Digging into the SEALS Platform

Miguel Esteban Gutiérrez, UPM

1st SEALS Tutorial
8th Extended Semantic Web Conference ESWC 2011
Heraklion, Greece
Digging into the SEALS Platform

SEALS PLATFORM OVERVIEW
Outline

• Objective of the SEALS Platform
• SEALS Platform Organization
• Structure of the SEALS Entities
• Evaluation Execution process
• Tool Life-cycle
• Architecture of the SEALS Platform
• SEALS Execution Infrastructure
Objective of the SEALS Platform

- Tools
- Test data
- Execution Request
- Evaluation descriptions
- Results
SEALS Platform Organization

SEALS users

SEALS Portal

Execution requests → SEALS Service Manager
Entity management requests → SEALS Service Manager

Runtime Evaluation Service

SEALS Repositories

Test Data Repository Service
Tools Repository Service
Results Repository Service
Evaluation Descriptions Repository Service

Software agents, i.e., technology evaluators

SEALS users
SEALS Platform Organization

SEALS Service Manager

• **Infrastructure Management**
  – Computing resources management

• **Repository Front-end**
  – Repository Services integration

• **Execution**
  – Execution requests management

• **Platform Administration**
  – Security management
  – Monitoring
Structure of the SEALS Entities
Evaluation Execution Process

- Infrastructure request
- Tools deployment
- Tools undeployment
- Infrastructure release
- Execution request analysis
- Execution environment preparation
- Evaluation description execution
- Execution environment clean-up
- Test data stage-in
- Activity Execution
- Results storage
- Test data stage-out
Tool Life-cycle (I)
Tool Life-cycle (II)

1. Bundled
   - «extract»
2. Deployable
   - «deploy»
3. Loadable
   - «start»
4. Prepared
   - «bootstrap»
5. Undeployed
   - «undeploy»
6. Removable
   - «stop»
7. Invokable
   - «bootstrap»

ESWC 2011, 29/05/2011
Architecture of the SEALS Platform
Logical Perspective

ESWC 2011, 29/05/2011
Architecture of the SEALS Platform
Physical perspective (I)
Architecture of the SEALS Platform
Physical perspective (II)
Architecture of the SEALS Platform
Physical perspective (I)
SEALS Execution Infrastructure
Dealing with heterogeneity and reproducibility

Processing Node
- SSM
- JBI Container
- Web Service
- RES Core

Execution Node
- Guest 1
- RES Worker

Virtualization Solution
- Virtual Machine

ESWC 2011, 29/05/2011
Doubts, comments, questions??
Digging into the SEALS Platform

HANDS-ON
Outline

- Introduction
- Setting up the local infrastructure
- Preparing the evaluation scenario
- Running the evaluation scenario
Introduction
Outline

• **Goals of the session**

• **Evaluation scenario**

• **Infrastructure set-up**
Introduction

Goals

• Show how to use current SEALS Platform components for running evaluation descriptions locally (@home)
Introduction
Evaluation scenario (I)

• Evaluation description:
  – *Purpose*: exercise an ontology engineering tool with “conformance-alike” test data
  – *Contract*:
    • Parameters:
      – conformanceTestSuite
      – conformanceTestSuiteVersion
      – tool
      – toolVersion
    • Outputs:
      – rawResult
      – numberToolBridgeFaults
      – numberToolFaults
Introduction
Evaluation scenario (II)

• **Evaluation description**, continued:
  – *Structure of the test data consumed:*

  ![Diagram](image)

  **NOTE:**
  • The `conf` namespace prefix maps to [http://www.seals-project.eu/ontologies/ConformanceTestSuite.owl#](http://www.seals-project.eu/ontologies/ConformanceTestSuite.owl#)
  • The individuals (tests and items) are defined in the namespace [http://www.seals-project.eu/Conformance/metadata.rdf#](http://www.seals-project.eu/Conformance/metadata.rdf#)
    – Conformance tests individuals are named with the `dc:identifier` value
Introduction
Evaluation scenario (III)

