



Semantic Evaluation at Large Scale Tutorial

Repositories Management

Michael Schneider, FZI Karlsruhe
Adrian Marte, STI Innsbruck

Outline

- Overview
- Architecture
- Managing Entities
- Test Data Repository Service
- Tools Repository Service
- Results Repository Service
- Links to Resources

SEALS Repositories

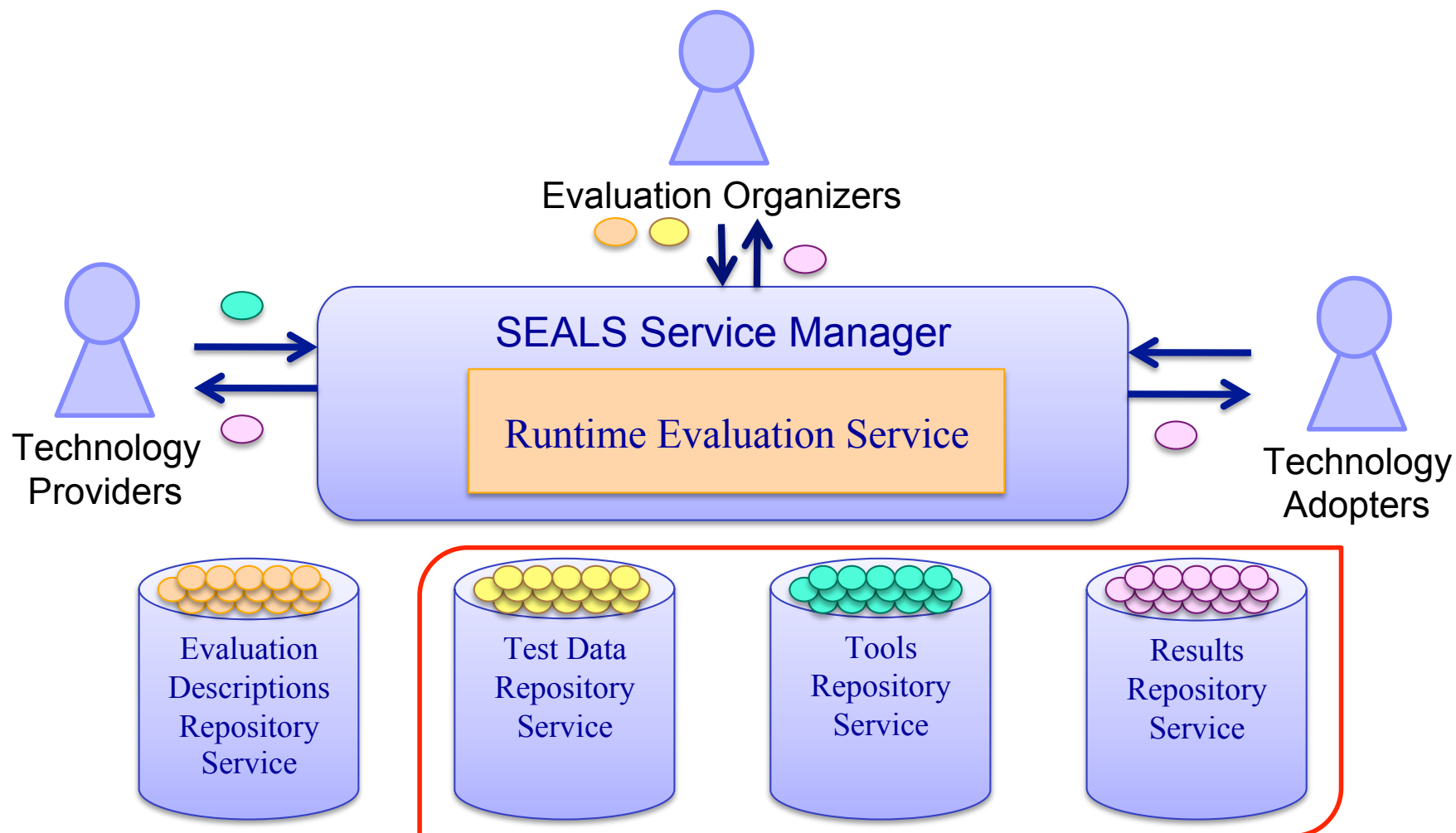
OVERVIEW

5/5/11

3

SEALS
Semantic Evaluation at Large Scale

Overall Architecture



SEALS Repositories

- Test Data Repository Service
 - Persistently stores test data sets
 - Stores references to external test data sets
 - Stores synthetic test data generators
 - Generates synthetic test data
- Tools Repository Service
 - Stores the tools involved in an evaluation
- Results Repository Service
 - Stores the raw result of an evaluation
 - Stores interpretations over one or more raw results

