Architecture Document

Quality Assessment on Enterprise Linked Data

Created By:

Carlos Montoya

Rauf Aghayev

Vugar Aghayev

Enterprise Information Systems Department

EIS-LAB

Bonn, September 2014

Content

Figure Index

1 INTRODUCTION

Comment [CM1]: This Section is created by Carlos Montoya

1.1 Problem Definition

During the development of Web Semantic technologies huge volume of data has being published on the web as a Linked Open Data. In order to use LOD for specific purpose, we should check its quality in advance. The quality term means in Semantic Web is fitness of use. Then we should have into account that not all the data is meaningful and if the quality is poor it leads to Enterprise problems such as data standardization, multiple data with duplicates in the data sets, meaningless information and so on.

To assure that the Enterprise is trustful, and then in the design we should take into account that the quality measurement depends on which domain the data will be used. On the other hand, we should assure that our LOD the data is measure for the correct quality metrics, which means that the given data in one LOD be useful for one of the use cases but not favorable for other ones. Therefore, for measurement of LOD we are going to use metrics and we identify the fitness of it use.

1.2 Objectives

The main goal of this document is to present the design of the application developed to accomplish the Quality Assessment on Enterprise Linked Data.

The process defines is compound by two sub problems:

- Streaming of Enterprise SPARQL Endpoints.
- Computation of the Quality Metrics over the SPARQL Endpoints.

Achieving these two parts, the values of the SPARQL endpoint should be visible through a user interface, which should be able to show the Metric Value against the SPARQL endpoint.

1.3 Document Structure

This document is structure as follow:

- The chapter one is the introduction, establishing the main purpose of the document and what to expect of it.

- The chapter two is the specification of the different scenarios (Use Cases), which were used to find the solution to the problem.
- The chapter three is the conceptual framework of the application, emphasizing on the technologies used in the project.
- The chapter four explains the structure of the Project and the detail of all the modules.
- The chapter five explained how to install the project and how to use it.
- The chapter six explained how to install the project and how to use it.
- The chapter seven describes the testing units that were used to test the classes and the performance of the project.

2 USE CASES

2.1 Use Case 1: Drugstore Online.

Comment [CM2]: This Use Case is created by Carlos Montoya

Based on the fact that there is some people who cannot go out because they suffer from some illness or they are recovering at home of some kind of medical treatment. Then they need a service that provide cheaper and trustful medicaments online that can be send easily to their home. For that reason the use case is related with the creation of a Drugstore Online which obtains information of different websites (providers) and then shows the best options to the user. Of course in order to offer a medicament or a treatment we must be able to check the quality of the data that we display on the screen.

We must be able to check in advance the data quality, for that reason we should be able to use the metrics (initial approximation):

1. Verifiability
2. Reputation
3. Believability
4. Consistency
5. Availability
6. Understandability
7. Conciseness
8. Volatility

The next image describes the basic uses cases that the enterprise system requires.

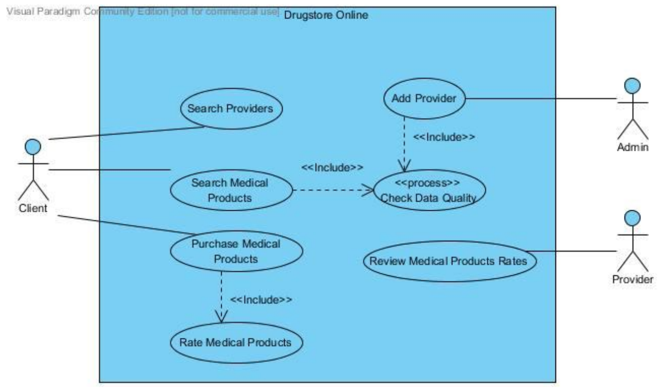

Visual Paradigm Community Edition [not for commercial use]
Drugstore Online
Search Providers
Add Provider
<<Include>>
Admin
Client
Search Medical Products
<<Include>>
<<process>> Check Data Quality
Purchase Medical Products
Review Medical Products Rates
Provider
<<Include>>
Rate Medical Products

Illustration 2-1, Use Case, Scenario

2.2 Use Case 2: Online Computer Store.

Comment [CM3]: Use case defined by Rauf Agayev

During the improvement of information technologies, approximately every person uses Computer, Tablet, Telephone, etc. In addition, they prefer, buy such kind of devices online for getting lower prices. Our use case related with this situation. We want to create Online Computer Store, which gets information about Computers from different web sites and shows customers. In order to achieve this, we should check quality of data in advance. On the other hand, we apply some metrics which fit our requirements, to the given data and then we decide data is usable or not.

Metrics:

1. Completeness
2. Amount of Data
3. Relevancy
4. Verifiability
5. Reputation
6. Believability
7. Response time
8. Availability
9. Understandability

2.3 Use Case 3: Hotel Reservation System.

Comment [CM4]: Created by Vugar Agayev

My use case is about intelligent hotel reservation system which aggregating data from several data sources, websites. It gets information about location of countries, cities and particular addresses from a spatial dataset and information about hotels from a hotel dataset. Additionally, all related information about hotels gathered from different booking services and represented as RDF. This integrated dataset allows user to find a suitable hotel between any arrival and departure time for his/her wonderful vacation.

1. The use case that we are going to develop is the related with Check Data Quality.
2. Availability
3. Licensing
4. Interlinking
5. Security
6. Performance
7. Accuracy
8. Consistency
9. Conciseness
10. Reputation
11. Believability

12. Verifiability
13. Objectivity
14. Understandability
15. Versatility

2.4 Metrics Definition

Comment [CM5]: Created by Carlos Montoya

2.4.1 Metrics definitions for the new Dimensions

The Enterprise systems have already other dimensions to assure the quality of the information that they use. For that reason here you can find a small list that is related with the quality measurement.