• **Evaluation description**, continued:
  – *Structure of the generated raw results*:

```
NOTE:
• The conf namespace prefix maps to http://www.seals-project.eu/ontologies/ConformanceResult.owl#
• The individuals (results and items) are defined in the namespace http://www.seals-project.eu/Conformance/metadata.rdf#
  – Raw result item individuals are named using the dc:identifier value, which matches the dc:identifier value of the associated conformance test
```
Introduction
Evaluation scenario (IV)

- Initiate
- Initialize
- While has next test case
  - Retrieve test data
  - Invoke tool
  - Succeds?
    - yes
      - Store intermediate result
      - End if
    - no
      - Tool Fault?
        - yes
          - Increase number of tool faults
          - End if
        - no
          - Tool Bridge Fault?
            - yes
              - Increase number of tool bridge faults
              - End if
            - no
              - Failed
    - End if
- End while
- Completed
- Store raw result
Introduction

Evaluation scenario (V)

• **Tool under evaluation:**
  – *Business logic*

![Flowchart diagram](image-url)
Introduction
Evaluation scenario (VI)

• Test data used:

  TestDataVersion
  ontologies
  ISA01.owl
  ISA02.owl
  ISA03.owl
  ISA04.owl
  ISA05.owl
  ISA06.owl
  ISA07.owl
  ISA08.owl
  ISA09.owl
  ISA10.owl

  metadata.rdf

  RDFS conformance test
  OWL_Lite conformance test
  OWL_DL conformance test
Introduction
Infrastructure set-up

- **APACHE TOMCAT**
  - TDRS
  - RRS
  - FireFox
  - Cygwin

- **SoapUI**
- **Groovy**

- **FUSE ESB (CORE)**
  - **EXECUTION ENGINE**
    - Evaluation
  - **RES-CORE**

- **FUSE ESB (WORKER)**
  - **EXECUTION WORKER**
  - **DATA TRANSFER WORKER**
  - **TOOL WRAPPER**
  - **TOOL**
Setting up the local infrastructure

Outline

• Prerequisites

• SEALS Repositories
  – Test Data Repository Service
  – Results Repository Service

• Runtime Evaluation Service
Setting up the local infrastructure

Prerequisites

• Required software:
  – Java SE Development Kit 6 update 18\(^1\)
  – SVN client compatible with the SEALS Subversion Server
  – Maven 2.2.1\(^2\), configured according Annex B of D9.3 v1.0-FR

  \textit{NOTE: The binaries of the required software should be included in the path environment variable}

• Required middleware:
  – Fuse ESB 4.2.0-fuse-02-00\(^3\)
  – Apache Tomcat 6.0.26\(^4\) (or higher)
    • OpenRDF Sesame 2.2.4
    • OpenRDF Workbench 2.2.4

• The middleware will be deployed in \texttt{D:\SEALS\environment} (from now on \%ENV\%)

\(^{1}\) http://www.oracle.com/technetwork/java/archive-139210.html
\(^{2}\) http://maven.apache.org/download.html
\(^{3}\) http://fusesource.com/downloads/
\(^{4}\) http://archive.apache.org/dist/tomcat/tomcat-6/v6.0.26/bin/
Setting up the local infrastructure
SEALS Repositories (I)

• Configuration details for the middleware used in the examples:
  – **Apache Tomcat:**
    • Installed in %ENV%\apache-tomcat-6.0.26 (from now on, %TOMCAT%)
    • Listening on port 8080
  – **OpenRDF Sesame:**
    • Deployed as openrdf-sesame application
  – **OpenRDF Workbench:**
    • Deployed as openrdf-workbench application
Setting up the local infrastructure
SEALS Repositories (II)

