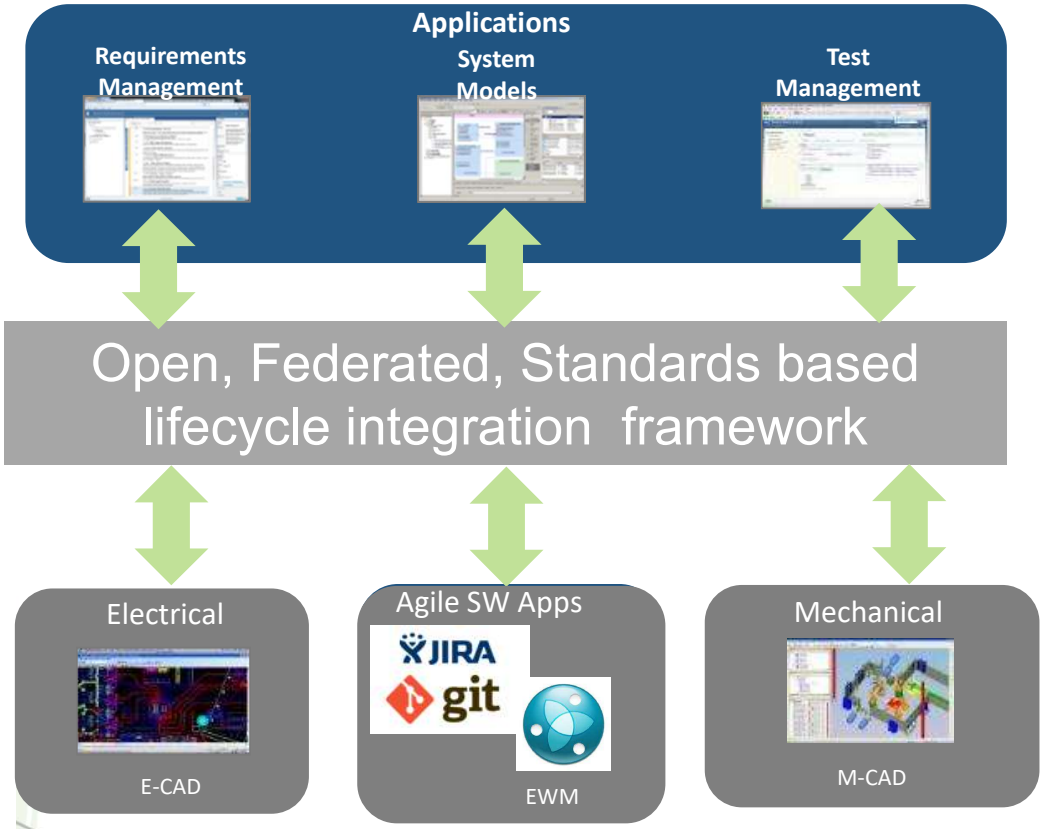
What is the overall energy consumption across all viewpoints?

How do I prove that my test plan covers all the requirements?

JEK0

Approaches for attaining the digital engineering vision

- Point to point integrations
  - Does not scale, no support for central view points
- Centralized - all domain data synchronized to a central repository
  - Often challenges to authoritative source concept, domain tools isolated
  - Proprietary – limited tool selection
- MBSE backbone - Import various domain data to an MBSE tool
  - Replication of data of authoritative tools
  - Does not scale
- Centralized Link management – central link repository
  - Domain tools not aware of links; versioning issues
- Linked federated data (OSLC approach)
  - No replications of data
  - Collaborative and Standardized
  - Supports digital viewpoints (as defined by DEIXWG)



JEK1

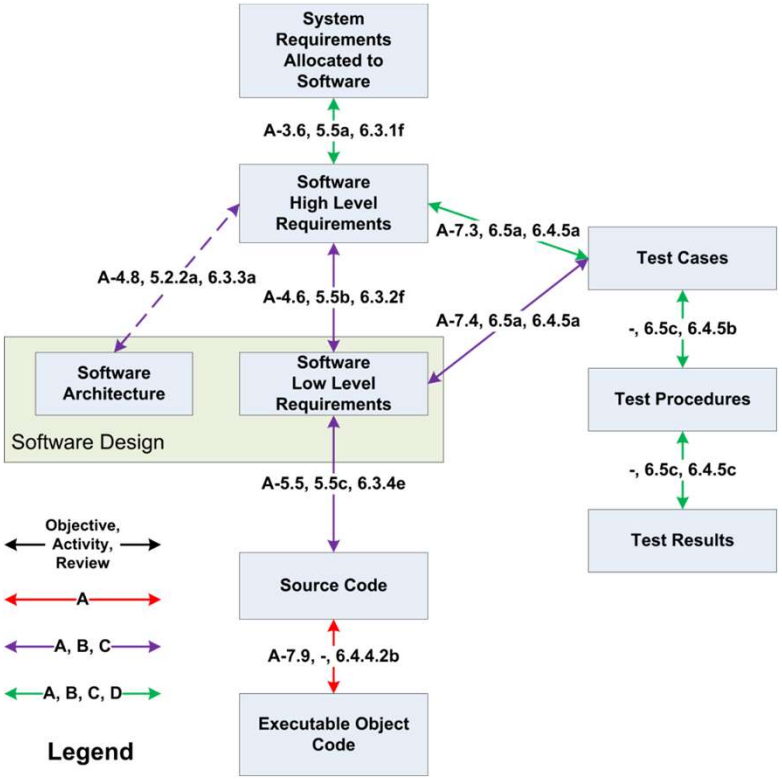
Slide 21

JEK0 What is ASOT? Spell it out?
Jad El-Khoury, 2023-06-12T11:20:59.421

JEK1 "Open Federated Standards based lifecycle integration" Does not read so well.
Why not call it "Federated Linked Data" like you do in the text?
Jad El-Khoury, 2023-06-12T11:22:04.178

Key digital threads enabler capabilities

- *Digital continuity*: establish digital information models based on standard resource types and relationships across all domains tools
- Enable *cross domain data exchange* through standard data representations
- *Global configuration management*: manage consistency across all engineering data sources using cross tools configuration management
- *Cross lifecycle analytics and viewpoints:* produce the necessary insights and evidence from across all domain tools
- *Integrated change and process management* across all engineering data and tools



information model required by DO178 DALs

JEK0

Agenda

1. OSLC goals and digital threads use cases
2. **OSLC foundations: W3C linked data**
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

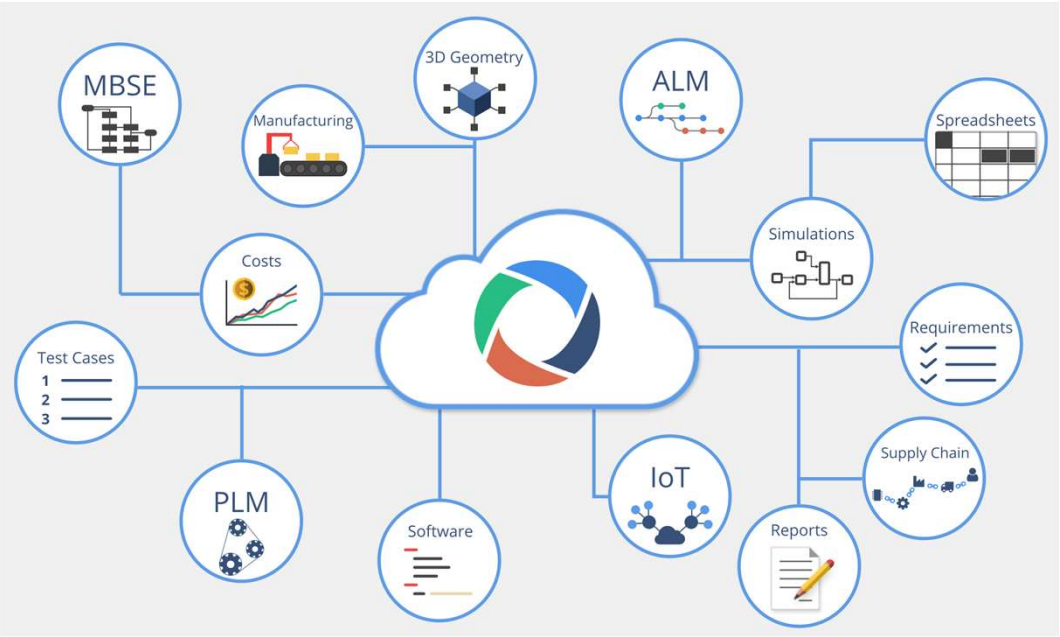
Slide 23

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

Open Services for lifecycle (OSLC)

A lifecycle integration framework based on open data model and services standards across a federated set of tools

- Specifies standard lifecycle information models based on W3C ontologies
  - Standard resource representation
  - Linking across resources
- Open world assumption: minimal assumptions on data models and services
  - Enable discovery
- Enables collaboration across tools based on standard REST services for complete modularity
- Enable integration of existing tools with no assumption of how they are implemented
- Enable services for cross lifecycle viewpoints and analytics



OASIS

OSLC Member Section

http://oasis-oslc.org

JEK0

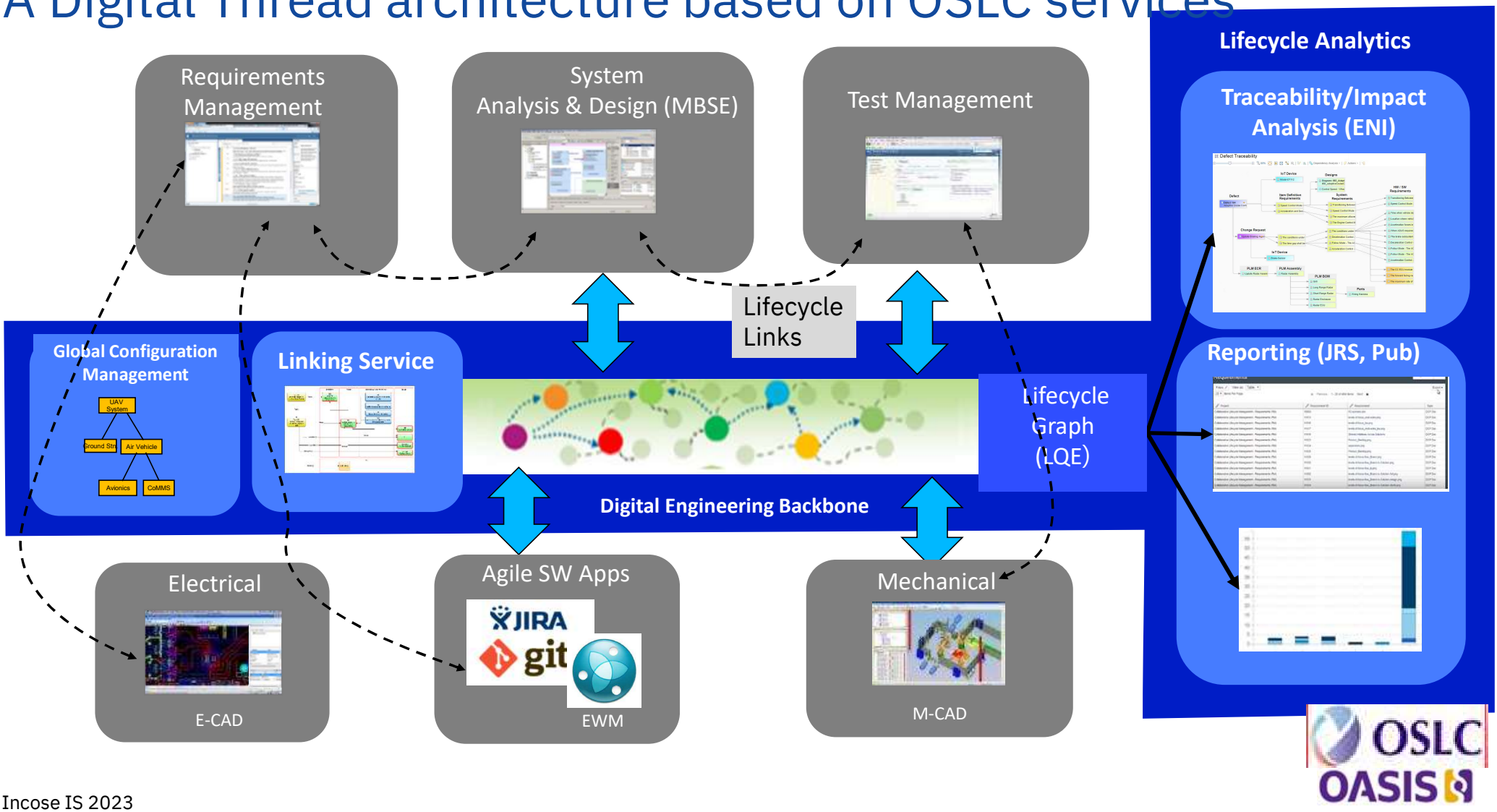
Who currently specifies OSLC at OASIS

Slide 25

JEK0 We Should be very careful if we wan to say OASIS in the title. This is not a formel list.

If it is informal, it would be nice to add KTH and Lynxwork as well? 😉
Jad El-Khoury, 2023-06-12T11:43:13.527

A Digital Thread architecture based on OSLC services



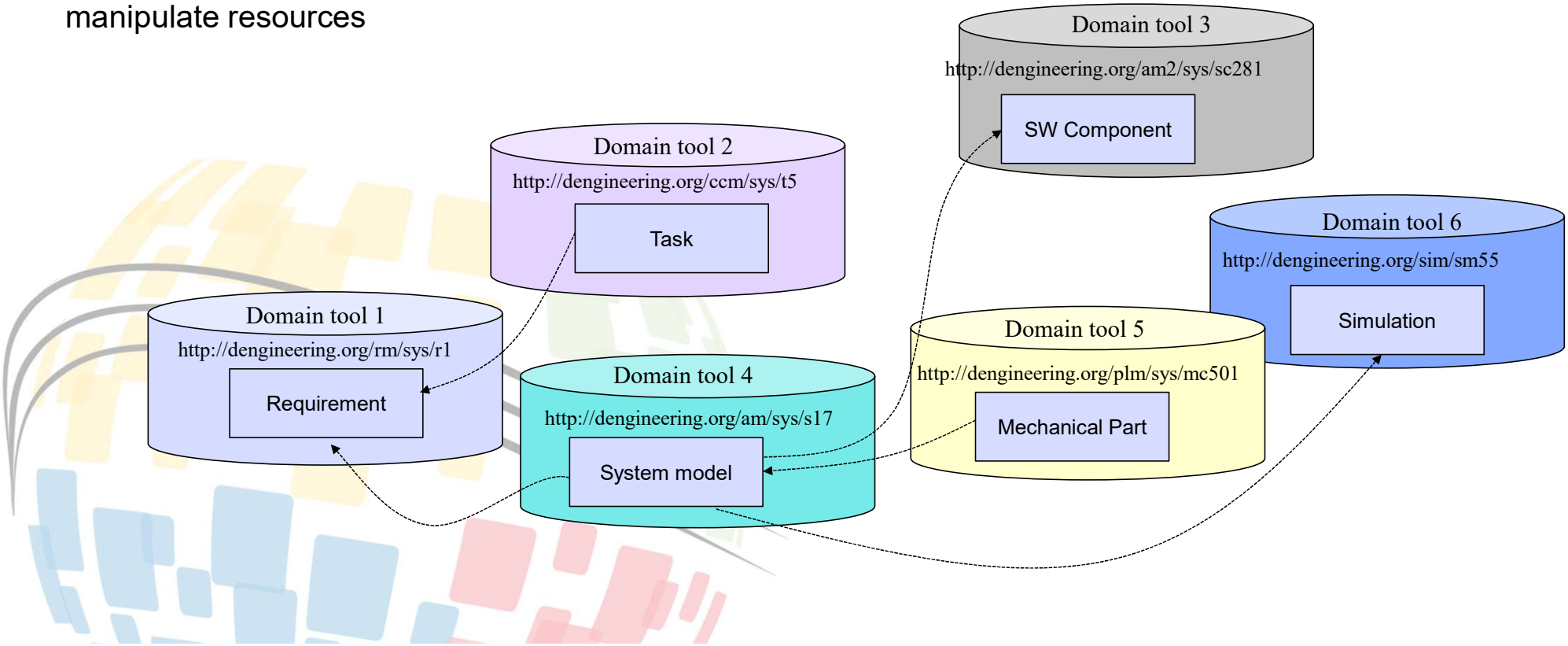
Incose IS 2023

JEK0

Linked data (w3c)

JEK1

- Lifecycle objects (resources) are identified by http URLs and described using vocabularies (ontologies)
- Enables lifecycle information models with relationships across all resources independent of their containers
- Data containers provide HJEK2 services to link access and manipulate resources



Slide 27

JEK0 URLS should be URLs (can also remove "http" before that.
Jad El-Khoury, 2023-06-12T11:22:58.815

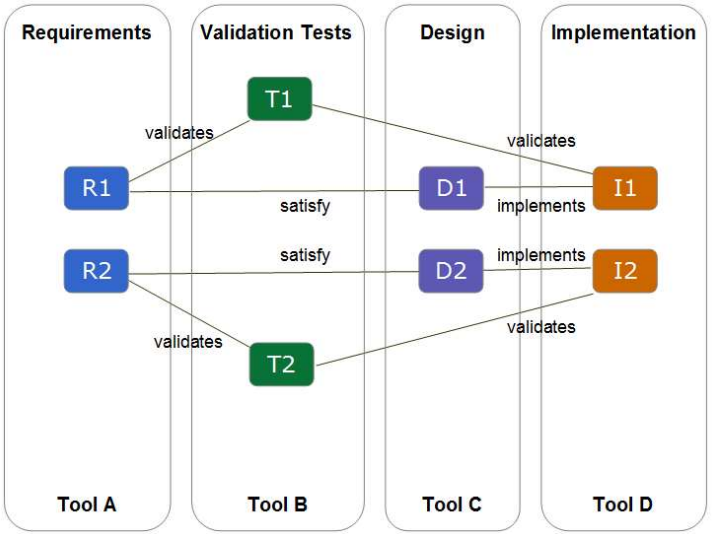
JEK1 Instead of "specified using vocabularies" it should be "described using vocabularies"
Jad El-Khoury, 2023-06-12T11:23:54.150

JEK2 "Best on http/REST architecture" what is that meant to mean?
Jad El-Khoury, 2023-06-12T11:24:44.501

