



UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE INFORMÁTICA

Master Thesis
MASTER IN ARTIFICIAL INTELLIGENCE RESEARCH

A SOFTWARE QUALITY MODEL FOR THE EVALUATION OF SEMANTIC TECHNOLOGIES

AUTHOR: FILIP RADULOVIĆ
SUPERVISOR: RAÚL GARCÍA-CASTRO
SUPERVISOR: ASUNCIÓN GÓMEZ-PÉREZ

June 2011

To my nephew Aleksa.

Acknowledgments

I would like to thank Raúl García-Castro for his valuable guidance and support during my work. His experience and patience helped me to extend my knowledge and further develop my research skills.

I am very grateful to Asunción Gómez-Pérez for giving me the opportunity to realize this thesis within the Ontology Engineering Group.

Also, I want to thank Daniel Garijo, Idafen Santana Pérez, and José Mora for helping me with the translation of some parts of the thesis into Spanish.

Last, but not least, I wish to thank my family for their everlasting support.

Abstract

In order to obtain high-quality software products, the specification and evaluation of quality during the software development process is of crucial importance. One important component in software evaluation is the software quality model, since it provides the basis for software evaluation and gives a better insight of the software characteristics that influence its quality. Furthermore, quality models also ensure a consistent terminology for software product quality and provide guidance for its measurement.

In recent years, semantic technologies have started to gain importance and, as the field becomes more and more popular, the number of these technologies is increasing exponentially. Just as with any other software product, the quality of semantic technologies is an important concern, and multiple evaluations of semantic technologies have been performed. However, the problem is that there is no consistent terminology for describing the quality of semantic technologies and it is difficult to compare them because of differences in the meaning of the evaluation characteristics used. Also, existing software quality models do not define those quality characteristics that are specific to semantic technologies.

This thesis presents a quality model for semantic technologies which aims to provide a common ground in the field of semantic technology evaluation. It also presents a new method for extending software quality models, based on a bottom-up approach, that is used to define the quality model for semantic technologies. Finally, this thesis describes the use of the semantic technology quality model in a web application that visualizes semantic technology evaluation results and provides semantic technology recommendations.

Resumen

Con el propósito de obtener productos software de alta calidad, es de capital importancia la especificación y evaluación de la calidad durante el proceso de desarrollo de software. Un componente importante en la evaluación del software es el modelo de calidad, puesto que provee los fundamentos para la evaluación del software y da una mejor perspectiva de las características del software que influyen en su calidad. Además, los modelos de calidad también aseguran una terminología sistemática para la calidad de productos software y proporcionan guías para su medida.

En los últimos años las tecnologías semánticas han comenzado a ganar importancia y, a medida que su popularidad ha crecido más y más, el número de estas ha crecido exponencialmente. Al igual que con otros productos software, la calidad de estas tecnologías es una cuestión de gran importancia y múltiples evaluaciones sobre ellas han sido llevadas a cabo. Sin embargo, el problema reside en que no existe una terminología consistente para describir la calidad de las tecnologías semánticas, lo cual dificulta compararlas, debido a las diferencias en el significado de las características usadas en la evaluación. Además, los modelos de calidad software no definen aquellas características que son propias de las tecnologías semánticas.

En esta tesis se presenta un modelo de calidad para tecnologías semánticas cuyo objetivo es elaborar una especificación común para la evaluación de dichas tecnologías. También se presenta un nuevo modelo para expandir modelos de calidad de software basado en una propuesta "bottom-up", que es usado para definir un modelo de calidad para tecnologías semánticas. Finalmente se describe el uso del modelo de calidad en una aplicación web que visualiza resultados de evaluación de tecnologías semánticas y aporta recomendaciones sobre las mismas.

Contents

Acknowledgments	i
Abstract	iii
Resumen	v
1 Introduction	1
1.1 Motivation	2
1.2 Structure of the Document	3
2 State of the Art	5
2.1 Semantic Technology Evaluation	5
2.1.1 The SEALS Project	6
2.2 Software Quality Models	7
2.2.1 McCall's Model	8
2.2.2 Boehm's Model	9
2.2.3 ISO 9126's Model	9
2.2.4 SQuaRE's Model	10
2.3 Approaches for Extending Software Quality Models	11
2.4 Conclusions	12
3 Approach	15
3.1 Insufficiencies of the State of the Art	15
3.2 Goals	15
3.3 Scope	16
3.4 General Vision for Approaching the Solution	17
4 Quality Model for Semantic Technologies	19
4.1 A Bottom-Up Method for Extending a Software Quality Model	19
4.2 Defining A Quality Model For Semantic Technologies	21
4.2.1 Identifying Basic Measures	21

4.2.2	Identifying Derived Measures	22
4.2.3	Identifying Quality Measures	23
4.2.4	Specifying Relationships Between Measures	23
4.2.5	Defining Domain-Specific Quality Sub-characteristic	24
4.2.6	Aligning Quality Sub-characteristics to a Quality Model	24
4.3	Complete Overview of the Quality Model	25
5	Detailed Description of the Quality Model	29
5.1	Ontology Engineering Tools	29
5.1.1	Test Data	31
5.1.2	Basic Measures	31
5.1.3	Derived Measures	32
5.1.4	Quality Measures	34
5.1.5	ISO 9126 Quality Characteristics	36
5.1.6	Semantic Quality Characteristics	36
5.2	Reasoning Systems	37
5.2.1	Test Data	37
5.2.2	Basic Measures	41
5.2.3	Quality Measures	42
5.2.4	ISO 9126 Quality Characteristics	45
5.2.5	Semantic Quality Characteristics	45
5.3	Matching Tools	46
5.3.1	Test Data	46
5.3.2	Basic Measures	48
5.3.3	Derived Measures	48
5.3.4	Quality Measures	48
5.3.5	ISO 9126 Quality Characteristics	49
5.3.6	Semantic Quality Characteristics	49
5.4	Semantic Search Tools	50
5.4.1	Test Data	50
5.4.2	Basic Measures	53
5.4.3	Derived Measures	53
5.4.4	Quality Measures	55
5.4.5	ISO 9126 Quality Characteristics	58
5.4.6	Semantic Quality Characteristics	59
5.5	Semantic Web Service Tools	60
5.5.1	Test Data	60
5.5.2	Basic Measures	61
5.5.3	Derived Measures	61
5.5.4	Quality Measures	63