SEALS Repositories

ARCHITECTURE

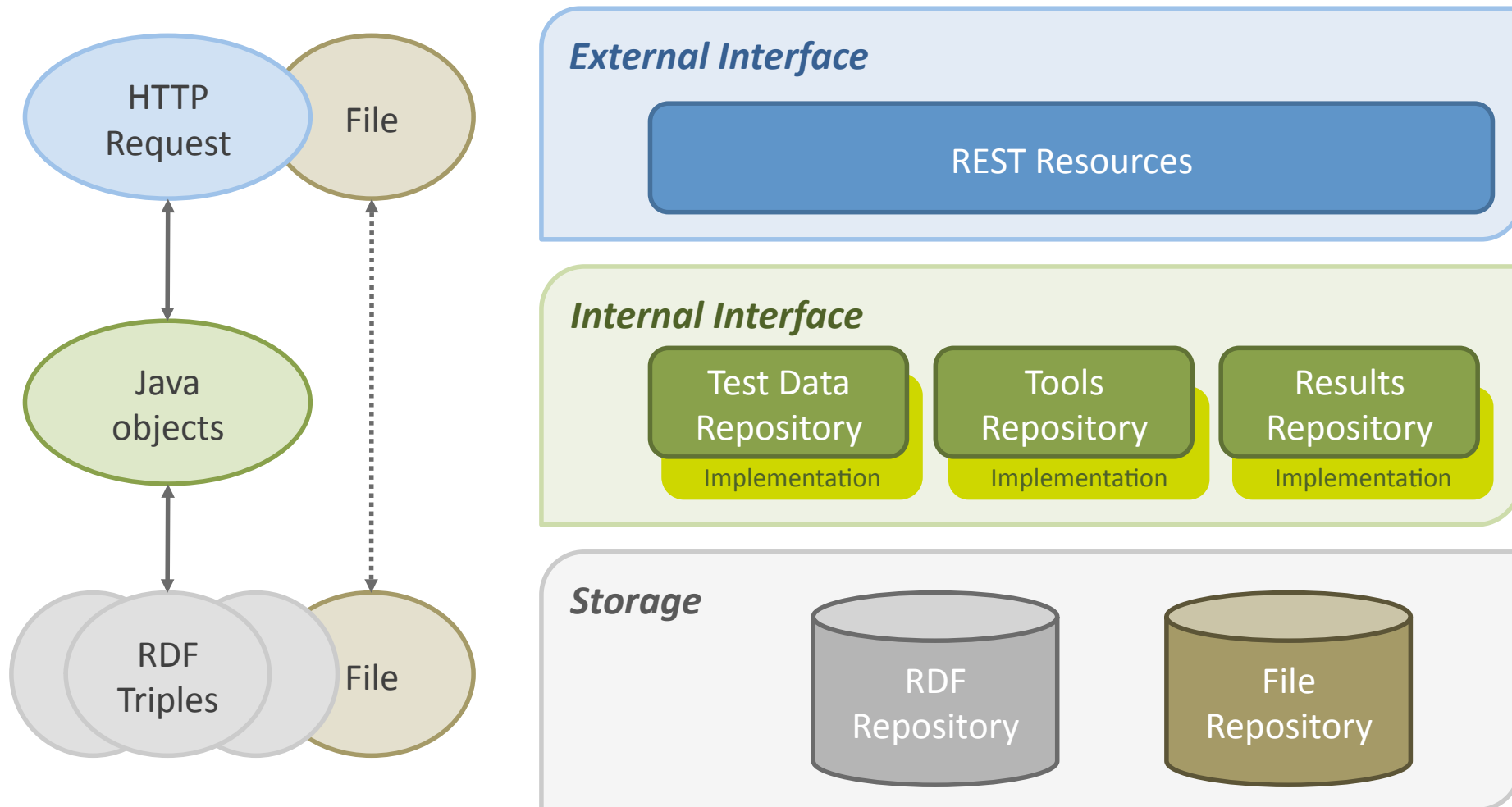
Artifacts and Artifact Versions

- Repositories store artifacts and artifact versions
 - Artifact
 - Usually a collection of versions
 - Usually does not have data associated with it
 - Test data collection, tool, result, interpretation, etc.
 - E.g., the tool “Protégé” is an artifact
 - Artifact version
 - Concrete version of an artifact
 - Comes with concrete data
 - E.g., the tool version “Protégé 4.1” is an artifact version

Metadata

- Artifacts and artifact versions can be described using the SEALS metadata ontology
- Repositories store data and metadata separately
- Used to identify, interlink and search for artifacts and artifact versions

Repositories Architecture

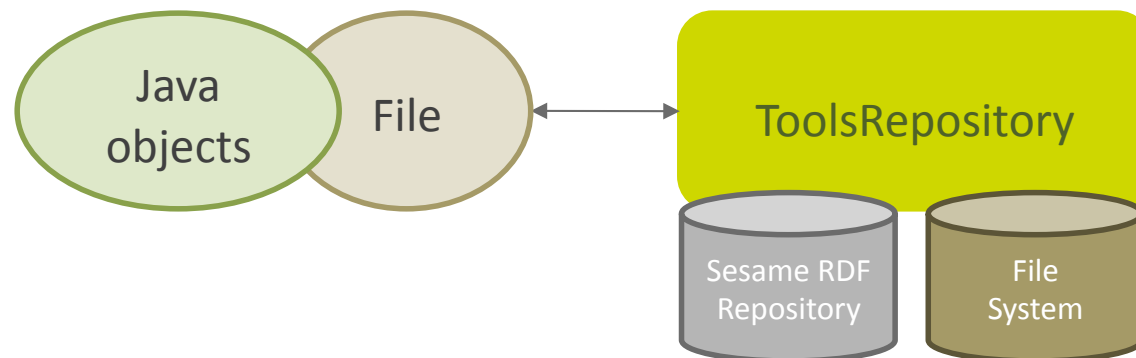


Storage Layer

- *File Repository*
 - Stores submitted tools, generators, test data, etc.
 - Uses underlying file system
 - Allows for storage of many large files
- *RDF Repository*
 - Stores RDF encoded metadata
 - Allows to pose SPARQL queries over metadata
 - Based on Sesame RDF repository
 - Can be accessed remotely over HTTP

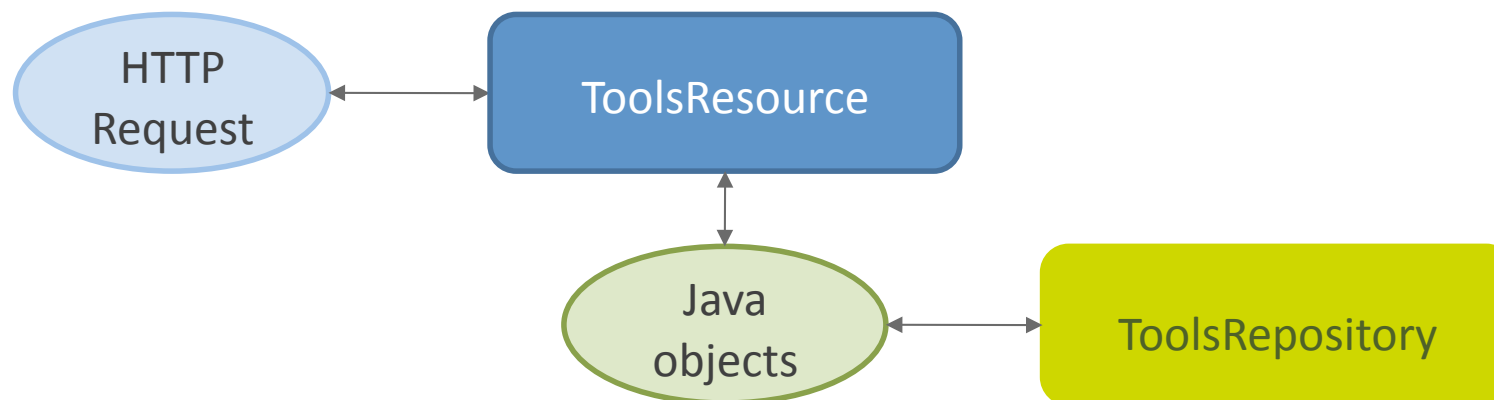
Internal Interface Layer

- Java interfaces for repositories
- Use underlying storage layer
- Allows repository functionality to be exposed using different protocols
- Allows other implementations that use different storage mechanisms



External Layer

- RESTful interface based on Restlet
 - Framework for RESTful applications
 - Open-source
- HTTP resources represent repository functionality
- Dispatch HTTP requests to corresponding repositories
- Use underlying repository interface implementations