- Deploying the Test Data Repository Service (version 1.1-b):
  - Grab the application war from the SEALS Shared Artifacts Repository:
  - Copy application war to Tomcat hot deploy directory (%TOMCAT%\webapps)
  - Create repository in OpenRDF Sesame:
    - Id: testdata
    - Type: Native Java Store RDF Schema
    - Triple indexes: spoc, posc, cpso
  - Configure the application (%TOMCAT%\webapps \tdrs-web-1.1-b\WEB-INF\classes\config.properties):
    - Sesame repository connection details:
      - TestDataRepositoryURL=http://localhost:8080/openrdf-sesame
      - TestDataRepositoryName=testdata
    - Local file store details:
      - TestDataSetFileDirectory=%ENV%/repositories/testdata/testdatasets/
      - GeneratorFileDirectory=%ENV%/repositories/testdata/generators/
      - TempFileDirectory=%ENV%/repositories/testdata/temp/
Setting up the local infrastructure
SEALS Repositories (III)

• Deploying the **Results Repository Service (version 1.1-b):**
  – Grab the application war from the SEALS Shared Artifacts Repository:
  – Copy application war to Tomcat hot deploy directory (%TOMCAT%/webapps)
  – Create repository in OpenRDF Sesame:
    • Id: results
    • Type: Native Java Store RDF Schema
    • Triple indexes: spoc, posc, cpso
  – Configure the application (%TOMCAT%/webapps/rrs-web-1.1-b/WEB-INF/classes/config.properties):
    • **Sesame repository connection details:**
      – ResultsRepositoryName = results
    • **Local file store details:**
      – RawResultFileDirectory=%ENV%/repositories/results/rawresults/
      – InterpretationFileDirectory=%ENV%/repositories/results/results/interpretations/
Setting up the local infrastructure
SEALS Repositories (IV)

- **Caveats:**
  - Tomcat has to deploy the application wars before in order to expose the configuration files
  - Restart Tomcat whenever the configuration is updated
  - Regarding the local file store directories:
    - They have to be created manually
    - Their **full** names in the configuration files have to be encoded as valid Java strings:
      - D:\SEALS\repositories ➔
        » D:/SEALS/repositories
        » D:\\SEALS\\repositories
Setting up the local infrastructure
Runtime Evaluation Service (I)

• Follow the guidelines provided Chapter 4 of the
deliverable **D9.3 Iterative design and implementation of the Evaluation Descriptions Repository Service v1.0-FR**
Setting up the local infrastructure
Runtime Evaluation Service (I)

• Configuration details used for the tutorial:
  – Used **source code** from the **trunk**
  – RES Resources deployed to %ENV%\resources
  – RES Core:
    • FUSE ESB instance installed in %ENV%\fuse-esb-4.2.0\core (from now on %FUSE_CORE%)
  – RES Worker:
    • FUSE ESB instance installed in %ENV%\fuse-esb-4.2.0\worker (from now on %FUSE_WORKER%)
Setting up the local infrastructure

Runtime Evaluation Service (II)

• Configuration details used for the tutorial, continued:
  – **RES Worker**, continued:
    • *Configuration file* (%FUSE_WORKER%\SEALS\configuration.properties):

```
environment=INTEGRATION TESTING
    # RES Worker Tools Services specific configuration
    tool.oet.location.package=SEALS/itest
tool.cmt.location.package=SEALS/packages/cmt
tool.srst.location.package=SEALS/packages/srst
tool.sst.location.package=SEALS/packages/sst
tool.swst.location.package=SEALS/packages/swst

    # RES Worker Repository Access Services specific configuration
    tdrs.url=http://localhost:8080/tdrs-web-1.1-b/
tdrs.tmp_directory=SEALS/tmp/repositories/tdrs

    rrs.url=http://localhost:8080/rrs-web-1.1-b/
rrs.tmp_directory=SEALS/tmp/repositories/rrs

    # RES Worker Utility Services specific configuration
    rc.connection_timeout=5000
rc.tmp_directory=SEALS/tmp/rc
```
Preparing the evaluation scenario