Linked Data Principles

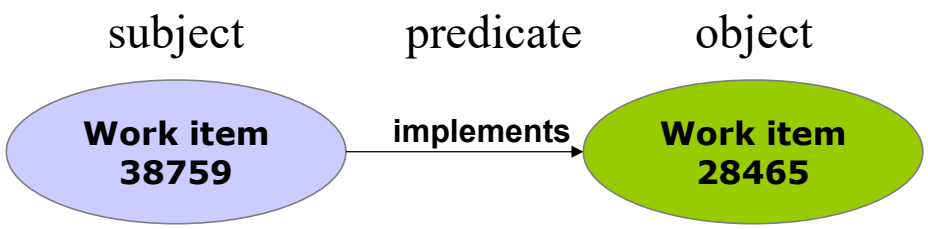
- Tim Berners-Lee's four principles for Linking Data:

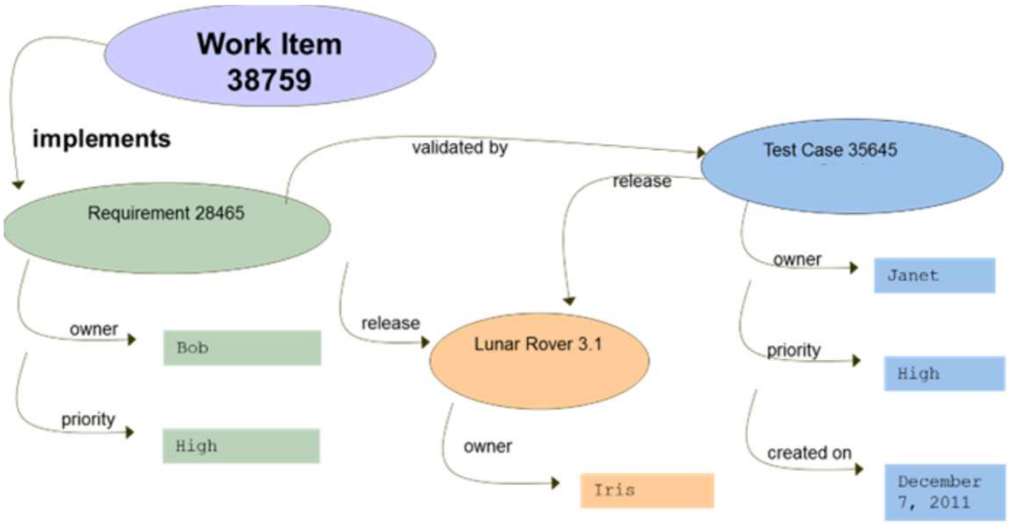
  1. Use URIs as names (*identity*) for things
  2. Use HTTP URIs so that people can look up those names
  3. When someone looks up a URI, provide useful information using the standards (RDF, SPARQL)
  4. Include links to other URIs so that they can discover more things



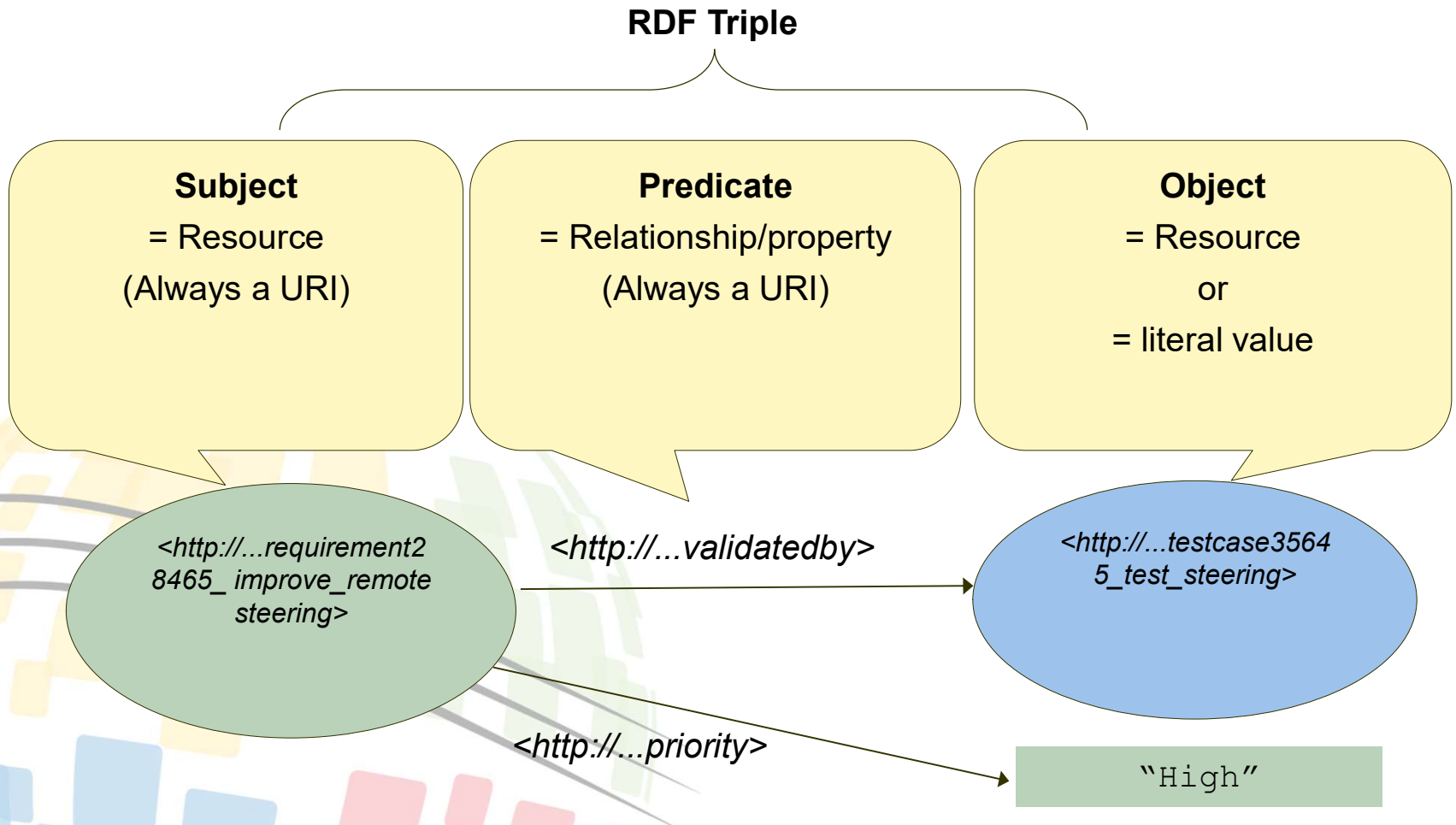
The RDF data model

- RDF – Resource Definition Framework
  - a standard to describe structured data on the web.
  - Generic description of linked data as a set of **triples**
  - RDF triples inspire a **graph**
- Basic structure of information: a triple
  - Each triple represents and edge
  - consisting of a subject, a predicate and an object.
  - The predicate denotes a relationship between the subject and object.
  - Graph nodes are resources or literals (values)
- RDF predicates are defined in RDF vocabularies identified by namespaces
  - e.g. rdf:about





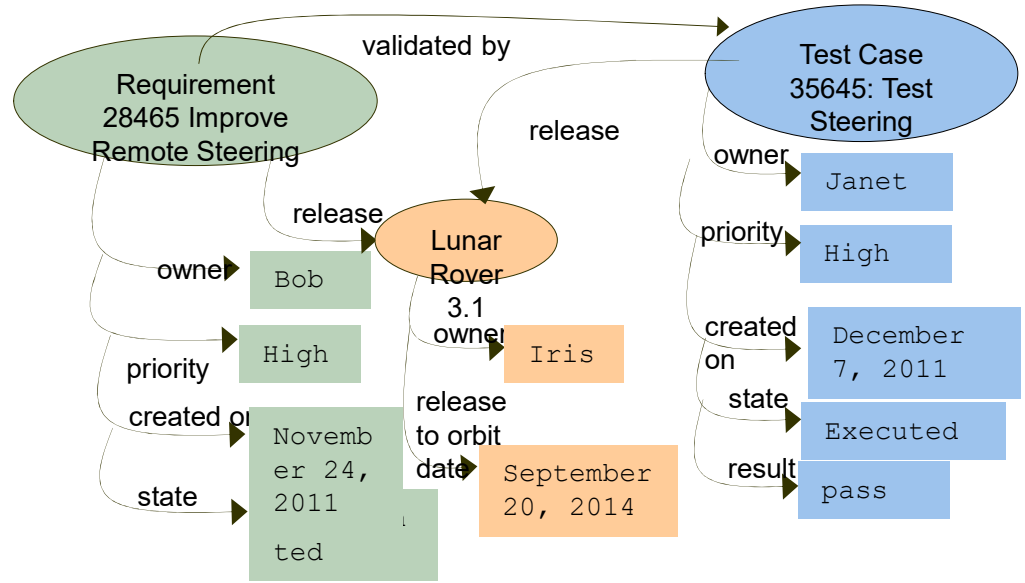
The Key Concepts - Everything is a URI



RDF graph data model - Compared to other data models

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.

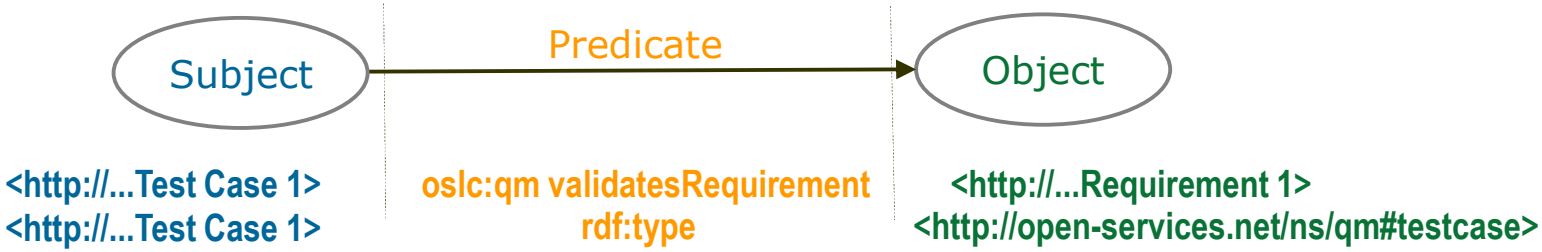


Requirement	Owner	Priority	…	Release	Validated by
R28464 …	…	…	…	…	
R28465 Improve Remote Steering	Bob	High	…	LR3.1	TC35645
R28466 …	…	…	…	…	…

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	…		…

RDF Textual Serialization formats




```
<rdf:description rdf:about="http://example.com/TestCases1>
        <oslc_qm:validatesRequirement rdf:resource="http://example.com/Requirements/1"/>
        <rdf:type rdf:resource="http://open-services.net/ns/qm#testcase"" />
</rdf:description>
```

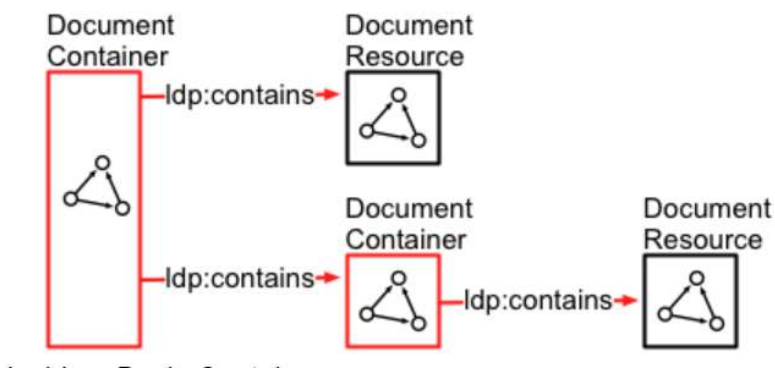
RDF/XML

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

Turtle

Linked Data Platform (LDP)

- A W3C Recommendation that provides clarifications and extensions of the 4 rules of Linked Data.
- Defines a set of rules for HTTP operations on web resources
  - to provide an architecture for accessing, updating, creating and deleting Linked Data resources from servers.
- Adds vocabulary and HTTP APIs to manage data containers
  - E.g. ldp:contains
- Supports basic, direct, and indirect containers
- Provides standard means to traverse and maintain hierarchical containers
- It is recommended for OSLC domains to implement the LDP patterns



@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix ldp: <http://www.w3.org/ns/ldp#>.
<http://example.org/alice/> a ldp:Container, ldp:BasicContainer; dcterms:title 'Alice's data storage on the Web' ;
ldp:contains <http://example.org/alice/foaf> , <http://example.org/alice/avatar> .

JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. **Service oriented RESTful HTTP based APIs**
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

Slide 34

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

HTTP based APIs (REST Architecture)

JEK0

- Using the HTTP (text based) protocol as an API across distributed applications

Key concepts:

- Requests and responses
- Resources: the targets of HTTP requests. May represent various things in the target application
  - E.g. A requirement, model-element, customer, defect, list of defets
- Resources: Identified by URIs
- 4 verbs for foundational API operations
  - Request = GET
  - Create = POST
  - Update = PUT
  - Delete = DELETE
- Message parameters
  - For example the expect format of the response. E.g. Accept=text/turtle
- message body: for example the resource content when creating a new resource

Slide 35

JEK0 REST instead of RESt

Jad El-Khoury, 2023-06-12T11:27:38.906

Examples: HTTP requests

- Get products from a bug tracker
  - GET /tracker/ldp-demo/ HTTP/1.1
  - Host: example.org
  - Accept: text/turtle; charset=UTF-8

  - Or:
  - HTTP://example.org/ tracker/ldp-demo?accet-text/turtle&method=get (URL encoded)
- Create a new defect on tracker
  - POST /tracker/ldp-demo/ HTTP/1.1
  - Host: example.org
  - Content-Type: text/turtle
  - <> a bt:BugReport;
  - dcterms:title "LDP Demo crashes when shutting down.";
  - dcterms:creator <http://example.org/tracker/users/johndoe> .

JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. **OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview**
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

Slide 37

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

The OSLC technology stack

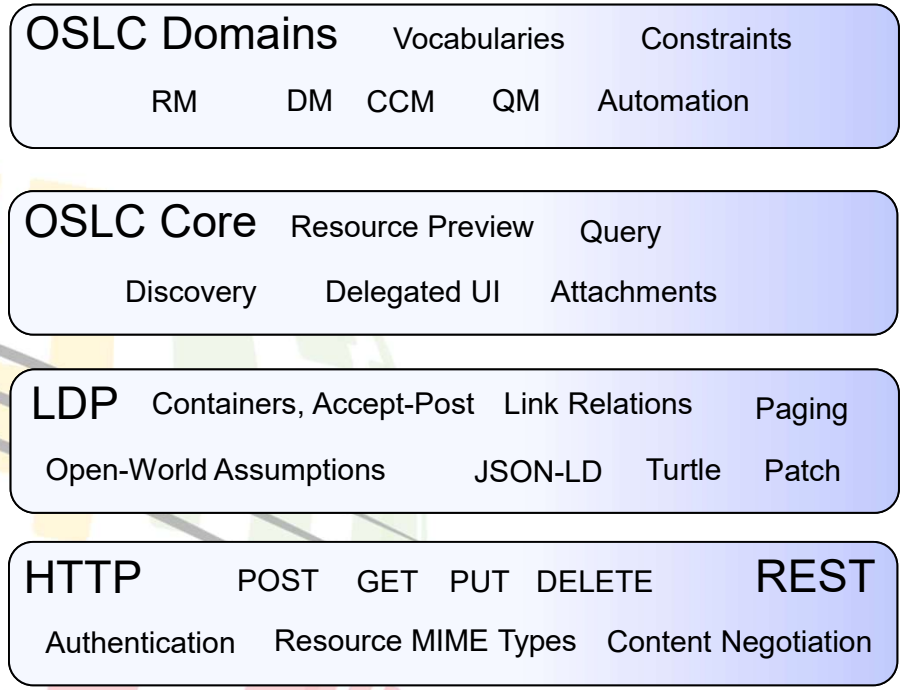
- OSLC Contributes with
  - **Core**: The standard rules and patterns for integrating lifecycle tools.
    - How to use HTTP & RDF to perform resource creation, queries, …
  - **Domains**: Common vocabulary for the lifecycle artifacts

Domains of interest that maintain separation of concerns and establish collaborative value streams through integration

Discoverability through Minimal, discoverable, self-describing capabilities to *enable* application integration

Reducing Variability through Self-describing, semantically rich, linked data resources leveraging HATEOAS

Address Complexity through HTTP and REST as the standard mechanism for distributed, loosely coupled APIs



Change Management
Configuration Management
Requirement Management
Quality Management

OSLC Core 3.0 Specification, OASIS

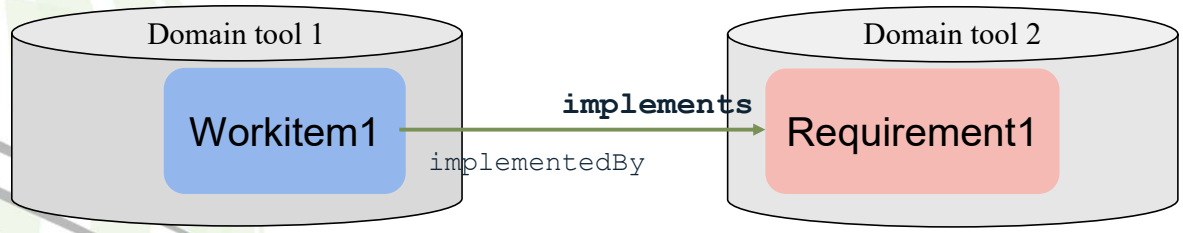
LDP 1.0 Specification, LDP.next Working Group, W3C

HTTP 1.1 Specification, IETF

Linked Data

OSLC resources and links

- OSLC resources are based on resource types specified in OSLC vocabularies
- OSLC links are OSLC properties referencing other resource URIs
- OSLC link types are described in OSLC vocabularies
- OSLC best practice is not to replicate links across tools and use link discovery for incoming (reverse) links



OSLC Vocabularies

- OSLC vocabularies describe types of resources and resource properties specified as OWL ontologies
- Vocabularies are described using RDFS (RDF Schema) which uses and RDF syntax
- The resource types and properties providing semantics to RDF data when used as types or predicates
- Note: Vocabularies do not impose constraints on resources, only provide semantics

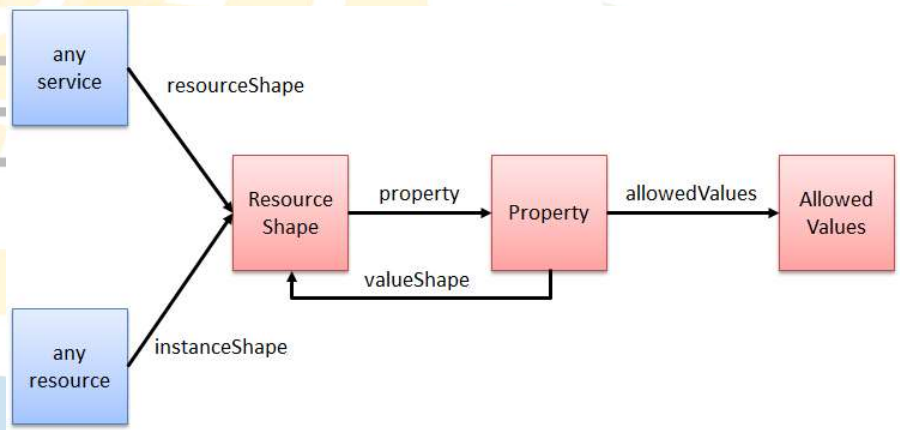
Example: requirements vocabulary

```
# declaring the vocabulary namespace
oslc_rm:  a owl:Ontology ;
      dcterms:title  "OSLC Requirements Management (RM) Vocabulary" ;
# declaring the requirement resource type
oslc_rm:Requirement  a   rdfs:Class ;
      rdfs:comment        "Statement of need." ;
      rdfs:isDefinedBy    oslc_rm: ;
      rdfs:label          "Requirement" .
# declaring an RM domain property
oslc_rm:specifies   a    rdf:Property ;
      rdfs:comment      "Expresses a specification relationship between entities, where the subject entity
                         specifies the object entity. For example, a model element specifies a requirement." ;
      rdfs:isDefinedBy  oslc_rm: ;
      rdfs:label        "specifies" .
```

These are rdfs metdata properties of the class, not of instances of the class.