5.5.5	ISO 9126 Quality Characteristics	64
5.5.6	Semantic Quality Characteristics	64
6	Application of the Semantic Technology Quality Model	67
6.1	Architecture	67
6.2	Dependencies	69
6.3	Instalation	69
6.4	Evaluation Results Visualization	70
6.4.1	Requirements	70
6.4.2	Use	70
6.5	Semantic Technology Recommendation	78
6.5.1	Requirements	80
6.5.2	Use	81
7	Conclusions and Future Work	85
	Publications	87
	Bibliography	89
A	Quality Model Ontology	93

List of Figures

2.1	ISO 9126 internal and external quality characteristics and sub-characteristics.	9
3.1	A general vision for approaching the solution.	17
4.1	The method for extending software quality models based on the bottom-up approach.	20
4.2	Steps of a conformance test execution.	22
4.3	Entities in the conformance scenario for ontology engineering tools.	25
4.4	External and internal quality characteristics of semantic technologies.	27
5.1	Conformance scenario.	29
5.2	Interoperability scenario.	30
5.3	Scalability scenario.	30
5.4	Quality characteristics of ontology engineering tools.	38
5.5	Classification scenario.	38
5.6	Class satisfiability scenario.	39
5.7	Ontology satisfiability scenario.	39
5.8	Entailment scenario.	40
5.9	Non-entailment scenario.	40
5.10	Quality characteristics of reasoning systems.	47
5.11	Accuracy scenario.	47
5.12	Quality characteristics of ontology matching tools.	50
5.13	Automated scenario (I).	51
5.14	Automated scenario (II).	51
5.15	User-in-the-loop scenario (I).	52
5.16	User-in-the-loop scenario (II).	52
5.17	Internal and external quality characteristics of semantic search tools.	60
5.18	Quality in use of semantic search tools.	60
5.19	Semantic web service discovery scenario.	61
5.20	Quality characteristics of semantic web service tools.	65

6.1	Application architecture.	68
6.2	Part of the table showing available conformance executions.	71
6.3	Part of the table showing the conformance results of a particular execution.	72
6.4	Information added and lost in a test execution.	72
6.5	Test results according to ontology language features.	73
6.6	Results for tools according to OWL DL ontology language.	74
6.7	Ontologies used in one test execution.	74
6.8	Statistics for a particular tool according to a particular test suite. . .	74
6.9	Part of the table showing available interoperability results.	75
6.10	Part of the table showing interoperability results for two particular tools according to a particular test suite.	76
6.11	Information added and lost in a test execution.	76
6.12	Test results according to ontology language features.	77
6.13	Results for tools according to the OWL DL test suite.	77
6.14	Ontologies used in one test execution.	77
6.15	Statistics for two particular tools according to a particular test suite. .	78
6.16	Software selection process.	81
6.17	Homepage of the semantic technologies recommendation system. . . .	82
6.18	Requirements specification form.	82
6.19	Results for the recommendation.	83
6.20	Results for a particular tool.	83
A.1	Graphical representation of the <i>QualityModel</i> ontology.	93

List of Tables

2.1	Measures used in conference publications.	6
2.2	Measures used in SEALS evaluation campaign.	8
2.3	ISO 9126 internal and external measures for Accuracy and Fault tolerance.	10
2.4	Elements of the described quality models.	11
4.1	Total number of measures obtained for semantic technologies.	23

Chapter 1

Introduction

When the World Wide Web (WWW) was created, people browsing the Web were only able to consume its content. Furthermore, the original Web was designed primarily for humans. It is also called a Web of Documents because it mainly consists of a huge number of documents in form of HTML pages. Every document carries some data, but those data are not connected because all connections exist on a document level only. The user that browses a web page knows if it is about food, musical instruments or football, but computers are not able to process and understand web page data and their meaning. But as the Web was evolving and there were more and more data present, various problems arose, like information overload and the fact that it was hard to find the right information. Furthermore, data was not easy interoperable between various sources.

The Semantic Web [27] is a new generation of the Web that introduces the semantics (meaning) of data and considers the Web to be a huge database. It is about extending the web of documents to a web of data and puts a focus on triples instead of pages, on RDF¹ instead of HTML and, while the original Web was designed for humans, the Semantic Web is designed for computer programs. Parts of the data that are present in various documents carry some meaning that machines are able to process and are linked with data from other documents.

Semantic technologies provide new ways to express in machine processable formats knowledge and data that can be exploited by computer programs, and their purpose is to enable the development of the Semantic Web. As the Semantic Web field becomes more and more popular, the number of these technologies is increasing exponentially. Different types of semantic technologies have been described in the literature [14]: for data and metadata management, querying and reasoning, ontology engineering, ontology customization, ontology evolution, ontology instance generation, and semantic web services.

¹<http://www.w3.org/RDF/>

1.1 Motivation

Software product quality has become an important concern in almost every domain or technology, and the specification and evaluation of quality during the software development process is of crucial importance for obtaining high quality software [3].

As it is the case with almost every software domain, the quality of semantic technologies is also of high importance. However, various problems exist regarding the evaluation of the quality of semantic technologies:

- There is no consistent terminology for specifying the quality of semantic technologies.
- Although there are plenty of existing evaluations, their nature varies significantly, from general evaluation frameworks [32] to tool-specific evaluations [18, 26] and even characteristic-specific evaluations [13].
- Because of the differences in the evaluations performed, evaluation results are very difficult or impossible to compare.