SEALS Repositories

MANAGING ENTITIES

5/5/11

13

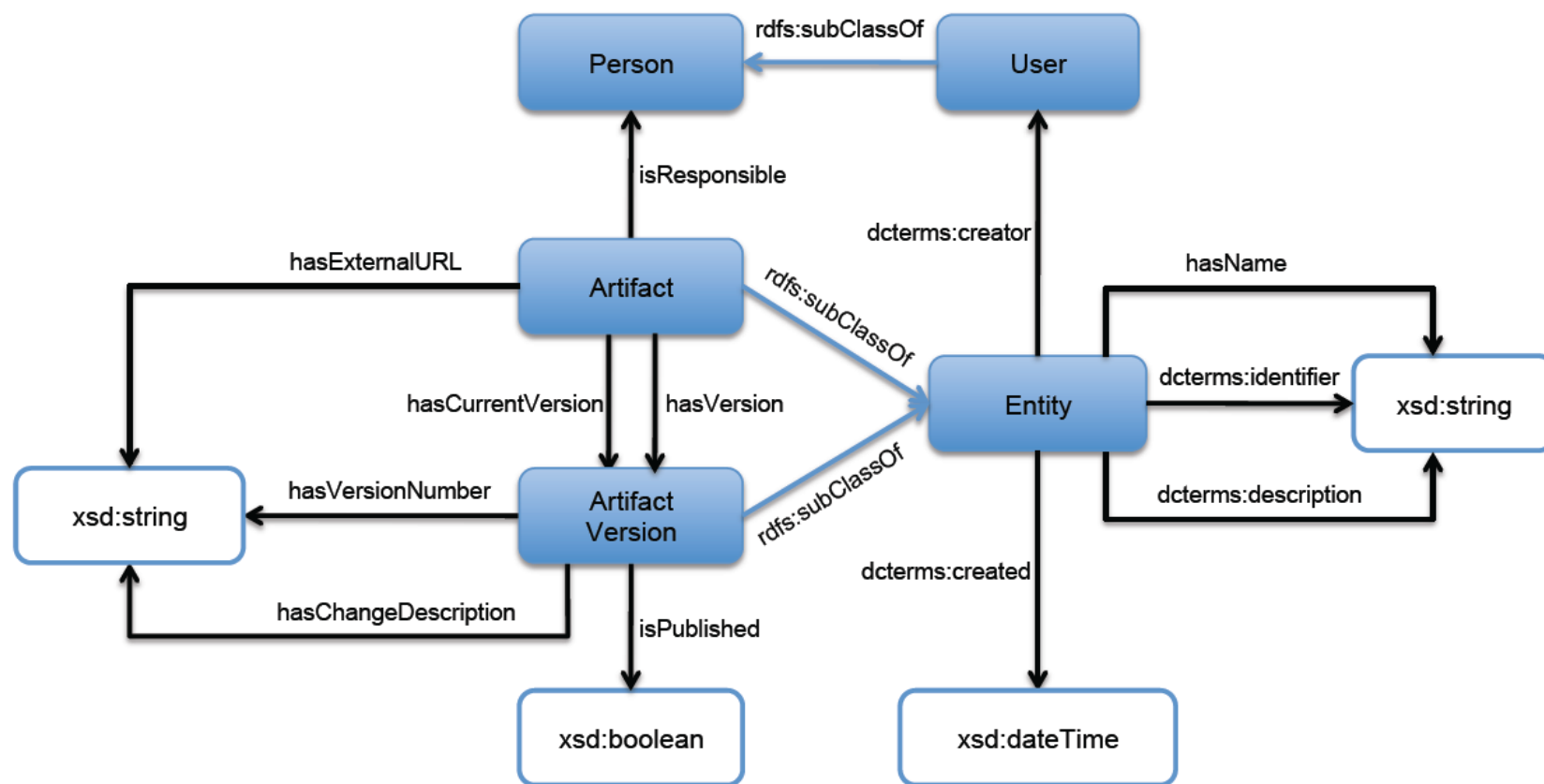
SEALS
Semantic Evaluation at Large Scale

SEALS Metadata Ontology

- Used to describe artifacts and artifact versions
- Supports *user-provided* and *platform-managed* metadata
- Reuses Dublin Core Metadata Initiative vocabulary
- Available as an OWL ontology¹

¹ <http://www.seals-project.eu/ontologies/SEALSMetadata.owl>

SEALS Metadata Ontology



Describing Entities

- User-provided metadata
 - Provided when entity is registered, created or updated in repository
 - Can be changed by user (or platform component)
 - E.g., textual description of test data, classification of a tool
- Platform-managed metadata
 - Automatically generated during evaluation lifecycle
 - Exclusively managed by platform
 - E.g., creation date, allocated computing resources

Describing Entities

- Metadata is RDF/XML encoded
- Submitted to repository when registering or updating entities in the repository
- Used to identify, interlink and search for entities stored in the repositories

Describing Entities: Example

- Metadata for tool “IRIS” (Datalog reasoner)

```
<rdf:RDF xmlns:rdfs="..." xmlns:rdf="..." xmlns:dcterms="..."  
    xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">  
  <seals:Tool>  
    <seals:hasName rdf:datatype="&xsd:int">IRIS</seals:hasName>  
    <dcterms:description rdf:datatype="&xsd:int">  
      Open-source Datalog reasoner implemented in Java.  
    </dcterms:description>  
    <seals:hasExternalURL rdf:resource="http://www.iris-reasoner.org" />  
    <seals:hasToolCategory rdf:resource="http://www.seals-project.eu/  
      ontologies/SEALSMetadata.owl#ReasonerOrStorageTool" />  
  </seals:Tool>  
</rdf:RDF>
```

Storing Entities

- Repository functionality exposed via RESTful HTTP interface
 - *POST*: Creates HTTP resource for an artifact or artifact version, which is returned in `Content-Location` header
 - *PUT*: Updates resource and respective artifact or artifact version
 - *GET*: Retrieves representation of resource, i.e., data or metadata of artifact or artifact version
 - *DELETE*: Removes resource and respective artifact or artifact version

Storing Entities

- Artifact along with its RDF/XML encoded metadata is registered in repository using a HTTP POST request
- Artifact version along with metadata and data (in the form of a ZIP file) is submitted to the resource of the artifact using a HTTP POST request
- GET, PUT, DELETE or POST can be used to retrieve, update, delete or add additional artifacts and artifact versions
- POST is also used to publish an artifact, in which case it is allowed to be used in an evaluation

Storing Entities: Example

- Add tool “IRIS 0.7.3” to Tools Repository Service (TRS)
 - Create metadata for tool IRIS
 - Register tool in TRS by submitting metadata with HTTP POST request

```
POST /trs-web/tools
Host: seals-test.sti2.at
Accept: application/rdf+xml
Content-Type: application/x-www-form-urlencoded
```

```
metadata=<?xml version="1.0" encoding="UTF-8"?><rdf:RDF
xmlns:seals=...
```

Storing Entities: Example

- Add tool “IRIS 0.7.3” to Tools Repository Service (TRS)
 - Create metadata for version 0.7.3 of IRIS and package tool in ZIP file
 - Add tool version along with metadata and ZIP file to repository by submitting HTTP POST request to resource of tool

```
POST /trs-web/tools/35ded060-287e-11e0-91fa-0800200c9a66/  
Host: seals-test.sti2.at  
Accept: application/rdf+xml  
Content-Type: multipart/form-data, boundary=AaB03x
```

```
--AaB03x  
Content-Disposition: form-data; name="metadata"
```

```
<?xml version="1.0" encoding="UTF-8"?><rdf:RDF xmlns:seals=...
```

```
--AaB03x  
Content-Disposition: form-data; name="data"; filename="IRIS_0.7.3.zip"  
Content-Type: application/zip
```

```
... contents of IRIS_0.7.3.zip
```