OSLC Resource Shapes

- Constraining the properties of domain resources
  - Meaning/purpose/usage of a property
  - Which properties are mandatory, which optional
  - Permitted values of a property
- Server uses `oslc:ResourceShape` to provide this information
  - Defined by OSLC Core Resource Shape
  - Constraints on the *shape* of a resource and those to which it is linked; how to associate a shape with a resource

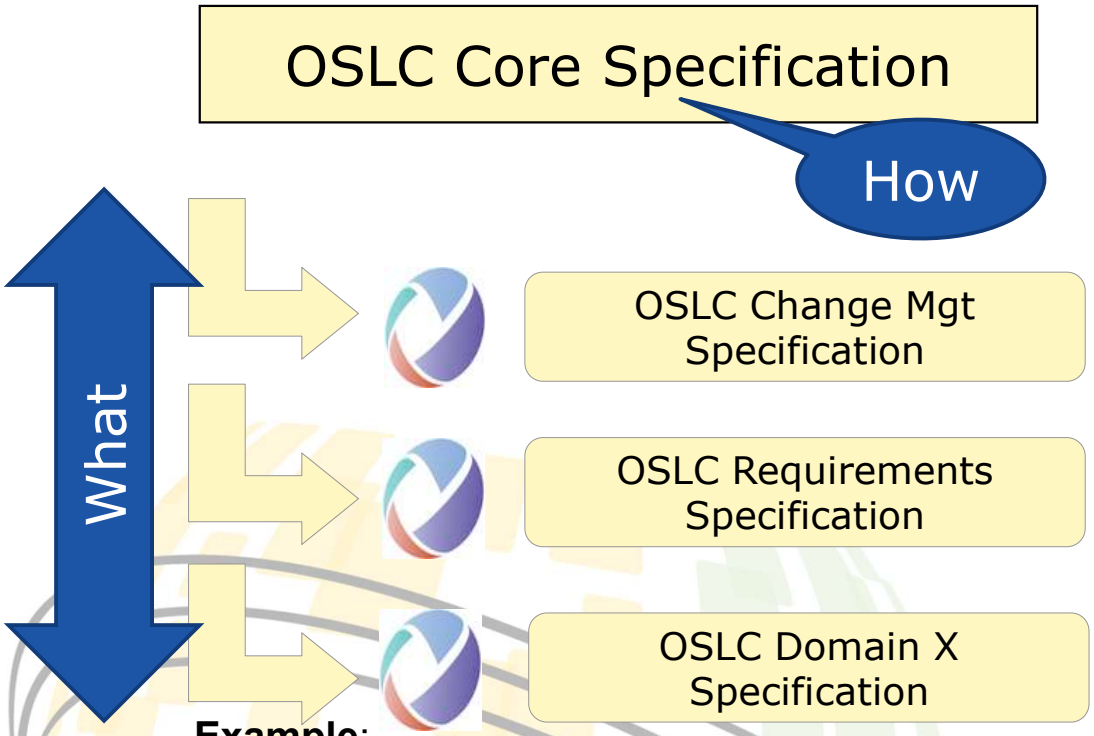


OSLC Vocabularies curate properties from other vocabularies

- Example: Dublin Core Metadata Initiative (DCMI)
  - Defines a set of properties for describing documents.
- https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

Property	Definition
dcterms:creator	An entity primarily responsible for making the content of the resource
dcterms:title	A name given to the resource
dcterms:format	The physical or digital manifestation of the resource
dcterms:date	A date of an event in the lifecycle of the resource
dcterms:publisher	An entity responsible for making the resource available
dcterms:subject	A topic of the content of the resource

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.

OSLC providers (domain tools)

OSLC Service Provider catalog

provides

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

example: IBM Engineering Workflow Manager (RTC)

OSLC Service Provider

provides an implementation of

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, …

OSLC Service

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

example: Change Management capability

manages

OSLC Resource

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 Core services

OSLC defines standard rules and patterns for integrating lifecycle tools

1. Discovery of capabilities

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

3. Querying for resources

4. Delegated UI for Selection

5. Delegated UI for Creation

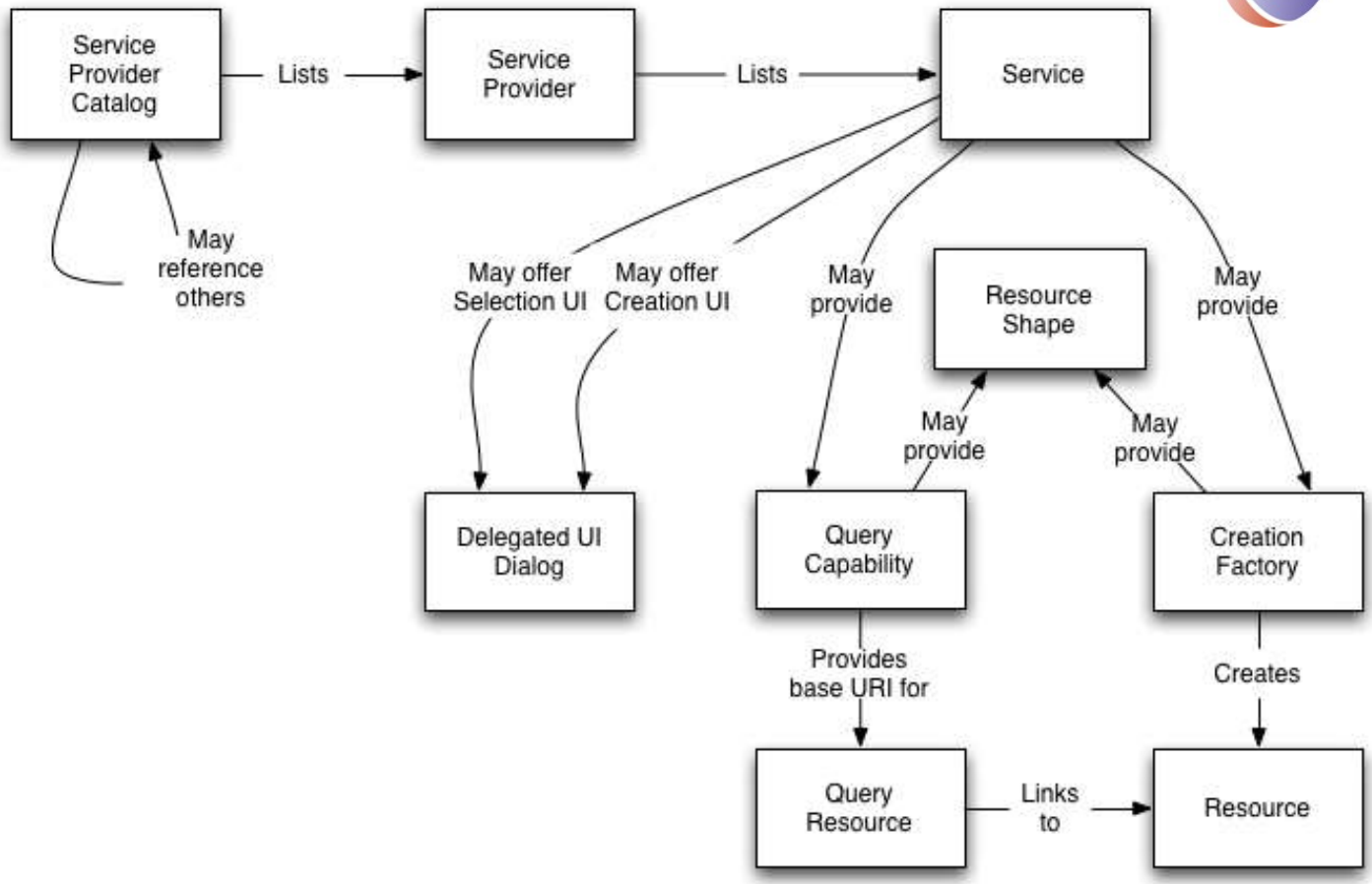
6. UI Previews for Resource Links

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



2a. 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)

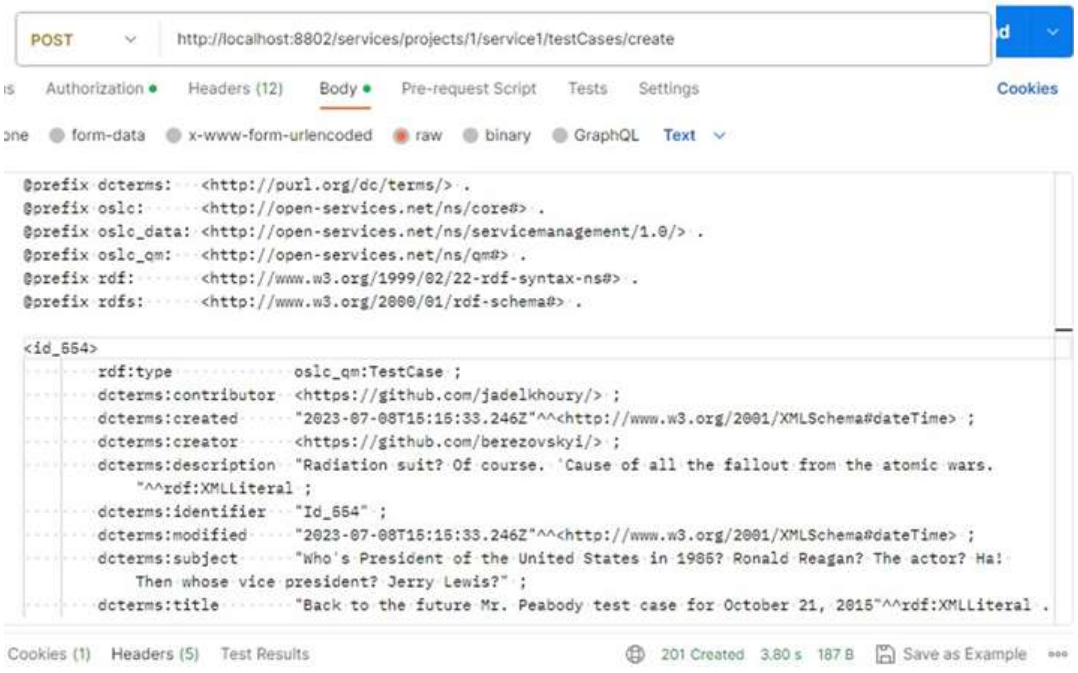
```
GET http://bugs/123
```

  - Get Defect 123 (just title and status)

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

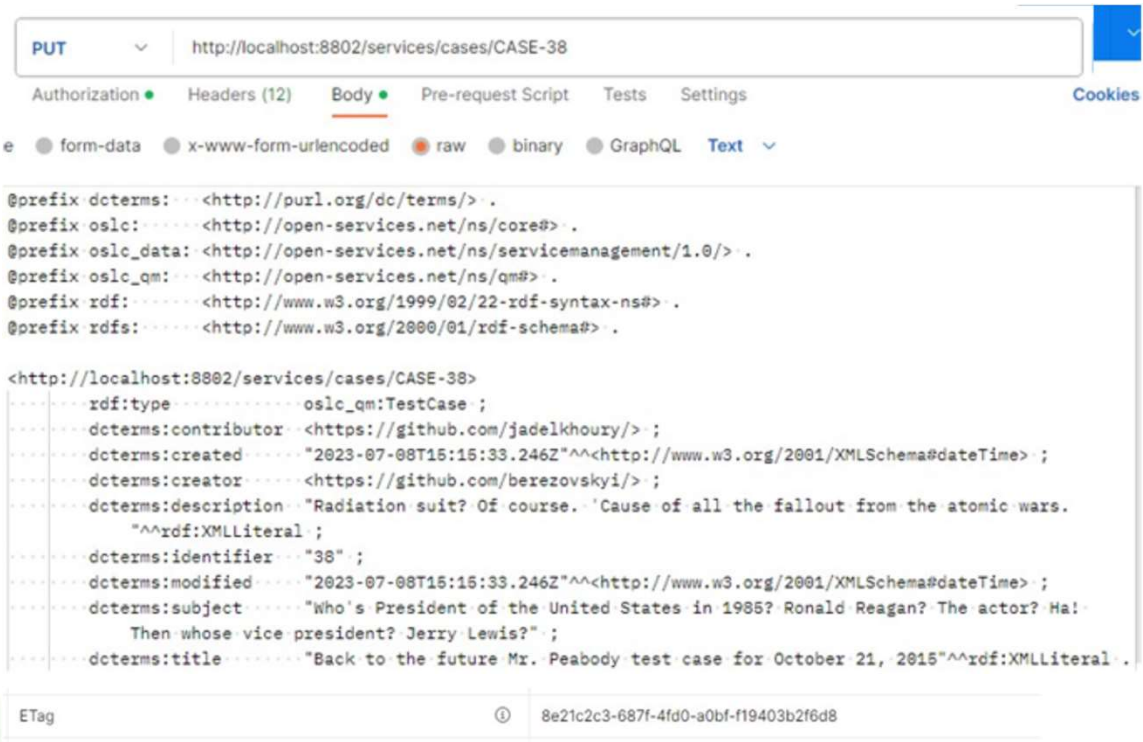
2b. Resource Creation (Create)

- Create a resource using HTTP POST
  - 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 may be defined by domain specification, or by service providers via a resource shape associated with creation factory



2c. Resource Modification (Update)

- Use HTTP GET to get resource properties to be updated
  - You'll get an ETag back
- Change only the property values you need to change
  - Clients must preserve unknown content
- 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

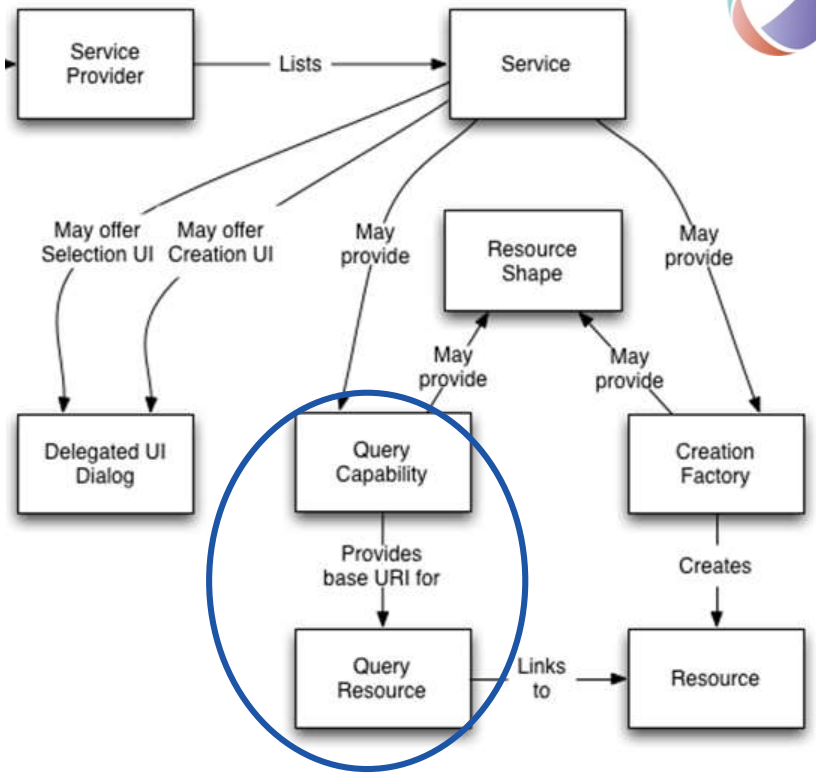


2d. HTTP C.R.U.D - Resource Deletion (Delete)

- Use HTTP DELETE on the resource identifier
- May not be allowed
- Response usually:
  - 200-OK
  - 204-No-Content
  - 400-Bad-Request
  - 403-Forbidden

3 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



3. 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

3. Query examples

- Find high severity bugs created after April fools day

```
http://example.com/bugs?oslc.where=
  cm:severity="high" and dcterms:created>"2023-04-01"
```