In the field of Software Engineering, these problems are solved by providing common frameworks for the specification of software quality and the definition and execution of software evaluations.

The main goal of this thesis is to provide a common ground for the evaluation of semantic technologies, which will enable the specification of quality requirements and the consequent comparison of evaluation results. To achieve this goal, we have developed a quality model for semantic technologies. Quality models provide the basis for software evaluation and give a better insight of the characteristics that influence software quality by specifying a consistent terminology for software quality and by providing guidance for its measurement.

This thesis also aims to provide an application that visualizes semantic technology evaluation results. Such application can be of great help in the analysis and comparison of evaluation results. Furthermore, this thesis also aims to provide an application for semantic technology recommendation, which will help users who are not very familiar with semantic technologies to select the ones that would best suite their needs.

The work in this thesis has been performed in the scope of the SEALS European project². The SEALS project is developing a platform which provides freely available services for the evaluation of semantic technologies, together with public evaluation campaigns which provide evaluation results. Evaluation campaigns cover five different types of semantic technologies: ontology engineering, ontology reasoning, ontology matching, semantic search, and semantic web services. Each type of

²<http://www.seals-project.eu>

these technologies is evaluated through several different evaluation scenarios, and against common test data.

In this thesis we have taken all the evaluation scenarios defined in the SEALS project as an input for the definition of the quality model. Besides, the results obtained in the first evaluation campaigns have been the ones used in the web applications.

1.2 Structure of the Document

This document is organized as follows:

- Chapter 2 analyses the state of the art in semantic technology evaluation and software quality models. First, a review of semantic technology evaluation efforts is presented and then some well-known software quality models are described. Afterwards, existing approaches for extending quality models are presented.
- Chapter 3 enumerates the inadequacies of the state of the art, presents the goals of this master thesis, defines the scope of the thesis, and describes the approach followed to provide the solution.
- Chapter 4 presents a new method for extending software quality models that is based on a bottom-up approach, and then describes in a concrete scenario how the method has been applied in the field of semantic technologies. Afterwards, a complete overview of the semantic technology quality model is presented.
- Chapter 5 presents the detailed description of the quality model for semantic technologies obtained from all evaluation campaigns in the SEALS project. For every type of semantic technology, details about the output of all the steps of the method applied for obtaining the quality model are presented.
- Chapter 6 presents applications that are based on the quality model. It includes applications for semantic technologies evaluation result visualization and semantic technology recommendation, and describes their architecture, details of implementation, and provides guidance for their usage.
- Chapter 7 draws some conclusions and presents ideas for future work.

Chapter 2

State of the Art

This chapter presents the work that is related to this thesis, with the purpose of providing an overview of the field that this thesis covers. Section 2.1 describes the state of the art in semantic technology evaluation. Then, some well-known software quality models are presented in Section 2.2., and Section 2.3 describes existing methods for extending software quality models.

2.1 Semantic Technology Evaluation

As mentioned in the introduction, multiple evaluations of semantic technologies have been performed. This section presents a literature review of semantic technology evaluations that we have performed according to the procedure described in [20]. The procedure consists of following several steps:

- *Background.* The purpose of the research is to review previous evaluations of semantic technologies, including the evaluation results and processes, in order to obtain the overview of the current state in semantic technology evaluation.
- *Research questions.* We stated two research questions: Is there a specification of the quality of semantic technologies? Which quality measures are used to evaluate those characteristics?
- *Strategy for searching previous studies.* We analyzed the proceedings of the two most relevant conferences in the semantic area: the International Semantic Web Conference (nine editions) and the European Semantic Web Conference (seven editions) to identify those publications that deal with semantic technology evaluation.
- *Study selection criteria and procedures.* We focused on those publications that describe evaluation methods or suggest measures for evaluation, as well as

publications that suggest new algorithms (e.g., for reasoning or semantic web service discovery) and that are also evaluated in the publication.

- *Data extraction strategy.* In the analysis of the existing evaluations, we extracted and classified the data according to the quality measures used.
- *Synthesis of the extracted data.* Gathered data was used to determine which quality measures are mostly used in semantic technology evaluations.

In total, we have analyzed fifty seven publications. Table 2.1 shows an overview of this analysis including, for each type of semantic technology, the evaluation measures used.

Table 2.1: Measures used in conference publications.

Ontology engineering tools (2)
execution (1), execution time (1), information added/lost (1)
Ontology matching tools (21)
precision (19), recall (19), f-measure (13), measure at cut-off point (1)
Reasoning and storage systems (18)
classification time (5), execution time (5), reasoning time (5), entailment time (1), labeling time (1), lattice operation time (1), justification time (1), loading time (4), reasoner errors (1), correct results (7), wrong classifications (1), fitness value (1)
Semantic search tools (5)
query execution time (2), speed (1), recall (4), precision (3), reciprocal rank (1), f-measure (1), relevance (1), loading time (1), usability (2)
Semantic web service tools (11)
precision (12), recall (8), f-measure (1), returned sources (1), binary preference (1), reciprocal rank (1)

2.1.1 The SEALS Project

The Semantic Evaluation At Large Scale (SEALS) project is an FP7¹ project which aims to create the infrastructure for semantic technology evaluation, and to perform evaluations of semantic technologies through public evaluation campaigns.

The first evaluation campaign of the SEALS project has produced evaluation results for several types of semantic technologies. For each type of the semantic technology, several tools were evaluated according to common test data in one or several evaluation scenarios:

¹http://cordis.europa.eu/fp7/home_en.html

- Three evaluation scenarios were defined for the ontology engineering tools:
 - conformance scenario
 - interoperability scenario
 - scalability scenario
- Five evaluation scenarios were defined for the ontology reasoning systems:
 - classification scenario
 - class satisfiability scenario
 - ontology satisfiability scenario
 - entailment scenario
 - non-entailment scenario
- One evaluation scenario was defined for the matching tools:
 - accuracy scenario
- Two evaluation scenarios were defined for the semantic search tools:
 - user-in-the-loop scenario (accuracy and performance)
 - automatic scenario (accuracy, performance, usability)
- One evaluation scenario was defined for the semantic web service tools
 - semantic web service discovery

Table 2.2 shows an overview of the measures used in the first SEALS evaluation campaign.