1. Easy of Manipulation
   a Keeping URIs short
   b Mixed metrics between believable and reputable dimensions

2. Free of Error
   a Dereferencability issues
   b Erroneous annotation/representation erroneous
   c Inaccurate annotation, labeling, classification

3. Value-Added
   a Evaluation by the expert users

4. Temporal Reliability
   a Meaning and semantics changes over time
   b Look for loss of history data with no record of previous values

2.5 Requirement Specification

Comment [CM6]: Created by Carlos Montoya

Requirement Name	Process Data Quality		
Main Users	System and Admin	**Priority**	High

Author	Carlos Montoya	**Creation Date**	14-05-204
Revision Date		**Modifications Date**	
Summary	The data that display the main system of the company, should be check in advance, this use case is in charge to measure the data sets quality, by measuring the dimensions of: Free of Error, Verifiability and Measurability. After verify those metrics, the system should be able to display the values obtained and the user should be able to compare those values between each other.		
Main Track Events			
User		**System**	
1. The user access the program.		2. The system displays the name of the dimensions to be evaluated, under the question "Please chose a Dimension", those are shown as a combo box, and the options are: 1st Verifiability, 2nd Free of Error and 3rd Measurability.	
3. The user selects one of the dimensions displayed.		4. Depending of the option selected, the system should display a new combo box with the metrics to be evaluated. The system has three different cases (depending on the dimension selected).	
		4.1 When the user choose Verifiability, then the system show the metrics: 1st Authenticity of the dataset, 2nd Usage of digital signatures.	
		4.2 When the user choose Free of Error, then the system show the metrics: 1st Dereferencability issues, 2nd Erroneous annotation/representation erroneous.	
		4.3 When the user choose Measurability, then the system show the metrics:	

5. The user select one of the metrics displays in the point 4.	6. The system now shows the different dataset available to be evaluated under those metrics. They show the options under a multiple select box. The datasets are: DATASET1, DATASET2, and DATASET3.
	7. When the user select at least one dataset the system display the bottom "evaluate"
8. The user press the button "evaluate".	9. The system should display a graph where is easy for the user to identify the metrics vs the datasets. And each dataset is easy to identify.
Alternatives Tracks	
Exception Tracks	
After the user in the point 8 select the information for evaluate and the info of the evaluation is not available, then the system should display a message "The information is not available right now, please choose another dataset or another metric."	
Extension Points	
Pre - Conditions	
To display easily and fast the information of the results of the evaluation of the metrics, those values (metric values) should be evaluated in advance and those result should be saved into some media that should be defined by the development group.	
The selected datasets to be evaluated should be free of access, and then the development is not going to violate any copyright of the information used.	

Post- Conditions
All the datasets are evaluated and these results are store in an accessible media to be used by another use cases.
Considerations

3 CONCEPTUAL FRAMEWORK

Comment [CM7]: Created by Carlos Montoya

Shown below are explained all the technologies uses for the development of the application of Quality Assessments.

3.1 RDF

RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.

RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.

This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations.[1]

3.2 Apache Jena

Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL and updated through SPARUL.[2]

Its Architecture includes:

- API to work (read, process, write) ontologies RDF and OWL.
- Engine of inference for rezoning over ontologies RDF and OWL.

[1] http://www.w3.org/RDF/

[2] http://en.wikipedia.org/wiki/Jena_(framework)

- A strategy to flexi storage for RDF triples in memory.
- Query engine compatible with SPARQL.

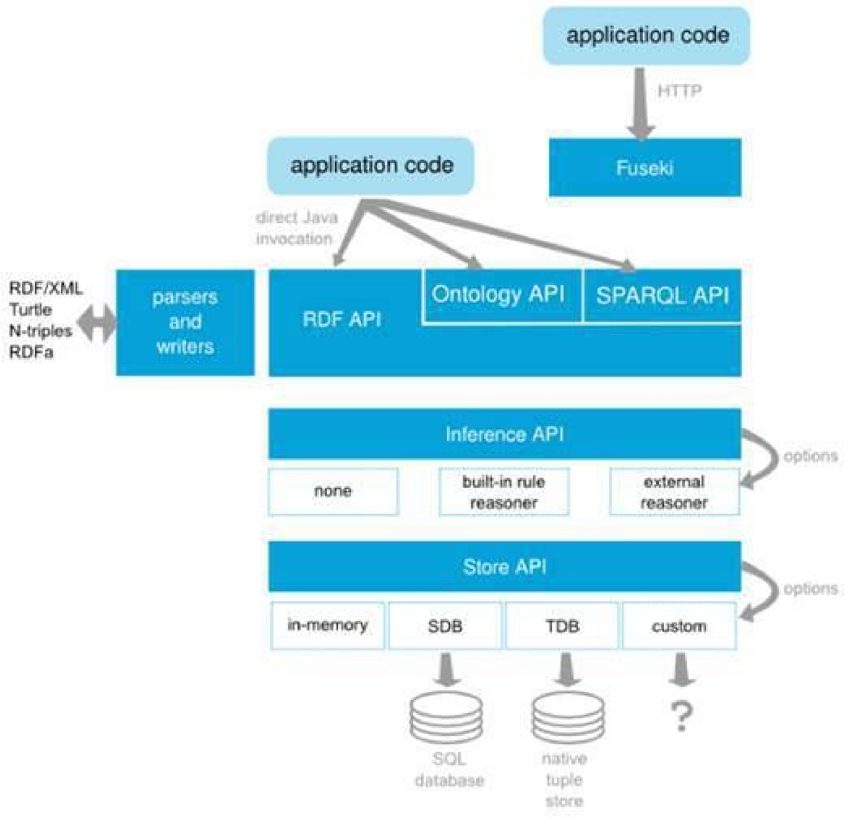


Illustration 3-1, Apache Jena Specification

3.3 SPARQL

SPARQL: acronym for Simple Protocol and RDF Query Language is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web[3].

[3] http://en.wikipedia.org/wiki/SPARQL

SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

3.4 MAVEN

Maven, a Yiddish word meaning accumulator of knowledge, was originally started as an attempt to simplify the build processes in the Jakarta Turbine project. There were several projects each with their own Ant build files that were all slightly different and JARs were checked into CVS. We wanted a standard way to build the projects, a clear definition of what the project consisted of, an easy way to publish project information and a way to share JARs across several projects.

The result is a tool that can now be used for building and managing any Java-based project. We hope that we have created something that will make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.[4]

Maven's Objectives

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Providing guidelines for best practices development
- Allowing transparent migration to new features

3.5 JUnit

JUnit is an open source framework designed for the purpose of writing and running tests in the Java programming language.

JUnit allows the developers to incrementally build test suites to measure progress and detect unintended side effects. Tests can be run continuously. Results are provided immediately. JUnit shows test progress in a bar that is normally green but turns red when a test fails. An ongoing list of unsuccessful tests appears in a space near the bottom of the display window. Multiple tests

[4] http://maven.apache.org/what-is-maven.html

can be run concurrently. No subjective human judgments or interpretations of test results are required. The simplicity of JUnit makes it possible for the software developer to easily correct bugs as they are found.

3.6 JSF

JavaServer Faces (JSF) is a Java specification for building component-based user interfaces for web applications.[1] It was formalized as a standard through the Java Community Process and is part of the Java Platform, Enterprise Edition[5].

JSF is based in the MVC pattern. The use of this framework is especially useful for the maintenance in comparison to the JSP and Servlets[6].

