Quality Assessment for Linked Open Data: Assessing the Trustworthiness and Relevancy Dimensions

A thesis presented to

Master of Science in Computer Science

Rheinische Friedrich-Willhelms-Universität Bonn

Under the supervision of

Prof. Dr. Sören Auer

Head of Department

Enterprise Information Systems Department

Institüt für Informatik III

Presented by

Carlos-Eduardo Montoya

Matriculation 2599128

Connollystr. 3/T45, 80809, München, BY

September, 2014

Acknowledgments

Firstly, I would like to thank Prof. Dr. Sören Auer for providing me with time and guidance during the process of elaboration of this master thesis, for responding to my questions, and being accessible whenever I needed. I also would like to thank Mrs. Jeremy Debatista, who performed as such an inspiring monitor of my work, given and thankfully to an agreement of work cooperation during the research. Without his patience through a consistent and dedicated feedback, this master thesis would have not been possible to achieve.

Secondly, I would like to thank the Colombian organization Colfuturo, for providing me with the scholarship that made the dream of achieving my master's education in Germany a reality.

Last but not least, I would like to thank my beloved parents for supporting me even when distance has been a huge barrier to overcome, to my daughter that provides the inspiration to get a better future and to my unique and irreplaceable wife, whose patience and love does not know limits.

Table of Contents

List of tables

Table of figures

ABSTRACT

Nowadays, as our world seems to be moving faster and be growing more rapidly every day, Web technologies have turned into a very useful tool for almost every human being, reason why the amount of information that can be used, requires some tools in order to check its quality to subsequently be trustful to produce new information enriched both in data and in analysis. Since unfortunately this so called quality is not fully assessed, a great amount of published data exerts very poor quality. Accordingly, this master thesis focuses on providing users with tools to assure that the quality of the data is properly established, measured and accomplished by the datasets. Moreover, the thesis responds to the main objective of assessing the Trustworthiness and Relevancy Dimensions and providing a tool capable of make this quality information explicit by providing quality metadata to the assessed datasets using the Dataset Quality Ontology (daQ), considering that evidently this quality assessment is not relaxed given the constant developments and changes of the data sources. To accomplish the last and develop this thesis, firstly a research was made on the current available tools, discerning their capabilities and restrictions, and secondly performing an implementation of various metrics evaluated both locally by using unitary test and by testing them against two data sets and comparing their results.

Keywords: *Rdf, Linked Open Data, Quality Metrics, Trustworthiness, Relevancy, SPARQL endpoints, Quality Metadata, Dataset Quality Ontology (daQ)* .

1. INTRODUCTION

1.1 Definition of the problem

With the rapid growth of Web technologies, data publishers started to reproduce more and more datasets as Linked Open Data (LOD) [Hartig, 2008]. At the current state, the LOD Cloud that covers eight different domains, including the Governmental and Geographical domains, discovered a total of 1014 datasets. Although these datasets follow the five star deployment guideline, data consumers are not able to identify which of the available datasets are "fit" for their use [Debattista, 2014][1] .

Data published on the Web only reveals a huge gap in the data quality, for that reason trust plays a really important role in the process of consuming data in many different circumstances such as communication between humans or data exchange between a human and a computer [Pattanaphanchai, 2011]. Untrustworthy data leads towards wrong decisions or may cause users misunderstand the concept or story, especially on the Web where an abundance of information is found. Unfortunately nowadays is more than evident a lack of control over this huge amount of data that has been published.

Consequently and since not all published data has been revised, it therefore cannot be considered fitness for use. With the aim to improve these gaps in the dimensions related with the trustworthiness and relevancy of published data, it is performed this master thesis.

1.2 Aim of the Master Thesis

As will be further discussed lately, the number of datasets that have been published has increased during the last decade, as well as the quantity of companies that have joined the initiative proposed by The Linking Open Data (LOD) project[2]. Both increases considered as successful behaviors have triggered a new set of requirements in order to manage this amount of data keeping as a focus the ability of getting advantages from it.

One of the key factors to manage data is to assure certain level of data reliability. This can be achieved through the development of a tool capable of differentiate datasets

[1] http://dl.acm.org/citation.cfm?id=2660525

[2] http://linkeddata.org/

given some specific conditions they should provide and that will be further discussed in the present document. Such a tool should be proficient in the evaluation of the Trustworthiness and Relevancy Dimensions, in which are involved both subjective and objective metrics.

In addition and in the aim of helping users to identify the main differences between datasets and its data quality, this master thesis will concentrate on measuring the values of the developed metrics for the mentioned dimensions.

By considering the results of the comparison between datasets, the user will be able to check the quality of data, which in conjunction with the comparison of the resultant metrics displayed in the developed User Interface (UI) will help to establish an important conclusion of which LOD is more fitness for use.

Listed below are the research questions that were formulated to analyze the results of this master thesis.

1.3 Research Questions

The two main research questions this master thesis aims to respond are:

a. Do the datasets published can be trusted? In relation to this question and the metrics involved it is important to highlight that for the objective metrics is possible to secure the accomplishment of the measurement of reliability. For the subjective metrics a fairly good approximation in the measurement of reliability will be made, producing the necessity of answering the following question.

b. Can subjective metrics be semi-automatized? In this direction, significant attention will be given to attain a comparison on the values obtained in the metrics to be applied to distinct datasets (at least two).

There are also various technical questions, which answers will be generated from the development of the software solution:

c. How should an online database be retrieved in order to be evaluated (data streaming)?

d. Which framework should be used to display the information?

e. Is there a tool that can be reused in the compute of the metrics defined?

1.4 Thesis Structure

The present master thesis will follow the following structure:

The chapter one consist of the introduction to the aim of the thesis and the research questions that were established in order to analyze the defined problem in relation to the reliability of published data and a brief of its implications. In this chapter the reader can understand the main purpose of the document and what to expect from it.

The chapter two comprises an explanation of the theoretical background in which relies the development of the master thesis, utilized in order to attain the objectives established, all within the framework of the revised current literature related to the topic. In this chapter the reader is able to comprehend the context before getting into specific aspects of the technological solution.

The chapter three is dedicated to explain how was structured the solution along with the construction of the required metrics, the stream processor and the user interface (UI). Here the reader will have access to the specifications of the solution generated by the master thesis.

The fourth chapter offers a validation of the obtained results from the solution through the running of tests and applying the quality metrics to two distinct datasets.

The fifth chapter provides a reflection on the limitations and strengths of the developed software.

The sixth and final chapter of the document delivers to the reader the conclusions and final discussion observed during the development of the thesis and some insights related to future work settled on the topic.

2. LITERATURE AND BACKGROUND REVIEW

2.1 Linked Open Data (LOD)

The term Linked Open Data (LOD) is related to using the Web to connect related datasets that was not previously linked, and/or to using the Web to lower the barriers to linking data currently linked using other methods. In addition, the main idea of LOD is to create typed links between data from different sources [Bizer, Heath & Berners-Lee, 2009]. More specifically, Wikipedia defines Linked Open Data as "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF"[3]

The main goal of LOD is to improve the value of the data available in public datasets by exposing them on the Web using standardized technologies, and by interlinking related items so that clients can easily combine information from various sources [Schndl, 2009]. To make this possible the LOD principles are integrated with the Web Architecture and technologies [Bizer, Cyganiak & Heath; (2007)]. These principles are a set of rules for publishing data on the Web in a way that all published data becomes part of a single global space. These are:

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

Accordingly to these principles the objective consists of people be able to look easily up on those names, and also provide a structured information that easily references and connects the information with another sources. This combination of facts make possible the discovery of new information related in different datasets. In this context the standard of Resource Description Framework (RDF) is introduced as a tool to describe the data.

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

Linked data is essentially about publishing structured data in RDF using the URIs rather than focusing on the existence of the data per se. This simplification decreases the obstacles for data providers, and therefore fosters a wide-spread adoption as it is mentioned by Hausemblas, [2009].

The Linked Open Data (LOD) project started out in early 2007 with a relatively modest number of datasets and participants, and it has grown since then both in terms of depth, impact and contributors. The Figure 1 represents the initial state of the cloud (developed by Richard Cyganiak and Anja Jentzsch)[4]

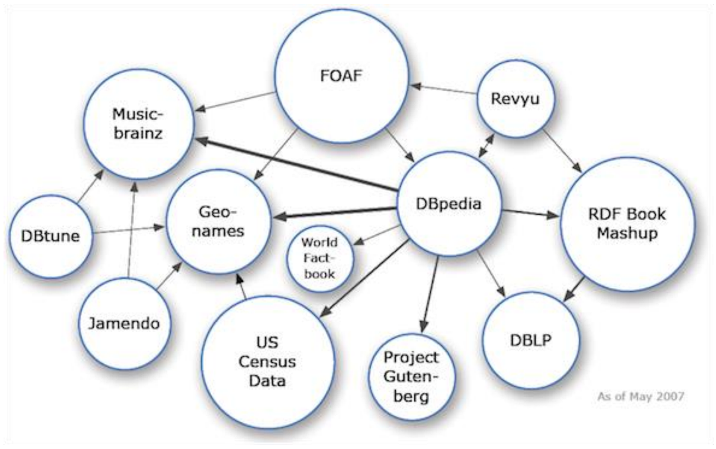


Fig. 1, State of the LOD community in 2007

The success of the project is evident as shown in the Figure 2 (LOD cloud, Cyganiak and Jentzsch)[3] as follows. Currently the project includes over 570 different datasets representing a gradually growing and open implementation of the linked data principles.

[4] Information retrieved from: http://lod-cloud.net/, as of 15-Sep-2014.

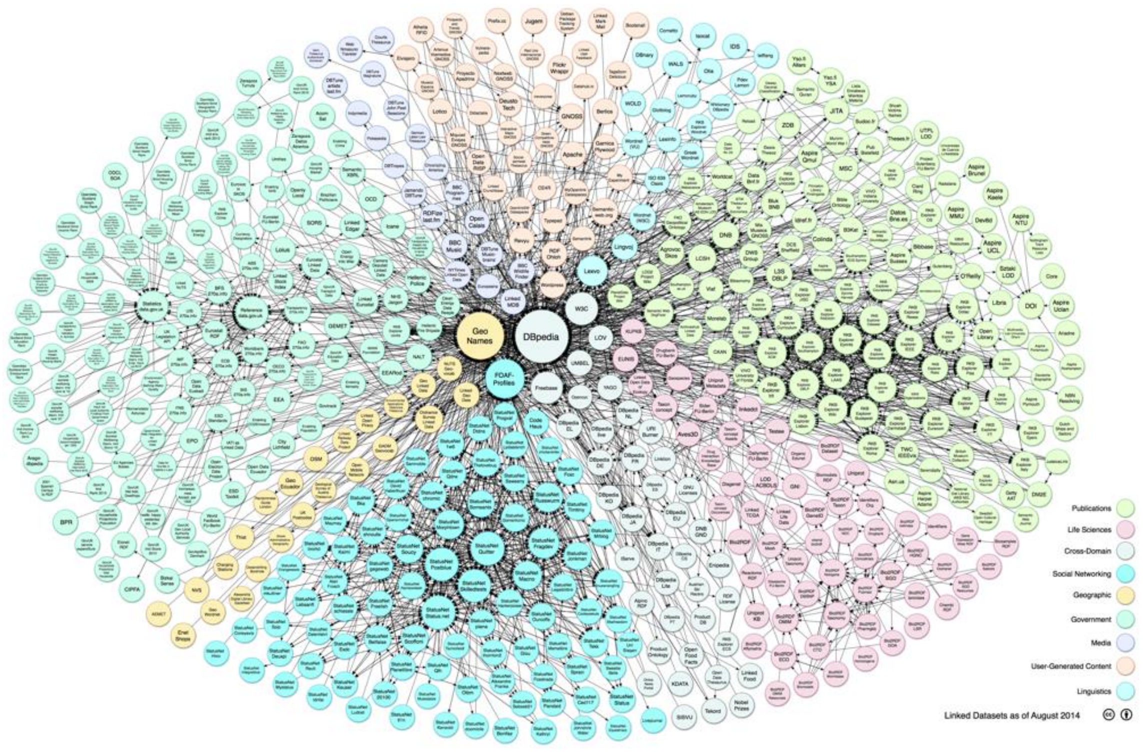


Fig. 2, State of the LOD community in Ago, 2014

2.2 Resource Description Framework – RDF

The Resource Description Framework (RDF) is a standard model for data interchange on the Web, or as it is mentioned in the W3C's recommendations[5], "is a framework for representing information in the Web". RDF has important characteristics that assist data integration even if the fundamental schemas diverge, and it specifically supports the evolution of schemas over time without demanding all the data consumers to be changed.

Furthermore and according to W3C, RDF covers the linking structure of the Web in order to use URIs to define what is usually referred as a "triple", meaning the relationship between things, as well as the two ends of the link[4]. As simple as this model may be seen, its use permits structured and semi-structured data to be mixed, exposed, and shared across different applications.

Each of the triples consists of a set of elements such as a subject, a predicate and an object. The subject and object are nodes and the predicate is the connector between them.

---

[5] http://www.w3.org/TR/rdf11-concepts/, retrieved on 20-Ago-2014

Each node can be one of three types: Internationalized Resource Identifier (IRIs)[6], literals or blank nodes. Likewise when these triples are associated in a set they are called RDF graph. The following image represents this concept.

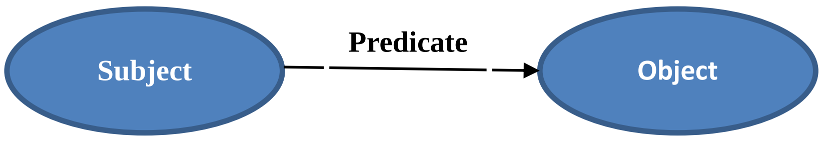


Fig. 3, Basic RDF graph

A RDF statement articulates an association between two resources, the subject and the object, and the predicate is their relationship.

The linking structure as shown in Figure 3 arranges a directed, labeled graph, in which the named link between two resources is represented by the edges, thus the graph nodes. This graph view is the simplest imaginable mental model for RDF, reason why it is commonly used in easy-to-understand visual explanations.

Each of the components of the triple can have a specific type, as follows:

- The subject, can be an IRI or a blank node.
- The predicate, is an IRI.
- The object, can be an IRI, a literal or a blank node.

An IRI within a RDF graph is a Unicode string that conforms to the syntax according to Dürst, M., Suignard, M., [2005]. The literals are usually specific values such as strings, numbers or dates. And the blank nodes are disjoint from IRIs and literals. Accordingly, the set of possible blank nodes is arbitrary.

In the intent of understanding the functionality of the RDF it is necessary to recognize the meaning of the RDF vocabulary concept, consisting of a collection of IRIs proposed to be used in RDF graphs. Some of the more common used examples are:

[6] Denotes something in the world (the "universe of discourse")

Table 1, RDF Vocabularies examples:

Namespace prefix	Namespace IRI
Rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
Rdfs	http://www.w3.org/2000/01/rdf-schema#
Xsd	http://www.w3.org/2001/XMLSchema#
foaf	http://xmlns.com/foaf/0.1/

An example of how a stored dataset looks like one of the many data editors (in this case turtle[7]) is:

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .

<#green-goblin>
   rel:enemyOf <#spiderman> ;
   a foaf:Person ;    # in the context of the Marvel universe
   foaf:name "Green Goblin" .

<#spiderman>
   rel:enemyOf <#green-goblin> ;
   a foaf:Person ;
   foaf:name "Spiderman", "Человек-паук"@ru .
```

In the example there are various namespaces defined such as rdf, rdfs, etc. Similarly the code includes some triples. The following graph (Figure 4) gives an example of the triple <#green-goblin> rel:enemyOf <#spiderman>:

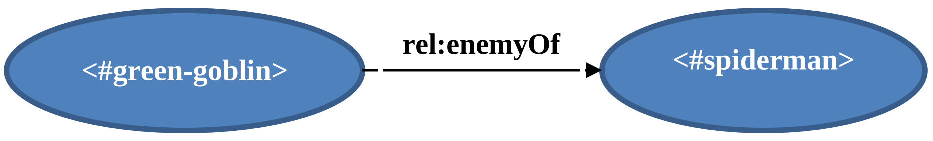


Fig. 4, RDF triple example

[7] http://www.w3.org/TR/turtle/

2.3 SPARQL

SPARQL is the acronym for Simple Protocol And RDF Query Language, which is a RDF query language, used for databases, able to retrieve and manipulate data stored in RDF format. A standard was made by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium[8], and is recognized as one of the key technologies of the semantic Web[9].

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

The definition of a SPARQL query[10] is given by a tuple (E, DS, R) where:

- E is a SPARQL algebra expression.
- DS is an RDF Dataset.
- R is a query form.

According to the definition SPARQL can be used to express queries that allow to consult different datasets, if the data is stored both as native RDF or even if the RDF is given by some middleware that publishes it as a RDF statement. Likewise, SPARQL contains the capabilities for the search of mandatory or optional graph patterns, both with the joints or disjoints. Its queries can be sets of results or RDF graphs.

The SPARQL protocol is closely related with the following specifications:

a. To send or receive queries, as defined by Clark, K. (2008)

b. Having the XML specification format of the queries response, as defined by Beckett, D. (2008)

A basic example of SPARQL query[11] consists of:

[8] http://en.wikipedia.org/wiki/World_Wide_Web_Consortium

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

[10] http://www.w3.org/TR/rdf-sparql-query/#sparqlDefinition

[11] http://skos.um.es/TR/rdf-sparql-query/, retrieved on 10-Ago-2014

The data stored in the dataset is:

```
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
_:a  foaf:name   "Johnny Lee Outlaw" .
_:a  foaf:mbox   <mailto:jlow@example.com> .
_:b  foaf:name   "Peter Goodguy" .
_:b  foaf:mbox   <mailto:peter@example.org> .
_:c  foaf:mbox   <mailto:carol@example.org> .
```

The query to retrieve some information is:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
  { ?x foaf:name ?name .
    ?x foaf:mbox ?mbox }
```

Then the answer to the formulated query is:

Name	Mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

2.4 Quality for Linked Open Data

As already stated RDF and SPARQL are brand new models that use all the resources provided by the Web and how this model needs a specification to interact with the users. It is essential to highlight that the model is created in such a way that one computer can understand the meaning of one sentence [Bizer, Heath & Berners-Lee, 2009].