2.2 Software Quality Models

Quality in general is a complex and multifaceted concept that can be described from different approaches depending on whether the focus is in the concept of quality, the product, the user of the product, how the product was manufactured, or the value the product provides [17].

The way we define software quality depends on the approach that we take [24]; software quality means different things to different people and therefore defining and measuring quality will depend on the viewpoint. Similarly, choosing one software quality model or another will also depend on the intended users and uses of such model in concrete evaluations.

Next, this section describes some well-known software quality models and identifies their elements.

Table 2.2: Measures used in SEALS evaluation campaign.

Ontology engineering tools
execution errors, import/export duration, information added/lost
Ontology matching tools
precision, recall, f-measure, harmonic measure
Reasoning and storage systems
classification time, class satisfiability time, ontology satisfiability time, entailment time, non-entailment time, loading time, classification correctness, class satisfiability correctness, ontology satisfiability correctness, entailment correctness, non-entailment correctness
Semantic search tools
execution time, input time, question time, precision, recall, f-measure, number of results, load time, load successful, usability, number of attempts, answer found rate, experiment time, satisfaction value
Semantic web service tools
precision, recall, f-measure, binary preference, reciprocal rank, number of retrieved documents, normalized discounted cumulative gain

2.2.1 McCall's Model

McCall's software quality model is represented as a hierarchy of *factors*, *criteria* and *metrics* [8]. Factors are at the highest level in the hierarchy and represent the characteristics of the software product. Criteria are the middle layer and are considered to be the attributes of the factors, so that for every factor a set of criteria is defined. At the bottom level, metrics provide measures for software attributes.

McCall's model predefines a set of eleven software quality factors that are classified according to the product life cycle in three different groups:

- Product transition: portability, reusability, and interoperability
- Product revision: maintainability, flexibility, and testability
- Product operations: correctness, reliability, efficiency, integrity, and usability

McCall's quality model also gives the relationships between quality factors and metrics in the form of linear equations based on regression analyses. This is considered one of the major contributions of this model; however, the omission of the functionality aspect is regarded as the main lack [3].

2.2.2 Boehm's Model

Like McCall's model, Boehm's one has a hierarchical structure. It consists of twenty four quality characteristics divided into three levels [4]. It also gives a set of metrics and, while McCall's model is more related to the product view, Boehm's model includes users' needs.

2.2.3 ISO 9126's Model

The International Organization for Standardization (ISO) identified the need for a unique and complete software quality standard and, therefore, produced the ISO 9126 standard for software quality [21].

The ISO 9126 standard defines three types of quality: internal quality, external quality, and quality in use.

Six main software quality characteristics for external and internal quality are specified: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability, which are further decomposed into sub-characteristics (see Fig 2.1) that are manifested externally when the software is used, and are the result of internal software attributes [21]. The standard also provides the internal and external measures for sub-characteristics (see Table 2.3 for an example for Accuracy and Fault tolerance).

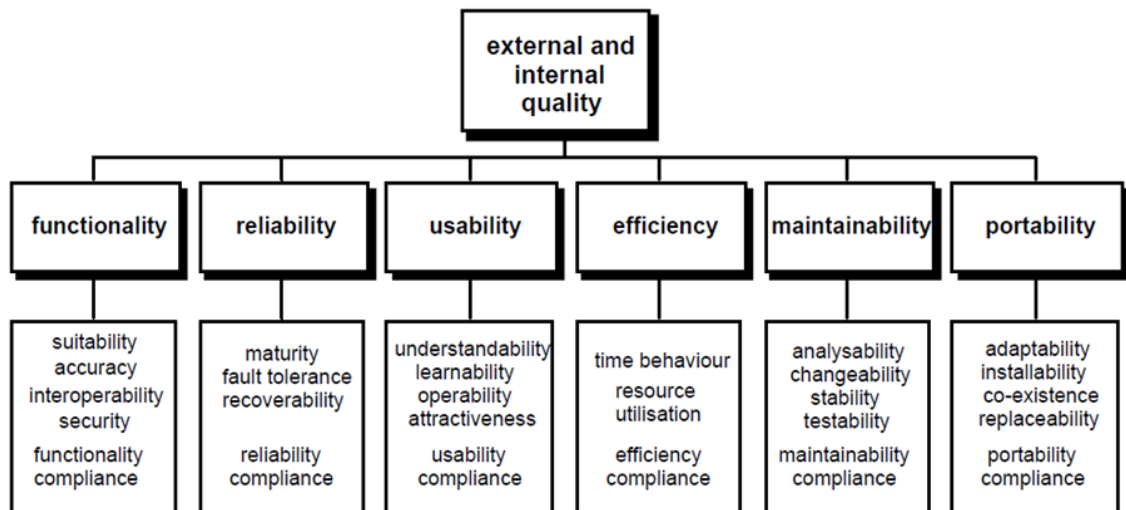


Figure 2.1: ISO 9126 internal and external quality characteristics and sub-characteristics.

Regarding quality in use, the model proposes four characteristics: Effectiveness, Productivity, Safety, and Satisfaction.

The ISO 9126 standard gives the complete view of software quality with evaluation criteria and definitions for all quality characteristics and sub-characteristics. Some authors also suggests that according to the nature of the product itself some new sub-characteristics can be added, the definitions of existing ones can be changed, or some sub-characteristics can be eliminated from the model [6].

However, as pointed out in [1], some practical problems with ISO 9126 arise, namely, the ambiguity in metric definitions and usability interpretation. Furthermore, the authors argue that the number of attributes and measures are missing, that some characteristics are too abstract, and that the standard itself is open to interpretations, which, according to the authors, questions its purpose.