- Find bugs related to test case 31459

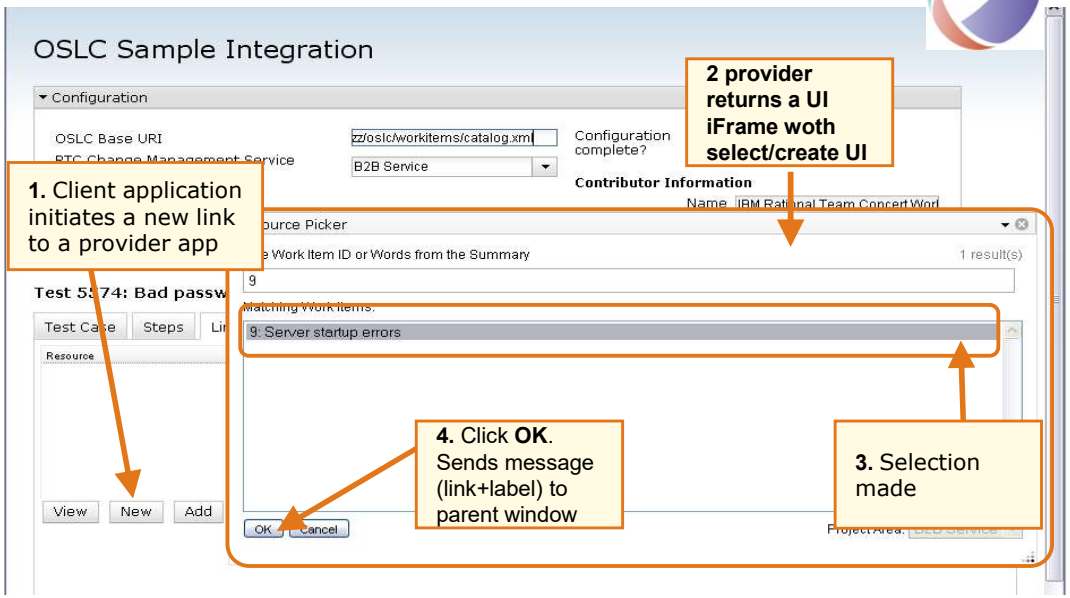
```
http://example.com/bugs?oslc.prefix=qm=
  <http://qm.example.com/ns>&
  oslc.where=qm:testcase=<http://example.com/tests/31459>
```

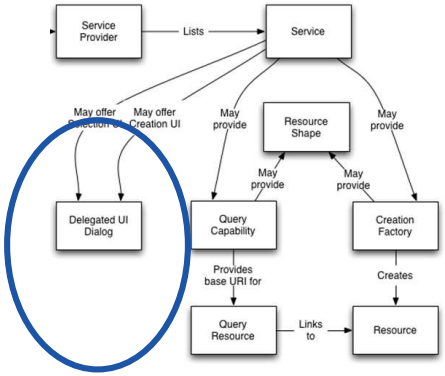
- Find all bugs created by John Smith

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

4. Delegated UI for Selection & Creation

- Delegated UI service renders UI of a provider application within a client application to enable easy selection/creation of resources from/by the provider
- Example: a test application needs to associate a test execution with a defect
- A delegated select/create UI is obtained from the defects application (CM)
- The tester looks for a matching defect using the UI provisions
- If no defect found, the tester creates a new defect via the delegated UI
- Ultimately a defect resource URI is returned to the test application
- Test application stores the link

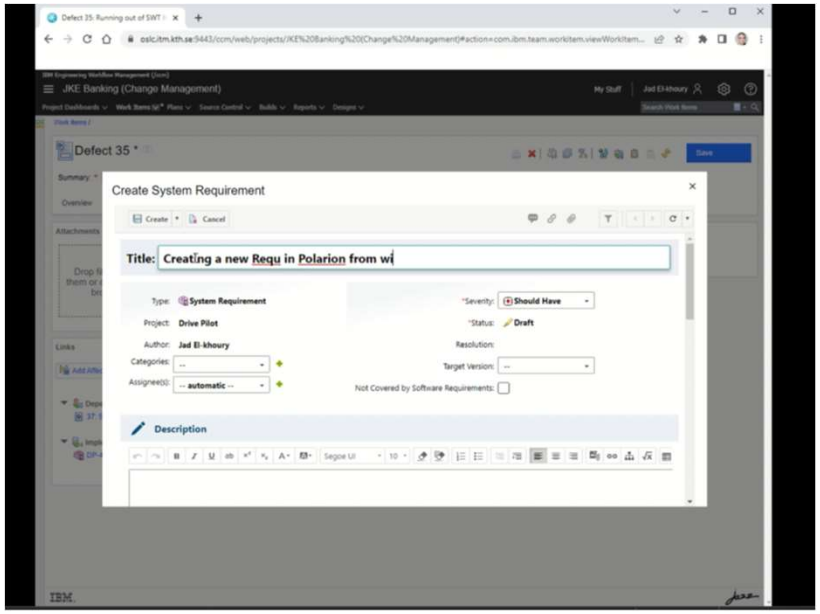


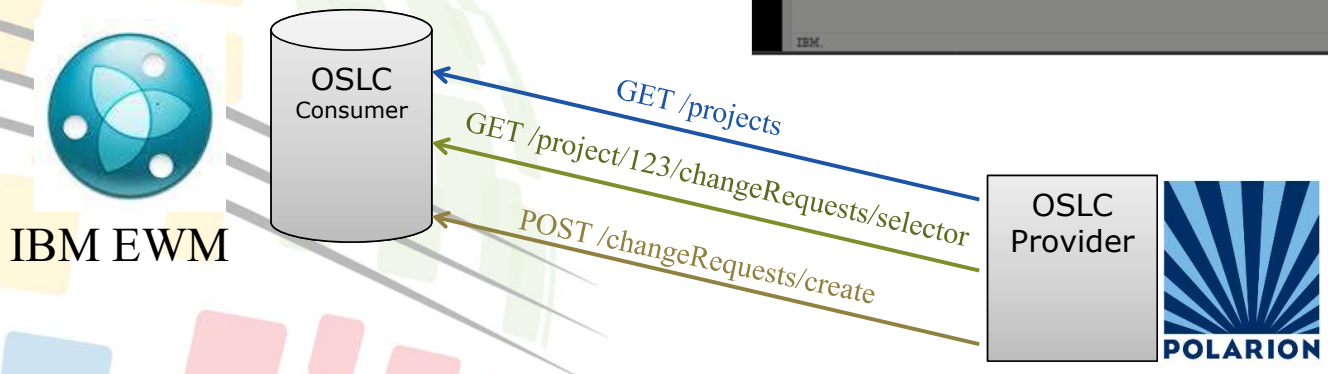


5. Delegated UI for Selection & Creation- Demo

The scenario: In Polarion, the user wants to link a Requirement to a Task in IBM EWM and also to Jira

- Alternatives:
  1. **JIRA Connector**: Import/export of JIRA artefacts into Polarion
  2. **OSLC Friendship**: Interoperate with JIRA OSLC service(s) that allow user to select/create a Task – from within Polarion.

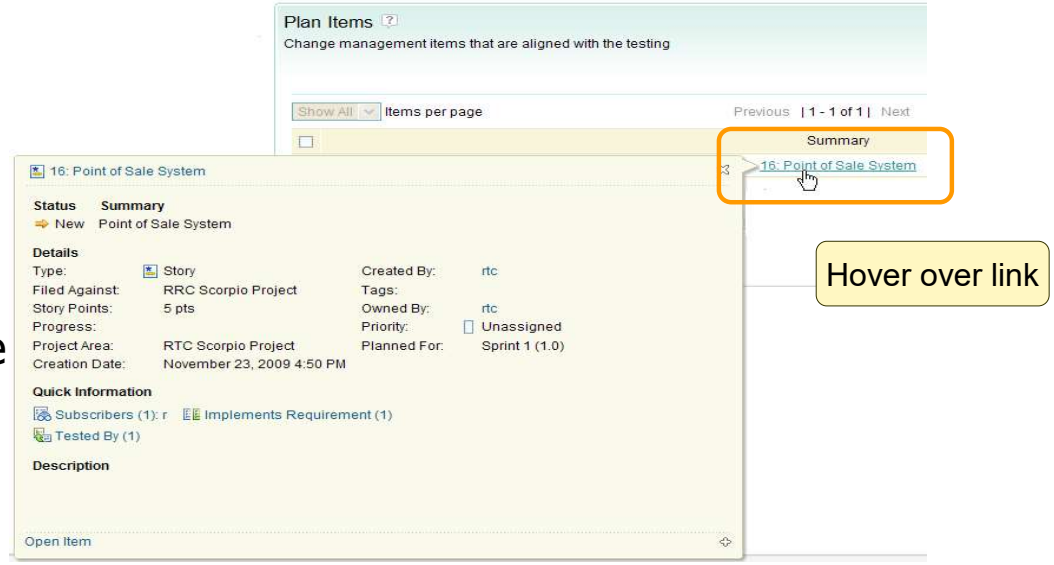




Link to video

6. Resource Preview

- Obtaining information about a linked resource by a client application
- The owning application determines what and how to visualize a preview for its resources
- Example: A requirements has a link to a story in a CM application. Hovering on the link brings up a preview of the story
- Obtaining a preview page is done by performing a get on the resource with a HTML MIME-type



JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. **OSLC domains and core lifecycle ontology**
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

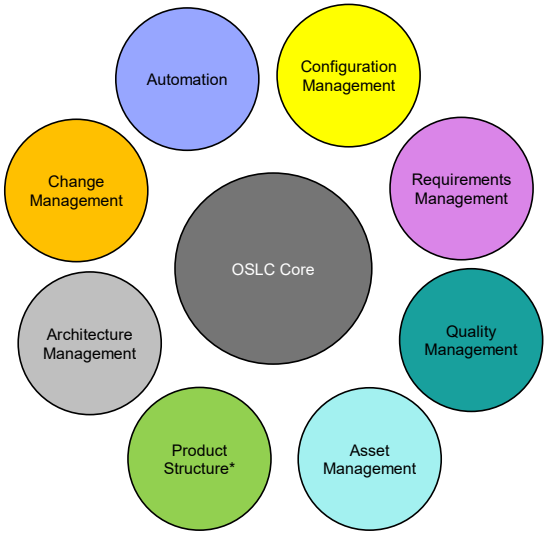
Slide 57

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

JEK0

OSLC Domain Specifications

- Core & Common
  - Configuration Management
  - Tracked Resource Set
  - Reporting
- Application Lifecycle Management (ALM)
  - Change Management
  - Quality Management
  - Requirements Management
  - Asset Management
  - Architecture Management
  - Automation
- (Software) Project Management
  - Estimation and Reporting
- Product Lifecycle Management (PLM)
  - ALM-PLM Interoperability



OSLC domain vocabularies

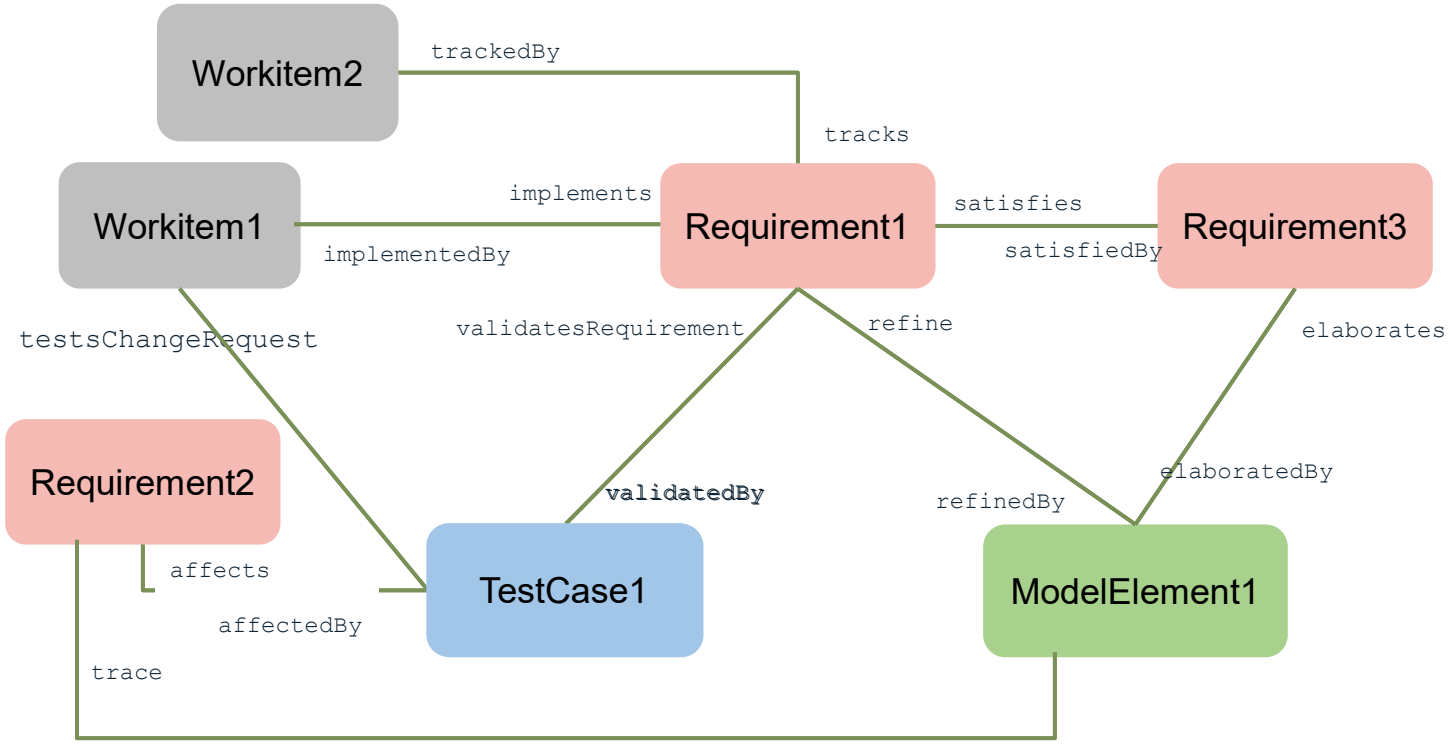
Slide 58

JEK0 Some of the domains on the list are obsolete.

Instead, why not mention the ongoing ones?
Jad El-Khoury, 2023-06-12T11:34:51.828

Standard OSLC domain links (partial)

- OSLC vocabularies specify domain link types as RDF properties
- A link type may constrain its source and target resource types
- Link types may specify labels and inverse labels



JEK0

Requirements Management domain

- Defines integration protocols for activities involving requirements, requirements collections, and traceability relationships
- Requirements Management v2.1 is an OASIS Standard
  - https://docs.oasis-open-projects.org/oslc-op/rm/v2.1/os/requirements-management-spec.html
- Requirement
  - A datum that can be represented as an `oslc_rm:Requirement`
  - Loosely, “A statement of need”
  - OSLC is not prescriptive of semantics
    - Example: An IBM DOORS Next textual artefact has an OSLC RM representation
    - Low bar for servers to expose wide variety of data
- Requirement Collection
  - A datum that can be represented as an `oslc_rm:RequirementCollection`
  - A collection of zero or more requirements.
  - OSLC is not prescriptive of semantics
    - Examples: IBM DOORS Next module (or “specification”) has such a representation

Slide 60

JEK0 Datum?
Jad El-Khoury, 2023-06-12T11:35:51.910

Example: Requirement resource http://openservices.net/ns/rm#Requirement

- Some properties are from the oslc_rm shape (green)
  - Eg dcterms:title property expresses a textual requirement, “Sample statement of need”
  - Vocabulary defined by Dublin Core
- Other vocabularies also used (black)
  - Eg process:projectArea property from jazz.net (see http://jazz.net/ns/process#)
- All properties are described by the oslc_core:instanceShape

- Distinction:
  - Vocabulary defines the term
    - Eg dcterms:identifier by Dublin Core
  - Shape describes usage/meaning of that term in the context of the resource
    - Eg That a dcterms:identifier represents the requirement’s identifier in the RM application

Example: Requirement resource in RDF

```
https://example.com/rm/resources/TX_kbvAUOA2Ee2EwpnLIAuuAQ
  a <http://open-services.net/ns/rm#Requirement> ;
  dcterms:modified "2023-04-21T12:20:49.637Z"^^xsd:dateTime ;
  oslc_core:instanceShape <https://example.com/rm/types/OT_BW8s8clCEe2jQ6SfLQN4Jw> ;
  process:projectArea <https://example.com/rm/process/project-areas/_-APP0MlBEe2jQ6SfLQN4Jw> ;
  oslc_config:component <https://example.com/rm/cm/component/_-DFvgMlBEe2jQ6SfLQN4Jw> ;
  dcterms:title "Sample requirement"^^rdf:XMLLiteral ;
  oslc_core:serviceProvider <https://example.com/rm/oslc_rm/_-APP0MlBEe2jQ6SfLQN4Jw/services.xml> ;
  dcterms:identifier "9"^^xsd:string ;
  dcterms:created "2023-04-21T12:20:49.637Z"^^xsd:dateTime ;
  dnext_nav:parent <https://example.com/rm/folders/FR_-FGhsclBEe2jQ6SfLQN4Jw> ;
dcterms:creator <https://example.com/jts/users/img> ;
  dcterms:description ""^^rdf:XMLLiteral ;
  jazz_acp:accessControl <https://example.com/rm/accessControl/_-APP0MlBEe2jQ6SfLQN4Jw> ;
  dcterms:contributor <https://example.com/jts/users/img> .
```

Requirements Management shapes

- See https://docs.oasis-open-projects.org/oslc-op/rm/v2.1/os/requirements-management-shapes.html for RM 2.1 Requirement shape

Prefixed Name	Occurs	Read-only	Value-type	Representation	Range	Description
dcterms:contributor	Zero-or-many	unspecified	AnyResource	Either	oslc:AnyResource	Contributor(s) to resource (reference: Dublin Core). It is likely th necessarily the case.
dcterms:created	Zero-or-one	true	dateTime	N/A	Unspecified	Timestamp of resource creation (reference: Dublin Core).
dcterms:creator	Zero-or-many	unspecified	AnyResource	Either	oslc:AnyResource	Creator(s) of resource (reference: Dublin Core). It is likely that th the case.
dcterms:description	Zero-or-one	unspecified	XMLLiteral	N/A	Unspecified	Descriptive text (reference: Dublin Core) about resource represe and suitable inside an XHTML element. [cc-5]
dcterms:identifier	Zero-or-one	true	string	N/A	Unspecified	An identifier for a resource. This identifier may be unique with a resource is created. Not intended for end-user display.
dcterms:modified	Zero-or-one	true	dateTime	N/A	Unspecified	Timestamp of last resource modification (reference: Dublin Core
dcterms:subject	Zero-or-many	false	string	N/A	Unspecified	Tag or keyword for a resource. Each occurrence of a dcterms:su
dcterms:title	Exactly-one	unspecified	XMLLiteral	N/A	Unspecified	Title (reference: Dublin Core) of the resource represented as ric XHTML element. [cc-6]
oslc_rm:affectedBy	Zero-or-many	false	Resource	Reference	oslc:AnyResource	The subject is affected by the object, such as a defect or issue.
[illegible]	Zero-or-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