With the creation of this model a lot of data sources have joined and therefore the amount of published data has become enormous, thus is really relevant to manage it, but before explaining what is the quality for Linked Open Data, it is necessary to answer the questions: is it necessary to manage all the information? and, is all the data really relevant?

The answer to these two last questions is no. This because when each of the data sources that has been published contained a specific purpose, some of these are, GEO location, BIO development information, academic research, and so on, then if the management of all information is not needed, is imperative to create tools that allow the users (whether humans and/or machines) can distinguish between what is good and bad information.

Knight, S.A. and Burn, J. (2005), make reference and describe how the information is conceived as fitness for use for only a certain application or use case. They mention also that some datasets with quality problems should be useful for some applications. In the paper "Quality Assessment for Linked Open Data: A Survey" by Zaveri, Rula, Mourino, Pietrobon, Lehmann & Auer, [2012], the authors gave a clear example with DBpedia[12], in which the data quality is sufficiently enough to enrich the Web search with facts or suggestions about entertainment topics. In this scenario, DBpedia can be used to display personal information, when any given user searches for an actor for example, in this case it is admissible that some personal facts of the actor are missing. Dissimilarly to achieve the development of a medical application the quality of DBpedia is insufficient, this because then the search engine most likely requires the complete information in relation to the medical cases and even more any lack of information could lead to important mistakes or even affecting human lives.

In the same vein it is critical to develop the so called tools able to help in the measurement of quality in published datasets and besides share this quality of information in such a way that can be useful for others.

There are several characteristics the datasets hold and that are called dimensions. These dimensions have been defined to measure quality. An interesting classification has been given by Zaveri et al., [2012], where the authors summarize the quality dimensions into a number of 18 that can be applied to LOD. These dimensions can be grouped also by its measure, as follows: accessibility dimensions, intrinsic dimensions, contextual dimensions, representational dimensions and inter-relationships between dimensions.

[12] www.dbpedia.com

Table 2, Quality Dimensions grouped

Group	Dimension
Accessibility	Availability
	Licensing
	Interlinking
	Security
	Performance
Intrinsic	Syntactic validity
	Semantic accuracy
	Consistency
	Conciseness
	Completeness
Contextual	Relevance
	Trustworthiness
	Understandability
	Timeliness
Representational-conciseness	Representational-conciseness
	Interoperability
	Interpretability
	Versatility

For each of the dimensions above mentioned, a set of quality assessments exist (for the purpose of this project they are renamed as ‘metrics’). These metrics are heuristic and are specifically designed to assess a particular situation that should be fulfilled by a given dataset.

This master thesis is focused on the assessment and development of the Contextual dimensions, specifically Trustworthiness and Relevance of the datasets.

2.4.1 Trustworthiness Dimension

Within this package of metrics there are some that measure credibility as explained by Bizer, [2007] in his doctoral dissertation, and further summarized and complemented by Zaveri et al., [2012]. These authors declared this metric as the “degree to which the information is accepted to be correct, true, real and credible”.

Correspondingly, within these dimensions there are some metrics related to the reputation of the dataset following the description made by Zaveri et al. [2012], where this metric is defined as “a judgment made by a user to determine the integrity of a data”. Dimensions based on the reputation that can gain one dataset through the experience of users is highly usable as is described by Gol, et al. [2007].

Lastly, another dimensions taking into account into this group of trustworthiness are the dimensions related with the verifiability of the dataset which are defined by Zaveri et al., [2012] as “the degree by which a data consumer can assess the correctness of a dataset”. The authors built this definition from the one that given by Naumann, [2002]. Based on Naumann´s definition the authors also established a set of seven quality metrics that are related in the table 3, as follows:

Table 3, Trustworthiness Dimension

Dimension	Metric
Trustworthiness	Trustworthiness of statements.
	Trustworthiness through reasoning.
	Trustworthiness of statements, dataset and rules.
	Trustworthiness of a resource.
	Trustworthiness of information provider
	Trustworthiness of information provider (content trust)
	Reputation of the dataset
	Trustworthiness of statements.

2.4.1.1 Trustworthiness of statements

The purpose of these metrics is to measure the provenance information that is provided by the dataset, as it is explained by Hartig, [2008] the trustworthiness of each dataset should be addressed. To accomplish this it is defined a “Trust Model for RDF data”, in which a function where the value of 1 represents the database being evaluated should be absolutely believable, meaning that the facts they contained are completely true. In contrast a value of -1 should be an absolute disbelieved.

The function is based on the relevant information that a dataset should have, such as publisher of the dataset, creation method, creation time, publication time of possible original sources, title of the dataset, summarized information of the dataset, etc. This function is based on the study of the named graphs and semantic sitemaps explained by Cyganiak, et al., [2008].

2.4.1.2 Trustworthiness through reasoning

This metric is intended to be used to assess information such as Blacklist or by an authority that repeatedly communicate if the dataset is reliable, as described by Bonatti, P., et al., [2011].

In this paper the authors defined and built the application based on previous experiences of the information retrieved. In order to achieve this they created a so called 'Blacklist' where they stored the datasets that they found to be harmful or to provide bad information. In comparison they created another list containing the datasets that are known to provide useful information.

During the development of this master thesis, it was found that this metric is very similar to the Trustworthiness Information Provider metric, which accomplishes a fairly equal goal but faces the problem with a different approach that will be explained in its description later below.

2.4.1.3 Trustworthiness of statements, dataset and rules.

These metrics are related to the construction of trust ontologies that assign trust values, 1 if the resource should be trusted or 0 if not.

The algorithm of the metric is described by Jacobi, I., Kagal, L. and Khandelwal, A., [2011], and mainly seeks for meta-information that must be on the datasets, like user ratings or something similar. The function is based on the definition for independent facts, rules and statements. For example we have one rule that should be trustful, then every new fact derived of that definition should be trustful.

The implementation of these metrics triggered a new problem in relation to every dataset having to contain the definition of a True Ontology allowing a consumer to evaluate if a resource should be trusted or not.

2.4.1.4 Trustworthiness of a resource

These metrics intend to compute trusted values between two entities. This is made by building or using a propagation algorithm that uses the trust ontologies to calculate if the resource can be trusted or not. As mentioned by Shelarpour and Katebi, [2010], the dataset should extend the dataset to store information related with the true (implementation of a true ontology).

Moreover this metric can be extended if there is the case of multiple paths existing, then these values can be aggregated into one value by a weighting mechanism that take all the paths and look for which one is trustful and which one not.

2.4.1.5 Trustworthiness Information Provider

The intention of these metrics is to check if the provider or some of the contributors of the dataset belong to an established list of trusted providers. This metric implements the list of trusted providers as a predefined list as suggested by Bizer, [2007], then the algorithm returns as true if the provider of the dataset is contained into the list of trusted providers, and false if not.

These metrics work on the premise of checking the provenance of graphs of the dataset, described and explained by Felmming, [2010], as the recognition of some attributes that bring information such as creator, contributor, provenance, publisher, source, if the data is derived from another one, etc.

2.4.1.6 Trustworthiness of information provided (content trust)

These metrics mainly look for a trusted fact to establish the accuracy of the dataset, this means that it should look for attributes such as creator, contributor, provenance, publisher, source, if the data is derived from another one, etc., and then if the dataset contained some of these values, it should be trusted, and accordingly if the dataset does not contain any of the attributes it should not be trusted.

These metrics can be measured by the provenance of digital signatures that can be contained into the dataset. If the dataset contains such digital signatures then they can be checked, and if they are true the dataset is trustful.

2.4.1.7 Reputation of the dataset

The last of the metrics that belong to the Trustworthiness dimension is based on the commentaries of experts' users, as explained by Mendes, et al., [2012]. The system should be capable of taking a decision based on the recommendation of an expert user.

To accomplish this the system should build data based on the opinions of experts' users that recommend the datasets and its veracity.

2.4.2 Relevancy Dimension

This dimension is defined by Zaveri et al., [2012] as "the provision of information which is in accordance with the task at hand and important to the users' query". They merged the amount-of-data dimension that is described by [Pipino et al, 2002] with the metric Coverage because both of them are almost equal. Once it is done, they came to the conclusion that the metrics belongs to the Relevancy Dimension. These metrics are referenced in the table 4, as follows:

Table 4, Relevancy Dimension

Dimension	Metric
Relevancy	Relevant terms within meta-information attributes
	Coverage

2.4.2.1 Relevant terms within meta-information attributes

The main purpose of this metric is to detect relevant terms into the dataset and based on this amount of terms establish a relevant value that can be understandable to the final user. This can be accomplished by taking into consideration three different approaches as described by Zaveri, et al., [2012], these are:

Firstly, count the number of occurrences of relevant terms within meta-data attributes [Bizer, 2007]. The algorithm should establish a base and return the percentage of attributes that contain attributes such as title, description and subject.

Secondly, a solution can be provided by the hyperlink analysis and by using information retrieval methods that measure the appearing of terms with meta-data information [Bizer, 2007].

Thirdly, it is proposed to build a ranking to determine the centrality of RDF documents and statements.

2.4.2.2 Coverage

This metric measures the coverage and the level of detail of a dataset, with the aim to fulfill the requirement that the level of detail should be considerable for certain task, for example in a medical application the detail level should be high, meanwhile in an entertainment application the level of detail can be low.

The algorithm return a percentage value of the number of attributes that contained the attributes that were defined by a master element.

In the following chapter it is described the architecture solution provided during the development of this master thesis.

3. ARCHITECTURE OF THE SOLUTION

The Quality Application designed through the development of this master thesis is a Web application that integrates the Java technology and Jena framework. As main pattern the application is built using the pattern MVC (Model, View and Controller).

In the next graphic is shown how the project architecture was conceived:

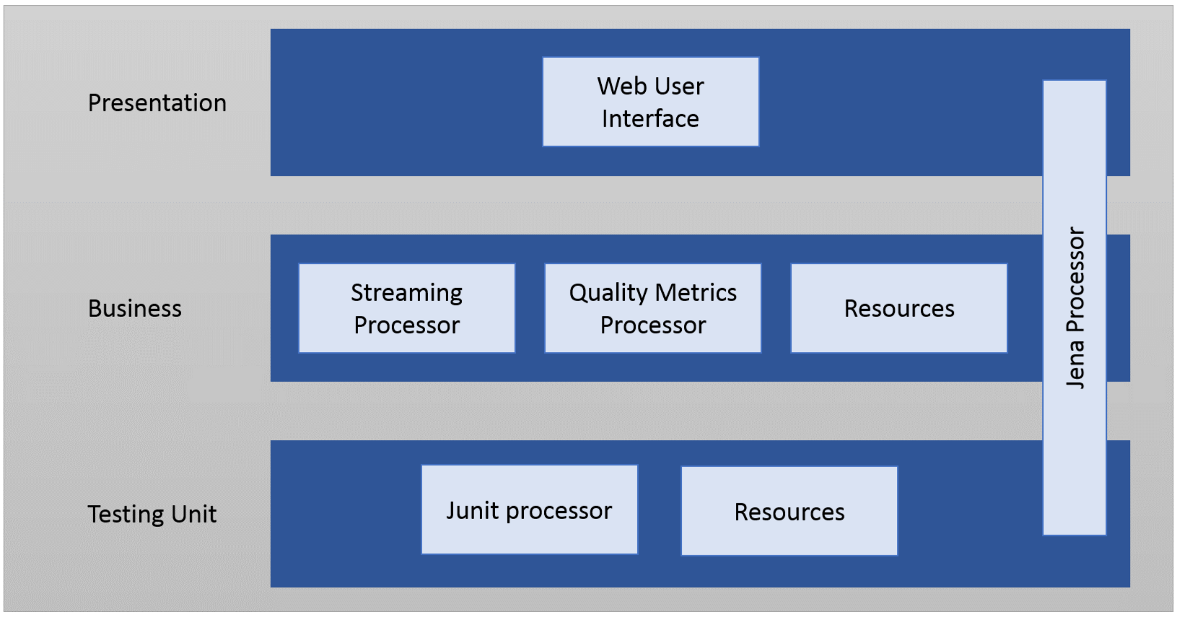


Fig. 5, General Architecture

The Presentation layer includes the component of Web User Interface (UI). This is the ownership of the interaction with the user, and it is in charge of showing the results when running all the metrics over the specified dataset.

The Business layer comprises three components: the first one is the Streaming Processor, which is the owner of the process to connect to the SPARQL endpoint and stream all the data to be used by the quality metrics processor. It is solved using the paradigm of producer-consumer to make it faster in its processing. The second component is the Quality Metrics Processor, which is the processor that computes the values for all the quality metrics developed and defined in the project. The third component is Resources, which is used to load the model into the diachronic data model, and work to load the advantages offered by Maven.

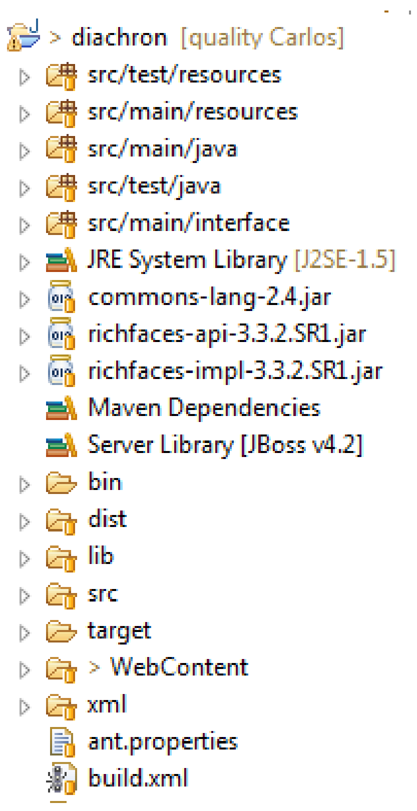
The Testing unit layer compounds the JUnit Processor that takes advantage of all the tools offered by JUnit and that was used to test the classes and assure their functionality. The Resources in this layer exemplifies datasets that were used by JUnit to make sure that the classes used found the correct value of the metrics.

To accomplish the required communication between all the modules we used the tools provide it by Jena, its classes and its tools to connect to the SPARQL endpoints. All this classes are assessed in all the application and they are independent from the other packages.

3.1 Project Structure

3.1.1 Quality Project

The Project is built by packages that contain all the modules necessaries to run the application and therefore be able to use it, as shown as follows:



```
> diachron [quality Carlos]
  src/test/resources
  src/main/resources
  src/main/java
  src/test/java
  src/main/interface
  JRE System Library [J2SE-1.5]
  commons-lang-2.4.jar
  richfaces-api-3.3.2.SR1.jar
  richfaces-impl-3.3.2.SR1.jar
  Maven Dependencies
  Server Library [JBoss v4.2]
  bin
  dist
  lib
  src
  target
  > WebContent
  xml
  ant.properties
  build.xml
```

Fig. 6, EIS-LAB Project structure.

The folder **Src/main/java** is the package containing all the classes that are needed to run the project, in this package the interface, the metric processor and the streaming processor are defined.

The folder **Src/main/resources** is the package containing all the files that are required by Maven to build the structure of the project.