Table 2.3: ISO 9126 internal and external measures for Accuracy and Fault tolerance.

Quality Characteristics	Quality Sub-Characteristics	External Measures	Internal Measures
Functionality	Accuracy	Computational accuracy	Computational accuracy
		Precision	Precision
Reliability	Fault tolerance	Failure avoidance	Failure avoidance
		Incorrect operation avoidance	Failure avoidance
		Breakdown avoidance	

2.2.4 SQuaRE's Model

Although the ISO 9126 standard has been accepted and used successfully, some problems and issues for its further use and improvement have been identified. They eventually arise mainly because of advances in technologies and changes of users needs. As pointed out by Azuma [2], the main problems were due to issues on metrics and lack of a quality requirement standard.

In order to address those issues, the existing standard is being redesigned and has been named SQuaRE. By the time of writing this thesis, the parts of the SQuaRE standard related to the quality model and evaluations are still under development (ISO 25010, ISO 25040) and their final versions will be published in 2011.

As a summary of this section, Table 2.4 presents the elements of the described quality models. In our work, we have decided to adopt terminology from ISO 9126 standard.

Table 2.4: Elements of the described quality models.

Structure/Model	McCall	Boehm	ISO 9126
First level	Factor	High level characteristic	Characteristic
Second level	Criteria	Primitive characteristic	Sub-characteristic
Third level	Metrics	Metrics	Measures
Relationships between entities	Factor-Metric	/	Measure-Measure

2.3 Approaches for Extending Software Quality Models

Some authors have proposed software quality models for various types of applications: B2B [3], mail servers [7], web-based applications [39], e-learning systems [33], and ERP systems [6]. All those authors have used the ISO 9126 standard as the basis software quality model, and have extended it to fit their particular domain.

Software quality model extensions can be performed following two main approaches [9]:

- A **top-down** approach that starts from the quality characteristics and continues towards the quality measures.
- A **bottom-up** approach that starts from the quality measures and defines the quality sub-characteristics that are related to each specific measure.

In their work, Franch and Carvallo proposed a method based on a top-down approach for customizing the ISO 9126 quality model [11]. After defining and analyzing the domain, the method proposes six steps:

1. *Determining quality sub-characteristics.* In the first step, according to the domain, some new quality sub-characteristics are added while others are excluded or their definitions are changed.
2. *Defining a hierarchy of sub-characteristics.* If it is needed, sub-characteristics are further decomposed according to some criteria.
3. *Decomposing sub-characteristics into attributes.* In this step abstract sub-characteristics are decomposed into more concrete concepts which refer to some particular software attribute (i.e., observable feature).
4. *Decomposing derived attributes into basic ones.* Attributes that are not directly measurable are further decomposed into basic ones.

5. *Stating relationships between quality entities.* Relationships between quality entities are explicitly defined. Three possible types of relationships are identified:
 - *Collaboration.* When increasing the value of one entity implies increasing of the value of another entity.
 - *Damage.* When increasing the value of one entity implies decreasing the value of another entity.
 - *Dependency.* When some values of one entity require that another entity fulfills some conditions.
6. *Determining metrics for attributes.* To be able to compare and evaluate quality, it is necessary to define metrics for all attributes in the model.

In building their quality model for B2B applications, Behkamal et al. proposed a method to customize the ISO 9126 quality model in five steps [3]. The main difference with the previous method is that in Behkamal's approach the quality characteristics are ranked by experts; the experts should provide weights for all quality characteristics and sub-characteristics, and these weights are later used to establish their importance. Besides, Behkamal's approach does not contemplate defining relationships between quality entities.

2.4 Conclusions

The analysis of the current state in semantic technology evaluation shows that different quality characteristics were evaluated in different evaluations. Also, in some cases, different measures have been applied when evaluating the same characteristics; therefore, merging different evaluation results and their analysis might lead to wrong and misleading conclusions.

Without a common ground to put the evaluations of semantic technologies under, it is very difficult to compare them and assess their quality. Furthermore, there is no consistent terminology for describing their quality, and available software quality models do not specify the quality characteristics specific to semantic technologies. In that sense, evaluation results and their analyses can often be misleading.

Using a software quality model may provide a good common basis for evaluation, since software quality models support the specification of quality and instructions on how to measure it. Existing software quality models provide terminology and description of characteristics that are general for almost every kind of software. However, in order to use a quality model in a specific domain, it usually has to be extended to include the particularities of such domain.

Various methods for extending quality models have been proposed in the literature; they all follow a top-down approach, starting from general characteristics to concrete measures. For some cases, however, a bottom-up approach would be more effective as is the case of those which have many of evaluations to extract the quality model (as in the semantic technologies field). However, we have not found any example of a bottom-up approach in the literature.

Chapter 3

Approach

This chapter presents the approach followed in the development of this thesis. First, Section 3.1 lists the insufficiencies of the state of the art and, then, Section 3.2 presents the goals that we want to achieve to overcome these insufficiencies. The scope of the work presented in this thesis is described in Section 3.3. Finally, Section 3.4 outlines the general guidelines that we have followed in the development of the work.

3.1 Insufficiencies of the State of the Art

The insufficiencies of the state of the art, based on the conclusions presented in Section 2.4, are the following:

- Lack of a method based on a bottom-up approach for extending software quality models.
- Lack of a consistent terminology for describing the quality of semantic technologies, as well as of a specification of quality characteristics for semantic technologies and the quality measures needed for obtaining them.
- Lack of a motivating example that shows the need of the quality model for semantic technologies

3.2 Goals

The objective of this master thesis is to define a semantic technology quality model, which helps to put semantic technology evaluations under a common ground and to describe quality characteristics, as well as the measures that influence them.