- JSF provide standard and reusable components for create user interfaces for web applications.
- JSF provide different libraries to the component's access and manipulation.
- JSF saves automatically the page information and it fill it out when they are deploy on the client side.
- Today there are many different interfaces to simplify de development of web application based on JSF Framework.

The User Interfaces that are created with the technology JSF, run in a web server and it response is shown in a web client. The development using the JSF framework allow us to be focus it the creation of components for the user interfaces, event managed, backing beans and their interactions. And not to be focus it in the requests and responses processes from the client or from the server side.

3.7 RichFaces

RichFaces is an open source library that enables the use of AJAX for JSF; RichFaces is developed by the JBoss.org community. This part allows the integration of capabilities of development for Enterprise applications.

RichFaces is not only a JSF component library, also it provides:

- Skin-ability (Capability to change and update the look and feel of the application).

---

[5] http://en.wikipedia.org/wiki/JavaServer_Faces

[6] https://javaserverfaces.java.net/

- Components Development Kit (CDK) to assist the development of new components.

In the project we used the RichFaces version 3.3.4[7].

3.8 EJB3

The Enterprise Java Bean (EJB) Technology is the component architecture of the server side JEE (Java Enterprise Edition). This technology allows the faster and easy development of distributed, transactional, safety and portables applications based on java.

The EJB specification tries to provide a standard way to develop the “back-end” of the business, typically is found in Enterprise applications. EJB was created with the aim of manage the common concerns related with persistence, transactional integrity and safety, in a standard form.

The specification show that the application should provide:

- Processing transactions.
- Persistence integration services offered by JPA.
- Concurrency control.
- Events using Java Message Service.
- Services Directory (JNDI[8]).
- Deployment of software components in an application server.
- Remote Procedure calls using RMI-IIOP.
- Business Methods Exhibition as Web Services.

3.9 Apache Ant

Apache Ant is a software tool for automating software build processes. It is similar to Make but is implemented using the Java language, requires the Java platform, and is best suited to building Java projects.

[7] The library can be download it in: http://www.jboss.org/richfaces

[8] Java Naming Directory Interface, API for the directory services. This allows clients to discover and searching objects and names through a name.

The most immediately noticeable difference between Ant and Make is that Ant uses XML to describe the build process and its dependencies, whereas Make uses Makefile format. By default the XML file is named build.xml.[9]

[9] http://en.wikipedia.org/wiki/Apache_Ant

4 APPLICATION STRUCTURE

Comment [MC8]: Created By Carlos Montoya

4.1 Modules Structure

The Quality application is a command line application with Java technology and Jena framework. As consequence the application contains uses the pattern MVC (Model, View and Controller).

In the next graphic we show how is built the project.

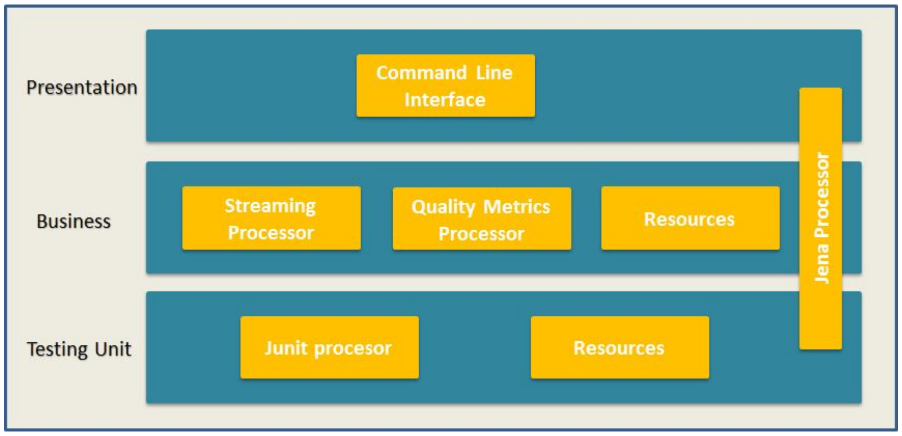


Illustration 4-1, General Architecture

Inside the presentation layer we present the following components:

- User Interface: Is the owner of the interaction with the user, and is in charge to show the results of run all the metrics over the specified dataset.

The Business layer has the next components:

- The Streaming Processor is the owner of the process to connect to de SPARQL endpoint and stream all the data to be use by the quality metrics processor; it is solved using the parading of producer-consumer to make it faster in its processing.

- The Quality Metrics Processor is the processor who computes the values for all the quality metrics defined in the project.

- The content of Resources is used to load the model into the diachron data model, and work to load the advantages offered by Maven.

The Testing unit is compound by:

- The JUnit Processor that take advantages of all the tools offered by JUnit and we used to test our classes and assure its functionality.
- The resources are example datasets that were used by JUnit to make sure that our classes find the correct value of the metrics.

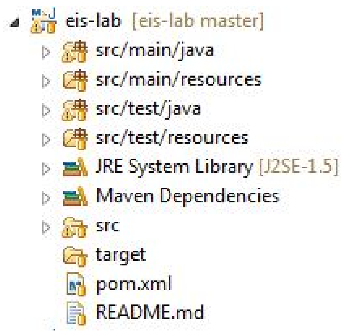
To make the communication between all the modules we used the tools provide it by Jena, such its classes and its tools to connect to the SPARQL endpoints.

All this classes are accessed in all the application and they are independent from the other packages.

4.2 Project Structure

4.2.1 EIS-LAB Project.

The Project is built by packages that contain all the modules necessaries to run the application and be able to use it. You can check in the next graphic.



```
eis-lab [eis-lab master]
  src/main/java
  src/main/resources
  src/test/java
  src/test/resources
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  src
  target
  pom.xml
  README.md
```

Illustration 4-2, EIS-LAB Project structure

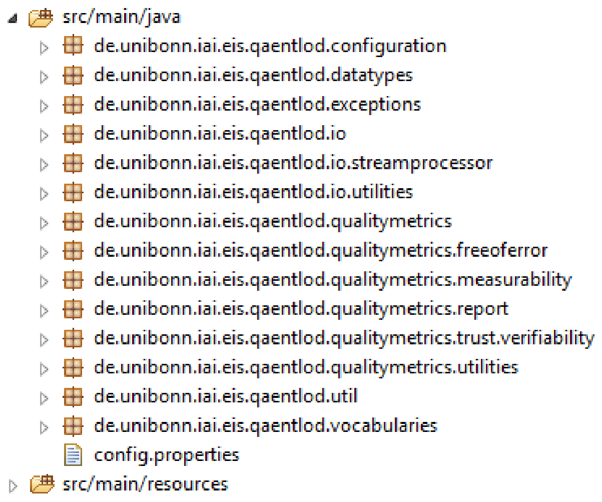
In the graphic you can see the folders:

- Src/main/java: This package contain all the classes that are need it to run the project, in in this package are defined the interface, the metric processor, the streaming processor.
- Src/main/resources: This package contains all the files that are required by maven to build the structure of the project.