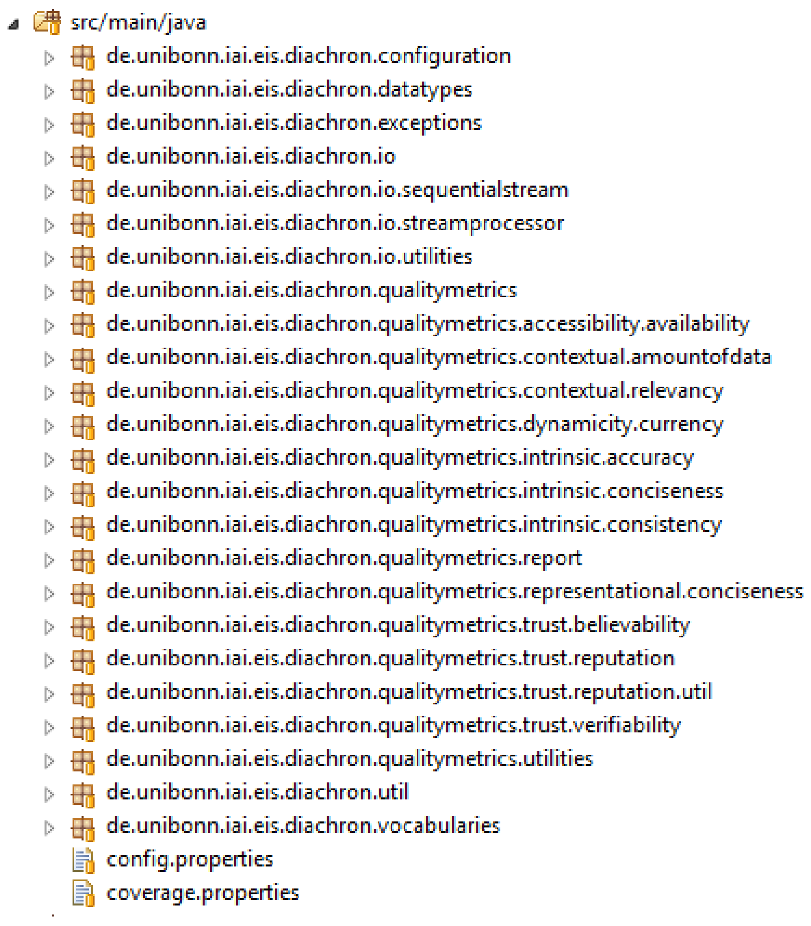
The folder **Src/test/java** is the package containing all the unit tests that were defined to test the classes contained in the main package.

The folder **Src/test/resouces** is the package that stores all the resources that are used to test the classes, it contain some examples for datasets.

It is important to mention that the project contains a reference to a library called Maven Dependencies, which contains all the libraries that are used by the packages described before.

3.1.1.1 src/main/java

As it was previously mentioned this package 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.diachron.configuration
  de.unibonn.iai.eis.diachron.datatypes
  de.unibonn.iai.eis.diachron.exceptions
  de.unibonn.iai.eis.diachron.io
  de.unibonn.iai.eis.diachron.io.sequentialstream
  de.unibonn.iai.eis.diachron.io.streamprocessor
  de.unibonn.iai.eis.diachron.io.utilities
  de.unibonn.iai.eis.diachron.qualitymetrics
  de.unibonn.iai.eis.diachron.qualitymetrics.accessibility.availability
  de.unibonn.iai.eis.diachron.qualitymetrics.contextual.amountofdata
  de.unibonn.iai.eis.diachron.qualitymetrics.contextual.relevancy
  de.unibonn.iai.eis.diachron.qualitymetrics.dynamicity.currency
  de.unibonn.iai.eis.diachron.qualitymetrics.intrinsic.accuracy
  de.unibonn.iai.eis.diachron.qualitymetrics.intrinsic.conciseness
  de.unibonn.iai.eis.diachron.qualitymetrics.intrinsic.consistency
  de.unibonn.iai.eis.diachron.qualitymetrics.report
  de.unibonn.iai.eis.diachron.qualitymetrics.representational.conciseness
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation.util
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability
  de.unibonn.iai.eis.diachron.qualitymetrics.utilities
  de.unibonn.iai.eis.diachron.util
  de.unibonn.iai.eis.diachron.vocabularies
  config.properties
  coverage.properties
```

Fig. 7, src/main/java packages.

For the project the most important packages are:

- de.unibonn.iai.eis.diachron.io: this package contains the main class to be executed. It provides the complete interface of interaction with the user and provides the user with different functionalities.

- de.unibonn.iai.eis.diachron.io.streamprocessor: this package contains the classes that solve the problem of streaming a SPARQL endpoint, it solves this issue by implementing the paradigma of producer-consumer.

- de.unibonn.iai.eis.diachron.qualitymetrics: this package contains all the metrics that were developed by the quality project. The ones that belong to the Trustworthiness and Relevancy dimensions are:

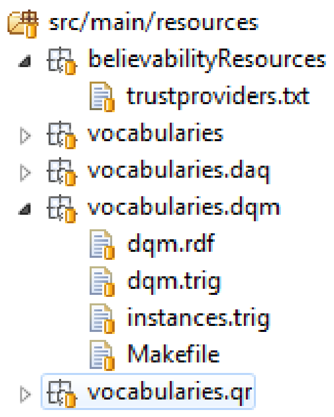
i. Contextual.relevancy.
ii. Trust.verifiability.
iii. Trust.believability.
iv. Trust.reputation.

- **de.unibonn.iai.eis.diachron.vocabularies**: this package includes all the information related with the vocabularies that were defined to be readable by the Maven processor.

Along with the src/main/java folders there are defined two property files (config and coverage), these are used within the metrics and by the streaming processor, and will be explained in detail in the next chapter.

3.1.1.2 src/main/resources

These packages contain all the resources that are necessary to run on Maven, and are compound by:



```
src/main/resources
  believabilityResources
    trustproviders.txt
  vocabularies
  vocabularies.daq
  vocabularies.dqm
    dqm.rdf
    dqm.trig
    instances.trig
    Makefile
  vocabularies.qr
```

Fig. 8, src/main/resources folders.

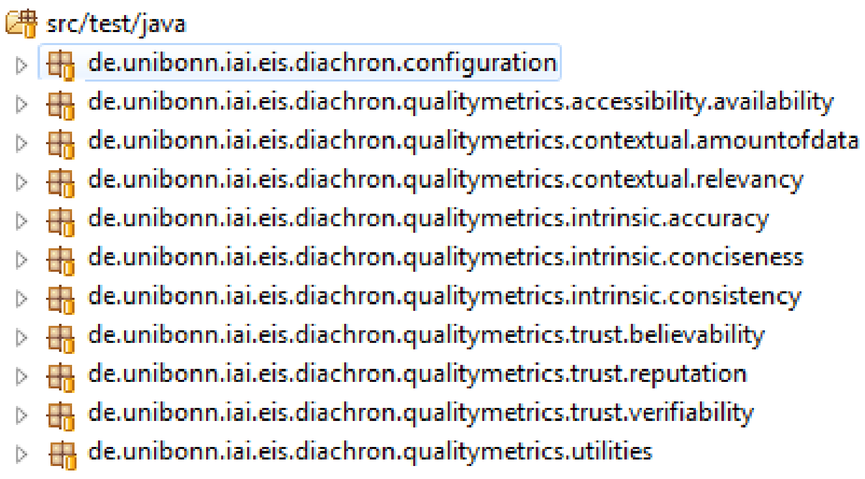
The main folder is called vocabularies and contains all the necessary files that specifies in Maven the languages that the project should read and use to create its vocabulary.

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

The file called trusproviders.txt contains the definition of all the trusted providers that are used by the quality metrics, specifically the Identity Information Provider Metrics that will be described in the next session

3.1.1.3 src/test/java

This folder is the one comprising 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.accessibility.availability
  de.unibonn.iai.eis.diachron.qualitymetrics.contextual.amountofdata
  de.unibonn.iai.eis.diachron.qualitymetrics.contextual.relevancy
  de.unibonn.iai.eis.diachron.qualitymetrics.intrinsic.accuracy
  de.unibonn.iai.eis.diachron.qualitymetrics.intrinsic.conciseness
  de.unibonn.iai.eis.diachron.qualitymetrics.intrinsic.consistency
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability
  de.unibonn.iai.eis.diachron.qualitymetrics.utilities
```

Fig. 9, src/test/java packages.

3.1.1.4 src/test/resources

This folder contains all the resources that were used to test the developed classes, this files are local data dumps that emulate a faster access to a SPARQL end point.

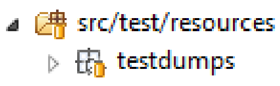

src/test/resources
testdumps

Fig. 10, src/test/resources testdumps.

3.1.1.5 src/main/interface

This package contains all the resources needed to run the application on the Web, here are defined the backing beans that are used by the UI following the pattern MVC. This layer correspond to the controller view.

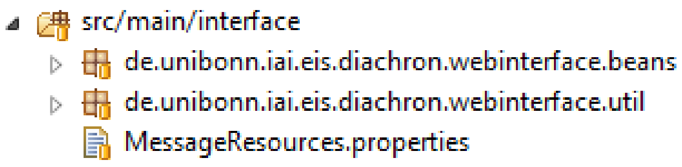
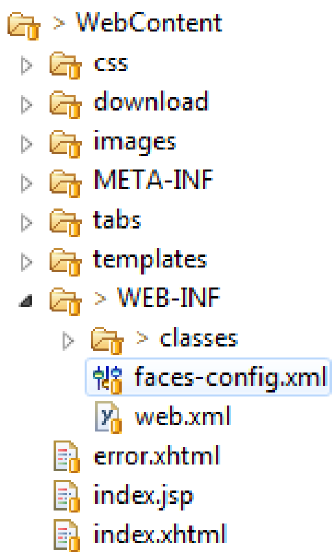

src/main/interface
de.unibonn.iai.eis.diachron.webinterface.beans
de.unibonn.iai.eis.diachron.webinterface.util
MessageResources.properties

Fig. 11, src/main/interface

3.1.1.6 WebContent

This folder contains all the files that are used by JBoss to show the UI into the pattern of MVC. This layer defines all the structure of the View. Their files implement the framework RichFaces that allows the developer to define a modern UI with an easy manipulation.



```
> WebContent
  ▷ css
  ▷ download
  ▷ images
  ▷ META-INF
  ▷ tabs
  ▷ templates
  ◢ > WEB-INF
      ▷ > classes
          faces-config.xml
          web.xml
    error.xhtml
    index.jsp
    index.xhtml
```

Fig. 12, WebContent structure.

The faces-config.xml contains all the definitions needed by the JBoss processor to identify the Backing beans and its locations.

The Web.xml file describes the behavior that should have the application once it is deployed, meaning that all the navigation rules should be contained here, such as links, filters, etc.

Within the download folder it is contained the diachron.jar file to be exposed in the Web as a .jar library. This .jar contains all the classes compiled of the quality metrics that could be used for another project, in other words this .jar is the Model layer from our project.

The tabs folder and templates are as their names imply the definition of the functionality of every of the tabs shown in the UI and the templates that are used.

3.2 Metrics Implementation

3.2.1 Trustworthiness Metrics Implementation

3.2.1.1 Trustworthiness of statements

The development of this metric was based on the definition given by Cyganiak, et al., [2008], and as a result the metric called TrustworhitnessRDFstatements.java was created. This is located within the folder src/java/main and within the package de.unibonn.iai.eis.diachron,qualitymetrics.trust.believability.

In this class the algorithm should look for specific information into the dataset, this means that based on the amount of information that the metric finds, the metric value or metric function retrieves a value in the interval [-1,1]. Specifically if the function retrieves '-1', the consumer should completely distrust the dataset, if in contrast the function returns the value '1' this means the consumer should completely trust the dataset.

Thus to solve the algorithm the focus was to establish which attributes should be contained for the dataset, these attributes are:

- **i.** Publisher
- **ii.** Creator
- **iii.** Created
- **iv.** Source
- **v.** Title
- **vi.** Content
- **vii.** Home Url
- **viii.** Provenance

Every one of these attributes are already defined as a meta-data information from the Dublin Core[13], therefore the central idea of the code is to count into the dataset how many of the resources are indeed present in the dataset, by making the comparison between elements against the DC core. The attributes used from the DC core are:

1. http://purl.org/dc/terms/creator
2. http://purl.org/pav/createdBy
3. http://purl.org/dc/terms/created
4. http://purl.org/dc/terms/publisher
5. http://purl.org/dc/terms/source
6. http://purl.org/dc/terms/title
7. http://purl.org/dc/terms/description
8. http://purl.org/dc/terms/provenance

[13] http://dublincore.org/documents/dces/

Since it is still missing one of the properties into the dataset the program is using the specification given by FOAF Ontology[14], where firstly it looks for their own attributes in the dataset then if it finds this information it proceeds and looks for the triples, then the program tries to find the homepage information by using the attribute: http://xmlns.com/foaf/0.1/homepage

Once we have the ontologies used to find the information needed from the provider, the metric establishes the function that must summarize how many of those attributes are present in the dataset. At end the function return a value following the next distribution:

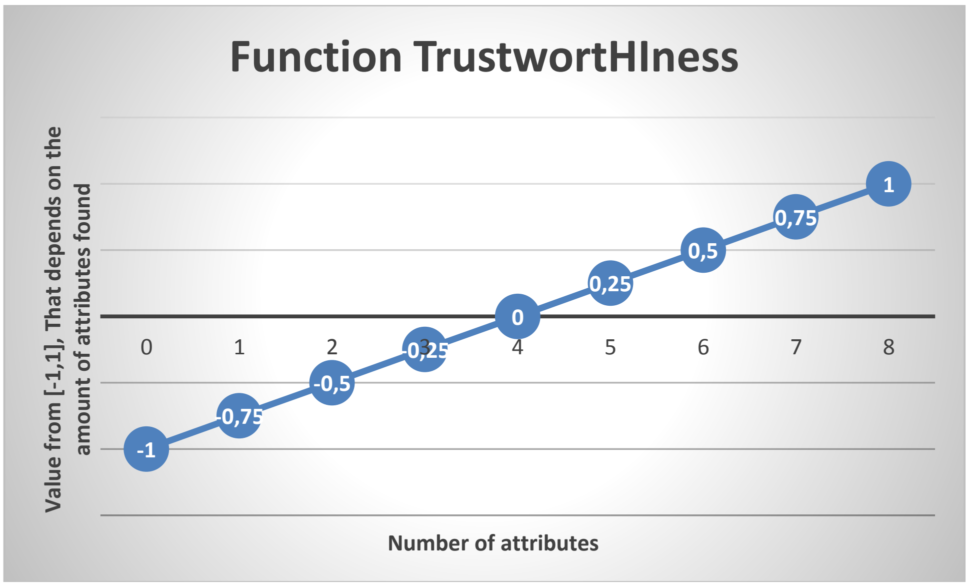


Fig. 13, Function for the Trustworthiness of statements metric.

3.2.1.2 Trustworthiness through reasoning

These metrics have two different approximations to be solved, the first one is based on the construction of one Blacklist, meaning that by the use or by the advice of some expert users the system should build a list that keeps the information of datasets that are known to provide untruthful information [Bonatti, P. et al., 2011].

The second approach that Bonatti exposed is that the system should use some kind of true ontology that gives the consumer an idea to derive a Boolean value if trusted or not into the given dataset.

[14] http://en.wikipedia.org/wiki/FOAF_(ontology)

The approximation used for the development of this master thesis is the one that constructs a Blacklist. Such implementation can be viewed within the folder src/main/java and within the package de.unibonn.iai.eis.diachron,qualitymetrics.trust.believability. The name of the class refers to the method that was decided to use, Blacklist.java.

To solve the problem of the Blacklist, a file that should be fulfilled by the system administrator was created. The file is located within the folder src/main/resources/believabilityResources and it is call blacklist.txt. This file comprises all the URIs of untrusted publishers or creators.

Once the file was created and when the metric is instantiated, this metric reads the file and keeps in memory the untrusted information of providers. When the consumer calls the metric, the program uses the DC Core ontology. The explanation to use this ontology is because it is well known and accepted to provide meta-information related with the provenance of the resources of the dataset.

The metric looks for the predicates: http://purl.org/dc/terms/creator and http://purl.org/dc/terms/publisher, and then it creates two distinct variables to summarize the number of creators or publishers found, and the number of publishers or creators of those that are store in the Blacklist, then it returns a value based in the function *f* that is applied over the dataset *x*, as follows:

$$f(x) = \begin{cases} 0.5 \quad , & where\ Number\ of\ publishers \\ & or\ creators\ is = 0 \\ 1 - \dfrac{\#\ of\ untrusted\ publishers\ \ or\ creators}{\#\ of\ publisher\ or\ creators}, & otherwise \end{cases}$$

Fig. 14, Function to stablish the value of the Blacklist metric measure.

The purpose of this function is that if in the dataset any of the publishers or creators are contained into the Blacklist the dataset is completely trustful, but in the other case if all of the publishers and creators are contained in the Blacklist then the dataset is not trustful at all.

In cases where the metric cannot find any information of any publisher or creator it returns the default value of 0.5, this value is given because the metric could not find any information, being this a probable risk, or it could be in reason to a new dataset that is still in process of completion, or some other particular situation, but this result is considered as fairly reliable information.