This thesis provides solutions to the insufficiencies of the State of the Art mentioned in the previous section. In particular we provide:

1. A method for extending software quality models, based on a bottom-up approach.
2. A quality model for semantic technologies, which describes the quality characteristics and measures that are relevant for semantic technologies, and provides instructions on how to obtain them.
3. A web application for the visualization of semantic technology evaluation results and for the recommendation of semantic technologies according to users' needs. This application uses the quality model proposed.

3.3 Scope

The scope of this Master thesis is the following:

1. With respect to the method applied for defining the quality model, we have used the method based on the bottom-up approach that we propose in this thesis.
2. With respect to evaluation results used in defining the quality model, we have used the evaluation results provided in the first evaluation campaign of the SEALS project, as well as the results obtained in the literature review.
3. With respect to the types of the semantic technologies, we have taken into account technologies that are evaluated in the SEALS project: ontology engineering tools, storage and reasoning systems, matching tools, semantic search tools, and semantic web service tools.
4. With respect to the web application for result visualization, we provide the possibility to visualize conformance and interoperability results of ontology engineering tools. The visualization of the results obtained for other types of semantic technologies is out of the scope of this thesis.
5. With respect to the web application for semantic technologies recommendation, the system only provides recommendations for ontology engineering tools, based on the evaluation results obtained in the SEALS project. Recommendations for other types of semantic technologies are out of the scope of this thesis.

3.4 General Vision for Approaching the Solution

In order to specify the quality characteristics for semantic technologies and the measures that influence them, we have developed a software quality model that extends the ISO 9126 quality model. The software quality model consists of a detailed description of the different quality entities that affect semantic technology quality and provides detailed specifications on how to measure them.

For developing the quality model, we have used existing evaluation results. Therefore, in order to use them, we first developed a method for extending software quality models that is based on a bottom-up approach.

At the end, we have used the quality model to develop a web application that will provide visualizations of evaluation results and recommendations of semantic technologies.

Figure 3.1 shows a general vision for approaching the solution.

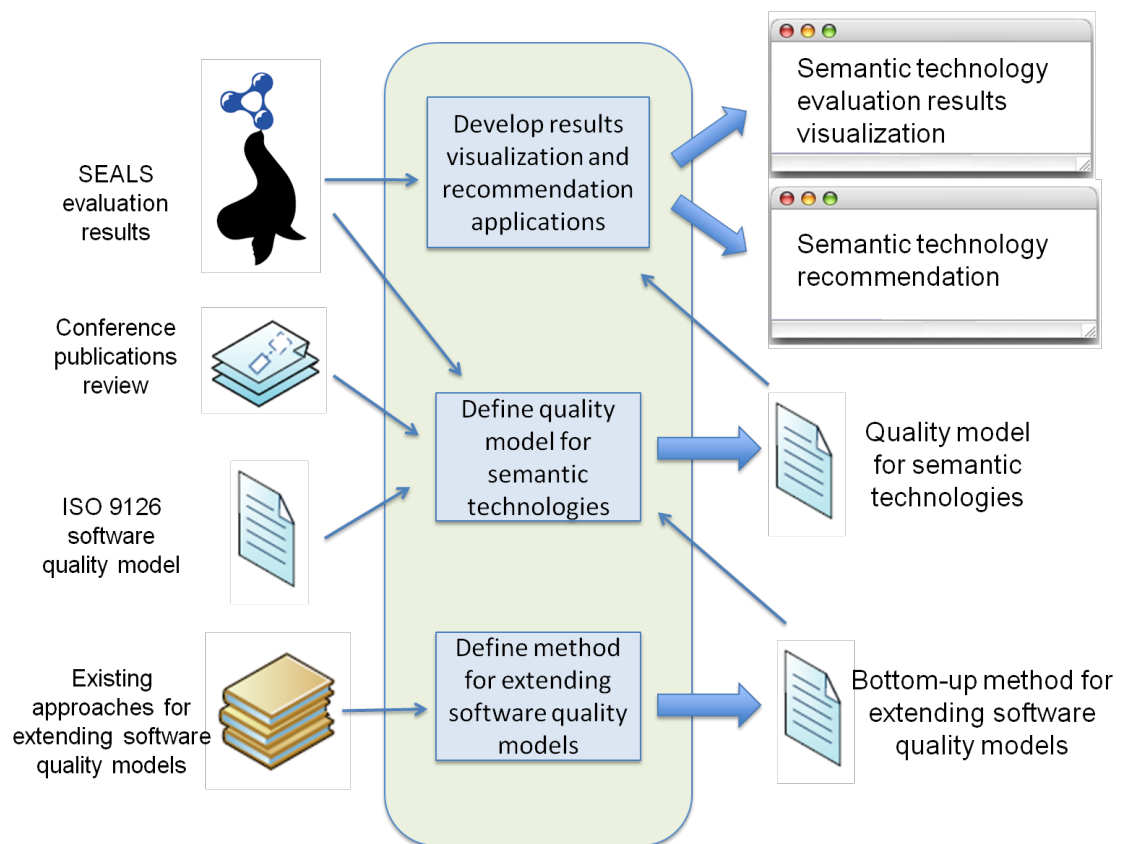


Figure 3.1: A general vision for approaching the solution.

Chapter 4

Quality Model for Semantic Technologies

This chapter presents a method for extending software quality models that is based on a bottom-up approach, starting from existing evaluation results, instead of following a top-down approach, as those presented in Section 2.3.

Afterwards, Section 4.2 describes how we have performed each step of the method for defining the quality model. As an example, we describe one concrete scenario, that of evaluating the conformance of ontology engineering tools. Finally, Section 4.3 presents the complete overview of the quality model for semantic technologies.

4.1 A Bottom-Up Method for Extending a Software Quality Model

In some scenarios, it can be helpful to base in real practices the extension of the quality model because of the existence of a significant body of software evaluations and evaluation results. An example of this occurs in the semantic technology area.