- Src/test/java: This package contains all the unit test that were defined to test the classes contains in the main package.
- Src/test/resouces: In this package is stored all the resources that are used to test the classes, it contain some example datasets.
- Also we can check that the project contain a reference to a library that is call Maven Dependencies, it contain all the libraries that are uses by the packages described before.

4.2.1.1 src/main/java

This package as mention before contains all the classes that are necessary to run the project and it is compound by the next packages:



```
src/main/java
  de.unibonn.iai.eis.qaentlod.configuration
  de.unibonn.iai.eis.qaentlod.datatypes
  de.unibonn.iai.eis.qaentlod.exceptions
  de.unibonn.iai.eis.qaentlod.io
  de.unibonn.iai.eis.qaentlod.io.streamprocessor
  de.unibonn.iai.eis.qaentlod.io.utilities
  de.unibonn.iai.eis.qaentlod.qualitymetrics
  de.unibonn.iai.eis.qaentlod.qualitymetrics.freeoferror
  de.unibonn.iai.eis.qaentlod.qualitymetrics.measurability
  de.unibonn.iai.eis.qaentlod.qualitymetrics.report
  de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.verifiability
  de.unibonn.iai.eis.qaentlod.qualitymetrics.utilities
  de.unibonn.iai.eis.qaentlod.util
  de.unibonn.iai.eis.qaentlod.vocabularies
  config.properties
src/main/resources
```

Illustration 4-3, src/main/java packages

In the project the most important packages are:

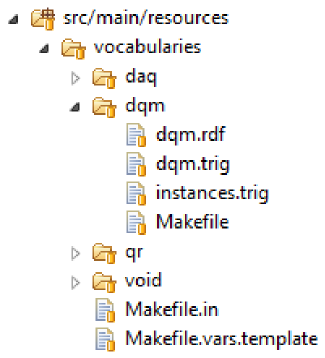
- de.unibonn.iai.eis.qaentlod.io: In this package is the main class to be executed. It provided all the interface of interaction with the user and provide to the user with different functionalities.
- de.unibonn.iai.eis.qaentlod.io.streamprocessor: In this package are the classes that solve the problem of streaming a SPARQL endpoint, it solves this problem implementing the problem of producer-consumer.
- de.unibonn.iai.eis.qaentlod.qualitymetrics: This package contain all the metrics that are defined in our project, it has several subpackages that are:
  - freeoferror: this implements the metrics defined to be into the freeoferror dimension.

- Trust.verifiability: this implements the metrics defined to be into the Trust dimension.
- Measurabilty: this implements the metrics defined to be into the Measurabilty dimension.

- de.unibonn.iai.eis.diachron.vocabularies: This package includes all the information related with the vocabularies that are defined to be readable by the maven processor.

4.2.1.2 src/main/resources

These packages contain all the resources that are necessary to run on maven, it is compound by:



```
src/main/resources
  vocabularies
    daq
    dqm
      dqm.rdf
      dqm.trig
      instances.trig
      Makefile
    qr
    void
    Makefile.in
    Makefile.vars.template
```

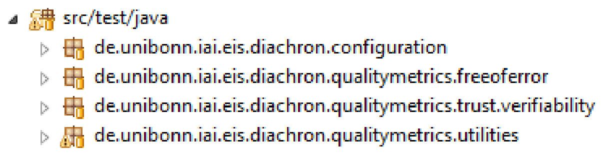
Illustration 4-4, src/main/resources folders

The main folder is call vocabularies, it contains all the necessary files to specify into maven the languages that it should read and create its vocabulary.

Into the dqm.trig is the RDF specification of all the metrics and dimensions that should be taking it into account in the moment of evaluating all the metrics.

4.2.1.3 src/test/java

This folder is the one that contain all the information related with the unit test of the source code. The structure is built by the next packages:



```
src/test/java
  de.unibonn.iai.eis.diachron.configuration
  de.unibonn.iai.eis.diachron.qualitymetrics.freeoferror
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability
  de.unibonn.iai.eis.diachron.qualitymetrics.utilities
```

Illustration 4-5, src/test/java packages

4.2.2 Metrics Explanation

Comment [MC9]: Created By Carlos Montoya and Rauf Agayev

Requirement Name	Quality Metric APIs documentation		
Priority	High		
Author	Rauf Agayev, Vugar Aghayev	**Creation Date**	30-08-2014
Revision Date	01-09-2014	**Modifications Date**	25-08-2014
Summary	This section of documentation explains functionality of packages, classes and methods of Quality metrics. For clear information screenshots of those things are given.		

Verifiability – Authenticity	**Explanation**
package: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.verifiability** classes: → *AuthenticityDataset* → *Digital signatures*	The package trust.verifiability package contains the classes that are responsible of check the authenticity of the information.

class: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.verifiability.AuthenticityDataset**
methods: → *public void compute(Quad quad){}* → *public double metricValue(){}*
	This class gets quads from some sources (sparql endpoint) and operates on them. The main purpose of the class is checking the provenance of the dataset.

```
@Override
void compute(Quad quad)
  divide quad three parts: subject, predicate, object
  if predicate is not null and predicate is URI and subject is not null
    then
    if setProvenanceProperties contains predicate's URI
      then
      String curSubjectURI = if subject is URI
                               then subject's URI
                               else String form of subject
                             end if

      put in mapProvenanceResources curSubjectUri and object
    end if
  end if
```

To compute the value it receives Quad by Quad and check it for each of them if the predicate is equal to one of the attributes of provenance, if it is contained then it store the info on the list of provenance resources of the data set.

```
@Override
double metricValue
        if mapProvenanceResources is not null
           then
             if mapProvenanceResources size is greater than zero
             return 1
             end if
        end if
return 0;
```

It returns 0 if the dataset doesn't have any provenance information, in contrast it return 1 if the dataset contained some of the provenance values.

Verifiability – Digital Signature	**Explanation**
class: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.verifiability.AuthenticityDataset** methods: → *public void compute(Quad quad){}* → *public double metricValue(){}*	This class gets quads from some sources (sparql endpoint) and operates on them. The main purpose of the class is checking the provenance of the dataset by checking if the document contained valid digital signatures.

```
@Override
void compute(Quad quad)
  divide quad three parts: subject, predicate, object
  if predicate is not null and predicate is URI and subject is not null
    then
    if setProvenanceProperties contains predicate's URI
       then
       String curSubjectURI = if subject is URI
                               then subject's URI
                               else String form of subject
                              end if

       put in mapSignatureResources curSubjectUri and object
    end if
  end if
```

To compute the value it receives Quad by Quad and checks it for each of them if the predicate is equal to one of the attributes well-known to contain the digital signatures value.