3.2.1.3 Trustworthiness of statements, dataset and rules and trustworthiness of a resource

From the moment these metrics started to be implemented a problem was found regarding the dataset having to contain some level of trusted ontology related to every of the statements. It was established that based on the queries the system had to be able to answer if the result is trustful, or if the new data or relationship generated can be trustful.

The found problem is related to the definition of the architecture because the project itself streams all the data from the dataset, this with the aim of establishing the quality of the complete dataset and not only of one part of it (query).

Since the metric Trustworthiness of a resource metric had the same issue previously explained, then the solution is to use the quality ontology to establish a path between two elements, one in which the trustworthiness is unknown and one in which it is known. As the main purpose of the project is to qualify a complete dataset the idea of finding a path between two resources became unnecessary, if it is found that the complete dataset is trustful then it means that all its resources should be trustful as well.

For the reasons just mentioned these two metrics were not taking into account into the development of the project.

3.2.1.4 Trustworthiness Information Provider

This metric can be achieved through different ways. This project made use of two of those ways, the first one is based on the concept given by Gamble, M. and Goble, C., [2011], where the dataset can be trustful and is given by the building of provenance graphs. In the authors´ explanation on how to measure this dimension, they looked for attributes related with the authenticity of the dataset. Based on this definition the metric called AuthenticityDataset.java was created, and it is located within the package de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability. The computation of the metric looks for specific attributes, these are:

1. http://purl.org/dc/terms/creator
2. http://purl.org/dc/terms/contributor
3. http://purl.org/dc/terms/publisher
4. http://purl.org/dc/terms/source
5. http://purl.org/dc/terms/provenance
6. http://purl.org/net/provenance/ns#DataPublisher[15]
7. http://www.w3.org/TR/prov-o/#wasDerivedFrom[16]
8. http://purl.org/pav/createdBy[17]

When some of these attributes are found the metric summarizes the value and then produces the final result of the metric. This value of the metric corresponds to the function *f* over the dataset *x*:

$$f(x) = \begin{cases} 1, & found\ at\ least\ one\ of\ the\ attributes \\ 0, & otherwise \end{cases}$$

Fig. 15, Function to stablish the metric value of Authenticity of the dataset metric.

The second approach to solve this metric is based on the suggestion given by Bizer, [2007], where the system must have a list of trusted providers. This was solved with the creation of the file called trustproviders.txt located within the folder src/main/resources//believabilityResources. This file contains a list of URIs for known trusted providers that must be created by an expert user, reason why it is not an objective metric because it requires a human interaction and every human interaction is attached to the subjectivity and even more depends on the domain where the dataset is situated.

The metric created for this second approach is called IdentityInformationProvider.java and it is located within the package de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability. The functionality of this

[15] Attribute found reading the Provenance Vocabulary Core Ontology Specification: http://trdf.sourceforge.net/provenance/ns.html#sec-intro, retrieved on Jul-2014

[16] http://www.w3.org/TR/prov-o/, Found on internet as a W3C specification into the Provenance Ontology, retrieved on Jul-2014.

[17] http://pav-ontology.googlecode.com/svn/trunk/pav.html - The Provenance, Authority and Versioning Ontology, retieved on Jul-2014

metric is similar to the Blacklist metrics, firstly it loads all the information of the trusted providers into memory, then when the computation of the dataset occurs, the metric looks for the attribute http://purl.org/dc/terms/contributor or for the attribute http://purl.org/dc/terms/creator.

If during the computation the metric finds some triples with those the two attributes defined, it increases the counter of creator or contributor. Also in parallel for every contributor or creator it increases the value of total number of creators. At the end of the function the computation of the metric is given by the function *f* over the dataset *x*:

$$f(x) = \begin{cases} \frac{\#\, of\ creators\ in\ the\ list}{\#\, of\ creators}, if\ the\ number\ of\ creator\ is\ at\ least\ 1 \\ 0 \qquad\qquad\qquad\qquad , if\ the\ number\ of\ creators\ is\ equal\ to\ 0 \end{cases}$$

Fig. 16, Function for the Identity Information Provider metric.

There is a third way to compute this metric that is suggested by Gil and Artz, [2007], and by Golbeck, et al., [2003], where the method is to develop a trusted ontology that can be used by the actual dataset, then the dataset should be capable to store this information, so the metric only have to read or have to find the resource where this value is saved and read the value from it.

The values proposed by these authors are defined as follows:

1. Distrusts absolutely
2. Distrusts highly
3. Distrusts moderately
4. Distrusts slightly
5. Trusts neutrally
6. Trusts slightly
7. Trusts moderately
8. Trusts highly
9. Trusts absolutely

For every resource in the dataset a value of the trusted ontology should be given by some expert users that know the domain of the dataset. Since it was found that the ontology should be implemented by the datasets, and that the metric just have to return the result consulted over these values. For this reason this third implementation was not developed into the project.

3.2.1.5 Trustworthiness of information provided (content trust)

This metric is solved by the development of two new metrics, the first one is based in the meta-information that is self-contained by the dataset, it should look for values such as title, content and URI.

This first developed metric is called ProvenanceInformation.java, it is located within the package de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability. After checking the validation given by Flemming, [2010], the dataset should contain at least the attributes listed below:

1. http://purl.org/dc/terms/creator
2. http://purl.org/dc/terms/title
3. http://xmlns.com/foaf/0.1/homepage

Only then if the dataset contains such attributes the computation can return as true, if by any chance one of the attributes is missing then the dataset cannot be trusted.

The function to return the value is defined as the computation *f* over the dataset *x*:

$$f(x) = \begin{cases} 1 \quad , & \mathit{Iff\ the\ title, the\ creator\ and} \\ & \mathit{the\ homepage\ are\ present.} \\ 0 \quad , & \mathit{otherwise} \end{cases}$$

Fig. 17, Function for Provenance Information metric.

The second metric that helps to reach the goal of this metric is called DigitalSignatures.java and is located within the package de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability. The aim of this metric is to check if the dataset owns one digital signature.

The ontology that fits the requirement to check the use of digital signatures is defined by Bizer, [2006], the SWP Ontology[18] provides meta-information like:

4. http://www.w3.org/2004/03/trix/swp-2/assertedBy

[18] http://wifo5-03.informatik.uni-mannheim.de/bizer/wiqa/swp/SWP-UserManual.pdf

5. http://www.w3.org/2004/03/trix/swp-2/authority
6. http://www.w3.org/2004/03/trix/swp-2/signature
7. http://www.w3.org/2004/03/trix/swp-2/signatureMethod
8. http://www.w3.org/2004/03/trix/swp-2/certificate

With these attributes provided by this ontology the goal is to check if the dataset contains some signatures that the consumer can use to verify if the dataset is trustful or not.

The function is defined by the computation value *f*, over the dataset *x*:

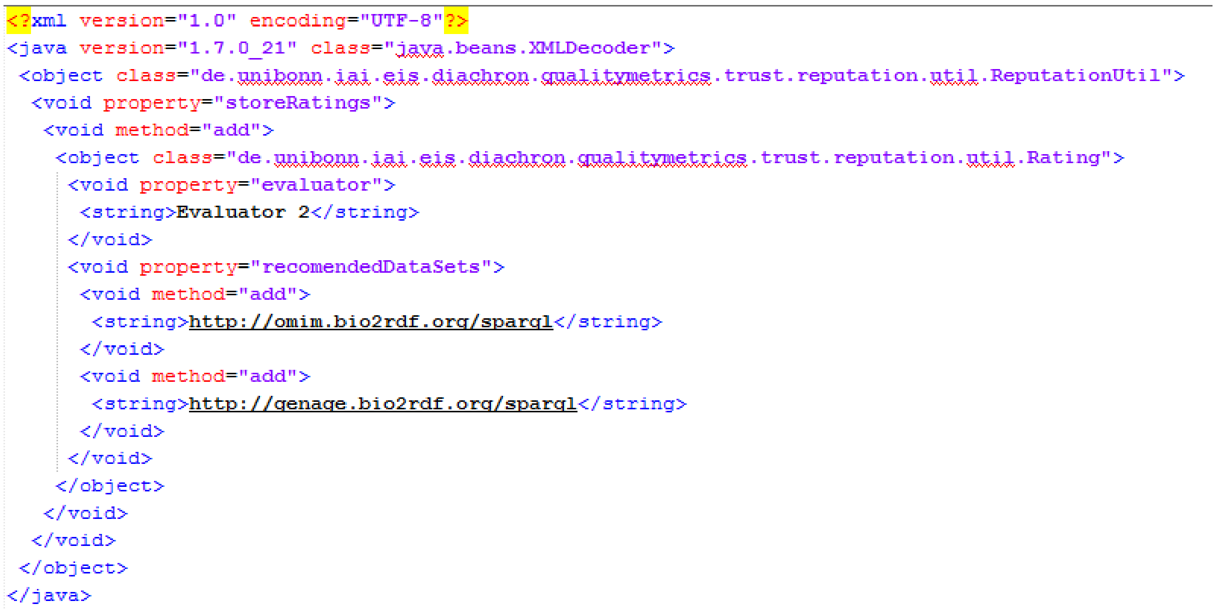
$$f(x) = \begin{cases} 1, & \text{some of the attributes is present} \\ 0, & \text{otherwise} \end{cases}$$

Fig. 18, Function for Digital Signatures metric.

3.2.1.6 Reputation of the dataset

The last one of the metrics that belongs to the Trustworthiness dimension is developed following the recommendation given by Mendes, et al., [2012], in which there were offered two different ways to measure the reputation of the dataset. The first one developed in this project was based on the creation of explicit rankings into the quality system, these rankings should be computed by expert users who recommend a set of given datasets. The second option given for the authors was based on the analysis of external links or by page ranks.

This project creates a metric called Reputation.java and is located within the package de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation. The main purpose of this metric is to provide a value that depends on the amount of expert users that recommend a certain dataset. To accomplish this, it is created a local file into the server based on the XML format which stores all the recommendations given by the expert users. The content of the file should be as follows:



```xml
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.7.0_21" class="java.beans.XMLDecoder">
 <object class="de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation.util.ReputationUtil">
  <void property="storeRatings">
   <void method="add">
    <object class="de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation.util.Rating">
     <void property="evaluator">
      <string>Evaluator 2</string>
     </void>
     <void property="recomendedDataSets">
      <void method="add">
       <string>http://omim.bio2rdf.org/sparql</string>
      </void>
      <void method="add">
       <string>http://genage.bio2rdf.org/sparql</string>
      </void>
     </void>
    </object>
   </void>
  </void>
 </object>
</java>
```

Fig. 19, Format of the ratings.xml file.

The file contains a list of rankings that should attain: a) every ranking should have an evaluator, it could be a string that stores the information of the given expert user that helps with the ranking, and b) every ranking must store a list of recommended datasets and every record should be an URI to the datasets.

The file contains the class definition because it takes advantages on the XML Encoder and XML Decoder to easily read or load the information in these classes. When the class is instantiated the first tasks that accomplish is to load all the rankings saved on the system, then when it computes the information based on the URI of the dataset the system returns the value of the metric. This value is given by the function *f* over the dataset *x*:

$$f(x) = \frac{\#\ of\ expert\ user\ that\ recomend\ the\ dataset\ x}{\#\ of\ total\ of\ expert\ users}$$

Fig. 20, Function for Reputation metric.

Once the value is calculated is possible to know if the dataset is recommended by some users. When the dataset is recommended by even more users the function value is more reliable. In comparison if the dataset is not recommended or if it is recommended by only a few users, then the consumer should take the decision to continue using the dataset or not.

Something that the consumer should be aware of is regarding the next two questions: who is an expert user?, and, is it the ranking of the expert user truly reliable?

Because the implementation is based on the direct interaction of the user with the system, the metric is categorized into the subjective metrics, this means that given the value of the expert users the metric value of one dataset may vary from one use of a system to another.

3.2.2 Relevancy Metrics Implementation

3.2.2.1 Relevant terms within meta-information attributes

As explained by Zaveri, et al., [2012], there are three different ways to solve these metrics. The first one explained by Bizer, [2007], consists of establishing a percentage of attributes that contain attributes such as title, description and subject. The second option also defined by Bizer consists of using methods of hyperlink analysis to look for terms with meta-data information. The last option is given by Bonatti, et al., [2011], and proposes the creation of one ranking similar to the page rank[19] that "determines the centrality of RDF documents and statemes" [Zaveri, et al., 2012].

The first solution described was the one adopted by this project. The goal is to find the occurrence of relevant terms within meta-information attributes. The first part of the problem is to find which meta-information attributes are relevant, to solve this in his dissertation Bizer, [2007] defines "Meta-information attributes such as title, description and subject classify and summarize content. The values of these attributes can be used as indicators for assessing whether content is relevant for a specific task". Besides it is proposed a function that counts the elements with those attributes.

This definition works as a starting point to create the metric called RelevantTermsWithinMetaInformation.java, that is located within the package de.unibonn.iai.eis.diachron.qualitymetrics.contextual.relevancy. Then the function that is defined to compute the value of the metric is given by the function *f* over the dataset *x*, where*:*

[19] http://en.wikipedia.org/wiki/PageRank, retrieved on 25-Sep-2014

$$f(x) = \frac{\#\, number\ of\ elements\ with\ meta - information\ (title, subject, description)}{\#\ of\ total\ terms\ in\ the\ dataset}$$

Fig. 21, Function for Relevant terms within meta-information metric.

The mentioned attributes, title, description and subject are well known and highly used by the current datasets and are provided by the DCTerm Ontology. These terms are defined as follows:

1. http://purl.org/dc/terms/title
2. http://purl.org/dc/terms/description
3. http://purl.org/dc/terms/subject

The algorithm that computes the value of the dataset for the given metric is defined as follows:

```
LET titles, description, subjects, terms ϵ List<String> and counterTerms ϵ Integer
LET triple ϵ RDFTriple
INIT titles, description, subjects, terms = new empty List
INIT counter, counterTerms = 0
COMPUTE (triple) – Repeat this for every quad in the dataset
DO
    counterTerms++
    IF (triple.predicate is not null) THEN
        IF (triple.predicate = "http://purl.org/dc/terms/title") THEN
            ADD (titles, triple.subject)
            IF (NOT CONTAINED (terms, triple.subject)) THEN
                ADD (terms, triple.subject)
            ENDIF
        ELSE IF (triple.predicate = "http://purl.org/dc/terms/description") THEN
            ADD (descriptions, triple.subject)
            IF (NOT CONTAINED (terms, triple.subject)) THEN
                ADD (terms, triple.subject)
            ENDIF
        ELSE IF (triple.predicate = "http://purl.org/dc/terms/subject")
            ADD (subjects, triple.subject)
            IF (NOT CONTAINED (terms, triple.subject)) THEN
```

```
                    ADD (terms, triple.subject)
                ENDIF
            ENDIF
        ENDIF
    END COMPUTE
```

Fig. 22, Algorithm to compute the relevant terms within meta-information metric.

At the end the metric computes the value based on the attributes found and applies the function defined in the Fig21, as it is defined in the next algorithm:

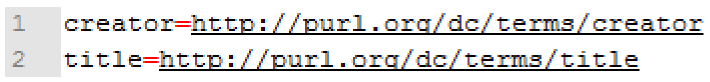
```
LET counter ϵ Integer
INIT counter = 0
METRICVALUE ()
    WHILE (HASNEXT (terms))
    DO
        term = NEXT(terms)
        IF (CONTAINED (titles, term) AND CONTAINED (descriptions, term) AND
                CONTAINED (subjects, term)) THEN
            counter++
        ENDIF
    ENDWHILE
    RETURN (counter/ counterTerms)
ENDMETRICVALUE
```

Fig. 23, Algorithm to get the Relevant terms within meta-information metric.

3.2.2.2 Coverage

This metric measures the coverage and the level of detail of a dataset. This is performed with the aim to fulfill the requirement that the level of detail should be considerable for certain task, for example in a medical application the detail level should be high, meanwhile in an entertainment application the level of detail can be low.

Then the idea firstly is to create a master file that provides the attributes that should be verified in every dataset. With this purpose in mind the project created a file called coverage.properties. In this file should be saved the URIs of the attributes that the system verifies. In the Fig. 24 is visualized the format of the file:



```
creator=http://purl.org/dc/terms/creator
title=http://purl.org/dc/terms/title
```

Fig. 24, Example of the coverage.properties file.