In this method, evaluation results are used as the starting point from which the quality measures, sub-characteristics and characteristics are specified.

The method for extending a software quality model consists in performing the following six consecutive steps (Fig 4.1):

1. *To identify basic measures.* The output of evaluating a software product with some input data (i.e., executing a test case) allows identifying the basic measures of a certain evaluation execution.
2. *To identify derived measures.* Basic measures can be combined to obtain derived ones, which are also related to one particular evaluation execution (i.e.,

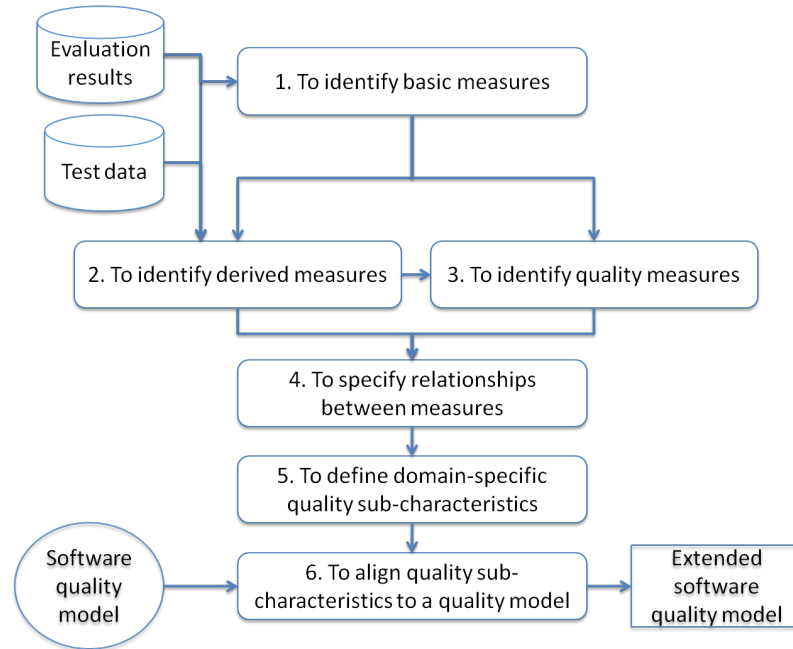


Figure 4.1: The method for extending software quality models based on the bottom-up approach.

test case).

3. *To identify quality measures.* Quality measures are measures related to a whole evaluation (i.e., multiple test cases using different input data) and are obtained by the aggregation of basic and derived measures.
4. *To specify relationships between measures.* In this step, which can be performed in parallel with the previous ones, relationships between measures are expressed either in an informal way (e.g., the collaboration, damage and dependency categories proposed in [11]) or more formally (e.g., with the formulas used for obtaining the measures, as proposed in [5]). For any derived measure defined, it is recommendable to specify the function (or set of functions) that allows obtaining such derived measure from the basic ones. Similarly, for any quality measure, it is also advisable to identify the function that defines it based in other measures. Also, it is important to note that one quality measure can be obtained using several different functions.
5. *To define domain-specific quality sub-characteristics.* Every software product from a particular domain has some sub-characteristics that are different from other software products and those sub-characteristics, together with more

generic ones, should be identified and precisely defined. Every quality measure provides some information about one or several software sub-characteristics; therefore, based on the software quality measures defined in the previous step, software quality sub-characteristics are specified. Furthermore, it is not necessary that every quality sub-characteristic has only one measure that determines it, but rather a set of measures. Finally, if needed, some quality sub-characteristics can be combined into more general ones.

6. *To align quality sub-characteristics to a quality model.* In this step, the alignment with an existing quality model is established; i.e., the software quality sub-characteristics that have been previously defined are related to others already specified in the existing model. Depending on the domain and nature of the software product, some new quality characteristics can be specified, or existing ones can be modified or excluded.

4.2 Defining A Quality Model For Semantic Technologies

This section describes in detail how we have applied the method in the example of conformance scenario of ontology engineering tools.

4.2.1 Identifying Basic Measures

The starting point for defining software quality measures has been the set of evaluation results obtained in the SEALS project, which provides evaluation results for different types of semantic technologies (ontology engineering tools [16], reasoning systems [38], ontology matching tools [10], semantic search tools [37], and semantic web service tools [36]).

For each type of technology, different evaluation scenarios were defined using in each of them different test data as input. In this step we identified the basic measures of each evaluation scenario (i.e., those outputs directly produced by the software during the evaluation).

Different test suites are used for evaluating the conformance of ontology engineering tools, which are composed of different test cases each containing:

- *Origin ontology.* The ontology to be used as input.

A test case execution consists of importing the file containing an origin ontology (O_i) into the tool and then exporting the imported ontology to another ontology file (O_i^{II}), as shown in Fig 4.2.

The basic measures obtained after a test case execution are:

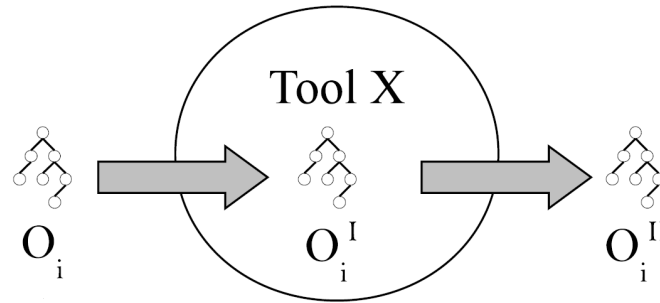


Figure 4.2: Steps of a conformance test execution.

- *Final ontology.* The ontology that is produced by the tool when importing and exporting the origin ontology.
- *Execution Problem.* Whether there were any execution problems in the tool when importing and exporting the origin ontology. Possible values are *true*, and *false*.

4.2.2 Identifying Derived Measures

In this step, basic measures identified in the previous step were combined together with the test data in order to obtain derived measures.