`@Override`
`double metricValue`
`if mapProvenanceResources is not null`
`then`
`if mapProvenanceResources size is greater than zero`
`return 1`
`end if`
`endif`
`return 0`
	It returns 0 if the dataset doesn't have any digital signature, in contrast it return 1 if the dataset contained some digital signature information.

Free Of Error	**Explanation**
package: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.freeoferror** class: *FreeOfError*	This package contains one class and responsible from "*Erroneous annotation/representation erroneous*"

class: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.freeoferror.FreeOfError**
methods: *public void compute(Quad quad){}*
public double metricValue(){}
public boolean isValidUri(Node node){}
	This class gets quads from some sources (sparql endpoint) and operates on them. The main purpose of the class is checking whether dataset contains syntax error or not.

`@Override`
`void compute(Quad quad)`
`boolean flag = true`
`increment numberOfTriples`
divide quad three parts: subject, predicate, object
`if subject is URI|`
`then`
`flag = isValidURI(subject)`
`else if subject is Blank Node`
`else`
`flag = false`
`end if`
 	
`if predicate is URI`
`flag = isValidURI(predicate);`
`else`
`flag = false`
`end if`
 	
`if object is URI`
`flag = isValidURI(object)`
`else if object is Blank Node`
`else if object is Literal`
`RDFDatatype dataType = object's Literal Data Type`
`flag = if dataType is not null`
`if dataType is valid Literal`
`true`
`else false`
`end if`
`else flag = false`
`if flag is true`
	Free of Error checking operations are being done by only this method. It separates subject, predicate and object of quad and checks their syntax.

`        counter++`
`      end if`	

`@Override`
`    double metricValue`
`        return counter * 1.0 / numberOfTriples`
	Returns free of error information by percentage. Number of free of error quads divided by number of all quads.

`boolean isValidURI(Node node)`
`        try`
`          connect to URI`
`        catch Exception`
`              return false;`
 	
`        return true;`
	In case of Subject or Predicate is URI, this method checks if they are valid or invalid.

Measurability	**Explanation**
package: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.measurability** class: → Measurability	Measurability package contains only one class and responsible from checking if data is measurable or not.

class: **de.unibonn.iai.eis.qaentlod.qualitymetrics.trust.measurability**
methods: → *public void compute(Quad quad){}*
→ *public double metricValue(){}*
	This class uses two approaches in order to evaluate dataset is measurable or not. For this purpose it checks if number of quads in dataset is more than threshold (you can set it) and does user have access to SPARQL endpoint.

`@Override`
`void compute(Quad quad)`
`        increment global variable counter`
`        set global variable sparqlEndPointWorks to true`
	When the quads come it increases variable for counting number of triples in dataset, and automatically it sets Boolean variable true, that user has access to SPARQL endpoint.

`@Override`
`double metricValue`
`    if counter >=threshold and sparqlEndPointWorks is true`
`    then return 1`
`    else return 0`
`    end if`
	Returns 0.0 if data is measurable and 1.0 if it is not. For being measurable two cases must hold.

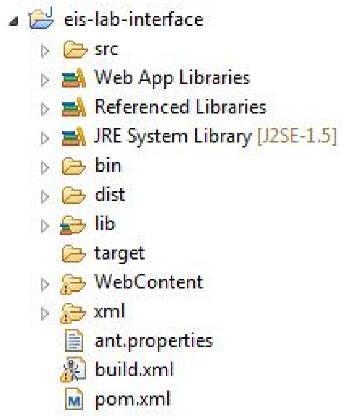
4.2.3 EIS-LAB-INTERFACE Project

Comment [MC10]: Created by Carlos Montoya

The main task of this Project is to manage the User Interface; the interface is a Web environment, for that reason it run over JBoss server. It uses the J2EE technologies of java Beans to run the application.

The interface import the EIS-LAB Project as a .jar File, then it is able to run the metrics and uses the functionality defined in the main project.

The structure of the Project is shown next:



```
eis-lab-interface
    src
    Web App Libraries
    Referenced Libraries
    JRE System Library [J2SE-1.5]
    bin
    dist
    lib
    target
    WebContent
    xml
    ant.properties
    build.xml
    pom.xml
```

Illustration 4-6, EIS-LAB-INTERFACE structure

The folders are defined as you can see in the image below are:

- Src: This package contains the implementation of the Backing beans that are used by de User Interface.
- Lib: In this folder are located all the libraries that the project need to run, including the eis-lab.jar
- Dist: Here are located all the files that are generated once you compile the project and execute the ant file (build.xml as you can see in the image).
- WebContent: In this folder are located all the files related with the User Interface (the css file, the. xhtml files and the configuration files need it to run the project on the server).

4.2.3.1 Src Folder:

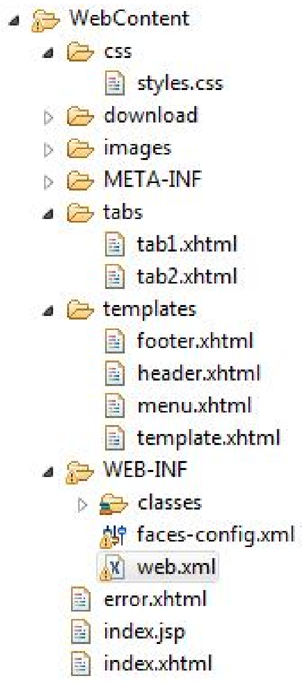
In this folder are located all the java classes that work as a Backing Beans, basically there are two classes ProcessDataSetBean.java and

QualityBean.java, both of them are in charge to managed the two different options that offer the UI, the first one is only provide the tools to show the results after run and evaluate the metrics over some datasets.

The second one is in charge to provide all the necessary things to evaluate a new dataset.