Outline

• *Populating the test data repository*

• *Deploying the evaluation description*
  – Physical deployment
  – Logical deployment

• *Deploying the tool under evaluation*
Preparing the evaluation scenario

Populating the test data repository (I)

• Registering the test data set:

```
> groovy AddTestData.groovy
    http://localhost:8080/tdrs-web-1.1-b/
    ../evaluation-scenario\test-data\TestDataSet
```

Adding persistent test data set '../evaluation-scenario\test-data\TestDataSet'.
- Target TDRS...........: 'http://localhost:8080/tdrs-web-1.1-b/'.
- Source metadata file: '../evaluation-scenario\test-data\TestDataSet.xml'.
SUCCESS [Created (201)]: Creado.
Collection published at 'http://localhost:8080/tdrs-web-1.1-b/testdata/persistent/Ontology+Engineering+Tools+OWL+Lite+Test+Data+Suite+Collection/'
...

NOTE:
- The Groovy scripts used are available in the groovy-scripts folder
- A Groovy 1.7 (or higher) distribution properly installed and configured is required for running the scripts (http://groovy.codehaus.org/Download)
Preparing the evaluation scenario
Populating the test data repository (II)

• Registering the test data set, continued:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <hasName xmlns="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#" rdf:datatype="http://www.w3.org/XMLSchema#string">Ontology Engineering Tools OWL Lite Test Data Suite Collection</hasName>
    <description xmlns="http://purl.org/dc/terms/" rdf:datatype="http://www.w3.org/XMLSchema#string">Ontology Engineering Tools OWL Lite Test Data Suite Collection</description>
    <hasTestDataCategory xmlns="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#" rdf:resource="http://www.seals-project.eu/someTestDataCategory"/>
</PersistentTestData>
</rdf:RDF>
```
Preparing the evaluation scenario
Populating the test data repository (III)

• Registering the test data set version:

> groovy AddTestDataVersion.groovy
  http://localhost:8080/tdrs-web-1.1-b/
  Ontology+Engineering+Tools+OWL+Lite+Test+Data+Suite+Collection
  ../evaluation-scenario\test-data\TestDataVersion

Adding persistent test data version '..\evaluation-data\binaries\TestDataVersion' to persistent test data set 'Ontology+Engineering+Tools+OWL+Lite+Test+Data+Suite+Collection'.
- Target TDRS.........: 'http://localhost:8080/tdrs-web-1.1-b/'.
- Source metadata file: '..\evaluation-scenario\test-data\TestDataVersion.xml'.
- Source data file....: '..\evaluation-scenario\test-data\TestDataVersion.zip'.
SUCCESS [Created (201)]: Creado.
Version published at 'http://localhost:8080/tdrs-web-1.1-b/testdata/persistent/Ontology+Engineering+Tools+OWL+Lite+Test+Data+Suite+Collection/v1.0/'
Preparing the evaluation scenario

Populating the test data repository (IV)

• Registering the test data set version, continued:

... Persistent test data set metadata:
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <hasName xmlns="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#" rdf:datatype="http://www.w3.org/XMLSchema#string">Ontology Engineering Tools OWL Lite Test Data Suite Collection</hasName>
      <description xmlns="http://purl.org/dc/terms/" rdf:datatype="http://www.w3.org/XMLSchema#string">Ontology Engineering Tools OWL Lite Test Data Suite Collection</description>
      <hasTestDataCategory xmlns="http://www.seals-project.eu/ontologies/SEALSMetadata.owl#" rdf:resource="http://www.seals-project.eu/someTestDataCategory/"/>
   </PersistentTestData>
</rdf:RDF>
Preparing the evaluation execution
Deploying the evaluation description (I)

• **Physical deployment:** Making the BPEL process that implements the evaluation description available in the same container as the RES Core.
  – Copy evaluation description service assembly (*res-itest-ed-sa-1.1-SNAPSHOT.zip*) to RES Worker hot deploy directory (%FUSE_CORE%\deploy)

• **Logical deployment:** Inform the RES Core about the endpoint from which the BPEL process that implements the evaluation description is accessible.
  – Invoke the **DeployEvaluationDescription** of the **Evaluation Description Deployer Service** of the RES Core.
Preparing the evaluation execution

Deploying the evaluation description (II)