JEKO

RM Requirement Shape – RDF format

```
<https://example.com/types/OT_BW8s8clCEe2jQ6SfLQN4Jw>
  a <http://open-services.net/ns/core#ResourceShape> ;
  dcterms:title "requirement"^^rdf:XMLLiteral ;
  oslc_core:serviceProvider <https://example.com/oslc_rm/_-APP0MlBEe2jQ6SfLQN4Jw/services.xml> ;
  oslc_config:component <https://example.com/cm/component/_-DFvgMlBEe2jQ6SfLQN4Jw> ;
  acc:accessContext <https://example.com/acclist#_-APP0MlBEe2jQ6SfLQN4Jw> ;
  oslc_core:describes <http://open-services.net/ns/rm#Requirement>,
                      <http://open-services.net/ns/rm#RequirementCollection> ;

oslc_core:property [
    a oslc_core:Property ;
    oslc_core:range <http://open-services.net/ns/rm#RequirementCollection>,
                    oslc_core:Resource,
                    <http://open-services.net/ns/rm#Requirement> ;
    oslc_core:propertyDefinition <http://www.ibm.com/xmlns/rdm/types/Link> ;
    dcterms:title "Link To"^^rdf:XMLLiteral ;
    oslc_core:occurs oslc_core:Zero-or-many ;
    dcterms:description "Tracks a general relationship between Requirements Management artifacts."^^rdf:XMLLiteral ;
    oslc_core:name "Link" ;
    oslc_core:representation oslc_core:Reference ;
    oslc_core:valueType oslc_core:Resource
  ],
```

cont…

- "Link To" traceability link
- Zero-or-many
- Target of link is either a Requirement, Collection or any OSLC Core Resource

Slide 64

JEK0 You showed another example a few slides beforehand. Keep this one, but remove the previous one?
Jad El-Khoury, 2023-06-12T11:37:01.505

Custom traceability properties

- User defines a traceability relationship in their application, as part of a curated custom vocabulary
- Application surfaces this user-defined property on OSLC API
  - RDFS descriptions of the vocabulary
  - ResourceShapes reflecting its usage

- Clients discover these custom properties via ServiceProvider and ResourceShape
  - Supports generic tooling (query builders, explorers etc.)
  - Supports better user experience (labelling of data, reports etc.)

Example: specifying a custom traceability property

RDFS description of the vocabulary term:

```
< http://example.com/ns/satisfaction >
  a rdf:Property ;
  oslc_config:component <https://example.com/rm/cm/component/_-DFvgMlBEe2jQ6SfLQN4Jw> ;
  oslc_core:serviceProvider <https://example.com/rm/oslc_rm/_-APP0MlBEe2jQ6SfLQN4Jw/services.xml> ;
  owl:sameAs <http://example.com/ns/satisfaction> ;
  rdfs:label "satisfies" ;
  oslc_core:inverseLabel "satisfied by" .
```

ResourceShape property:

```
  a oslc_core:Property ;
  dc:title "satisfies"^^rdf:XMLLiteral ;
  oslc_core:propertyDefinition <http://example.com/ns/satisfaction> ;
  oslc_core:range oslc_core:Resource,
                  <http://open-services.net/ns/rm#Requirement>,
                  <http://open-services.net/ns/rm#RequirementCollection> ;
  oslc_core:occurs oslc_core:Zero-or-many ;
  oslc_core:valueType oslc_core:Resource ;
  oslc_core:representation oslc_core:Reference
```

Example: Representation of requirement with a trace link

```
<https://example.com/rm/resources/TX_kbvAUOA2Ee2EwpnLIAuuAQ>
  a oslc_rm:Requirement ;
  dc:identifier "9"^^xsd:string ;
  dc:created "2023-04-21T12:20:49.637Z"^^xsd:dateTime ;
  dc:creator <https://example.com/jts/users/img> ;
  dc:description ""^^rdf:XMLLiteral ;
  dc:modified "2023-04-21T12:20:49.637Z"^^xsd:dateTime ;
  oslc_core:instanceShape
<https://example.com/rm/types/OT_BW8s8clCEe2jQ6SfLQN4Jw> ;
  oslc_config:component <https://example.com/rm/cm/component/_-
DFvgMlBEe2jQ6SfLQN4Jw> ;
  dc:title "Sample requirement"^^rdf:XMLLiteral ;
  oslc_core:serviceProvider <https://example.com/rm/oslc_rm/_-
APP0MlBEe2jQ6SfLQN4Jw/services.xml> ;
  oslc_rm:satisfies
<https://example.com/rm/resources/TX_05DFwM67Ee2tvoEyrwG3bQ> ;
  dc:contributor <https://example.com/jts/users/img> .
```

JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. **OSLC configuration management**
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

Slide 68

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

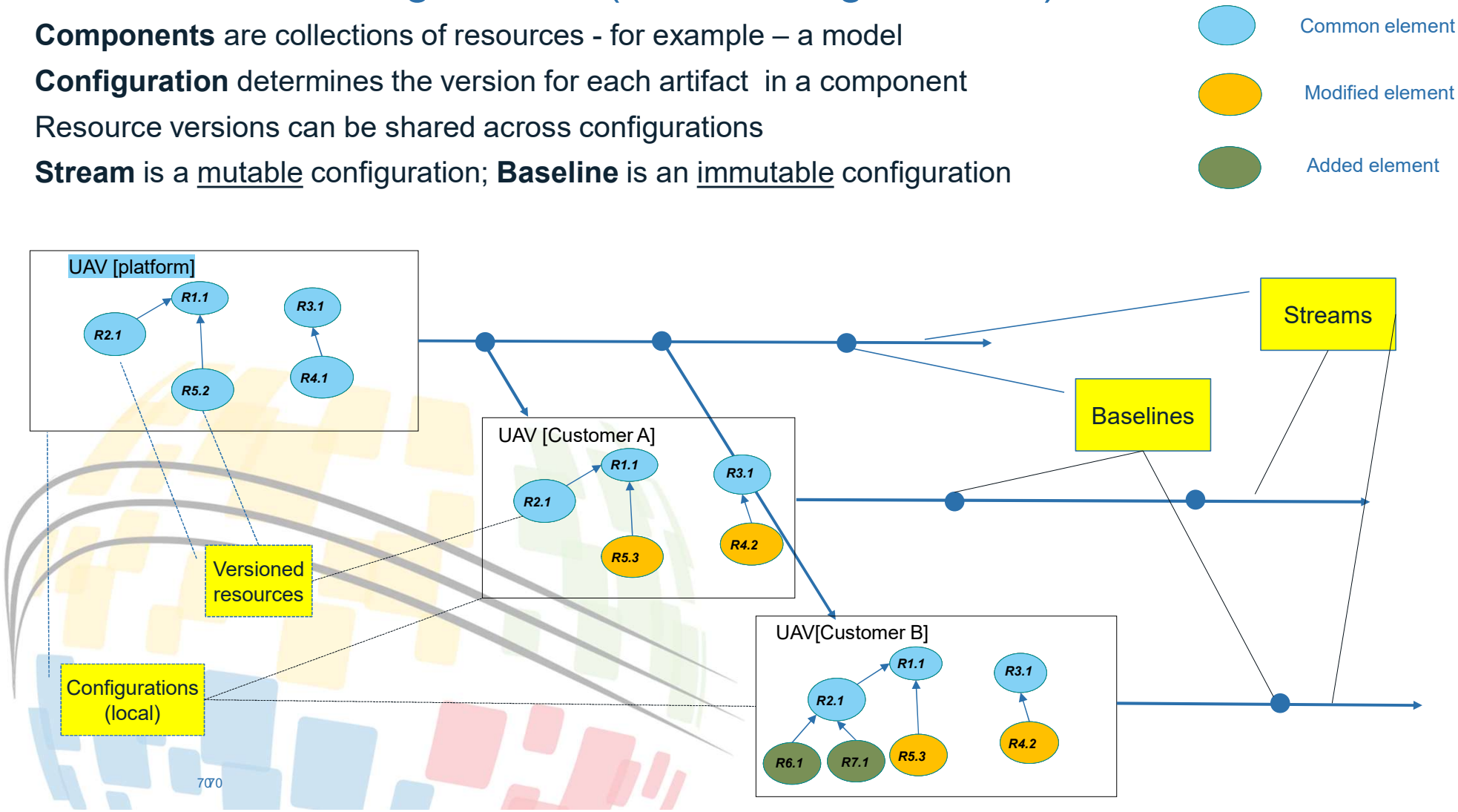
OSLC Configuration management : Key concepts

- Standardizing referential and publishing of **versioned engineering data** by domain providers
- Orchestrating multiple domain configurations into global contexts with global configurations
- Configuration providers: manage versioned data
  - E.g.: requirements, models, source-code, CAD data

- Concept resource: a referential identity of a resource that has versions
  - E.g. requirement#27
- Version resource: a concrete state of a concept resource
  - E.g. requirement#27 version 5

OSLC domain configurations (“local configurations”)

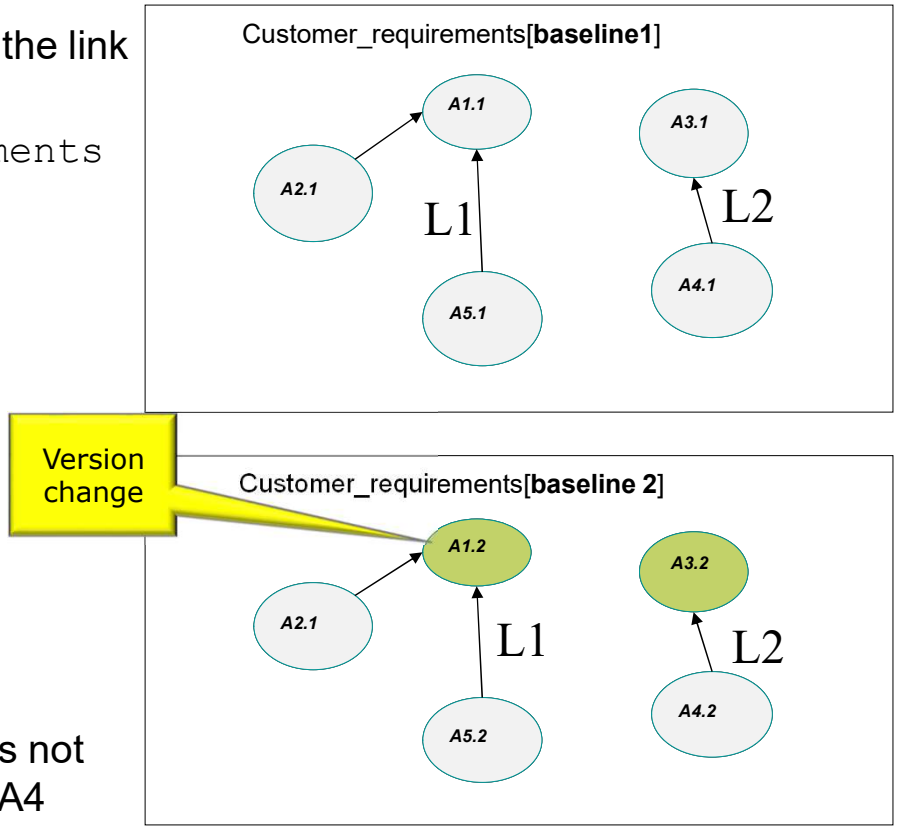
- **Components** are collections of resources - for example – a model
- **Configuration** determines the version for each artifact in a component
- Resource versions can be shared across configurations
- **Stream** is a <u>mutable</u> configuration; **Baseline** is an <u>immutable</u> configuration



Conceptual Links and Link resolution

- In OSLC we use conceptual links to maintain validity of the link in case of version updates using link resolution
- Example: Two configurations of `customer_requirements`

  `A1` and `A3` were modified in `baseline2`
- Conceptual links refer to concept resources, e.g
  - `L1` refers to `A1`
  - `L2` refers to `A3`
- In the context of [`baseline1`]
  - L1 **resolves** to `A1.1`
  - L2 **resolves** to `A3.1`
- In the context of [`baseline 2`]
  - L1 resolves to A1.2
  - L2 resolves to A3.2
- Changing to context from [baseline1] to [baseline2] does not require any change to the tracing requirements A5 and A4



Conceptual Links persist when versions of objects are replaced in a configuration

RDF representation of versioned resources with RDF graphs

- RDF graphs are containers of RDF statements
- In presence of multiple versions of the same resource, the version resource graph scopes the relevant statements of a specific version
- In this example `requirementA` has two versions `v1,v2,` and `requirementB` has one version `v1`
- Each version resource has two sets of triples enclosed by the graph:
  - Version triples
  - Resource triples
- Domain providers may implement graph scoping in different ways

```
:requirementA-v1
        a oslc_config:VersionResource ;
        dcterms:isVersionOf :requirementA .
    :requirementA
        a oslc_rm:Requirement ;
        oslc_config:versionId "v1" ;
        oslc_config:component :rmComponent1 ;
        dcterms:identifier "A" ;
        dcterms:title "Requirement A"^^rdf:XMLLiteral ;
        dcterms:description "A description of requirement A version 1"^^rdf:XMLLiteral .
```

Graph requirementA-V1

```
:requirementA-v2
        a oslc_config:VersionResource ;
        dcterms:isVersionOf :requirementA ;
        prov:wasRevisionOf :requirementA-v1 .
    :requirementA
        a oslc_rm:Requirement ;
        oslc_config:versionId "v2" ;
        oslc_config:component :rmComponent1 ;
        dcterms:identifier "A" ;
        dcterms:title "Requirement A"^^rdf:XMLLiteral ;
        dcterms:description "A description of requirement A version 2 (changed description)"^^rdf:XMLLiteral .
```

Graph requirementA-V2

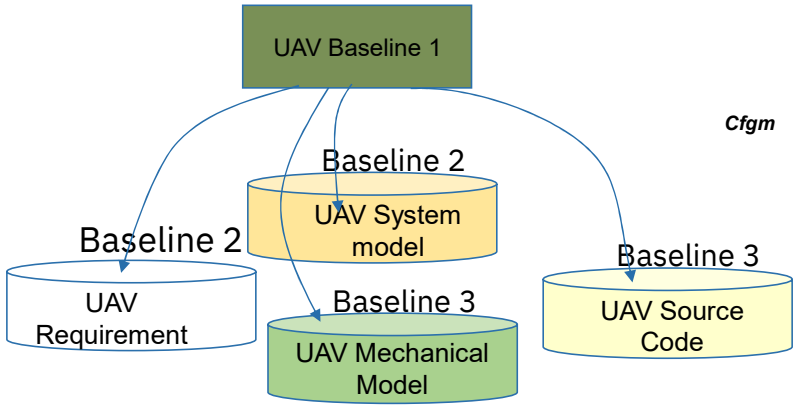
```
:requirementB-v1
        a oslc_config:VersionResource .
        dcterms:isVersionOf :requirementB .
    :requirementB
        a oslc_rm:Requirement ;
        oslc_config:versionId "v1" ;
        oslc_config:component :rmComponent1 ;
        dcterms:identifier "B" ;
        dcterms:title "Requirement B"^^rdf:XMLLiteral ;
        dcterms:description "A description of requirement B version 1"^^rdf:XMLLiteral ;
        oslc_rm:refines :requirementA.
```

Graph requirementB-V1

Global configuration management
Orchestrating configurations across multiple providers

- Consistent evolution of data across engineering disciplines: common baselining
- A breakdown structure across the entire design space
  engineering assets across variants and programs
- Reuse all engineering assets from the platform: requirements, design, implementation, test
- Manage changes across variants and programs

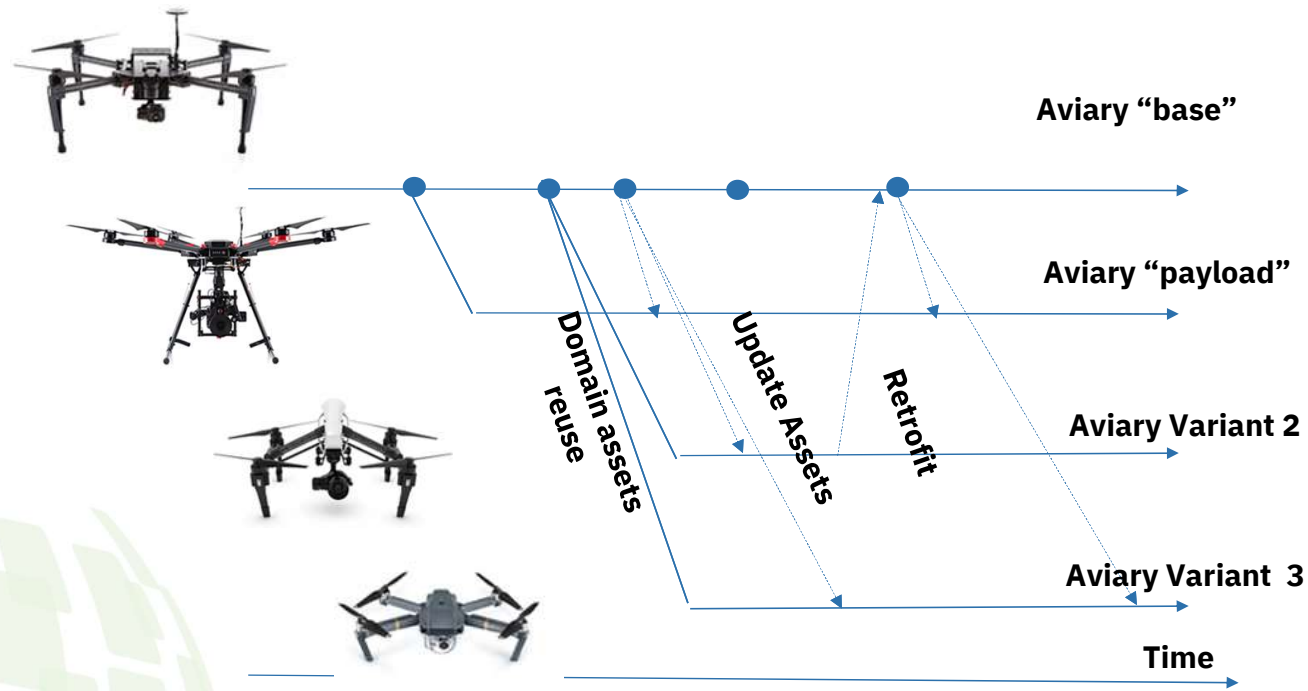
A Global Configuration across domain providers