4.2.3.2 WebContent Folder:

This Folder provides all the structure files that are need it to run the process in the JBoss server, here is in detail the folder structure:



```
WebContent
  css
    styles.css
  download
  images
  META-INF
  tabs
    tab1.xhtml
    tab2.xhtml
  templates
    footer.xhtml
    header.xhtml
    menu.xhtml
    template.xhtml
  WEB-INF
    classes
    faces-config.xml
    web.xml
  error.xhtml
  index.jsp
  index.xhtml
```

Illustration 4-7, WebContent structure

- The Css folder contained the style sheet that is used by the web application.
- The download folder contained the eis-lab.jar file that can be download it, on the web interface once the project is running in the server.
- The images folder is located the images that use the UI.
- The META-INF folder is needed for the server to provide information once it is running.
- The tabs folder contained the information of every tab of the main page of the project and provides the functionality that is display to the user.
- The templates folder provides the template that is used by the UI.
- In the WEB-INF folder are located all the classes after compilation, once the project is deploy on the server it contained the .class files.
- Also in the WEB-INF folder are located the files faces-config.xml and web.xml, the first one contains the definition of the backing beans (Is the one that configure the jsf), the second one save the configuration related with navigation rules of the system or the configuration of the Web application.
- Also in the folder are located three independent files, the main one is the one that is call index.xhtml that is the one that is process it once the user access to the system.

4.2.3.3 Ant Files

The ant file is in charge to build the entire project automatically and help the users to build the project and its dependencies.

The ant.properties contained the paths for the server of the maven installation directory and the JBoss installation directory.

The build.xml file is the executable ant, which the user should run to build the project.

5 TESTING

Comment [MC11]: Created by Rauf Agayev

5.1 JUnit Test

Requirement Name	JUnit testing		
Priority	High		
Author	Rauf Aghayev, Vugar Aghayev Carlos Montoya	**Creation Date**	02-08-2014
Revision Date		**Modifications Date**	
Summary	This section of documentation shows JUnit testing results for metric classes. For every metric class result of JUnit testing will be shown. For this purpose local data dumps was used.		

Verifiability – Authenticity	**Explanation**
Operations have been done in 801 milliseconds Metric value positive: 1.0 Metric value negative: 0.0	The test made the assertions loading two datasets, one to assert if the dataset contain information related with the authenticity, the metric return 1 if the information is contained, and 0 if not.
Verifiability – Digital Signature	**Explanation**
Operations have been done in 801 milliseconds Metric value positive_1: 1.0 Metric value positive_2: 1.0 Metric value negative: 0.0	The test made the assertions loading three datasets, two of them are use to assert if the dataset contain digital signatures, the metric then should return 1 if the digital signature is contained, and 0 if not.

Free Of Error	**Explanation**
Operations have been done in 767 milliseconds Metric value : 0.8368	This screenshot shows output of "testFreeOfError" method. 98 triples was loaded from data dump in 22 seconds and 84% of them are free of error. This percentage calculated by number of free of error triples divided by number of all triples.
Measurability	**Explanation**
Operations have been done in 1 seconds Metric value: 1.0	The screenshot depicts results of computation. The program run very fast, even it doesn't take 1 second. The reason behind metric value was introduced during documentation of metric classes. It outputs 1.0 if data is measurable 0.0 if it is not.

5.2 Performance Test

Requirement Name	Performance Test		
Priority	High		
Author	Rauf Aghayev, Vugar Aghayev	**Creation Date**	10-09-2014
Revision Date		**Modifications Date**	
Summary	This section of documentation shows Performance Testing. For this two different approaches was used. First approach is storing RDF triples in apache server, in repositories and accessing them from Eclipse by using http://localhost:8080/openrdf-sesame/repositories/Data_Set_Name. Second approach is accessing RDF triples by using global IP - http://es.dbpedia.org/sparql		

Apache Server		
Source	**Number Of Triples**	**Running Time**
http://localhost:8080/openrdf-sesame/repositories/Data_Set Data dump was taken from- https://developers.google.com/freebase/data	In data set there are 2.1M number of triples but for testing 500.000 of them was loaded.	35 seconds
Global Store		
Source	**Number Of Triples**	**Running Time**
http://es.dbpedia.org/sparql	In data set there are more than 10M number of triples but for testing 500.000 of them was loaded.	53 seconds

Comment [MC12]: Created by Carlos Montoya

6 INTALLATION AND USER MANUAL

6.1 INSTALLATION FOR DEVELOPMENT

With the aim to maintain or to extend the eis-lab application there is need to configure the development environment, which consist on the following software:

- Java JDK 5.0
- Eclipse
- JBoss 2.3.GA

After complete the installation of the need it software, it is need it to configure each of them.

6.1.1 Eclipse

Eclipse is an IDE (integrated development environment); this one is the tool that allows to modify or to change the java code.

6.1.1.1 Configuration

The source code[10] that is attached to the Project, then you are going to find a folder that contained the two projects that are necessary to run the program on the development environment.

To import the Project to eclipse you should import the Projects to the IDE.

Once it is done you should have something like the next image:

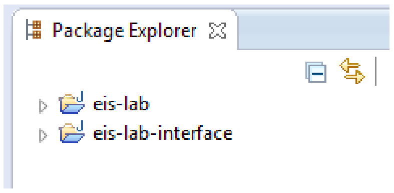
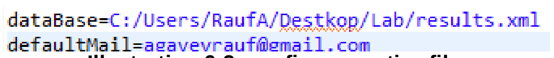


Illustration 6-1, Projects imported

On the path eis-lab/src/main/java you should have one file that is call config.properties (see Illustration 4-3).

On this file you should configure the two properties displayed, they look like this:

[10] The source code can be download from: https://github.com/Punkar22/Eis_Lab



```
dataBase=C:/Users/RaufA/Destkop/Lab/results.xml
defaultMail=agaveyrauf@gmail.com
```

Illustration 6-2, config.properties file

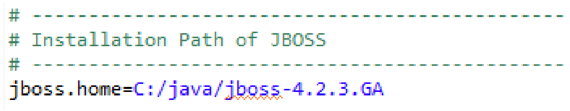
The first one is the directory where the program is going to store the results, and the default user mail to send the results after running the program (the program takes a lot of time while it runs, it has to stream millions of data).

6.1.2 JBoss 4.2.3 GA

The JBoss is the applications server where it deployed the eis-lab application.

6.1.2.1 Configuration

After the JBoss installation you should configure the ant.properties file that is located in the eis-lab-interface project, to specify the installation path.



```
# -----------------------------------------------
# Installation Path of JBOSS
# -----------------------------------------------
jboss.home=C:/java/jboss-4.2.3.GA
```

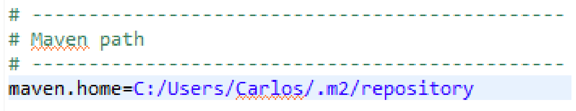
Illustration 6-3, ant.properties - JBoss path

6.1.3 Maven

Maven is used by the project to provide all the libraries need it to run the Project over RDF datasets.

6.1.3.1 Configuration

Also you should configure the ant.properties file that is located in the eis-lab-interface Project.