```xml
<?xml version="1.0" encoding="UTF-8"?>
  xmlns:rrs="http://www.seals-project.eu/resources/res/repositories/rrs/wsdl/v1"
  xmlns:tdrs="http://www.seals-project.eu/resources/res/repositories/tdrs/wsdl/v1"
  xmlns:oet="http://www.seals-project.eu/resources/res/tools/oet/wsdl/v2"
  xmlns:eval="http://www.seals-project.eu/resources/res/engine/evaluation/wsdl/v2">
  <process name="eval:integration-testing-evaluation">
    <!-- Contents removed for brevity -->
    <provide partnerLink="client">
      <service name="eval:customEvaluationService" port="customEvaluationPort"/>
    </provide>
    <!-- Contents removed for brevity -->
  </process>
</deploy>
```

Evaluation Description Deployment Descriptor

```xml
<v1:DeployEvaluationDescription
  xmlns:v1="http://www.seals-project.eu/resources/res/engine/deployer/wsdl/v1"
  xmlns:oet="http://www.seals-project.eu/resources/res/tools/oet/wsdl/v2"
  xmlns:eval="http://www.seals-project.eu/resources/res/engine/evaluation/wsdl/v2">
  <v1:evaluationDescriptionId>urn:ed:6ba7b810-9dad-11d1-80b4-00c04fd430c7</v1:evaluationDescriptionId>
  <v1:deploymentInformation>
    <v1:entryPointService>
      <v1:name>customEvaluationService</v1:name>
      <v1:port>customEvaluationPort</v1:port>
    </v1:entryPointService>
    <v1:deploymentInformation>
      <!-- Contents removed for brevity -->
    </v1:deploymentInformation>
  </v1:deploymentInformation>
</v1:DeployEvaluationDescription>
```

DeployEvaluationDescription Payload
Preparing the evaluation execution
Deploying the tool under evaluation

• Unzip tool package (res-itest-tool-1.1-SNAPSHOT-tool-package.zip) to path specified in the RES Worker configuration file (%FUSE_WORKER%\SEALS\configuration.properties)
  – tool.oet.location.package=SEALS/itest
Running the evaluation scenario

Outline

• Preparing the execution request context
• Preparing the request message
• Triggering the evaluation execution
• Analyzing the execution response
• Inspecting generated results
  – From the browser
  – From the shell
Running the evaluation scenario

Preparing the execution request context (I)

• Inform the Runtime Evaluation Service about the resources that should participate in the enaction of the execution request.
  – Which is the evaluation description associated to the execution request
  – Where are the RES Workers which will expose the tools under evaluation*

• Invoke the \texttt{RegisterExecutionRequestContext} of the \texttt{Execution Request Context Registry Service} of the RES Core.
Preparing the evaluation execution  
Preparing the execution request context (II)

```
<v1:RegisterExecutionRequestContext
 xmlns:v1="http://www.seals-project.eu/resources/res/engine/registry/wsdl/v1">
  <v1:executionRequestId>urn:erq:6ba7b810-9dad-11d1-80b4-00c04fd430c7</v1:executionRequestId>
  <v1:evaluationDescriptionId>urn:ed:6ba7b810-9dad-11d1-80b4-00c04fd430c7</v1:evaluationDescriptionId>
  <v1:computingResource>
    <v1:computingResourceId>tool</v1:computingResourceId>
    <v1:computingResourceLocation>127.0.0.1</v1:computingResourceLocation>
  </v1:computingResource>
</v1:RegisterExecutionRequestContext>
```
Running the evaluation scenario
Preparing the request message (I)

• The payload:

```xml
<snap1:initiate
 xmlns:snap1="http://www.seals-project.eu/resources/res/engine/evaluation/wsoy/v2"
 xmlns:snap2="http://www.seals-project.eu/resources/res/common/types/xsd/v1">
  <snap1:argument>
    <snap2:name>conformanceTestSuite</snap2:name>
    <snap2:value>Ontology Engineering Tools OWL Lite Test Data Suite Collection</snap2:value>
  </snap1:argument>
  <snap1:argument>
    <snap2:name>conformanceTestSuiteVersion</snap2:name>
    <snap2:value>v1.0</snap2:value>
  </snap1:argument>
  <snap1:argument>
    <snap2:name>tool</snap2:name>
    <snap2:value>ProtegeOWL</snap2:value>
  </snap1:argument>
  <snap1:argument>
    <snap2:name>toolVersion</snap2:name>
    <snap2:value>ProtegeOWLVersion1</snap2:value>
  </snap1:argument>
</snap1:initiate>
```
Running the evaluation scenario
Preparing the request message (II)

• The header:

```xml
<header:Header xmlns:header="http://www.seals-project.eu/resources/res/common/header/xsd/v1">
  <header:Destination>
    <header:HttpURI>http://localhost</header:HttpURI>
  </header:Destination>
  <header:ExecutionRequestId>urn:erq:6ba7b810-9dad-11d1-80b4-00c04fd430c7</header:ExecutionRequestId>
</header:Header>
```
Running the evaluation scenario
Triggering the evaluation execution (I)

NOTE: SoapUI 3.5.1 (or higher) is required for running the project (http://www.soapui.org/)

ESWC 2011, 29/05/2011
Running the evaluation scenario
Triggering the evaluation execution (II)
Running the evaluation scenario
Analyzing the execution response (I)
Running the evaluation scenario
Analyzing the execution response (II)

• Payload:

```xml
<completed xmlns="http://www.seals-project.eu/resources/res/engine/evaluation/wsd1/v2">
  <tns:result xmlns:tns="http://www.seals-project.eu/resources/res/engine/evaluation/wsd1/v2">
    <commons:name xmlns:commons="http://www.seals-project.eu/resources/res/common/types/xsd/v1">rawResult</commons:name>
    <commons:value xmlns:commons="http://www.seals-project.eu/resources/res/common/types/xsd/v1">integration-testing-evaluation1</commons:value>
  </tns:result>
  <tns:output xmlns:tns="http://www.seals-project.eu/resources/res/engine/evaluation/wsd1/v2">
    <tns:numberToolBridgeFaults>3</tns:numberToolBridgeFaults>
    <tns:numberToolFaults>2</tns:numberToolFaults>
  </tns:output>
</completed>
```
Running the evaluation scenario
Inspecting generated results

• Simple inspection
  – With a browser:
    • FireFox (3.6.13)
    • REST Client add-on (1.3.3)

• Complex inspection
  – With shell commands (*nix shell or Cygwin on Windows platforms):
    • curl
    • unzip
Running the evaluation scenario
Inspecting generated results with a browser (I)
Running the evaluation scenario
Inspecting generated results with a browser (II)
Running the evaluation scenario
Inspecting generated results with a browser (III)
Running the evaluation scenario
Inspecting generated results with a browser (IV)
Running the evaluation scenario
Inspecting generated results with a browser (V)
Running the evaluation scenario
Inspecting generated results with a browser (VI)
Running the evaluation scenario
Inspecting generated results with a browser (VIII)
Running the evaluation scenario
Inspecting generated results from the shell (I)
Running the evaluation scenario
Inspecting generated results from the shell (II)
Running the evaluation scenario
Inspecting generated results from the shell (III)
Running the evaluation scenario
Inspecting generated results from the shell (IV)
Doubts, comments, questions??
Digging into the SEALS Platform

Miguel Esteban Gutiérrez, UPM

1st SEALS Tutorial
8th Extended Semantic Web Conference ESWC 2011
Heraklion, Greece