Storing Entities: Example

- Add tool “IRIS 0.7.3” in Tools Repository Service (TRS)
 - Retrieve metadata using a HTTP GET request on the resource of the tool version
 - Accept header defines what type of representation is requested (metadata or data)

```
GET /trs-web/tools/35ded060-287e-11e0-91fa-0800200c9a66/  
    2d3c6d00-287e-11e0-91fa-0800200c9a66/  
Host: seals-test.sti2.at  
Accept: application/rdf+xml
```

```
GET /trs-web/tools/35ded060-287e-11e0-91fa-0800200c9a66/  
    2d3c6d00-287e-11e0-91fa-0800200c9a66/  
Host: seals-test.sti2.at  
Accept: application/zip
```

Accessing Entities

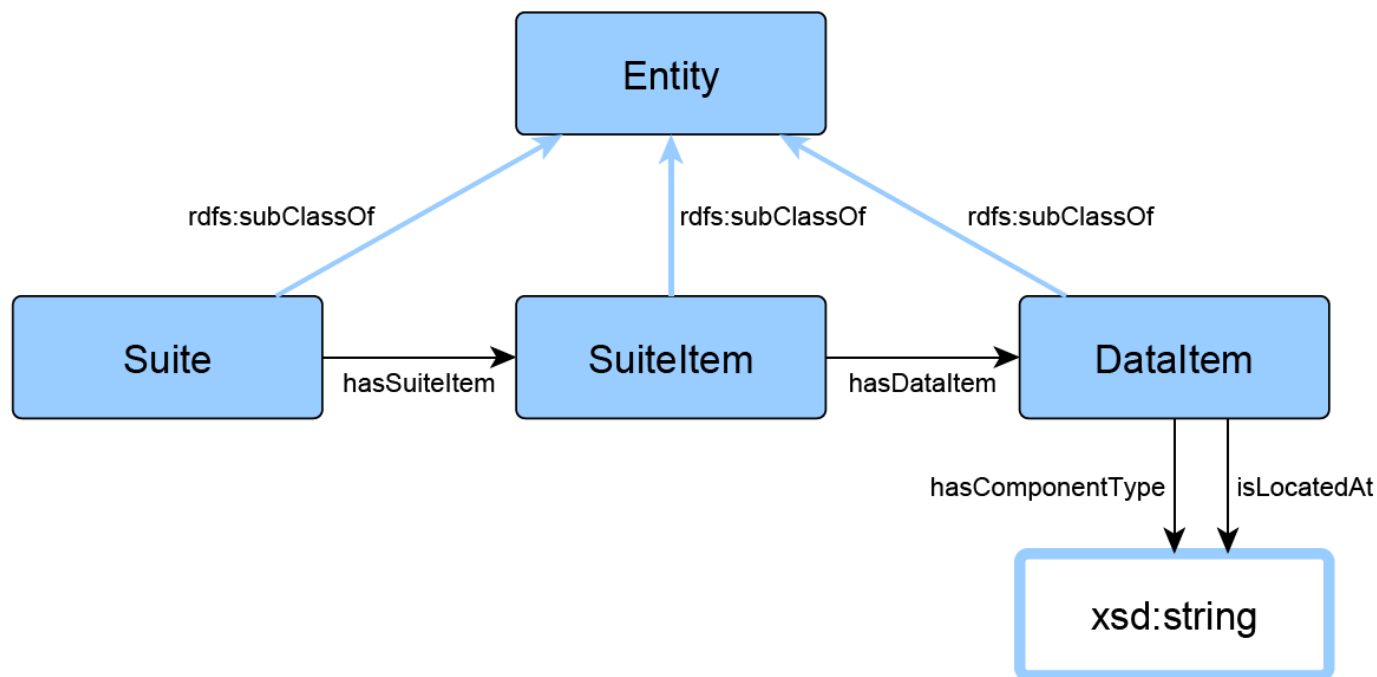
- Repositories create unique HTTP resources for submitted artifacts and artifact versions using the (auto-generated) `dcterms:identifier` value in the metadata
- Identifier is auto-generated by repository if not defined in the metadata
- Each repository defines URL schemes to access and manage the stored entities

Describing Suites

- Suite metadata describes suites of test data, raw results and interpretations according to suite ontology
- Enables fine-grained access via REST interface on contents of data file, using the identifiers and component types of the items defined in the suite metadata
- RDF/XML encoded suite metadata is stored separately in a file inside the data file

SEALS Suite Ontology

- Extends SEALS metadata ontology



Describing Suites: Example

- Test data suite for ontology matching: conference test suite

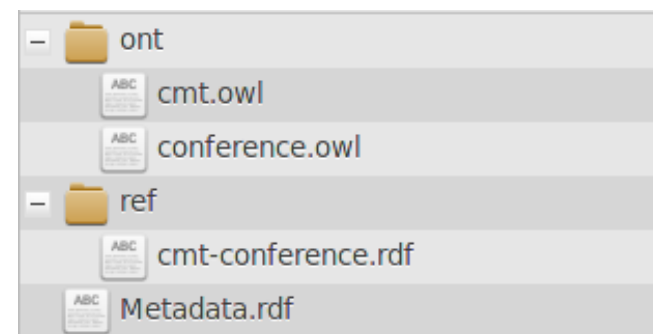
```
conference:ConferenceSuite
  a seals:Suite;
  seals:hasSuiteItem conference:cmt-conference.

conference:cmt-conference
  a seals:SuiteItem;
  dc:identifier "cmt-conference";
  seals:hasDataItem
    conference:cmt-source,
    conference:conference-target,
    conference:cmt-conference-ref.

conference:cmt-source>
  a seals:DataItem;
  dc:identifier "cmt-source";
  seals:hasComponentType "source";
  seals:isLocatedAt "../ont/cmt.owl".

conference:cmt-conference-ref
  a seals:DataItem;
  dc:identifier "cmt-conference-ref";
  seals:hasComponentType "reference";
  seals:isLocatedAt "../ref/cmt-conference.rdf";.

...
```