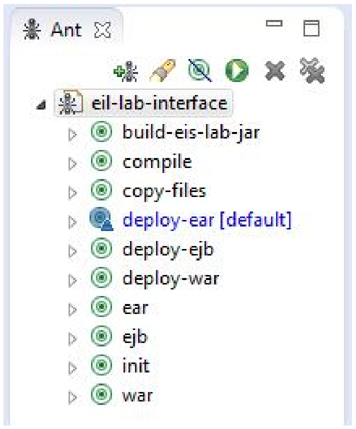


```
# -----------------------------------------------
# Maven path
# -----------------------------------------------
maven.home=C:/Users/Carlos/.m2/repository
```

Illustration 6-4, ant.properties - Maven path

6.1.4 Run the program on the server

Now that everything that is necessary, the project it is able to be deployed into the JBoss server, to do that you should go to the file that is located in the project eis-lab-interface. Your IDE should have installed the tools to run the ant files.



eil-lab-interface
- build-eis-lab-jar
- compile
- copy-files
- deploy-ear [default]
- deploy-ejb
- deploy-war
- ear
- ejb
- init
- war

Illustration 6-5, ant build view

In this view the user only have to give double-click on the mark on blue (deploy-ear), this process creates all the necessary files and copy them to the JBoss server. Also this process copy all the files to the directory eis-lab-interface/dist, this is useful if you want to send only the .ear files or the .jar files to another user.

After execute the ant build, it is necessary to run the JBoss server, to do this you have to go to the JBOSS_HOME\bin and then start the server.

To see the program you need to open the web page: http://localhost:8080/EIS_LAB_QAENTLOD/index.jsf.

6.2 USER MANUAL

6.2.1 Flow of the program.

To understand how it works the processing of the program you should understand the producer-consumer problem[11]. In this lab, the problem is solved in the project eis-lab in the package de.unibonn.iai.eis.qaentlod.io.streamprocessor.

This is implemented to improve the times that takes to stream one SPARQL endpoint, because usually these endpoints contained millions of RDF points of information.

[11] http://en.wikipedia.org/wiki/Producer%E2%80%93consumer_problem

Then the producer in in charge to establish the connection to the SPARQL endpoint, then using SPARQL queries it retrieved all the information by batches of size 10000 triples per time[12].

While the producer is calling the information, the consumer is running and waiting for all the information that is published by the consumer.

In the consumer are located all instances of the Quality Metrics, so in every lecture that the program does it send to every metric to compute the value of the metrics.

This process continue while the producer is publishing information, when this one stops, then the consumer received the message that there is no more information pending to be publish. So it continues to end the program.

After ending the consumer, with all the values obtained by the metrics, it goes and save the information of every metric into the File that is design for this (the path was configured in the section 6.1.1.1).

The final step is that the client sends a mail of confirmation when the process is done. Next is a sequence diagram to explain the process.

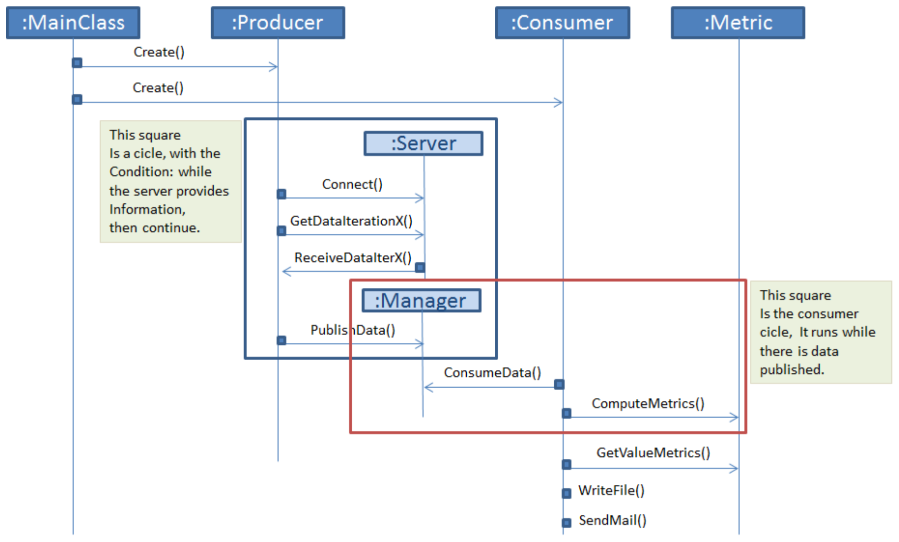


Illustration 6-6, Sequence diagram

6.2.2 How to Access and use the program.

To access to the web application the user need to access on the Web navigator to the link: http://localhost:8080/EIS_LAB_QAENTLOD/index.jsf.

[12] To accomplish this the project use LIMIT and OFFSET, see: http://www.w3.org/TR/rdf-sparql-query/#modOffset

Once it is running the user should be able to see the next things.

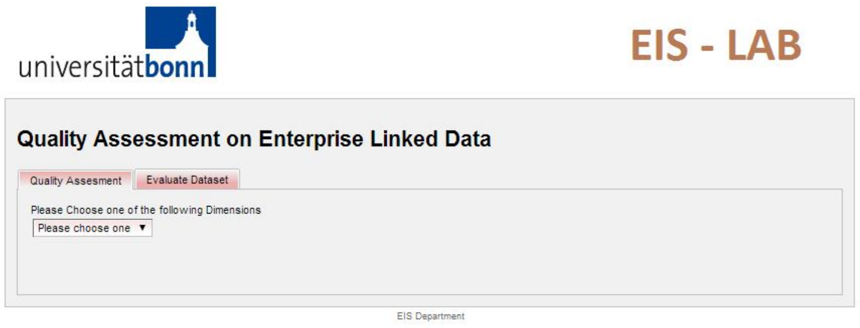

universitätbonn
EIS - LAB
Quality Assessment on Enterprise Linked Data
Quality Assesment
Evaluate Dataset
Please Choose one of the following Dimensions
Please choose one
EIS Department

Illustration 6-7, Init screen

There are two different functionalities available in the system:

6.2.2.1 Visualize results

The first tab was created with the aim to visualize the results; to make it work the user should follow the next steps.

First the user should choose one of the dimension that are display on the select panel, the next image show how it works.