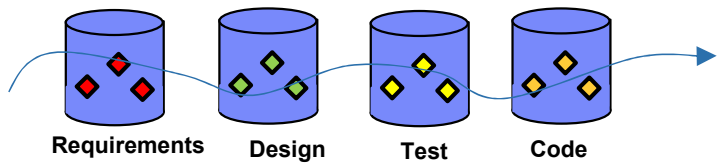
Example: Hierarchical configuration of the “Aviary” system

Component reuse across variant streams

- Enabling artifacts and component reuse across concurrent variants and programs using GCM streams
- Controlling undesired interference across variants
- Propagate changes across variants and programs



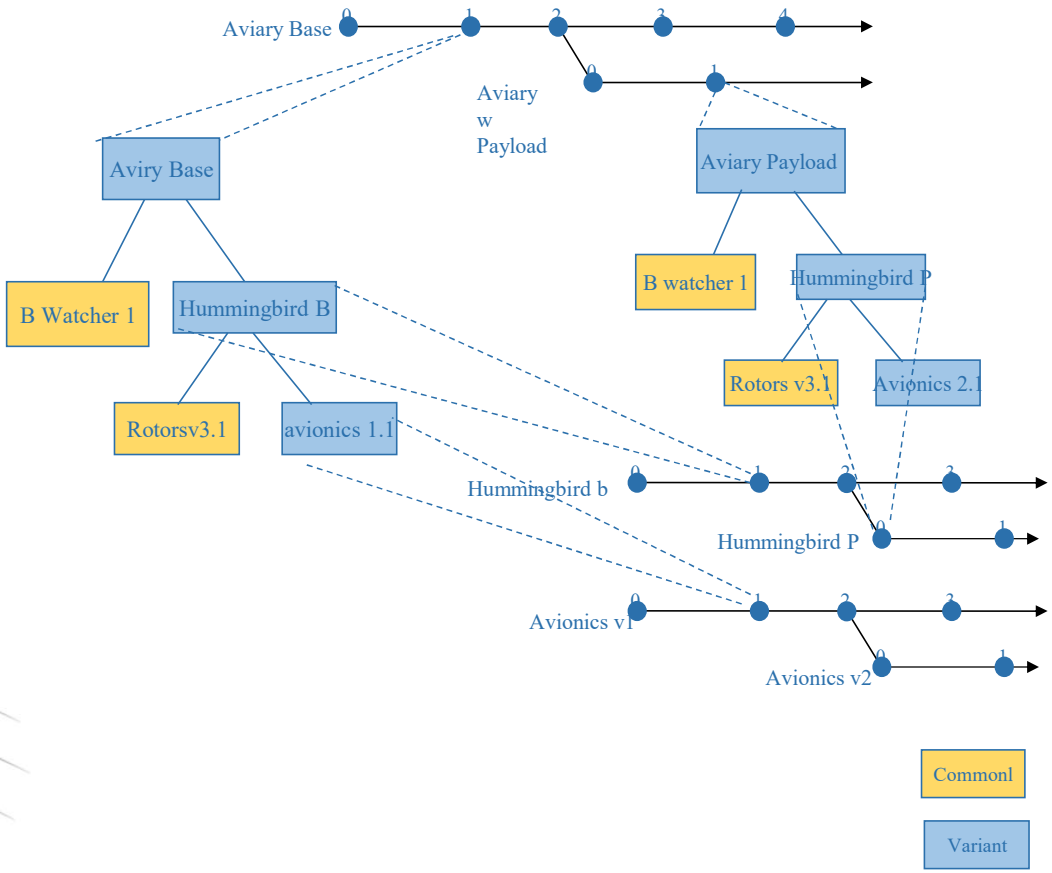
Example: Aviary Variants



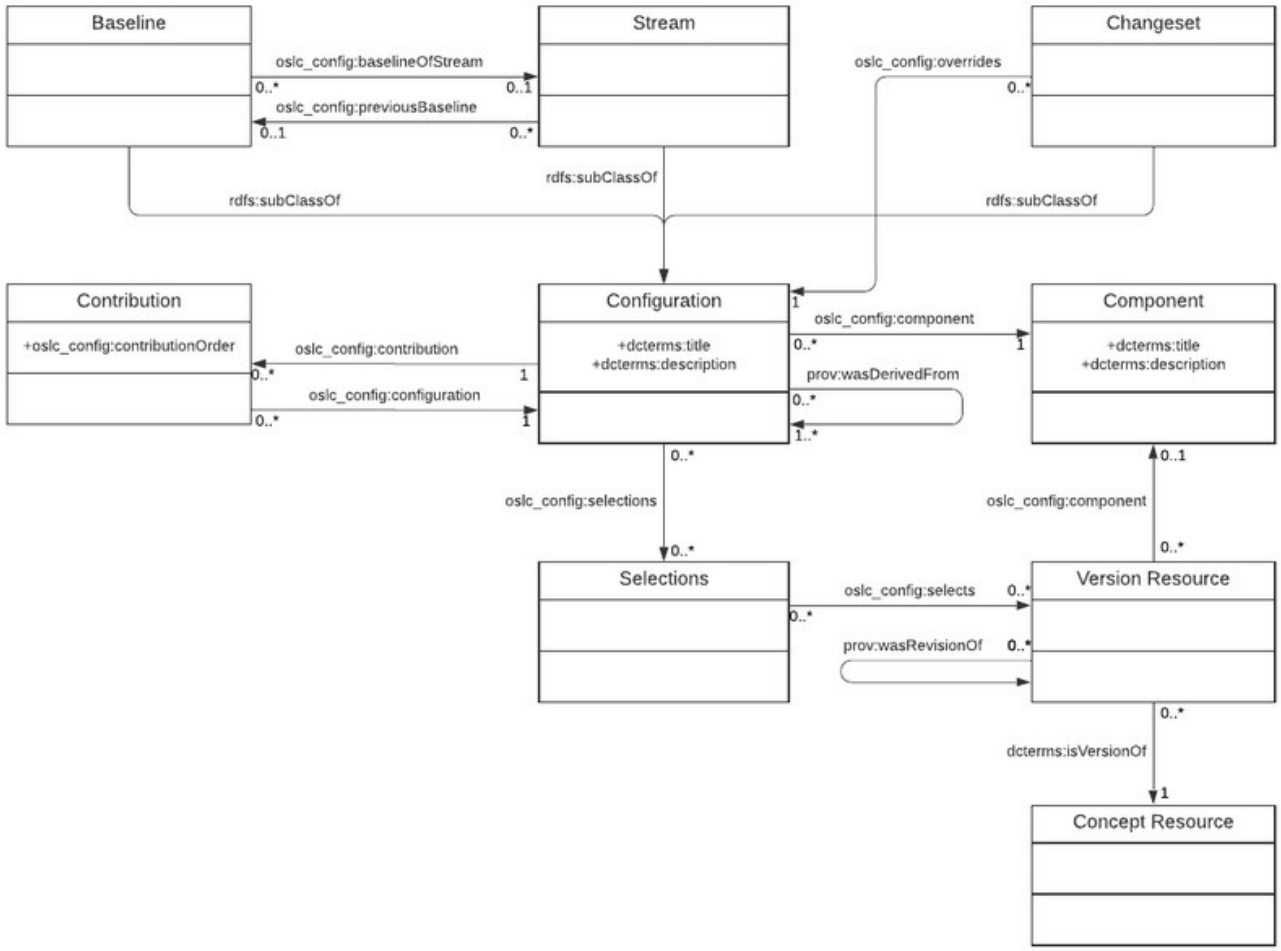
Systems engineering artifact reuse across programs and variants

- Federated configuration management enables baselining and reuse of configuration items across all lifecycle disciplines
- Configuration items are organized in hierarchical configurations
- Configuration items can have variants to realize variability across programs and products
- Configuration items can be reused across programs and products

Example: component reuse across a UAV system (Aviary) variants



Global configurations conceptual domain model



JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. **OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics**
8. Creating OSLC adapters - Eclipse LYO and others
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

Slide 77

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

Tracked Resource Set

- Tracked Resource Set v3.0 is an OASIS Project Specification
  - See https://docs.oasis-open-projects.org/oslc-op/trs/v3.0/tracked-resource-set.html
- Need: Digital Thread Central Reporting and Analysis
  - Reporting
  - Impact & Trace analyses
  - Metrics

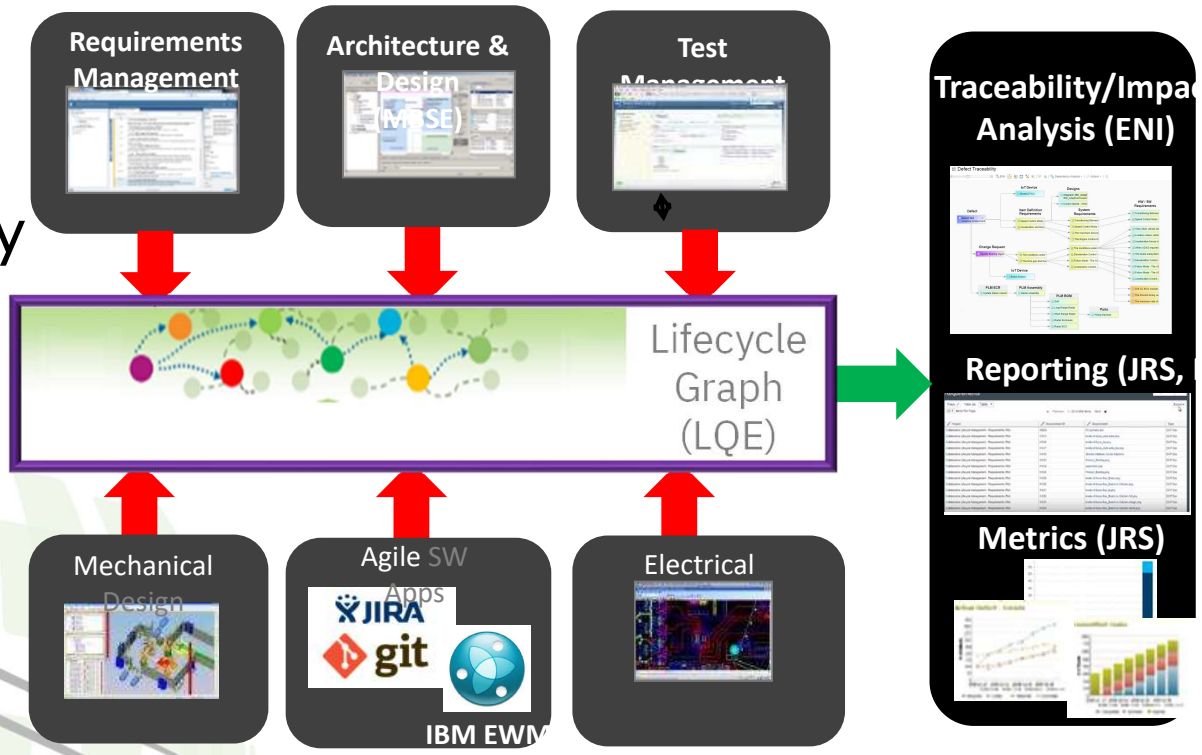
- Data is *replicated* to central repository
  - TRS is the OSLC mechanism for *replication* from domain application to this central repository
- Central repository provides query endpoint
  - Can be SPARQL, SQL etc. (currently specified by OSLC)

OSLC Tracked Resource Set

Domain TRS

Central repository (a TRS client)

Result sets



Visualization and export Tools

Tracked Resource Set

- Provides
  - Means to enumerate the set of resources in a server
  - Means to monitor changes over time to this set of resources
  - Protocol characteristics
    - TRS clients can join anytime and catch up
    - Access controls imposed by each TRS server
    - Uses OSLC representations
    - Robust to arbitrarily large sets of resources
    - Changes processed asynchronously
- Usage: Digital Thread reporting/analysis services is a TRS client
  - Replicate these data in a form optimized for reporting into a single reporting database
  - Track and react to resource changes in order to maintain freshness in that single database
  - Expose visualization services over that single database

Tracked Resource Set and its clients

- TRS contains all OSLC resources that are to be exposed to TRS client
- Generally, a server has more than one TRS
  - Eg resource category: Eg requirements and process definitions
  - Eg TRS client responsibility
    - “all domain content” or “only traceability data”
- A TRS client is typically the client of multiple TRS servers
  - Responsible for replicating resources from many servers

Tracked Resource Set

- Logical set comprises two parts
  - Base: a set of resources that belong to the set at a fixed time t0
  - Change Log: a record of all changes made to the logical set since t0
- TRS has representations of both base and change log
  - Base and change log are shared across all TRS clients
- A TRS client first reads the entire base, then polls the change log for changes
  - Changes are ordered and have identity
  - Enables TRS client to consume all changes exactly once
    - Creation -> Fetches new resource and inserts it
    - Modification -> Fetches latest resource representation and updates it
    - Deletion -> Removes current resource
  - Central repository is *eventually consistent* with all of the contributing domain servers

TRS base – tracking starting point

- Contains all resources appropriate to the scope of the TRS
  ResourceShapes
  Vocabulary terms
  Domain resources (here, OSLC RM resources

- Example of a base response

```
<ldp:DirectContainer rdf:about=".../rm/trs2/base">
    <ldp:member rdf:resource=".../rm/workflow/attrdef/_-APP0MlBEe2jQ6SfLQN4Jw/DefaultWorkflow"/>
    <ldp:member rdf:resource=".../rm/versionedResources/TX_cWKscMlCEe2jQ6SfLQN4Jw"/>
    <ldp:member rdf:resource=".../rm/cm/baseline/_QsUUcMlCEe2jQ6SfLQN4Jw"/>
    <ldp:member rdf:resource=".../rm/versionedShapes/LT_YAaNIclCEe2jQ6SfLQN4Jw"/>
    ...
```

Example: TRS change log

```
# Resource: http://cm1.example.com/trackedResourceSet
@prefix trs: <http://open-services.net/ns/core/trs#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://cm1.example.com/trackedResourceSet>
  a trs:TrackedResourceSet ;
   trs:base <http://cm1.example.com/baseResources/> ;
  trs:changeLog [
    a trs:ChangeLog ;
    trs:change <urn:example:6e8bc430:cm1.example.com:2010-10-27T17:39:33.000Z:103> ;
    trs:change <urn:example:6e8bc430:cm1.example.com:2010-10-27T17:39:32.000Z:102> ;
    trs:change <urn:example:6e8bc430:cm1.example.com:2010-10-27T17:39:31.000Z:101> .
  ] .
<urn:example:6e8bc430:cm1.example.com:2010-10-27T17:39:33.000Z:103>
  a trs:Creation ;
  trs:changed <http://cm1.example.com/bugs/23> ;
  trs:order "103"^^xsd:integer .
<urn:example:6e8bc430:cm1.example.com:2010-10-27T17:39:32.000Z:102>
  a trs:Modification ;
  trs:changed <http://cm1.example.com/bugs/22> ;
  trs:order "102"^^xsd:integer .
<urn:example:6e8bc430:cm1.example.com:2010-10-27T17:39:31.000Z:101>
  a trs:Deletion ;
  trs:changed <http://cm1.example.com/bugs/21> ;
  trs:order "101"^^xsd:integer .
```

Lifecycle query

- TRS client can expose query across all lifecycle resources
  - CQRS pattern
    - Experience shows that this architecture scales better and is easier to manage than federated query
  - Eg IBM ELM LQE exposes SPARQL endpoint
  - Eg IBM ELM LDX exposes services to discover trace links between OSLC resources
- Eg TRS client can trigger workflow transitions/alerts when a domain resource changes

JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. **Creating OSLC adapters - Eclipse LYO and others**
9. Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations

Slide 86

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

Eclipse Lyo

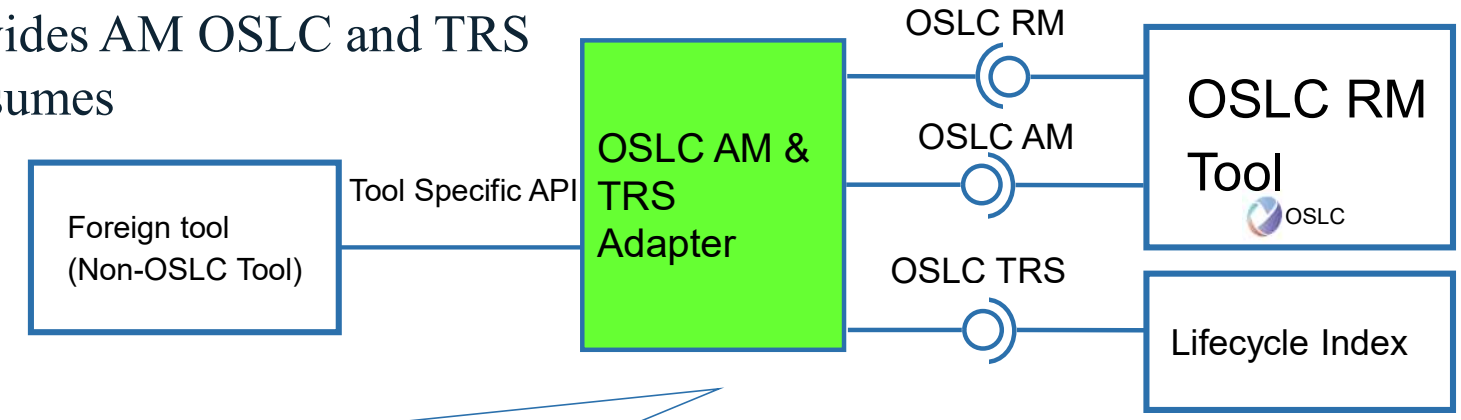
An Eclipse project aimed at accelerating the development of OSLC-compliant solutions.