The metric called Coverage.java located within the package de.unibonn.iai.eis.diachron.qualitymetrics.contextual.relevancy implements this solution. Firstly it loads all the attributes from the file, and for every attribute it creates a hashmap list[20]. The next algorithm specifies how the computation method from the class works:

```
LET propertiesLists ϵ HashMap<String, List<String>>
LET counterTerms ϵ Integer
LET terms ϵ List<String>
LET triple ϵ RDFTriple
LET properties ϵ SET<String> -- Set of properties specified in the file
INIT counter, counterTerms = 0
INITMETRIC ()
DO
    properties = LOADPROPERTIESFROMFILE()
    WHILE HAS NEXT (properties) DO
            property = NEXT(properties)
            ADD (propertiesList, property, NEW List<String>)
    ENDWHILE
ENDINITMETRIC

COMPUTE (triple) – Repeat this for every quad in the dataset
DO
    counterTerms++
    IF (triple.predicate is not null) THEN
```

[20] Decision was to use Hashmap instead List for the time and space that is needed to keep multiple lists on memory and therefore get its values

```
            WHILE HAS NEXT (properties) DO
                property = NEXT(properties)
                IF (triple.predicate == property) THEN
                    Aux = GETLIST (propertiesList, property)
                    IF (NOT CONTAINED (Aux, triple.subject))
                        ADD (Aux, triple.subject)
                    ENDIF
                    IF (NOT CONTAINED (terms, triple.subject))
                        ADD (terms, triple.subject)
                    ENIF
                ENDIF
            ENDWHILE
        ENDIF
    END COMPUTE
```

After computing the values with the algorithm, it is time to calculate the value of the metric, the function *f* over the dataset x is given by the formula:

$$f(x) = \frac{\#\ of\ terms\ that\ contain\ all\ the\ attributes\ defined\ in\ the\ properties\ file}{\#\ of\ terms\ of\ the\ dataset}$$

Fig. 26, Function for Coverage metric

Using this function the class calculates the value of the metric by using the list of terms. It goes through all the data storage in the terms list, and checks if every of these terms is created in every of the list of attributes, if the term is contained in all the lists then it increases the counter of terms that is used for the final computation.

3.3 Stream Processor

Once that all the metrics for the mentioned dimensions are developed, then they should be applied to the dataset. This solves one part of the problem regarding the creation of the metrics. The other part of the problem regarding the application of these metrics to some datasets must be considered.

One of the issues of working with online datasets relies in how to stream all the data from the dataset. As it was mentioned earlier nowadays the datasets contain millions of data and the intention of downloading all the information at once can be really difficult, even more given that the same dataset probably has some limitations that did not allow the consumer to consult more that certain amount of information at a time.

In the Website http://sparqles.okfn.org/ it is deployed a tool designed to monitor all the SPARQL endpoints that are registered in Datahub.IO[21], in which it is determined that the most common result-size threshold is to retrieve around 10.000 triples at a time. Taking this value as an example the project developed in this thesis must be able to retrieve information at an equal level, by 10.000 triples at a time.

Knowing this information we should create only a SPARQL query that retrieves all the information. The SPARQL query is a really simple one, and can be processed by every SPARQL technology, as follows:

```
SELECT DISTINCT *
{ ?s ?p ?o}
```

Where the main purpose is to retrieve all the triples from the dataset only making the distinction that every triple should be unique (avoid repetitions).

Since the average of information that must be retrieved is 10.000 triples at a time, this requires the completion of the query using the sentences of LIMIT[22] and OFFSET[23]. By increasing the OFFSET in every iteration of the algorithm by 10.000 and having the LIMIT always fixed to 10.000 the result is shown in the next algorithm:

```
LET iteration = 0 ϵ INTEGER
LET Triples ϵ SET OF TRIPLES
WHILE (NOT REACH THE LIMIT)
DO
    Iteration ++
    Triples = “SELECT DISTINCT *
              { ?s ?p ?o}
```

[21] http://datahub.io/

[22] http://www.w3.org/TR/rdf-sparql-query/#modResultLimit

[23] http://www.w3.org/TR/rdf-sparql-query/#modOffset

```
                LIMIT 10000
                OFFSET Iteration*10000”
        PROCESSTRIPLES (Triples)
ENDWHILE
```

Fig. 27, Sequential Producer algorithm.

At this point the algorithm just have to make a certain number of iterations until it reaches the limit, and for every set achieved it should pass to another method that is in charge to use this information and process all the triples. This process is sequential and if the processing of all the triples is high it will take a fair amount of time to call again to the server and so on.

The algorithm does not take into account the time that takes to connect to a dataset and download the data. For that reason the problem now is how to improve the time of processing all the information, in other words how to improve the processing time?

This problem has already been resolved by the well-known paradigm of producer-consumer problem[24], which creates a solution for a multi-process synchronization. But what does it mean multi-process synchronization?, multi-process synchronization creates different threads that work in parallel, one of these should be the producer of information, while the remaining ones should be the consumers of the information produced by the producer.

In this sense, meanwhile the producer is consulting the dataset, at the same time the consumer is using the information published by the producer in the last iteration. Both processes also are alive while the information is available in the system.

The algorithm of the producer now change a little bit, as shown next:

```
WHILE (NOT REACH THE LIMIT)
DO
        Iteration ++
        Triples = “SELECT DISTINCT *
             { ?s ?p ?o}
      LIMIT 10000
```

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

```
                OFFSET Iteration*10000”
                 PUBLISH (Triples)
          ENDWHILE
```

Now the triples are published within a synchronization mechanism and the client is listening and when there are some information available it consumes this information. This solution improves the time for one main reason, the time that takes the producer to retrieve the information from the Web is used by the consumer to process the data.

With this information the project creates its own producer and consumer. This problem is solved by the classes Producer.java, Consumer.java and StreamManager.java, that are located into the package de.unibonn.iai.eis.diachron.io.streamprocessor.

The process is described by the next sequential diagram:

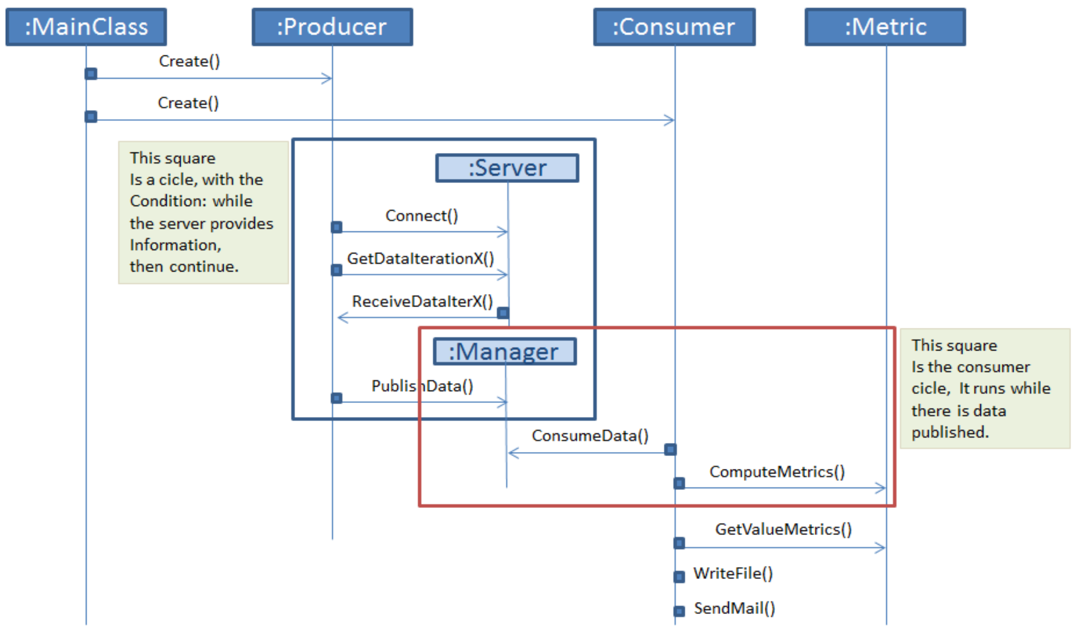
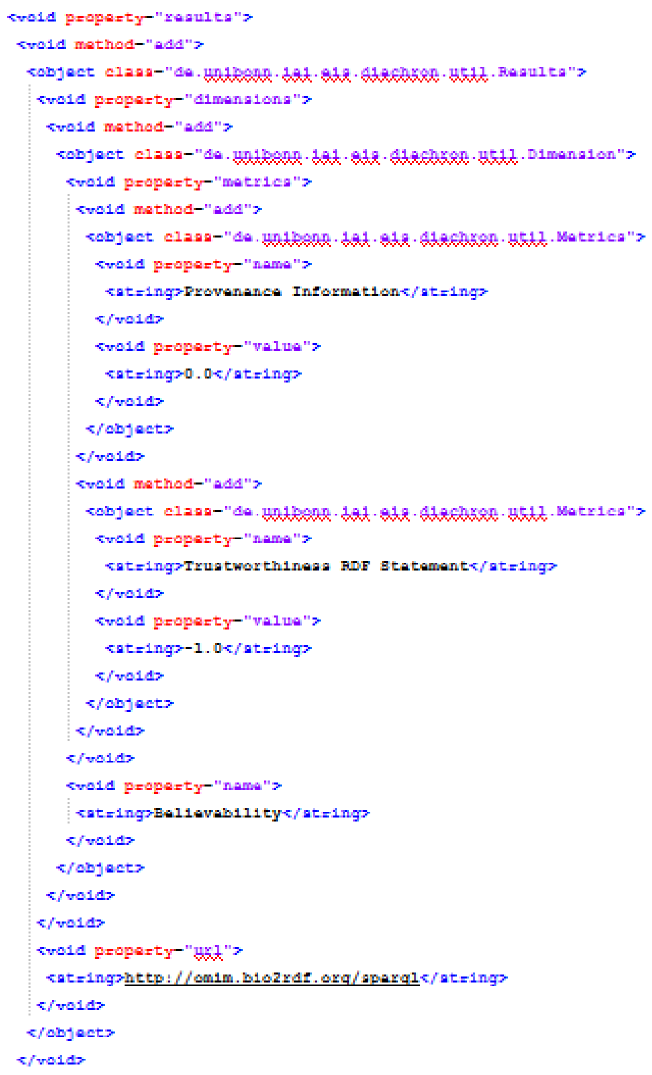


Fig. 28, Consumer-Producer sequential diagram.

The class Producer.java is in charge to create the invocations to the dataset and to keep in mind the number of iterations. The StreamManager.java class is in charge to manage all the information that is communicated between producer and consumer, this means that it provides some methods to put and to get the information that is being published. The last class that solves the problem is the consumer class, in the project this class consumes all the information that is being published and instantiated all the metrics that are going to be evaluated, then for all the triples that it found, it computes all of the metrics. When this class detects that the producer stopped publishing data, it writes into a local file the result of the processing of the dataset. This file is defined by the admin of the system into the file config.properties in the attribute “dataBase”. The file is located within the folder src/main/java/.

This file is being used by the UI to show the information obtained during the processing for every dataset. The format of such file is shown in the next example:



```
<void property="results">
 <void method="add">
  <object class="de.unibonn.iai.eis.diachron.util.Results">
   <void property="dimensions">
    <void method="add">
     <object class="de.unibonn.iai.eis.diachron.util.Dimension">
      <void property="metrics">
       <void method="add">
        <object class="de.unibonn.iai.eis.diachron.util.Metrics">
         <void property="name">
          <string>Provenance Information</string>
         </void>
         <void property="value">
          <string>0.0</string>
         </void>
        </object>
       </void>
       <void method="add">
        <object class="de.unibonn.iai.eis.diachron.util.Metrics">
         <void property="name">
          <string>Trustworthiness RDF Statement</string>
         </void>
         <void property="value">
          <string>-1.0</string>
         </void>
        </object>
       </void>
      </void>
      <void property="name">
       <string>Believability</string>
      </void>
     </object>
    </void>
   </void>
   <void property="url">
    <string>http://omim.bio2rdf.org/sparql</string>
   </void>
  </object>
 </void>
```

Fig. 29, Results file example.

In the local file the information of the dataset is stored, along with the dimensions that were evaluated and the metrics for those dimension with its values.

3.4 User Interface

The main task of this module is to manage the User Interface. The interface is a Web environment oriented, for that reason it runs over JBoss server. As a base it uses the J2EE technologies of Java Beans to run the application. This module corresponds to the layer VIEW on the MVC pattern.

The User Interface is in charge to promote and provide a friendly interaction with the user and the system. As a foundation it uses the definition of Java Server Pages which interacts with the sever, and produces the final view to the consumer. It also uses the framework define by RichFaces[25] that provides tools to make maintainable and easy to read sources.

The interface is compound by two folders, the classes that are located into the forlder src/main/interface and the folder WebContent. The first one is owner of the backing beans that are in charge to manage the interaction of the user with the server. And the second folder includes all the resources needed by JBoss to deploy the application.

Into the folder WebContent\WEB-INF\ are located the configuration files for the interface. A very relevant file is the one called Web.xml since it holds the configuration of all the behaviors that should have the application on the server. Another very relevant file is the one called faces-config.xml, in which are configured all the resources the server needs to identify the resources and to manage the internal behavior of the application.

Every screen that is displayed in the Web are those that end with the format .xhtml, this gives to the server the instruction to process before showing it to the consumer. Within the folder WebContent\tabs\ are located the behavior of the main interface that is shown to the final user.

To run the interface the user should have installed JBoss server to be able to run the application. If the reader wishes to learn specific directions on how to deploy and how to use the interface, is advised to refer to the Installation and User manual provided in the Appendix as attachment to this master thesis.

[25] http://richfaces.jboss.org/

4. RESULTS

4.1 Testing

In order to assure that the results of the development of the quality metrics system is accurate and reliable, a series of unit tests were designed to verify the system and these were performed using the JUnit framework[26]. Furthermore, a test was designed to check the reliability of the stream processor, such results are explained in the folllowing.

4.1.1 JUnit Test

For each one of the metrics a test was designed able to load information for some exemplified datasets, and given such information retrieve a metric value. In the following table is found a summarized list of the tests applied to the code, how long it took to run a given test, and which data dump was used (the data dumps are provided in the folder src/test/resources/datadumps).

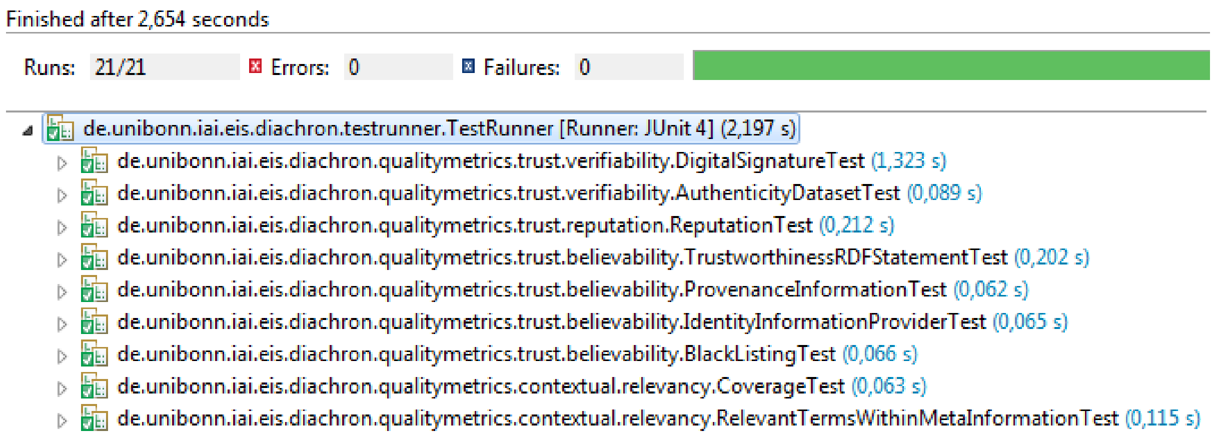
Table 5, JUnit test summary

Metric	Test Name	Running Time (s)	Datadump Used
DigitalSignature	testNeutralCase	0,889	CurrencyDocumentStatements CM2
	testPositiveCase	0,29	CurrencyDocumentStatements CM
	testNegativeCase	0,101	CurrencyDocumentStatements
AuthenticityDataset	testPositiveCase	0,047	DuplicateInstance
	testNegativeCase	0,031	CurrencyDocumentStatements
Reputation	testMiddleCase	0,084	CurrencyDocumentStatements CM
	testPositiveCase	0,06	CurrencyDocumentStatements
	testNegativeCase	0,059	DuplicateInstance
TrustwothinessRDFStatements	testNeutralCase	0,066	CurrencyDocumentStatements CM
	testPositiveCase	0,067	CurrencyDocumentStatements
	testNegativeCase	0,052	DuplicateInstance
ProvenanceInformation	testPositiveCase	0,029	CurrencyDocumentStatements
	testNegativeCase	0,03	DuplicateInstance
IdentityInformationProvider	testPositiveCase	0,028	ResearchGroup
	testNegativeCase	0,031	CurrencyDocumentStatements
BlackListing	testPositiveCase	0,032	CurrencyDocumentStatements

[26] More documentation about the framework can be found on this link http://junit.org/

	testNegativeCase	0,027	ResearchGroup
Coverage	testPositiveCase	0,033	CurrencyDocumentStatements
	testNegativeCase	0,027	DuplicateInstance
RelevantTermsWithinMetaInformation	testPositiveCase	0,052	CurrencyDocumentStatements
	testNegativeCase	0,056	CurrencyDocumentStatements CM

A total of 21 tests were ran in a total time of 2.654 seconds. The image below shows the visual tool that provides eclipse. All the tests were successful with neither Errors nor Failures:



```
Finished after 2,654 seconds
Runs: 21/21    Errors: 0    Failures: 0

de.unibonn.iai.eis.diachron.testrunner.TestRunner [Runner: JUnit 4] (2,197 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability.DigitalSignatureTest (1,323 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.verifiability.AuthenticityDatasetTest (0,089 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.reputation.ReputationTest (0,212 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability.TrustworthinessRDFStatementTest (0,202 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability.ProvenanceInformationTest (0,062 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability.IdentityInformationProviderTest (0,065 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.trust.believability.BlackListingTest (0,066 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.contextual.relevancy.CoverageTest (0,063 s)
  de.unibonn.iai.eis.diachron.qualitymetrics.contextual.relevancy.RelevantTermsWithinMetaInformationTest (0,115 s)
```

Fig. 30, Result of the tests in eclipse.