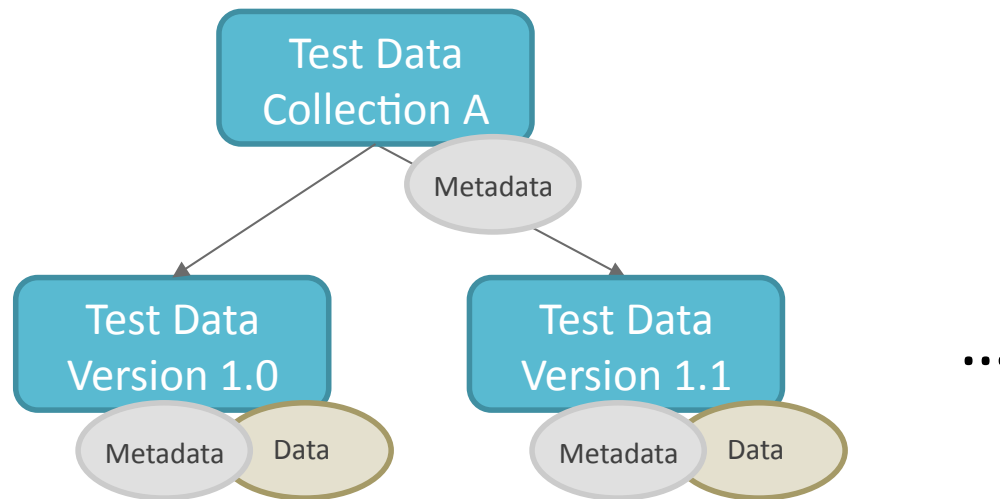


SEALS Repositories

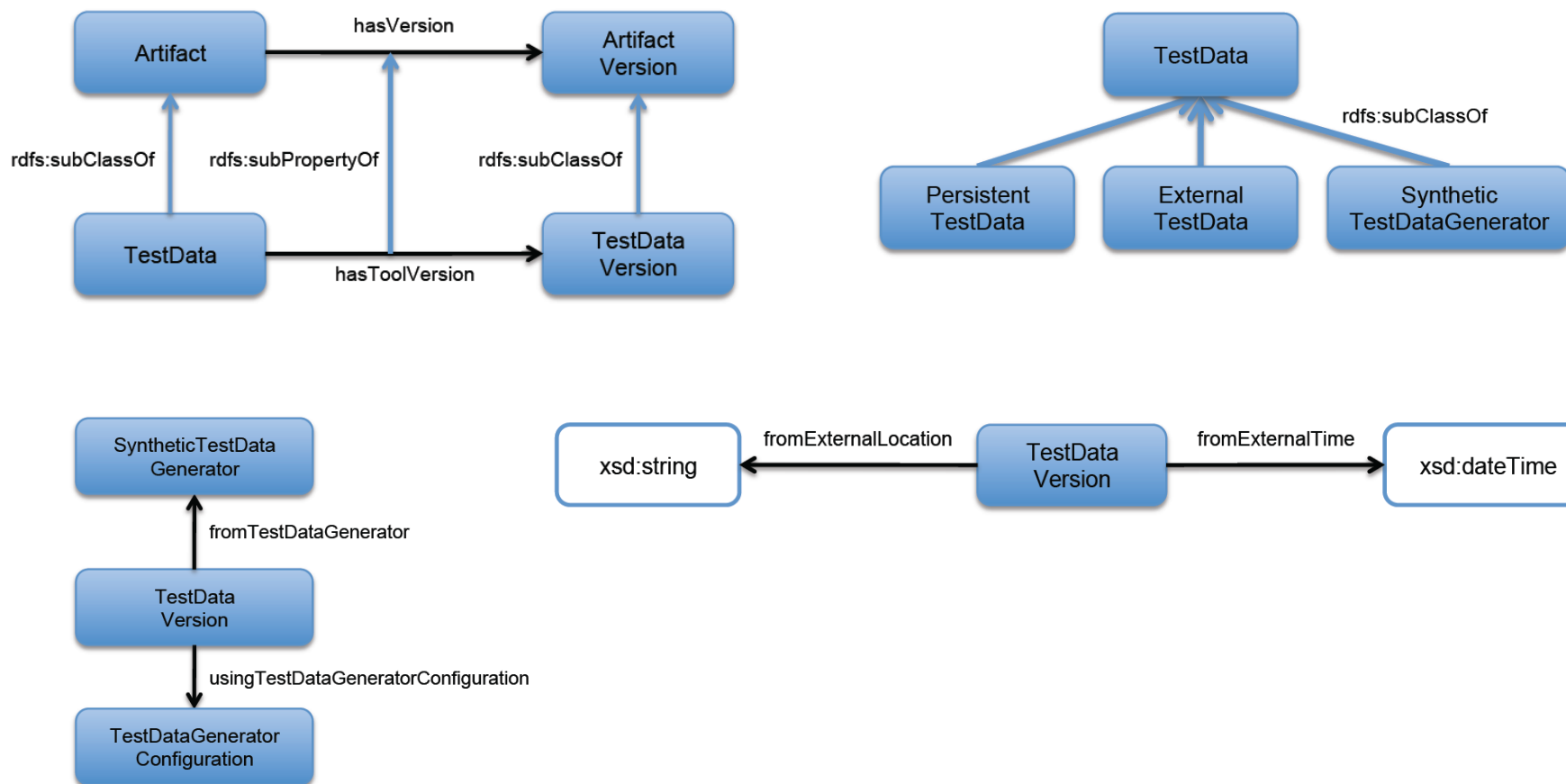
TEST DATA REPOSITORY SERVICE

Test Data Repository Service

- Manages *test data collections* and *test data versions* used for evaluations
- Stores persistent test data (suites)
- Allows to reference and persist external test data
- Provides means for generating and storing synthetic test data using a user-submitted test data generator



Describing Test Data



Describing Test Data: Example

- Ontology matching test data: conference testsuite

```
<rdf:RDF xmlns:rdfs="..." xmlns:rdf="..." xmlns:dcterms="..."
  xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">
  <seals:PersistentTestData>
    <seals:hasName rdf:datatype="&xsd:string">Conference Testsuite</seals:hasName>
    <dcterms:description rdf:datatype="&xsd:string">
      The conference testsuite.
    </dcterms:description>
  </seals:PersistentTestData>
</rdf:RDF>

<rdf:RDF xmlns:rdfs="..." xmlns:rdf="..." xmlns:dcterms="..."
  xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">
  <seals:TestDataVersion>
    <seals:hasVersionNumber rdf:datatype="&xsd:string">2010</seals:hasVersionNumber>
    <dcterms:description rdf:datatype="&xsd:string">
      Version of the conference dataset used in 2010.
    </dcterms:description>
  </seals:PersistentTestData>
</rdf:RDF>
```

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata	GET	RDF	Search/browse
[domain]/testdata/persistent	POST	RDF	Register new persistent test data collection
[domain]/testdata/persistent/[collection]	GET	RDF	Retrieve metadata of test data collection
	GET	ZIP	Retrieve current test data version
	PUT	RDF	Update metadata of test data collection
	POST	RDF	Add test data version

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata/persistent/[collection]/[version]	GET	RDF	Retrieve metadata of test data version
	PUT	RDF	Update metadata of test data version
	DELETE		Remove test data version
	POST	RDF	Publish test data version
[domain]/testdata/persistent/[collection]/ [version]/ pub	GET	RDF	Retrieve metadata of test data version
	GET	ZIP	Retrieve data of publish test data version
	DELETE	ZIP	Unpublish test data version