- Supports Java developers
- Lyo relies on the Eclipse foundation's governance and hosting support
  - Lyo is NOT dependent on the use of the Eclipse IDE
- Licensing
  - Open-source, under the Eclipse Public License (EPL)
  - Commercial-friendly license

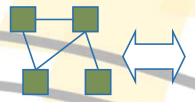
http://eclipse.org/lyo

Example: Integration of a “foreign” design tool (AM)

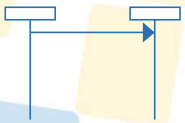
Adapter provides AM OSLC and TRS
Adapter consumes



Data Transformation
(internal storage to RDF)



Conversion of API Calls

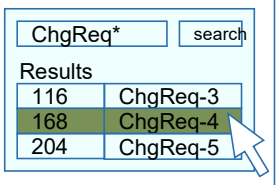


Data Publication
(via TRS spec for
indexing)

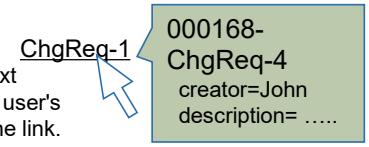
Tracked Resource Set
+ Change Log

Linked data
Association of CM resources

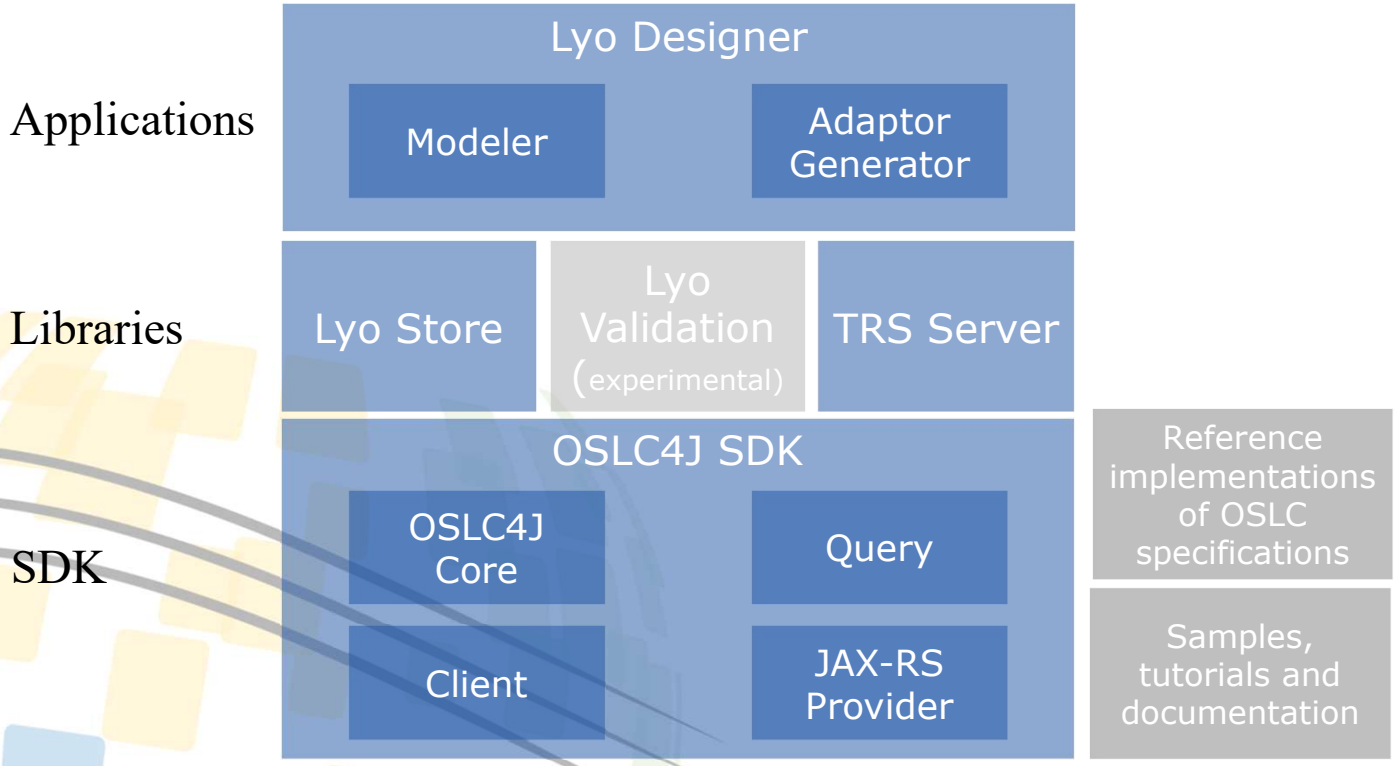
Selection UI
Allow a user to pick a
resource that is managed by
an OSLC Service



Preview UI
Show a user in-context
information when the user's
mouse hovers over the link.



Lyo Features - An Overview



JEKO

Eclipse/Lyo Designer

A modelling environment to develop OSLC domain tools

Model-Driven OSLC development

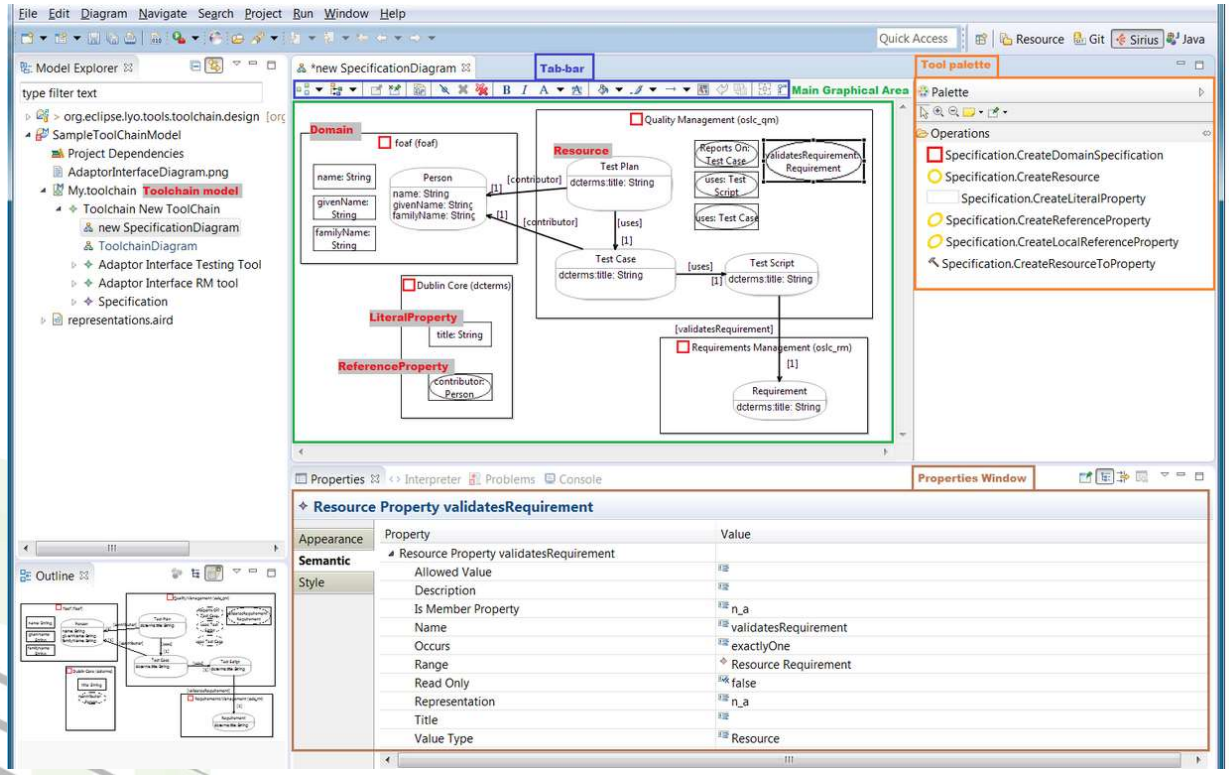
- Graphical notation for Linked Data and OSLC (DSL)
- Generates OSLC server Java code

Complete lifecycle support

- Specifications
- Implementation
- Adaptor testing
- Integration testing
- Automatic code & test generation

Support for multiple perspectives

- Tool-Data ownership
- Domain-data specifications
- Adaptor implementation model



Slide 90

JEK0 These are some very very old pictures.
Will you integrate the new slides I shared?
Jad El-Khoury, 2023-06-12T11:41:12.304

Lyo in use

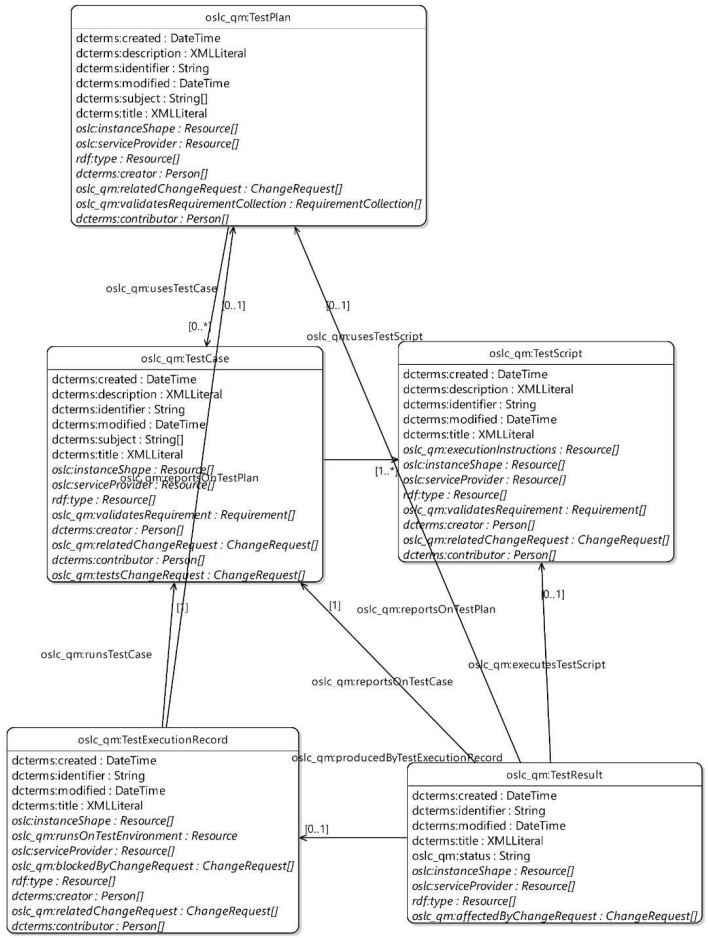
- The eclipse Lyo project
- Lynx designer as an example extension of Lyo
  - https://youtu.be/Quh8T6SvGuA

Working with LYO designer

1. Develop the Domain Specification(s)
   a. Model the Domain Specification(s)
   b. Generate corresponding Java source code
2. Develop the OSLC Server
   a. Model the server
   b. Generate implementation
   c. Develop code to connect to backend application
   d. Fine tune generated code – if necessary
3. Deploy & run
4. Repeat 1-3 (Lyo generation supports incremental development)

1a. Model Domain specifications

- You can import existing models of the OSLC Domain Specifications
  - https://github.com/eclipse/lyo
- You can also define your own extensions, or specific domain specifications.



1b - Generating code from Domain specifications

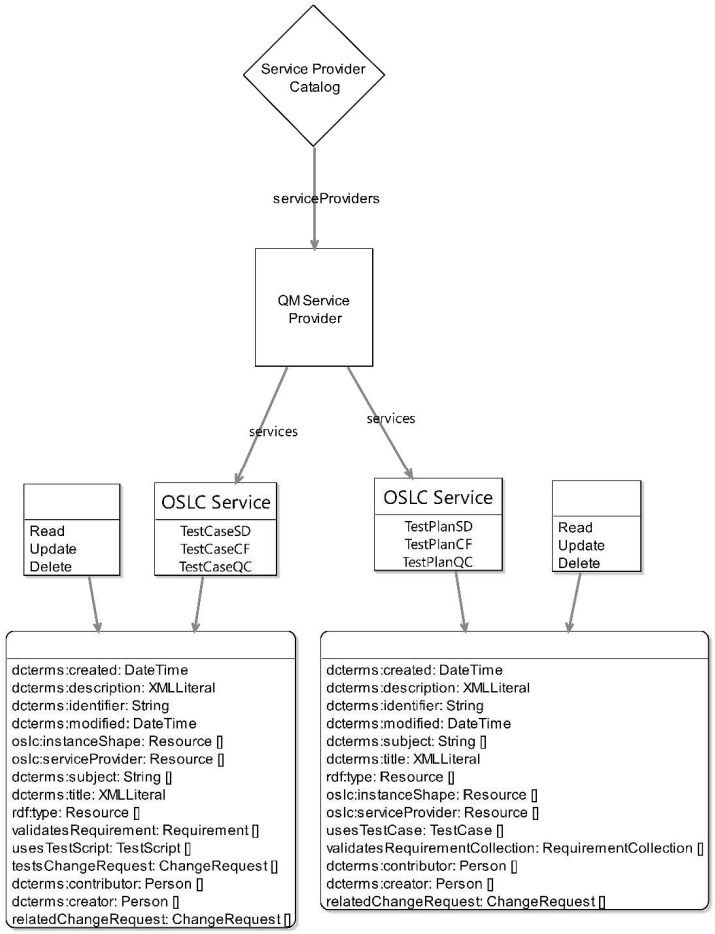
Java classes corresponding to the modelled OSLC resources

- All necessary attributes to create & handle instances of OSLC Resources.
- OSLC-annotations define the mapping between Java instances and RDF resources.
- OSLC4J can marshal/unmarshal such instances to/from any RDF-format

```
@OslcNamespace(Oslc_rmDomainConstants.REQUIREMENT_NAMESPACE)
@OslcName(Oslc_rmDomainConstants.REQUIREMENT_LOCALNAME)
@OslcResourceShape(..., describes = Oslc_rmDomainConstants.REQUIREMENT_TYPE)
public class Requirement extends AbstractResource {
    @OslcName("identifier")
    @OslcPropertyDefinition(DctermsDomainConstants.DUBLIN_CORE_NAMSPACE + "identifier")
    @OslcOccurs(Occurs.ExactlyOne)
    @OslcValueType(ValueType.String)
    @OslcReadOnly(false)
    public String getIdentifier()
    {
        return identifier;
    }
    ...
}
```

2a. Model OSLC Server

- Specify
  a. Discovery capabilities: Catalog & Service Providers, Services
  b. Desired capabilities: Creation Factories, Delegated UIs, Query capabilities.
  c. Resource Operations: Read, Update, Delete
  d. Authentication



2b. Generate OSLC Server

- Produce an almost-complete OSLC4J-compliant running implementation.
- OSLC4J-Annotated Java classes to handle
  - All OSLC service capabilities (complete from end-user request to RDF-response)
  - Discovery capabilities
  - Authentication
  - Basic JSP pages for the html-representation of the dialogs & previews
  - Swagger (OpenAPI) support

```
@OslcService(Oslc_rmDomainConstants.REQUIREMENTS_DOMAIN)
@Path("requirements")
public class ReqWebService {
    ....
    @OslcQueryCapability (

    @GET
    @Path("query")
    @Produces({OslcMediaType.APPLICATION_RDF_XML, ...})
    public Requirement[] queryReq(@QueryParam("where") String
where, ...) {
        ...;
        return resources;
    }

    @GET
    @Path("{requirementId}")
    @Produces({OslcMediaType.APPLICATION_JSON_LD, ...})
    public Requirement getRequirement(... final String
requirementId) {
        Requirement aReq = ...;
        return aReq;
    }

    @OslcCreationFactory (...)
    @POST
    @Path("create")
    public Response createRequirement(...) {
        ...
    }
}
```

2c. Connect to backend Application

- Develop code to connect to backend application, and obtain necessary data to handle each OSLC operation
  - Search items
  - Get/Update/Delete/Create

- Supports incremental development
  - Any manual code changes are preserved upon changes to the model, and subsequent code regeneration.