All the data dumps were previously loaded with information reliable to assure that the tests were able to measure the correct attributes in every case. The tests that have only two cases are those where the function is defined to return only 1(trust) or 0 (no trust). For those metrics were created two tests, one for the positive case, meaning that the dataset is trustful by the measure of that metric, and a second one, and as in comparison, for the negative case where the metric could not find the condition and for that reason return a no trust in that data set.

The next case is when the metric could return more than one value, usually a value between a range, e.g. the TrustworthinessRDFStatementTest metric value can be situated between in the range [-1,1]. Another case of the value of the metric being in a range is when the function returns a percentage, e.g. the metrics Coverage and RelevantTermsWithinMetaInformation, where the range varies between [0,1].

For those cases the tests were designed to produce three results, one result with positive data, the second result with information that can be trustful or not (usually was in the middle of the function), and a third test designed to test the negative case where the dataset cannot be trusted.

All the computation ran over the datadump “CurrencyDocumentStatements” which contains 4.169 triples. The higher time to compute all the triples is 0.889 sec. in the test of the class DigitaSignature for the neutral case. We can conclude that the computation is efficient and then we can continue with the next testing unit.

4.1.2 Performance Test

This test was designed to test the streaming processor and to check if it really improves the streaming time. To accomplish this task during the development of the project was created also a sequential streaming processor with the aim of compare both cases.

The first scenery in which the test is only using the system locally, relied on the assumption that the problem of data connectivity trough the net could be an overload for the system, then locally the project created a local RDF endpoint[27]. The datadump loaded contains 2’100.000 triples[28]. The results of running this process locally are summarized in the next table:

Table 6, Tests results from the local configuration.

local service http://localhost:8081/openrdf-sesame/repositories/test					
Test No.	**No. of triples**	**Sequential Streaming**	**Average per triple**	**Consumer-Producer Stream.**	**Average per triple**
1	500000	41,031	0,000082062	43,249	0,000086498
2	500000	41,198	0,000082396	47,831	0,000095662
3	500000	40,204	0,000080408	43,199	0,000086398
4	500000	40,402	0,000080804	45,855	0,00009171
5	500000	47,72	0,00009544	54,698	0,000109396
6	500000	55,254	0,000110508	64,403	0,000128806
			0,000088603		**0,000099745**

[27] http://openrdf.callimachus.net/

[28] The data was taking from https://developers.google. com/freebase/data

In all possible scenarios tested the sequential streaming was faster, this can be observed by the average of time to pass a triple through all the developed metrics where in the sequential streaming the average time was $8.8\,e^{-5}$ seconds, meanwhile the consumer-producer streaming was $9.9\,e^{-5}$ seconds. Clearly the average time of processing a triple is better in the sequential streaming.

Nonetheless this result led us to the second scenario where the goal was to test against real data, which immediately increased the time of information retrieval of the dataset, because every package of information should travel from the server to the local machine. For this reason the project tested the two streaming implementations against two SPARQL endpoint, http://genage.bio2rdf.org/sparql, and http://beta.sparql.uniprot.org/.

The results of this test is summarized in the next table:

Table 7, Test results over a remote SPARQL endpoint.

http://omim.bio2rdf.org/sparql					
Test No.	**No. of triples**	**Sequential Streaming**	**Average per triple**	**Consumer-Producer Streaming**	**Average per triple**
1	100000	13,268	0,00013268	6,789	0,00006789
2	100000	12,727	0,00012727	7,96	0,0000796
3	200000	25,327	0,000126635	13,756	0,00006878
4	200000	24,198	0,00012099	13,555	0,000067775
5	500000	64,106	0,000128212	38,055	0,00007611
6	500000	62,16	0,00012432	40,069	0,000080138
7	500000	63,132	0,000126264	37,552	0,000075104
8	1000000	140,339	0,000140339	82,729	0,000082729
9	1000000	141,639	0,000141639	83,268	0,000083268
10	1000000	141,753	0,000141753	84,151	0,000084151
			0,00013101		**0,0000765545**

In the first case, clearly the result shows that the consumer-producer implementation is better in the time cause it only took to process a triple $7.6\,e^{-5}$ seconds, whilst in contrast the sequential streaming decreased considerably and now the average time is $1.3\,e^{-4}$ seconds. At this point we have checked that the noise that the network adds to the sequential processor makes that the better solution to stream the data for the system be the consumer-producer streaming.

However to be sure that the noise added by the net represents a much better than the streaming solution another round of tests were ran over the second dataset, the results are summarized next:

Table 8, Test results over a second remote SPARQL endpoint

http://beta.sparql.uniprot.org/					
Test No.	**No. of triples**	**Sequential Streaming**	**Average per triple**	**Consumer-Producer Streaming**	**Average per triple**
1	100000	13,684	0,00013684	7,222	0,00007222
2	100000	14,472	0,00014472	7,303	0,00007303
3	200000	21,155	0,000105775	13,594	0,00006797
4	200000	21,101	0,000105505	14,89	0,00007445
5	500000	57,969	0,000115938	37,112	0,000074224
6	500000	54,884	0,000109768	33,531	0,000067062
7	500000	55,908	0,000111816	37,555	0,00007511
8	1000000	126,104	0,000126104	68,208	0,000068208
9	1000000	127,722	0,000127722	65,66	0,00006566
10	1000000	125,352	0,000125352	66,702	0,000066702
			0,000120954		**7,04636E-05**

In the second remote SPARQL endpoint, it is clear that the time still remains smaller in the consumer-producer solution, in this case the average for this solution is $7.04\ e^{-5}$ seconds, and the time that takes for the sequential processing is $1.2\ e^{-5}$ seconds.

In both cases the time is almost half of the time that takes to process a triple, so based on these results is possible to conclude that the consumer-producer is better to retrieve data from a remote SPARQL endpoint.

4.2 Data Sets Evaluation

To check the results of the metrics over real datasets, two were selected from the geographic domain. The next table summarizes this evaluation. Over every dataset all the metrics were streamed using the consumer-producer streaming which just previously was clarified to be the best option to stream a big amount of information.

The datasets selected to be evaluated are: http://geo.linkeddata.es/sparql and http://resource.geolba.ac.at/PoolParty/sparql/GeologicUnit.

The table shows all the information retrieved from the processing of the two datasets selected, and shows furthermore the value retrieved by the computation of the metric for those datasets, at the end it is found the summary of how long it took to process the complete dataset, and how many triples the system retrieved and evaluate from the remote source.

Remote SPARQL endpoint 1 = http://geo.linkeddata.es/sparql

Remote SPARQL endpoint 2 = http://resource.geolba.ac.at/PoolParty/sparql/GeologicUnit

Table 9, Evaluation of two datasets

	Remote SPARQL endpoint 1	**Remote SPARQL endpoint 2**
DigitalSignature	0	0
AuthenticityDataset	0	1
Reputation	0	0
TrustworthinessRDFStatements	-1	0.25
ProvenanceInformation	0	1
IdentityInformationProvider	0	1
BlackListing	0.5	1
Coverage	0	4.92 e-4
RelevantTermsWithinMetaInformation	0	2.49 e-4
Time to Stream all the data set	07h 28m 33s	00h 01m 2,706 s
Number of triples evaluated	13’100.000	16.060

Whit these results we can check the importance to store information about the provenance. In both cases the difference between numbers of triples is of millions, but the second SPARQL endpoint implements the ontologies that are defined to show information related with the provenance of the resources.

The results can be read as follows, the second SPARQL endpoint is trustful because it has information related to the author of the dataset, also stores information related with the provenance of the resources. The 4.92 e-4 percent of elements contain the terms with meta-information defined in the coverage.properties (title and creator attributes). Moreover 2.49 e-4 percent of elements are stored with relevant meta-information (title, description and subject).

The second SPARQL endpoint can also be trusted because its metric Trustworthiness of RDF statements is in 0.25, which indicates that the dataset provides information relevant for the terms and relationships that it stores.

In comparison the first SPARQL endpoint does not provide any of the information related with provenance, following the definition given by Zaveri et al., [2012], in which provenance refers to the “degree to which the information is accepted to be correct, true, real and credible“. We can then made the conclusion that the data should not be trusted because the consumer is not able to check any of the sources, provenance, author, etc. of the dataset.

The only way to trust the first SPARQL endpoint is through the constructions of the ranking by expert users, then if they include the endpoint in the list of trusted providers the metric of Identity Information Provider retrieves based on the experience that the dataset should be trusted.

Since the first SPARQL did not retrieve any information we concluded that one possible reason is because the endpoint did not use the ontologies FOAF or DCterms, for that reason was necessary a research through the datahub.io, aiming to look for datasets containing information related with its provenance, once it was founf it (a new SPARQL endpoint), the process consisted of running all the metrics over it. The results of this process is summarized in the next table.

Remote SPARQL endpoint 3 = http://ndc.bio2rdf.org/sparql

Table 10, Results over the third dataset

	Remote SPARQL endpoint 3
DigitalSignature	0
AuthenticityDataset	1
Reputation	0
TrustworthinessRDFStatements	-0.5
ProvenanceInformation	1
IdentityInformationProvider	0
BlackListing	0.5
Coverage	0
RelevantTermsWithinMetaInformation	0
Time to Stream all the data set	02h 45m 15s
Number of triples evaluated	6’120.562

From these results it was possible to reach the conclusion that the third SPARQL endpoint can be reliable in terms of the verifiability of the dataset. It has information of the author or contributor, this can be verified because the authenticity of the dataset metric retrieves “1”. In terms of believability the dataset is not fully trusted because it contains some metadata information that can be used to describe its resources, but not all the necessary to be trusted without hesitation.

In comparison, in terms of the relevancy, the dataset is not relevant, this can be observed because in the retrieved values by the coverage function and Relevant terms within meta-information metrics it retrieves 0 in both cases, meaning that the dataset is not useful when the information needs to be accurate or complete, its terms do not contain the list of minimum elements defined by the system to be relevant.

5. LIMITATION AND STRENGHTS

As in any type of research and development of new software various limitations were observed. Even though the process produced some interesting insights and findings when the metrics were applied to the datasets and when the process of streaming the information contained on them was executed.

The biggest limitation observed is regarding to the ontologies that were used into the current datasets. For example only the 35.77%[29] of the current datasets use some kind of provenance vocabulary, this restriction makes that the remaining 64.33 % of datasets could not have been measured by the system. Even more restricted is the use for the solutions that were developed in the system, because the project used the DC Terms ontology, and only the 28.37% of datasets use this ontology. Nevertheless one of the interesting findings of the project is that even when the dataset uses an ontology, this does not imply that the dataset uses all the set of properties defined.

Another relevant limitation was observed in relation to the datasets publishing its own ontology, and since every day new ones are proposed, these new ones do not apply any of the well-known ontologies. In this direction, it was established that the tools that intent to generalize the measure, are always restricted to the ontologies that are used to create the system.

In terms of the capability to measure all the information from one dataset another limitation was detected. Nowadays the information that has been published is about of millions of triples, and in some of them even billions of triples (DBpedia is in the order of 3 billion of pieces of information[30]), the problem then consists of that retrieving all the information should be done by batches. This approximation requires that the server should not block the request made by the server where the service is running. Within this project there are servers able to detect too many request and block petitions, making then impossible to retrieve all the data and therefore to be measured. When the remote servers allow this kind of recurrent calls, the faced problem is how to transport the information over internet and then manage the procedure on the local machine where the system is running, this

[29] Information retrieve on 27-Ago-2014, from http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

[30] Information retrieved on 27-Ago-2014, from http://dbpedia.org/About

demands that the connection to the Web should be reliable and the configuration of the local server should have enough storage to process all the information.

In relation to the perceived strengths of the project, one refers to the system being able to perform the streaming of information managed by the sized of the batch that is retrieved by the system, this with the aim to check which batch size should be optimal to retrieve the information efficiently.

A strength of the system is that there are some quality metrics that do not depend on the ontologies, and are based on the subjectivity that empowers the users to describe what should be trusted and what not. Though these depend on the configuration of the administrator, one example of this is the metric called Relevancy, that takes into account the experience of the expert users, and saved their opinion into a file of trusted publishers, then the system can be easily extensible to the beliefs of the consumer, and the consumer can by him/herself decide about what information should be trusted and what not.

The metric Coverage represents as well another of the strengths of the system, this because the fact that the attributes that the system should look into the dataset can be configured, and this parametrization allows to adapt the system to every goal that should be accomplished, meaning that the system is able to ask for many attributes of any ontology as the consumer wants, and based on this evaluate the dataset and check if it can be reliable or not.

6. CONCLUSIONS

The aim of this master thesis consisted of creating a tool capable of measure automatically the quality dimensions of trustworthiness and relevancy, defined into those dimensions are included both subjective and objective metrics.

With the objective metrics a relevant problem was faced since almost all of them look for certain attributes that the dataset should contain. In our case almost all of the metrics depended on the specification of some attributes that were attached to the ontologies that were found and used during the development of the master thesis. Bearing in mind an important disadvantage previously mentioned in the last chapter, regarding to not all the datasets that have been published using the same ontologies, in many cases they defined new ones, or simply they did not use ontologies at all.

Given that the subjective metrics are easily parameterized by the consumer, but as its name implies are subjective as “existing in the mind; belonging to the thinking subject rather than to the object of thought”[31], meaning that at the end every human being is owner of its own beliefs and disbeliefs on what is trustful or what not. This represents a huge risk when we talk about sensible information, as mentioned in this document not necessarily when talking of topics of culture or entertainment, but especially when the information relates to sensible topics such as medicine and therefore human beings lives.

The goal is then to create instances that allow the control of dataset publications with the aim of decreasing the gap between differences into the ontologies, and propagate the mechanism of unification of sources. For example increasing the number of datasets that contain or use information related with its provenance (now set only at 35.77%)[32].

The final idea then is to propagate such a tools like the daQ model that allows the consumers to have an impartial way to measure the quality of the datasets evaluating the quality dimensions of its concerns.

[31] http://dictionary.reference.com/browse/subjective

[32] http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/, retrieved on 27-Ago-2014

REFERENCES

Beckett, D. (2008). *SPARQL Query Results XML Format*, D. Beckett, Editor, W3C Recommendation, 15 January 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-XMLres-20080115/ . Latest version available in http://www.w3.org/TR/rdf-sparql-XMLres/.

Bizer, C. (2006). Semantic Web Publishing Vocabulary (SWP), User Manual, November 2006. Availabel at http://wifo5-03.informatik.uni-mannheim.de/bizer/wiqa/swp/SWP-UserManual.pdf, retrieved 20-Jul-2014.

Bizer, C. (2007). *Quality-Driven Information Filtering in the Context of Web-Based Information Systems*. PhD thesis, Freie Universität Berlin, March 2007.

Bizer, C., Cyganiak, R. & Heath, T. (2007). *How to Publish Linked Data on the Web*. Available at http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/, retrieved 19-Ago-2014.

Bizer, C., Heath, T. & Berners-Lee, T. (2009). Linked Data – The Story so far. *International Journal on Semantic Web and Information Systems, Special Issue on Linked Data*.

Bonatti, P., Hogan, A., Polleres, A., and Sauro, L. (2011). Robust and scalable linked data reasoning incorporating provenance and trust annotations. *Journal of Web Semantics, 9(2):165 – 201*.

Clark, K. *(2008). SPARQL Protocol for RDF*, W3C Recommendation, 15 January 2008, http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/. Latest version available in http://www.w3.org/TR/rdf-sparql-protocol/.