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata/persistent/[collection] [version]/suite/	GET	RDF	Retrieve suite metadata
[domain]/testdata/persistent/[collection]/ [version]/suite/[suite item]/item/[data item]	GET		Retrieve data item
[domain]/testdata/persistent/[collection]/[version]/ suite/[suite item]/component/[component type]	GET		Retrieve component

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata/synthetic	POST	RDF	Register new synthetic test data generator
[domain]/testdata/synthetic/[generator]	GET	RDF	Retrieve metadata of synthetic test data generator
	GET	ZIP	Generate transient test data version
	PUT	RDF	Update metadata of synthetic test data generator
	DELETE		Deregister synthetic test data generator
	POST	RDF	Publish synthetic test data generator

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata/synthetic/[generator]/pub	DELETE		Unpublish synthetic test data generator
[domain]/testdata/persistent/[collection]	POST	RDF	Persist generated test data version

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata/external	POST	RDF	Register new external test data collection
[domain]/testdata/external/[collection]	GET	RDF	Retrieve metadata of external test data collection
	GET	ZIP	Retrieve current external test data version
	PUT	RDF	Update metadata of external test data collection
	DELETE		Deregister external test data collection
	POST	RDF	Add external test data version

Accessing Test Data

URI	Op.	Accept	Effect
[domain]/testdata/external/[collection]/[version]	GET	RDF	Retrieve metadata of external test data version
	PUT	RDF	Update metadata of external test data version
	DELETE		Remove test data version
[domain]/testdata/persistent/[collection]	POST	RDF	Persist external test data version

Synthetic Test Data Generator

- Test Data Repository Service supports storage and on-the-fly generation of synthetic test data using a user-defined generator
- Synthetic Test Data Generator API enables implementation of a generator

Synthetic Test Data Generator

- *ZIP* file contains manifest, binaries and required libraries for execution
 - *bin* folder: contains synthetic test data generator implementation packaged in a JAR file
 - *libs* folder: default folder containing required libraries as JAR files
 - *resources* folder: contains resources used during test data generation
 - manifest.xml: defines required information for execution
 - Root element: `<generator>...</generator>`
 - Executable class: `<executable>org.example.Generator</executable>`
 - Libraries: `<libraries><library>some/folder/</library></libraries>`

Synthetic Test Data Generator API

- *TestDataGenerator* interface
 - Must be implemented by synthetic test data generator
 - Allows generation of test data (e.g. ontologies) for specified configuration files using a data manager

```
public interface TestDataGenerator {  
  
    public void generate(File... configurations);  
    public void setDataManager(DataManager dataManager);  
  
}
```

Synthetic Test Data Generator API

- *DataManager* interface
 - Allows retrieval of external and internal (inside generator package) resources
 - Creates test data according to suite ontology
 - An instance of this interface is passed to the test data generator before execution

```
public interface DataManager {  
  
    public DataItem getDataItem(String dataItemId);  
    public DataItem getDataItem(String suiteItemId, String dataItemId);  
    public DataItem getDataItem(URL url);  
    public DataItem createDataItem(String suiteItemId, String dataItemId);  
  
}
```

Synthetic Test Data Generator API

- *DataItem* interface
 - Provides access to existing or newly created data items

```
public interface DataItem {  
  
    public boolean isEmpty();  
    public OutputStream getOutputStream();  
    public InputStream getInputStream();  
    public boolean delete();  
  
}
```

Synthetic Test Data Generator: Example

- Example of a synthetic test data generator
- Wraps data generator of LUBM benchmark
- Generates instances of University Ontology for a specified RDF/XML encoded configuration

```
<rdf:RDF xmlns:rdfs="..." xmlns:rdf="..."  
    xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">  
<seals:TestDataGeneratorConfiguration>  
    <lubm:univNum rdf:datatype="&xsd:int">1</lubm:univNum>  
    <lubm:startIndex rdf:datatype="&xsd:int">0</lubm:startIndex>  
    <lubm:seed rdf:datatype="&xsd:int">1337</lubm:seed>  
    <lubm:ontology rdf:datatype="&xsd:string">  
        http://www.example.org/ns/  
    </lubm:ontology>  
</seals:TestDataGeneratorConfiguration>  
</rdf:RDF>
```

Synthetic Test Data Generator: Example

```
package eu.sealsproject.generator.uba
public class UbaGenerator implements TestDataGenerator {
    private DataManager dataManager;

    @Override
    public void generate(File... configurations) {
        // Create the configuration object, which initializes the parameters with default values.
        UbaConfiguration configuration = new UbaConfiguration();

        if (configurations.length > 0) {
            // Load the RDF encoded configuration file and overwrite the parameters in the configuration object.
            configuration.load(configurations[0]);
        }

        // Retrieve the parameters from the configuration.
        int univNum = configuration.getUnivNum();
        int startIndex = configuration.getStartIndex();
        int seed = configuration.getSeed();
        String ontology = configuration.getOntology();

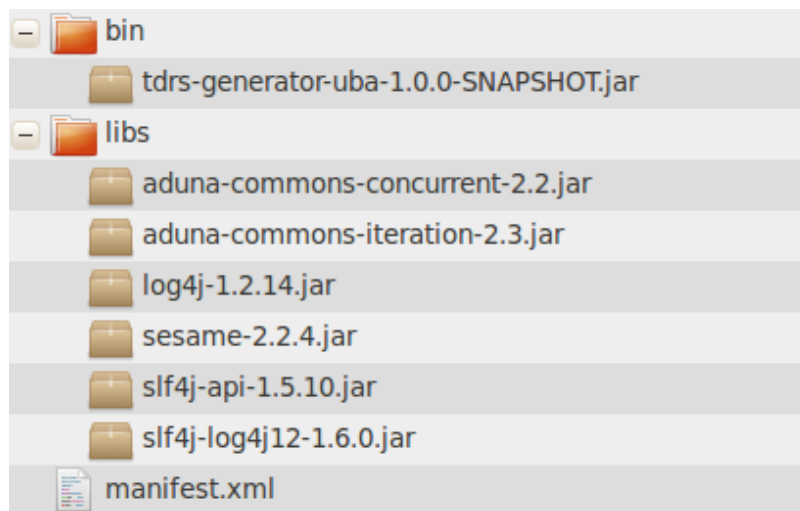
        // Create the UBA generator.
        Generator generator = new Generator();

        // Start generating the test data and store it using the data manager.
        generator.start(univNum, startIndex, seed, ontology, dataManager);
    }

    @Override
    public void setDataManager(DataManager dataManager) {
        // Initialize the data manager.
        this.dataManager = dataManager;
    }
}
```

Synthetic Test Data Generator: Example

- *ZIP* file contains manifest, binaries and required libraries for execution



manifest.xml :