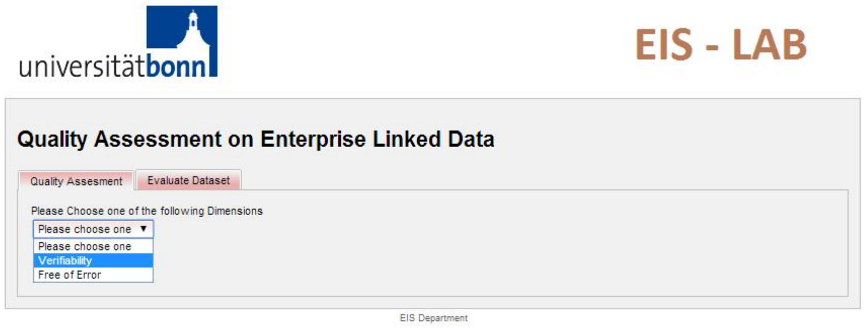
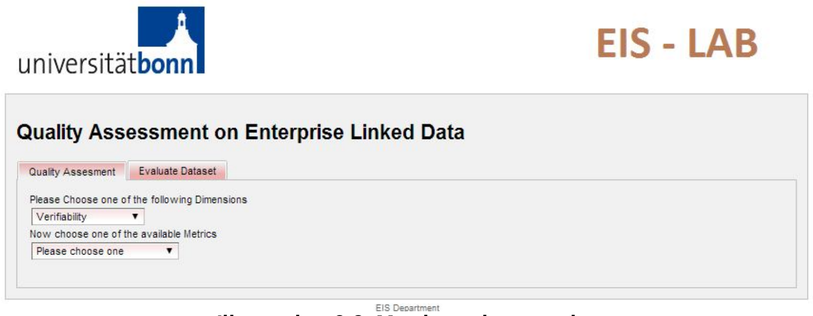


Illustration 6-8, Available dimensions

When the user has selected one of the dimensions then the system shows a new selectable combo box as shown in the next image.

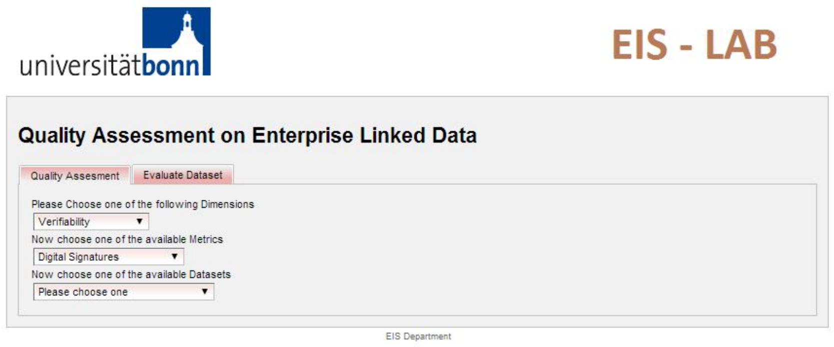


Quality Assessment on Enterprise Linked Data

Please Choose one of the following Dimensions
Verifiability
Now choose one of the available Metrics
Please choose one

Illustration 6-9, Metrics select combo

The user should select one of the available dimensions, once this is done, then the system display a new combo box, to show the available Data Sets that already had been evaluated.

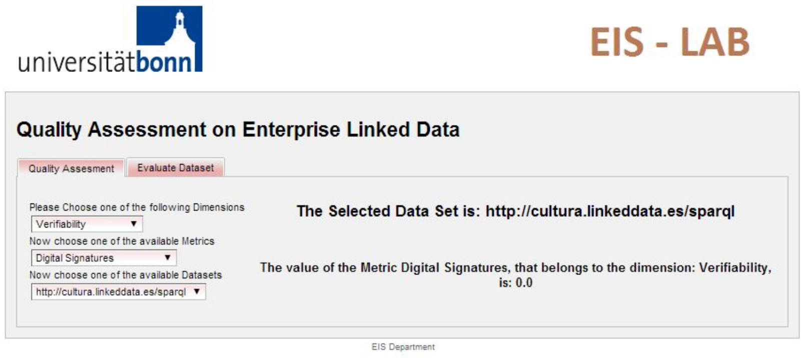


Quality Assessment on Enterprise Linked Data

Please Choose one of the following Dimensions
Verifiability
Now choose one of the available Metrics
Digital Signatures
Now choose one of the available Datasets
Please choose one

Illustration 6-10, Available Datasets

Then the user should select one of the datasets to see the results of this metric over the data set.



Quality Assessment on Enterprise Linked Data

Please Choose one of the following Dimensions
Verifiability
Now choose one of the available Metrics
Digital Signatures
Now choose one of the available Datasets
http://cultura.linkeddata.es/sparql

The Selected Data Set is: http://cultura.linkeddata.es/sparql

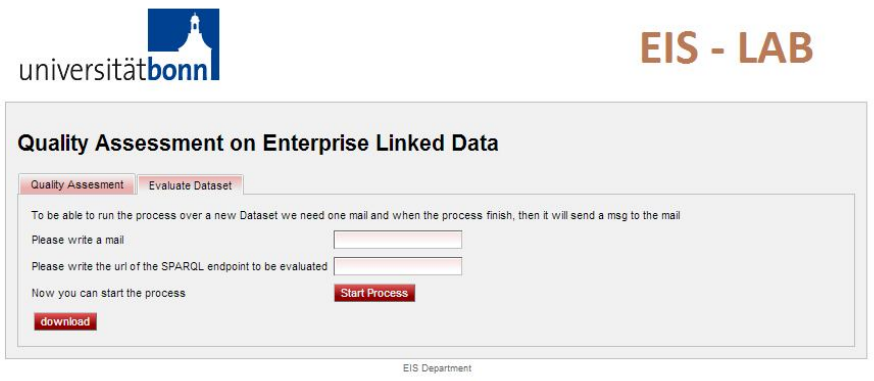
The value of the Metric Digital Signatures, that belongs to the dimension: Verifiability, is: 0.0

Illustration 6-11, Result Image

6.2.2.2 Evaluate new Dataset

The second tab provide by the system is in charge to evaluate a new data set or can be used to reevaluate one of the existing datasets.

When the user access to the second tab, the system shows the next screen:



To be able to run the process over a new Dataset we need one mail and when the process finish, then it will send a msg to the mail

Please write a mail

Please write the url of the SPARQL endpoint to be evaluated

Now you can start the process

Illustration 6-12, Evaluate new dataset

Then the user should to fill the information of user mail and SPARQL endpoint and then activate the “Start Process” button, this is going to start the process of evaluate the data set, once the process is done then the program will send an automatic mail to the specified mail.

Also the user can download the eis-lab.jar to work with it if they want to do it. To accomplish this, the user only has to click on the bottom “download” automatically it will be save to the computer.

7 REFERENCES

[i]. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer. (2012). *"Quality Assessment for Linked Open Data: A Survey"*. URL: http://www.semantic-web-journal.net/content/quality-assessment-linked-open-data-survey