Cyganiak, R., Stenzhorn, H., Delbru, R. and Tummarello, G. (2008). *Semantics Sitemaps: Efficient and Flexible Access to Datasets on the Semantic Web*. Volume 5021, 2008, pp 690-704

Dürst, M., Suignard, M. (2005), *Internationalized Resource Identifiers (IRIs)*. January 2005. RFC. URL: http://www.ietf.org/rfc/rfc3987.txt

Flemming, A. (2010). *Quality characteristics of linked data publishing datasources*. http://sourceforge.net/apps/mediawiki/trdf/index.php?title=Quality_Criteria_for_Linked_Data_sources.

Gamble, M. and Goble, C. (June 2011). *Quality, trust, and utility of scientific data on the web: Towards a joint model*. In ACM WebSci, pages 1–8.

Gil, Y. and Artz, D. (2007). *Towards content trust of web resources*. Web Semantics, 5(4):227 – 239.

Golbeck, J., Parsia, B. and Hendler, J. (2003). *Trust networks on the semantic web*. In CIA.

Hartig, O. (2008). Trustworthiness of data on the web. *In STI Berlin and CSW PhD Workshop*, Berlin, Germany.

Hausemblas, M. (2009). Exploiting Linked Data for Building Web Applications. Available at *http://wtlab.um.ac.ir/images/e-library/linked_data/other/exploit-lod-webapps-IEEEIC-preprint.pdf*, retrieved 19-Ago-2014.

Jacobi, I., Kagal, L. and Khandelwal, A. (2011). *Rule-based trust assessment on the semantic web*. In RuleML, pages 227 – 241.

Knight, S.A. and Burn, J. (2005). Developing a framework for assessing information quality on the World Wide Web. *Information Science, 8, pp 159–172*.

Mendes, P., Mühleisen, H. and Bizer, C. (2012). *Sieve: Linked data quality assessment and fusion*. In LWDM.

Naumann, F. (2002). *Quality-Driven Query Answering for Integrated Information Systems*, volume 2261 of Lecture Notes in Computer Science. Springer-Verlag.

Pattanaphanchai, J. (2011). DC Proposal: Evaluation Trustworthiness of Web Content using Semantic Web Technologies. *The Semantic Web-ISWC 2011, Lecture Notes in Computer Science Volume 7032, pp 325-332.*

Pipino, L., Lee, Y. and Wang, R. (2002). Data Quality Assessment. *Communications of the ACM, 45(4).*

Schandl, B. (2009). Representing Linked Data as Virtual File Systems. *WWW2009 Workshop on Linked Data on the Web. Session 1.1*, Madrid, Spain.

Shekarpour, S. and Katebi, S.D. (2010). *Modeling and evaluation of trust with an extension in semantic web*. Web Semantics: Science, Services and Agents on the World Wide Web, 8(1):26 – 36, March 2010.

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*

APPENDIX

APPENDIX

APPENDIX A - INSTALLATION AND USER MANUAL

The aim of this document is to explain to the final user how to install, deploy and modify the contents of the system developed. The document is structure in three sections, the first one explains how to install the project into a development environment, which allows the user to maintain the system or to make new developments. The second section is to explain how to access into the software and how to interact with the system, and the last section provides useful information for the systems administrator, providing the explanation how to deploy the system from the compiled files.

INSTALLATION FOR DEVELOPMENT

With the aim to maintain or to extend the diachron application the user needs 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 software, it is necessary to configure each element of them.

Eclipse

Eclipse is an IDE (integrated development environment); that is the tool that allows to modify, change and compile the java code.

Configuration

To be able to run the project in eclipse the user should import the project into the IDE, the user have two options: a)the source code is attached into the CD of the project or b) the source can be found online in the github repository[33] of the project.

Once it is done you should have something like the next image into the package ecplorer of eclipse:

[33] The source code can be download from: https://github.com/diachron/quality/tree/Carlos

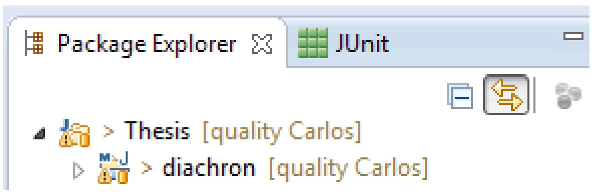
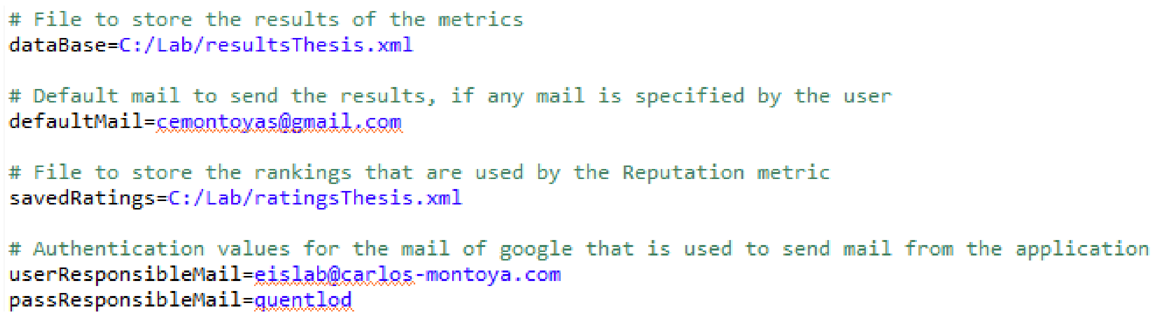


Fig. A - 1, Project Imported into eclipse IDE

Once inside of the project, in the path diachron/src/main/java, the user must locate one file that is call config.properties (see Fig. 2).

With this file located, please open it, it should look like this:



```
# File to store the results of the metrics
dataBase=C:/Lab/resultsThesis.xml

# Default mail to send the results, if any mail is specified by the user
defaultMail=cemontoyas@gmail.com

# File to store the rankings that are used by the Reputation metric
savedRatings=C:/Lab/ratingsThesis.xml

# Authentication values for the mail of google that is used to send mail from the application
userResponsibleMail=eislab@carlos-montoya.com
passResponsibleMail=quentlod
```

Fig. A - 2, config.properties file

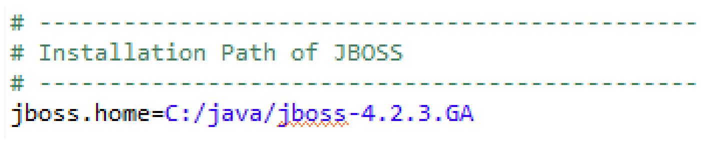
The properties in the file configure some functionalities of the system. The first one is the directory where the program is going to store the results. The second property is the default user mail that the system send the results after running the program (this is because the time to compute takes a lot of time and this is the mechanism to announce that the process is already finished). The third property is defined to be the path were the system is going to look for the rankings file that are used by the Reputation metric. The last two properties configure the access to the mail that is being used by the program to send the resultant mail.

JBoss 4.2.3 GA

The JBoss server is the applications where the system deploys the diachron application.

Configuration

After the JBoss installation the user should configure the ant.properties file that is located in the diachron project, to specify the JBoss installation path.



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

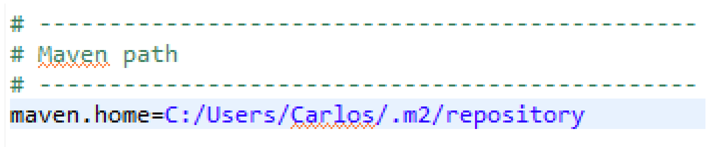
Fig. A - 3, ant.properties - JBoss path

Maven

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

Configuration

The user should configure the ant.properties file that is located into the diachron Project.

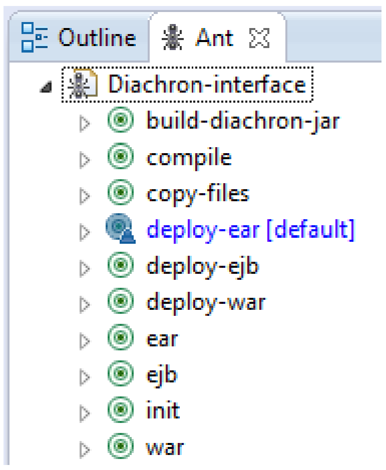


```
# -----------------------------------------------
# Maven path
# -----------------------------------------------
maven.home=C:/Users/Carlos/.m2/repository
```

Fig. A - 4, ant.properties - Maven path

Run the program on the server

Now that everything is installed and configured, the project can be deployed into the JBoss server, to perform this task the user should locate the file that is called build.xml, this file is located in the project diachron. Also your IDE should have installed the tools to run the ant files.



```
Outline  Ant
Diachron-interface
  build-diachron-jar
  compile
  copy-files
  deploy-ear [default]
  deploy-ejb
  deploy-war
  ear
  ejb
  init
  war
```

Fig. A - 5, ant build view

In the ant view the user only must give double-click on the line with the blue mark (deploy-ear), this instruction creates all the necessary files and copy these files to the JBoss server. Also this process creates a copy of all the files into the directory diachron/dist, this is useful if you want to send only the compile files to the servers administrator.

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 access the program in the browser the user needs to open the link http://localhost:8080/diachron/index.jsf.

USER MANUAL

How to Access and use the program.

To access to the web application the user need to copy the following link in the web browser: http://localhost:8080/diachron/index.jsf.

If the JBoss server is running the user should be able to see the next screen.

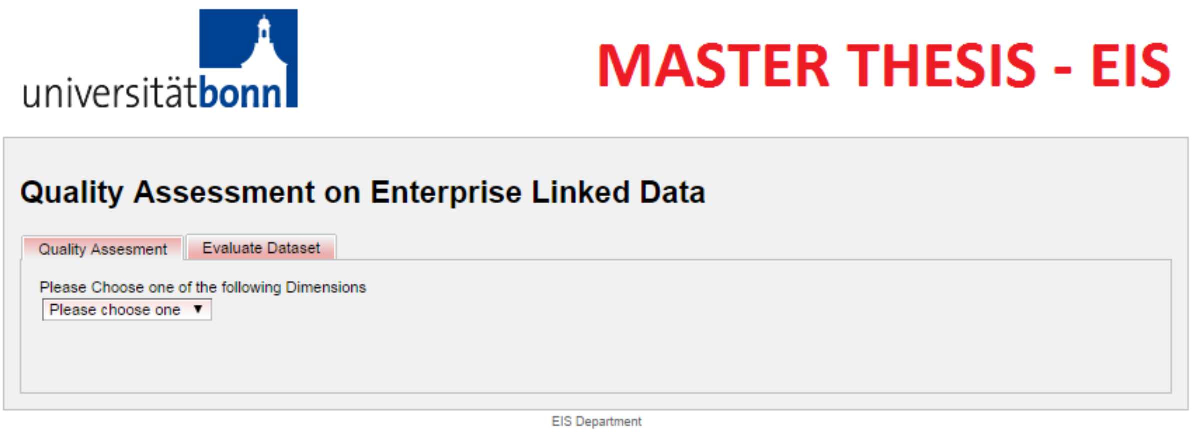


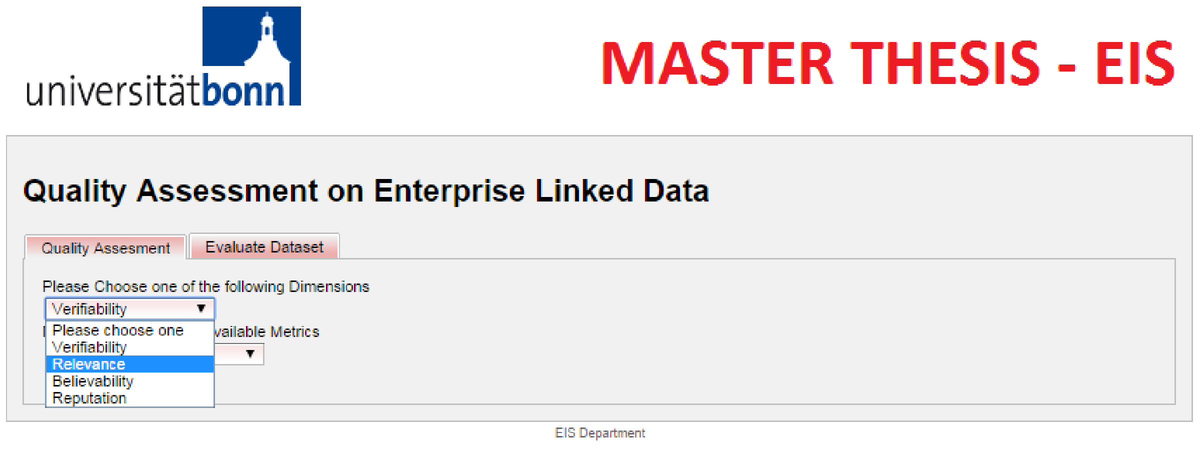
Fig. A - 6, Init screen

There are two different functionalities available in the system:

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.



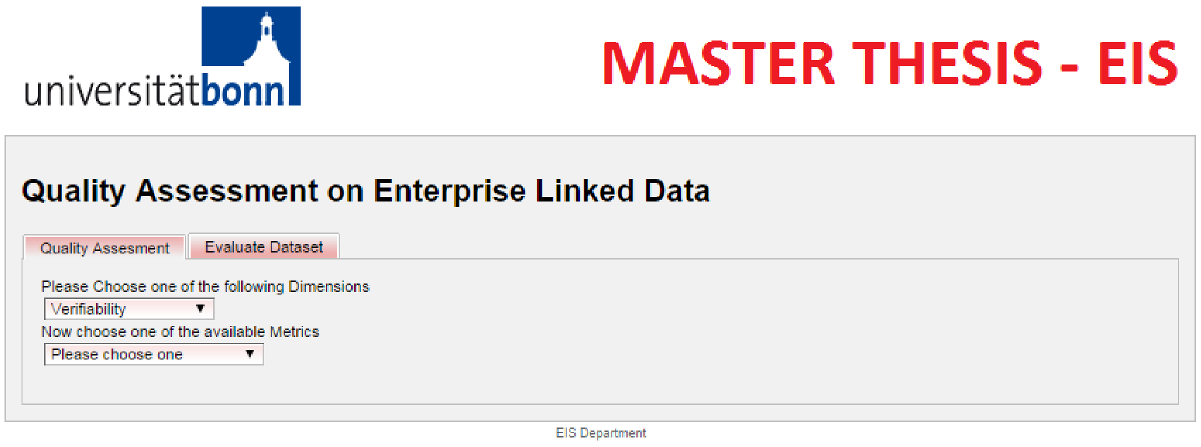
Quality Assessment on Enterprise Linked Data

Please Choose one of the following Dimensions
Verifiability
Please choose one
Verifiability
Relevance
Believability
Reputation
vailable Metrics

EIS Department

Fig. A - 7, 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.



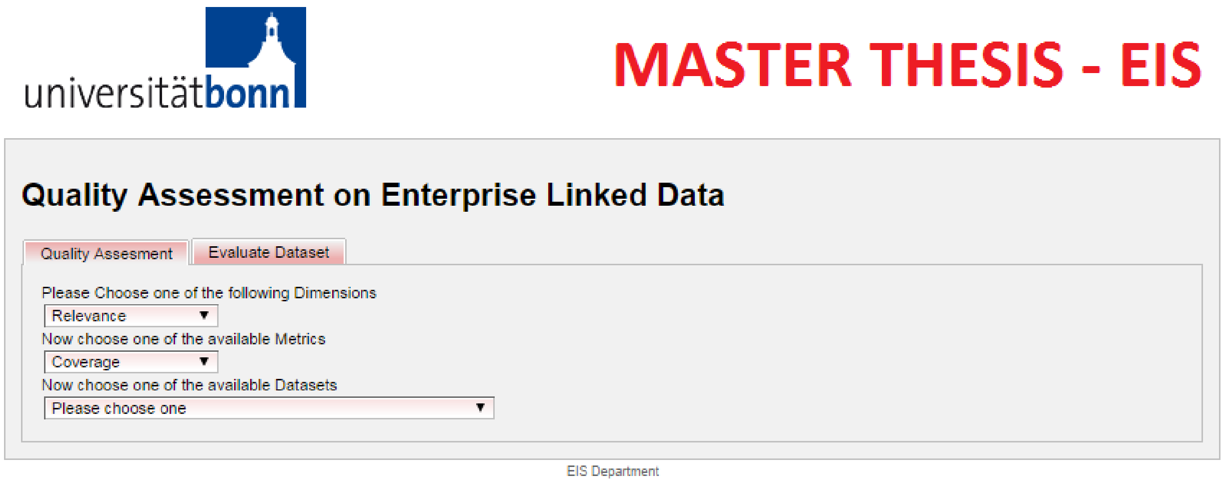
Quality Assessment on Enterprise Linked Data

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

EIS Department

Fig. A - 8, Metrics select combo

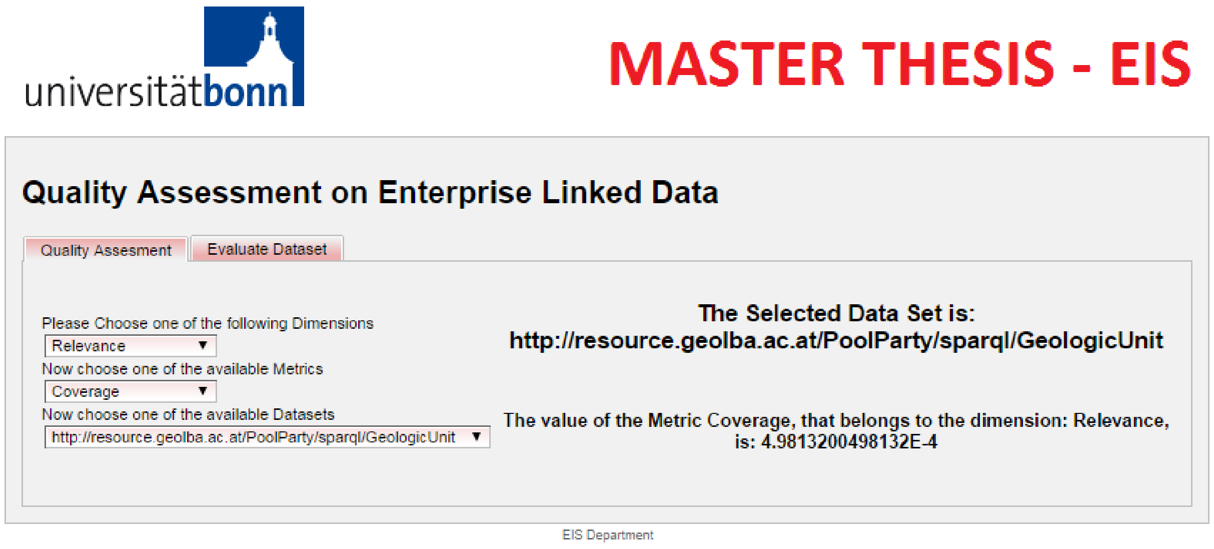
The user should select one of the available dimensions. Once this is done the system display a new combo box, to show the available Data Sets that already had been evaluated.