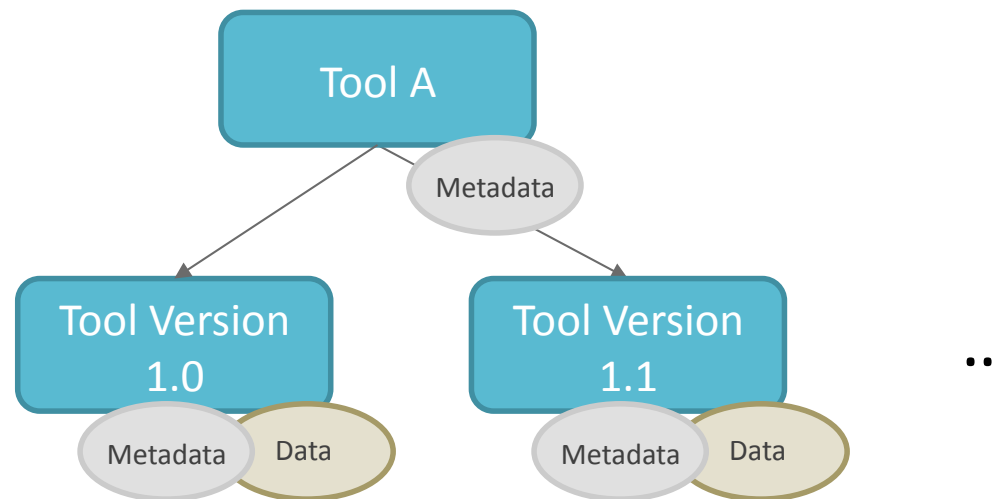
```
<generator>
<executable>
eu.sealsproject.generator.uba.UbaGenerator
</executable>
</generator>
```

SEALS Repositories

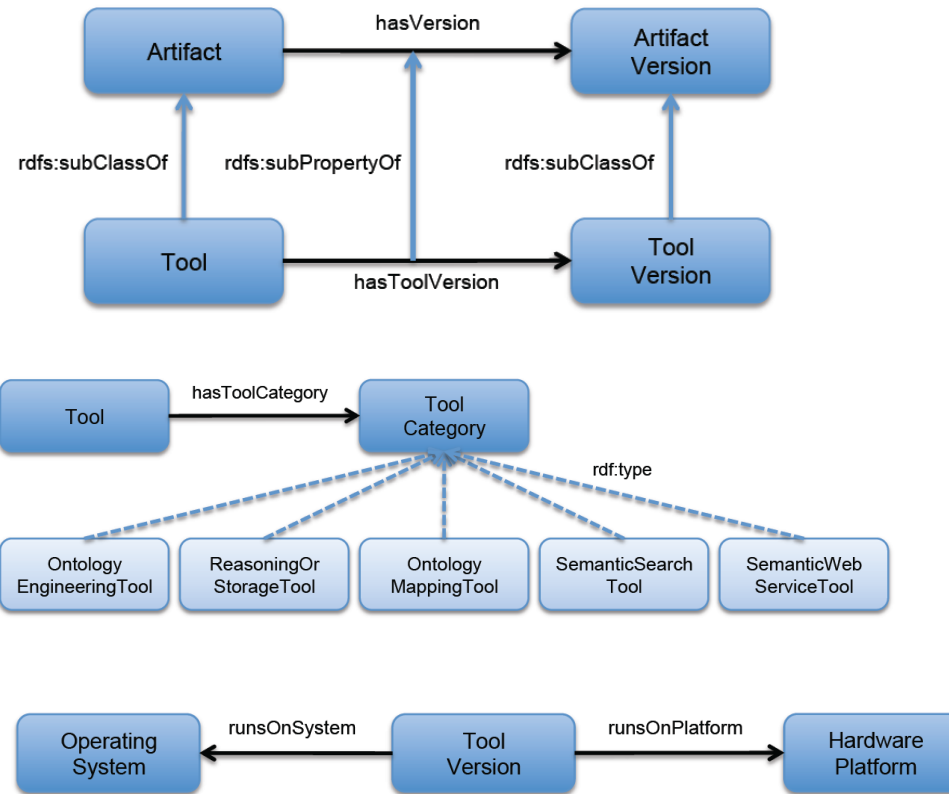
TOOLS REPOSITORY SERVICE

Tools Repository Service

- Manages *tools* and *tool version* that are evaluated on the platform
- Tools are expected to be packaged such that they can be integrated in an evaluation workflow



Describing Tools



Describing Tools: Example

- IRIS Datalog reasoner

```
<rdf:RDF xmlns:rdf="..." xmlns:dcterms="..."
  xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">
  <seals:Tool>
    <seals:hasName rdf:datatype="&xsd:int">IRIS</seals:hasName>
    <dcterms:description rdf:datatype="&xsd:int">
      Open-source Datalog reasoner implemented in Java.
    </dcterms:description>
    <seals:hasExternalURL rdf:resource="http://www.iris-reasoner.org" />
    <seals:hasToolCategory rdf:resource="http://www.seals-project.eu/
      ontologies/SEALSMetadata.owl#ReasonerOrStorageTool" />
  </seals:Tool>
</rdf:RDF>
```

Describing Tools: Example

- IRIS Datalog reasoner

```
<rdf:RDF xmlns:rdf="..." xmlns:dcterms="..."  
  xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">  
  <seals:ToolVersion>  
    <seals:hasVersionNumber rdf:datatype="&xsd:int">  
      0.7.3  
    </seals:hasVersionNumber>  
    <dcterms:description rdf:datatype="&xsd:string">  
      Version 0.7.3 of the IRIS reasoner  
    </dcterms:description>  
  </seals:ToolVersion>  
</rdf:RDF>
```

Accessing Tools

URI	Op.	Accept	Effect
[domain]/tools	GET	RDF	Search/browse
	POST	RDF	Register new tool
[domain]/tools/[tool]	GET	RDF	Retrieve metadata of tool
	GET	ZIP	Retrieve current tool version
	PUT	RDF	Update metadata of tool
	DELETE		Deregister tool
	POST	RDF	Add new tool version

Accessing Tools

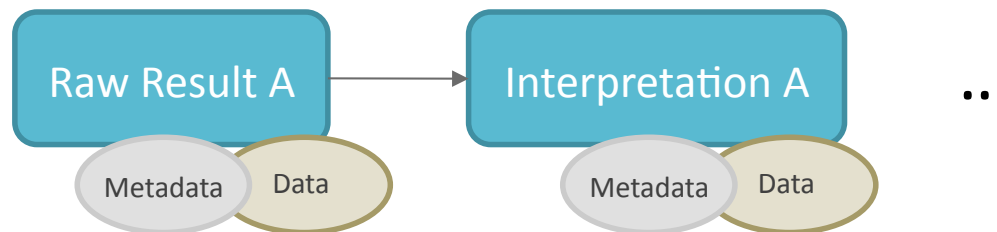
URI	Op.	Accept	Effect
[domain]/tools/[tool]/[version]	GET	RDF	Retrieve metadata of tool version
	PUT	RDF	Update metadata of tool version
	DELETE		Remove tool version
	POST	RDF	Publish tool version
[domain]/tools/[tool]/[version]/pub	GET	RDF	Retrieve metadata of published tool version
	GET	ZIP	Retrieve data of published tool version
	DELETE		Unpublish tool version