JEK0

Agenda

1. OSLC goals and digital threads use cases
2. The foundations: W3C linked data
3. Service oriented RESTful HTTP based APIs
4. OSLC core services - discovery, Create/Read/Update/Delete (CRUD), OSLC Query, resource selection, resource preview
5. OSLC domains and core lifecycle ontology
6. OSLC configuration management
7. OSLC TRS (track resource set): What is the purpose of TRS, how it works, and how TRS supports global analytics
8. Creating OSLC adapters - Eclipse LYO and others
9. **Putting it all together: an OSLC taking a tour of an OSLC implementation using a concrete system (UAV) and its cross-domain integrations**

Slide 98

JEK0 This new agenda point does not exist in the previous agenda slides.
Jad El-Khoury, 2023-06-12T11:39:38.689

The Aviary System

Surveillance system based on a drone (UAV), a control unit, and viewer devices

UAV supports manual and autonomous flight. Allows control of camera. Transmits video stream

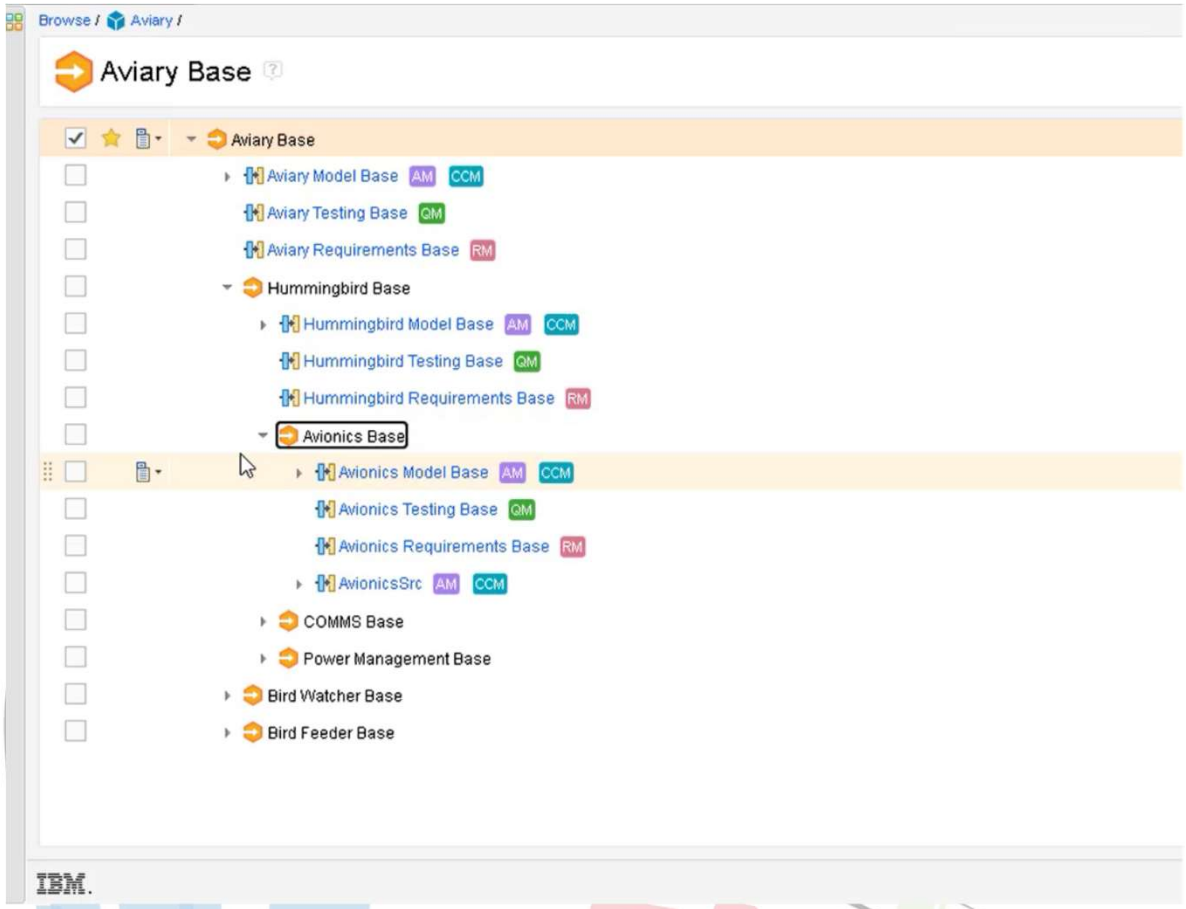
A variant of the base system would also allow moving payload to a target area

Bird Control

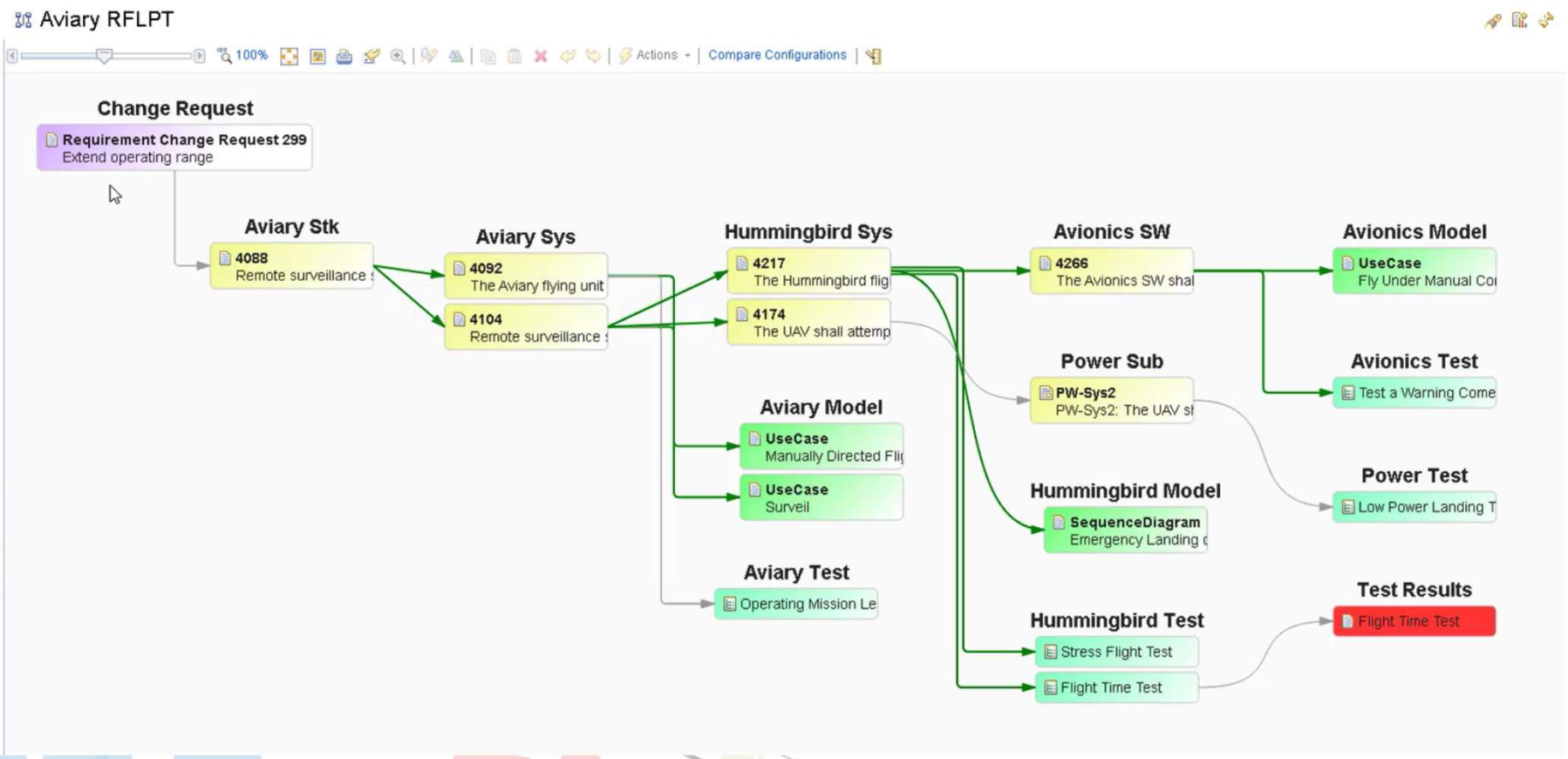
Hummingbird

Bird Watcher

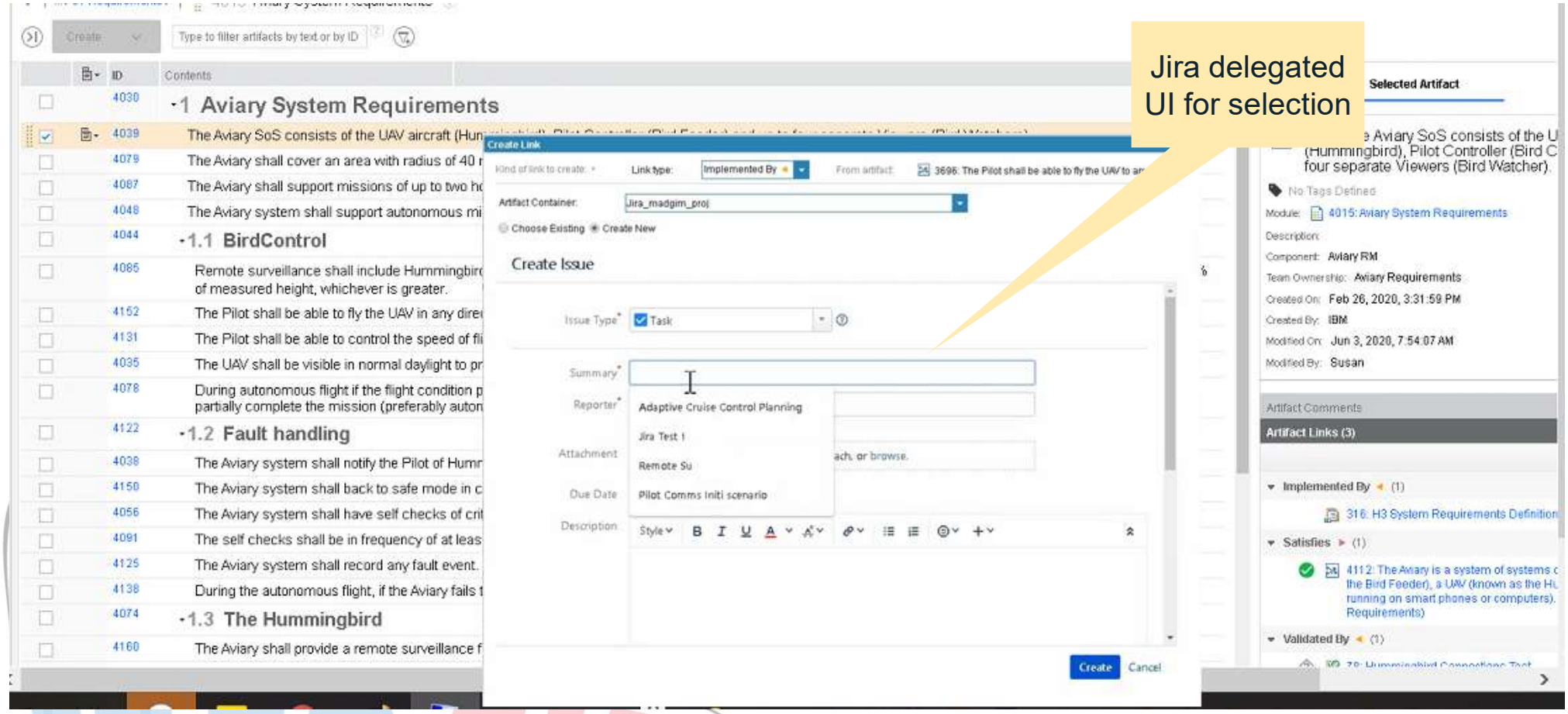
The Aviary system domains described by a global configuration



A digital thread across Aviary originated in a Range change request



Example: Selection/Creation service(1) between DOORS NG and Jira



Visualizing and creating links in a domain tool – requirements to model elements

IBM Engineering Requirements Management DOORS Next (/rm)

Aviary Requirements | Hummingbird

Aviary Base

Project Dashboard Artifacts Reviews Reports

Search Artifac

Hummingbird / | 4164 Hummingbird System Requirements

Create | View: Traceability to Models | Type to filter artifacts by text or by ID

Views

Search Views

- Artifact Type
- forReview
- RM - References DOORS
- RM - Satisfied By
- RM - Satisfies
- RQA
- Surveillance reqs
- Traceability to Functional Analysis
- Traceability to Models
- Tracked By
- Use Case - Manage Power
- Use Case - Manually Directed Flight
- Validated by Test Case
- Variation

ID	Contents	Traced By Architecture Element	Refined By Architecture Element	Satisfied By Architecture Element
4168	COMM_ACTIVE mode of the aircraft shall be defined to be a condition in which there has been communication from the Pilot in the last 5 seconds. In this mode, the aircraft responds to pilot commands.			Block: CommsSubsyst...
4195	The aircraft shall go into COMM_LOST mode should communication be lost between the Pilot and the aircraft for more than 5s.		SequenceDiagram: Co...	
4192	When entering COMM_LOST mode, the aircraft shall stop lateral and forward motion and station-keep at its current position with 0 ground velocity. In this mode, the camera, if on, shall continue to broadcast its video stream to the best of its ability.		SequenceDiagram: Co...	
4216	All communication with the pilot controller shall be encrypted with a secure connection including uploaded commands and downloaded data and video stream.	UseCase: Securely Co...	SequenceDiagram: Adj...	
5122	All communication with the viewers shall be encrypted with a secure connection including uploaded commands and downloaded data and video stream.		SequenceDiagram: Adj...	
4202	Video stream feed transmission shall			

Showing 56 of 56 (100%)

IBM.

Linking from Jira to Doors NG with a requirement preview

Dashboards | User Dashboards | Aviary | Practice Content | UAF

Projects | Issues | Boards | Create | Search

Aviary 2.0 / AV-5

Video Feed

Edit | Add comment | Assign | More | To Do | In Progress | Done | Admin | Export

Details

Type:	Story	Status:	TO DO (View Workflow)
Priority:	Highest	Resolution:	
Affects Version/s:	Aviary 2.0	Fix Version/s:	
Labels:	None		

People

Assignee: Dave

Description

Click to add description

Attachments

Drop files to attach, or browse.

Issue Links

tracks requirement

4166: The UAV video feed shall have 720p resolution and frame rate (25fps at ...

Activity

All | Comments | Work Log | History | Activity

There are no comments yet on this issue.

4166: The UAV video feed shall have 720p resolution and frame rate (25fps at 1280x72...

Location

Aviary Requirements | Hummingbird | Hummingbird System Requirements

Attributes

Type:	System Requirement	Format:	Text
Accepted:	false	Clarity:	
cmint_applied_RCR_numbers:		Description:	
Issues Found by RQA:		Need:	
Priority:		pvRestriction:	
Questions:		Requirement Type:	Functional
RQA Score:		Schedule:	
Source:		Stability:	
Status:		Team Ownership:	Aviary Requirements
Test Criteria:	None	Test Status:	
Verifiability:	false	Verification Method:	Not Set

In Modules

System Specification: 4164

Show More

Dashbords with traceability views to Jira stories

All Hummingbird SAFe - Program and Team Development Dashboards /

Hummingbird - Program Management

Auto-save | Save

Program Overview | Reviews | DO178 Traceability | DO178 Traceability Gap Analysis | RFLPT | Lifecycle Statistics | Continuous Integration | Jira | Add Widget

Jira Stories for Aviary

Filters | Configuration : Aviary Base

10 Items Per Page | Previous 1 - 5 of 5 items Next

Jira Story	Elaborated by Architecture Element	Tested by Test Case	Tracks Requirement
Camera Zoom	Surveil	Camera Zoom Test	The Hummingbird camera zoom shall be commandable via Pilot commands from -4x to + 10x wi fidelity of at least 5 levels of zoom.
Camera Focus		Camera Focus Test	The hummingbird camera focus shall be settable from 10m to infinity.
Video Feed			The UAV video feed shall have 720p resolution and frame rate (25fps at 1280x720 pixel resolu tion) or better.
Camera Direction			The Hummingbird camera shall b able to point straight down from the Hummingbird and up to 60% from vertical down in any direc- tion, under Pilot control, includin roll, pitch and yaw angles inde- pendent of the attitude of the U

Jira Traceability

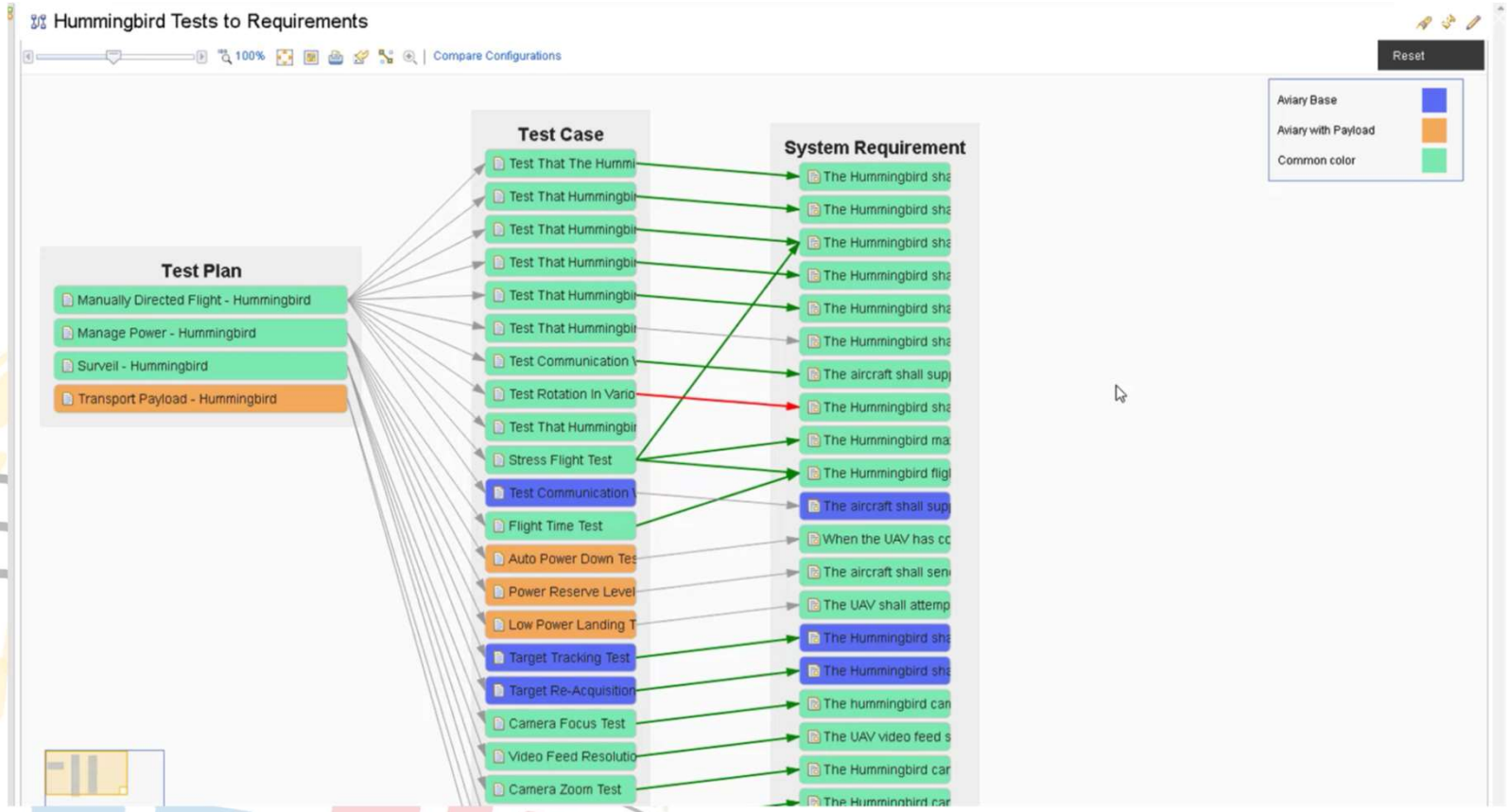
100% | Dependency Analysis

Jira Open Stories by Priority for Aviary

Filters | View as: Graph | Configuration : Aviary Base

Jira Bugs

Comparing two Aviary digital threads configurations focusing on tests to requirements traceability



Thank You!

33rd Annual INCOSE
international symposium

hybrid event

Honolulu, HI, USA
July 15 - 20, 2023