Please Choose one of the following Dimensions
Relevance
Now choose one of the available Metrics
Coverage
Now choose one of the available Datasets
Please choose one

Fig. A - 9, Available Datasets

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



Please Choose one of the following Dimensions
Relevance
Now choose one of the available Metrics
Coverage
Now choose one of the available Datasets
http://resource.geolba.ac.at/PoolParty/sparql/GeologicUnit

The Selected Data Set is:
http://resource.geolba.ac.at/PoolParty/sparql/GeologicUnit

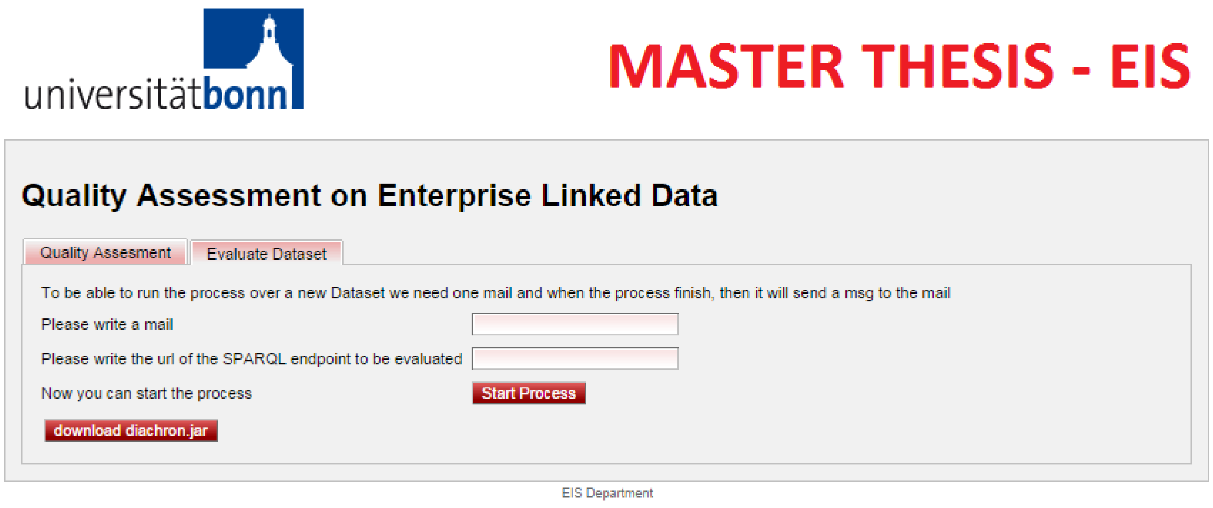
The value of the Metric Coverage, that belongs to the dimension: Relevance, is: 4.9813200498132E-4

Fig. A - 10, Result Image

Evaluate new Dataset

The second tab provide by the system is in charge to evaluate a new dataset or to re-run the process over one of the existing datasets.

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


MASTER THESIS - EIS

Quality Assessment on Enterprise Linked Data

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

Fig. A - 11, Evaluate new dataset

Then the user must complete the information that is asking: user mail and SPARQL endpoint. After this the user must start the process by clicking the "Start Process" button, the systems start the process of evaluate the given dataset. When the process is finish the system sends an automatic mail to the given mail.

Also in this screen the user can download the diachron.jar to work with the libraries if they want to do it. To download the file, the user only has to click on the bottom "download" automatically it will be save it to the computer.

INSTALLATION IN THE SERVER

If you the user is the admin of the server and only wants to deploy the system into the server, the user should do:.

Locate the folder that is called "dist", this folder is located into the CD of the project, after located open the folder, then the system should show something like this:

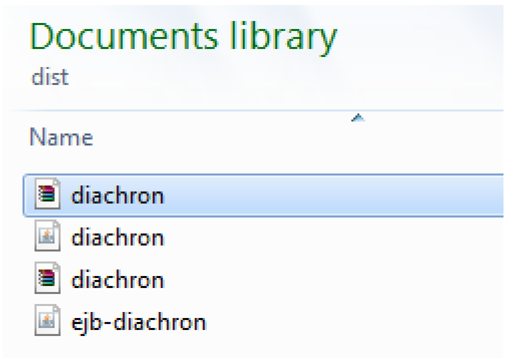
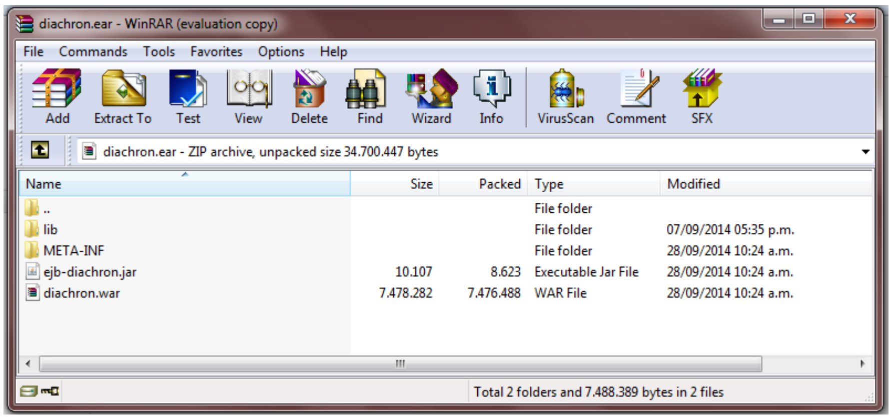


Fig. A - 12, dist folder

From this folder you should copy the file “diachron.ear” and copy into the folder $JBOSS_HOME\server\default\deploy.

Once the file was copied, the admin should open the “diachron.ear” file, to open it the user can use programs like winzip or winrar.

The user should see something similar to the next image:


diachron.ear - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment SFX

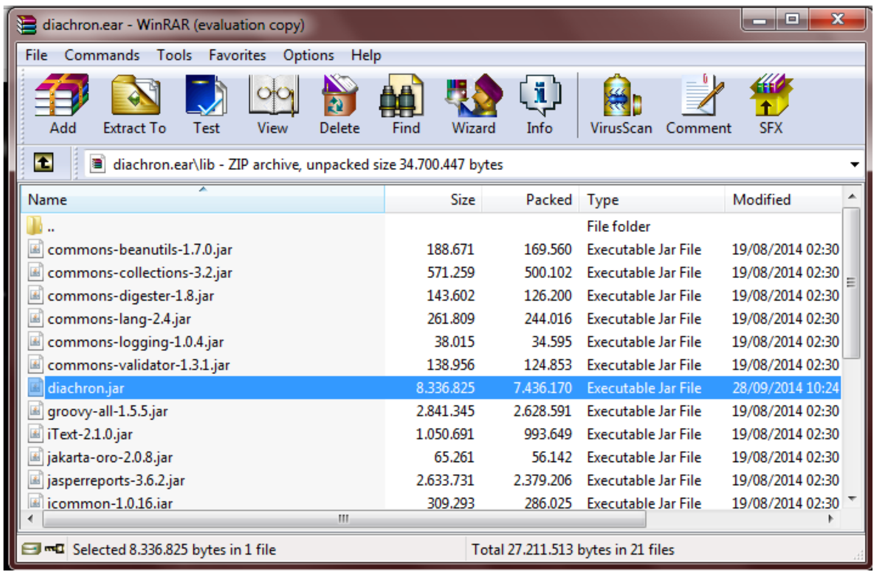
diachron.ear - ZIP archive, unpacked size 34.700.447 bytes

Name	Size	Packed	Type	Modified
..			File folder	
lib			File folder	07/09/2014 05:35 p.m.
META-INF			File folder	28/09/2014 10:24 a.m.
ejb-diachron.jar	10.107	8.623	Executable Jar File	28/09/2014 10:24 a.m.
diachron.war	7.478.282	7.476.488	WAR File	28/09/2014 10:24 a.m.

Total 2 folders and 7.488.389 bytes in 2 files

Fig. A - 13, diachron.ear opened

Now the user should search into the lib folder the file that is call “diachron.jar” and open it, also this can be done using programs like winzip or winrar.


diachron.ear - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info VirusScan Comment SFX

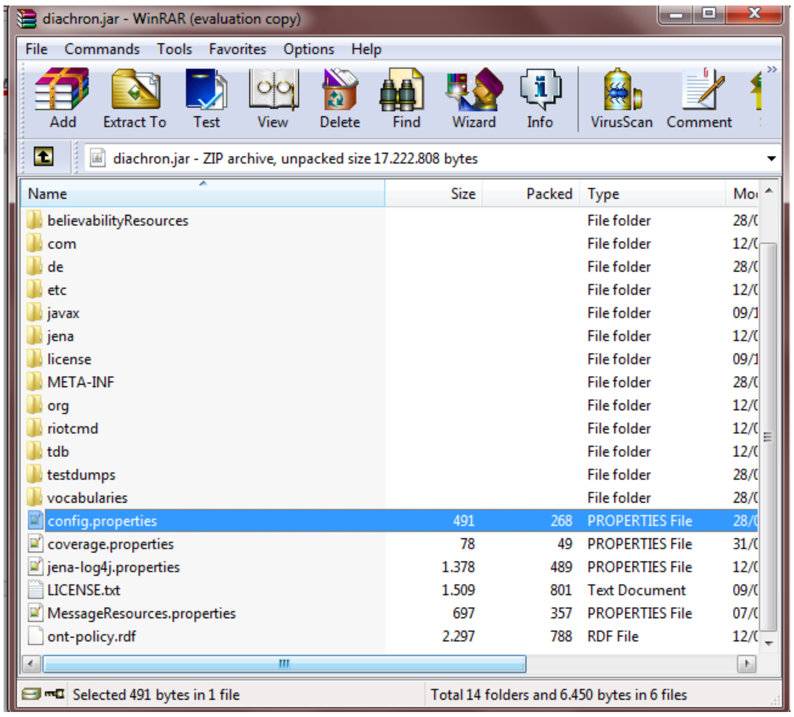
diachron.ear\lib - ZIP archive, unpacked size 34.700.447 bytes

Name	Size	Packed	Type	Modified
..			File folder	
commons-beanutils-1.7.0.jar	188.671	169.560	Executable Jar File	19/08/2014 02:30
commons-collections-3.2.jar	571.259	500.102	Executable Jar File	19/08/2014 02:30
commons-digester-1.8.jar	143.602	126.200	Executable Jar File	19/08/2014 02:30
commons-lang-2.4.jar	261.809	244.016	Executable Jar File	19/08/2014 02:30
commons-logging-1.0.4.jar	38.015	34.595	Executable Jar File	19/08/2014 02:30
commons-validator-1.3.1.jar	138.956	124.853	Executable Jar File	19/08/2014 02:30
diachron.jar	8.336.825	7.436.170	Executable Jar File	28/09/2014 10:24
groovy-all-1.5.5.jar	2.841.345	2.628.591	Executable Jar File	19/08/2014 02:30
iText-2.1.0.jar	1.050.691	993.649	Executable Jar File	19/08/2014 02:30
jakarta-oro-2.0.8.jar	65.261	56.142	Executable Jar File	19/08/2014 02:30
jasperreports-3.6.2.jar	2.633.731	2.379.206	Executable Jar File	19/08/2014 02:30
jcommon-1.0.16.jar	309.293	286.025	Executable Jar File	19/08/2014 02:30

Selected 8.336.825 bytes in 1 file   Total 27.211.513 bytes in 21 files

Fig. A - 14, Diachron.jar file located

Once this file is opened the user should see a list of folders and files, but the only one that needs to be modified is the one that is call config.properties, like it is shown in the next figure:


diachron.jar - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
VirusScan
Comment

diachron.jar - ZIP archive, unpacked size 17.222.808 bytes

Name	Size	Packed	Type	Mo
believabilityResources			File folder	28/(
com			File folder	12/(
de			File folder	28/(
etc			File folder	12/(
javax			File folder	09/1
jena			File folder	12/(
license			File folder	09/1
META-INF			File folder	28/(
org			File folder	12/(
riotcmd			File folder	12/(
tdb			File folder	12/(
testdumps			File folder	28/(
vocabularies			File folder	28/(
config.properties	491	268	PROPERTIES File	28/(
coverage.properties	78	49	PROPERTIES File	31/(
jena-log4j.properties	1.378	489	PROPERTIES File	12/(
LICENSE.txt	1.509	801	Text Document	09/(
MessageResources.properties	697	357	PROPERTIES File	07/(
ont-policy.rdf	2.297	788	RDF File	12/(

Selected 491 bytes in 1 file   Total 14 folders and 6.450 bytes in 6 files

Fig. A - 15, config.properties file location

On this file the user should configure the set of properties displayed, the content of the file look like this:

```
# File to store the results of the metrics
dataBase=C:/Lab/resultsThesis.xml

# Default mail to send the results, if any mail is specified by the user
defaultMail=cemontoyas@gmail.com

# File to store the rankings that are used by the Reputation metric
savedRatings=C:/Lab/ratingsThesis.xml

# Authentication values for the mail of google that is used to send mail from the application
userResponsibleMail=eislab@carlos-montoya.com
passResponsibleMail=quentlod
```

Fig. A - 16, config.properties file

The properties in the file configure some functionalities of the system. The first one is the directory where the program is going to store the results. The second property is the default user mail that the system send the results after running the program (this is because the time to compute takes a lot of time and this is the mechanism to announce that the process is already finished). The third property is defined to be the path were the system is going to look for the rankings file that are used by the Reputation metric. The last two properties configure the access to the mail that is being used by the program to send the resultant mail.

Now the user can close all the windows opened and start the JBoss server. Once these is done, the user can access to the location: http://localhost:8080/diachron/index.jsf[34].

[34] The port should be configure to the JBoss server port, it depends on the configuration of every server.

APPENDIX B – CD STRUCTURE

Attached to this thesis it is a CD with all the sources of the project, the structure of the CD folders are:

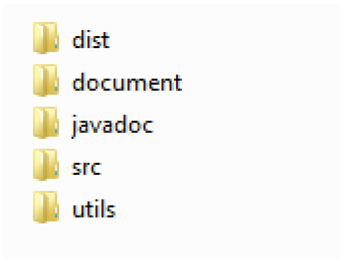


Fig. B - 1, CD folder structure

i. The “dist” folder: contains all the compile classes and files need it to deploy the project directly into the JBoss server.
ii. The “Javadoc” folder as its name mention is located the documentation of the code and can be access through the index.html that is inside the folder.
iii. The “document” folder contain the main thesis document in digital (PDF and Word), to be store, processed or publish.
iv. Into the “src” folder are the source files of the project, these files are saved as a project of eclipse IDE, then if the consumer want to add it to the eclipse IDE, just have to imported as a java project.
v. The “Utils” folder contained all the utilities that are need or can be useful at the moment of deploy the project, such as jboss-4.2.3.GA, openrdf-sesame-2.7.13-sdk