SEALS Repositories

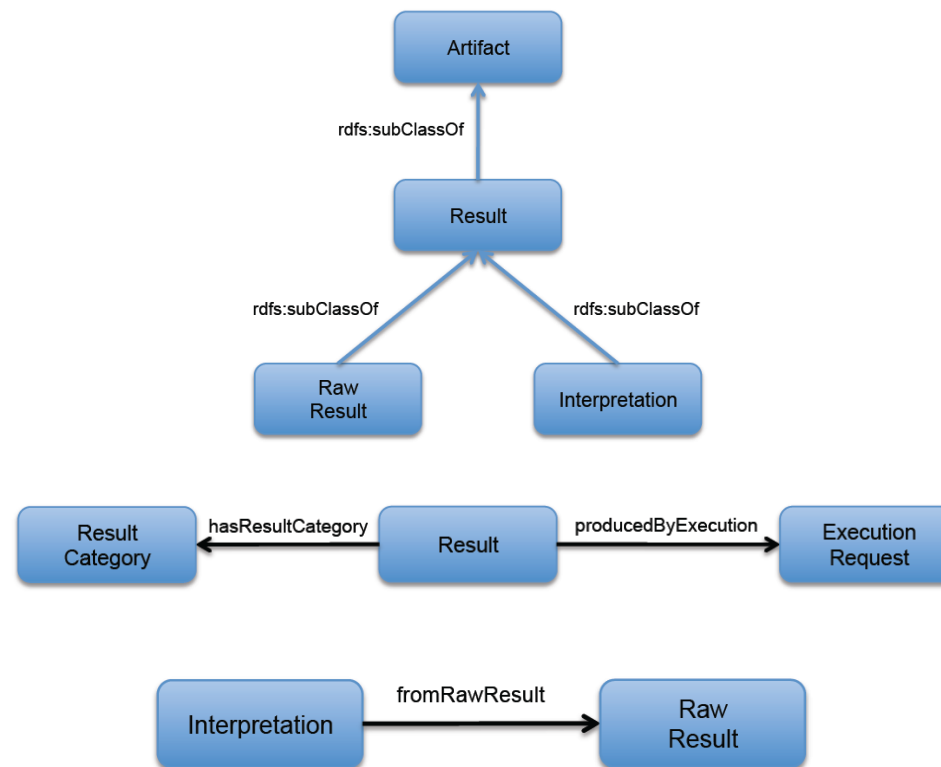
RESULTS REPOSITORY SERVICE

Results Repository Service

- Manages *raw results* and *interpretations* thereof that are created during an evaluation of a tool
- Can be described using suite ontology



Describing Results



Describing Raw Results: Example

- IRIS Datalog reasoner

```
<rdf:RDF xmlns:rdf="..." xmlns:dcterms="..."
  xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">
  <seals:RawResult>
    <dcterms:identifier rdf:datatype="&xsd:string">
      2d3c6d00-287e-11e0-91fa-08002000c9a66
    </dcterms:identifier>
    <seals:hasName rdf:datatype="&xsd:string">
      IRIS OpenRuleBench result
    </seals:hasName>
    <dcterms:description rdf:datatype="&xsd:string">
      Result of the OpenRuleBench evaluation.
    </dcterms:description>
    <seals:producedByExecution rdf:resource="...">
  </seals:RawResult>
</rdf:RDF>
```

Describing Interpretations: Example

- IRIS Datalog reasoner

```
<rdf:RDF xmlns:rdf="..." xmlns:dcterms="..."
  xmlns:seals="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#">
  <seals:Interpretation>
    <seals:hasName rdf:datatype="&xsd:string">
      IRIS OpenRuleBench interpretation
    </seals:hasName>
    <dcterms:description rdf:datatype="&xsd:string">
      Interpretation of the OpenRuleBench evaluation.
    </dcterms:description>
    <seals:fromRawResult rdf:resource="http://seals.sti2.at/rrs-web/
      results/2d3c6d00-287e-11e0-91fa-08002000c9a66/">
  </seals:Interpretation>
</rdf:RDF>
```

Accessing Raw Results

URI	Op.	Accept	Effect
[domain]/results	GET	RDF	Search/browse raw results
	POST	RDF	Add new raw result
[domain]/results/[rawresult]	GET	RDF	Retrieve metadata of raw result
	GET	ZIP	Retrieve data of raw result
	DELETE		Remove raw result

Accessing Raw Results

URI	Op.	Accept	Effect
[domain]/results/[rawresult]/suite	GET	RDF	Retrieve suite metadata of raw result suite
[domain]/results/[rawresult]/suite/ [suite item]/item/[data item]	GET		Retrieve data item of raw result suite
[domain]/results/[rawresult]/suite/ [suite item]/component/[component type]	GET		Retrieve component of raw result suite

Accessing Interpretations

URI	Op.	Accept	Effect
[domain]/interpretations	GET	RDF	Search/browse interpretations
	POST	RDF	Add new interpretation
[domain]/interpretations/[interpretation]	GET	RDF	Retrieve metadata of interpretation
	GET	ZIP	Retrieve data of interpretation
	PUT	RDF	Update interpretation
	DELETE		Remove interpretation

Accessing Interpretations

URI	Op.	Accept	Effect
[domain]/interpretations/[interpretation]/suite	GET	RDF	Retrieve suite metadata of interpretation suite
[domain]/interpretations/[interpretation]/suite/[suite item]/item/[data item]	GET		Retrieve data item of interpretation suite
[domain]/interpretations/[interpretation]/suite/[suite item]/component/[component type]	GET		Retrieve component of interpretation suite

SEALS Repositories

LINKS TO RESOURCES

Online Documentation

- Up-to-date documentation of repository interfaces and functionalities in the Wiki
 - http://www.seals-project.eu/wiki/index.php/Test_Data_Repository_Service_REST_API
 - http://www.seals-project.eu/wiki/index.php/Tools_Repository_Service_REST_API
 - http://www.seals-project.eu/wiki/index.php/Results_Repository_Service_REST_API
- Guidelines for creating metadata
 - [http://www.seals-project.eu/wiki/index.php/Repository_Metadata_\(New\)](http://www.seals-project.eu/wiki/index.php/Repository_Metadata_(New))
 - http://www.seals-project.eu/wiki/index.php/Suite_Metadata

Production Environment

- Repositories available at
 - Tools Repository
<http://seals.sti2.at/trs-web/>
 - Results Repository
<http://seals.sti2.at/rrs-web/>
 - Test Data Repository
<http://seals.sti2.at/tdrs-web/>

Test Environment

- Repositories for testing purposes available at
 - Tools Repository
<http://seals-test.sti2.at/trs-web/>
 - Results Repository
<http://seals-test.sti2.at/rrs-web/>
 - Test Data Repository
<http://seals-test.sti2.at/tdrs-web/>