In the conformance scenario, based on the test data and the basic measures of one test execution, the following derived measures were specified:

- *Information added.* The information added to the origin ontology after importing and exporting it.
- *Information lost.* The information lost from the origin ontology after importing and exporting it.
- *Structurally equivalent.* Whether the origin ontology and the final one are structurally equivalent. Possible values are *true*, and *false*.
- *Semantically equivalent.* Whether the origin ontology and the final one are semantically equivalent. Possible values are *true*, and *false*.
- *Conformance.* Whether the ontology has been imported and exported correctly with no addition or loss of information. Possible values are *true*, and *false*.

4.2.3 Identifying Quality Measures

Quality measures are related to a set of test executions, and are obtained using basic and derived measures. In some cases, however, quality measures can be specified using only basic measures.

From the derived measures in the conformance scenario, the following quality measures were obtained:

- *Ontology language component support.* Whether the tool fully supports an ontology language component.
- *Ontology language component coverage.* The ratio of ontology components that are shared by a tool internal model and an ontology language model.
- *Ontology information change.* The ratio of information additions or loss when importing and exporting ontologies.
- *Execution errors.* The ratio of tool execution errors when importing and exporting ontologies.

Similarly to the example of the conformance evaluation presented above, we have defined measures for the other types of tools. Table 4.1 summarizes the obtained results.

Table 4.1: Total number of measures obtained for semantic technologies.

Tool/Measures	Basic	Derived	Quality Measures
Ontology engineering tools	7	20	8
Ontology matching tools	1	3	4
Reasoning systems	7	0	8
Semantic search tools	12	11	21
Semantic web service tools	5	10	11
Total	27	40	50

4.2.4 Specifying Relationships Between Measures

We have identified the relationships between measures in a formal way by defining the formulas used for obtaining derived and quality measures.

For example, in the conformance scenario the formula for the *Information added* derived measure calculates the structural difference between the origin and final ontologies:

$$\text{final ontology} - \text{origin ontology}$$

Similarly, the formula for the *Execution errors* quality measure calculates the percentage of tests with execution problems:

$$\frac{\# \text{ tests where execution problem} = \text{true}}{\# \text{ tests}} \times 100$$

4.2.5 Defining Domain-Specific Quality Sub-characteristic

In this step, from the quality measures previously identified, we defined the set of quality sub-characteristics that are affected by those measures. In some cases we were able to reuse existing quality sub-characteristics but, in others, we had to define domain-specific ones.

In the conformance scenario, based on the measures and analysis presented above, we have identified three quality sub-characteristics:

- *Ontology language model conformance.* The degree to which the knowledge representation model of the software product adheres to the knowledge representation model of an ontology language. It can be measured using two different measures, which are obtained in the conformance evaluation:
 - *Ontology language component coverage*
 - *Ontology language component support*
- *Ontology processing accuracy.* The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when processing ontologies. It can be measured using:
 - *Ontology information change*
- *Ontology Processing Robustness.* The ability of the software product to process ontologies correctly in the presence of invalid inputs or stressful environmental conditions. It can be measured using:
 - *Execution errors*

Fig. 4.3 presents the basic measures, derived measures, quality measures, and quality characteristics of the conformance evaluation for ontology engineering tools.

4.2.6 Aligning Quality Sub-characteristics to a Quality Model

Since ISO 9126 is a widely adopted and used standard, we have also adopted it for constructing the quality model for semantic technologies.

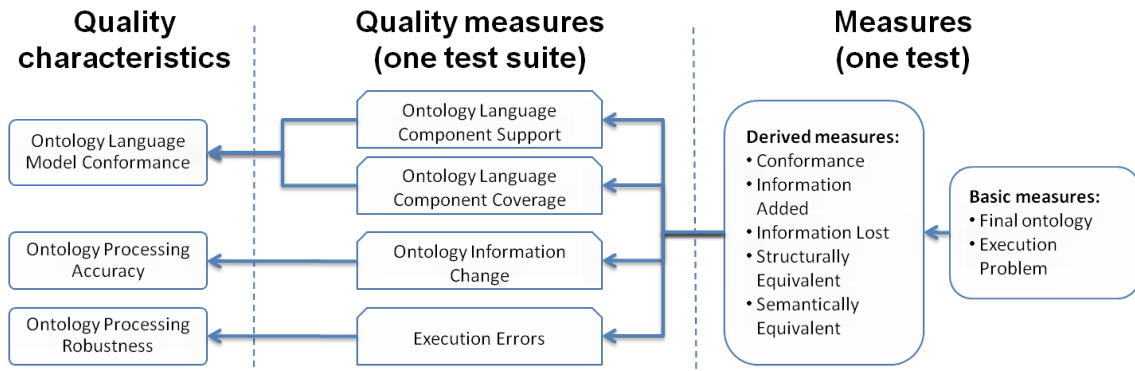


Figure 4.3: Entities in the conformance scenario for ontology engineering tools.

In the previous step we have identified the set of quality sub-characteristics that are specific for semantic technologies. In this step, all the identified sub-characteristics were properly assigned to ones that already exist in the ISO 9126 quality model.

In the case of the conformance scenario, identified quality sub-characteristics are aligned as follows:

- *Ontology language model conformance* is defined as a sub-characteristic of *Functionality compliance* (i.e., the capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality).
- *Ontology processing accuracy* is defined as a sub-characteristic of *Accuracy* (i.e., the capability of the software product to provide the right or agreed results or effects with the needed degree of precision).
- *Ontology processing robustness* is defined as a sub-characteristic of *Robustness* (i.e., the ability of the software product to function correctly in the presence of invalid inputs or stressful environmental conditions).

4.3 Complete Overview of the Quality Model

Fig. 4.4 shows a complete overview of the quality model for semantic technologies that was obtained after analyzing all the evaluation scenarios and results from the SEALS evaluation campaigns.

In the final version of the model, we have also taken into account the analysis presented in the state of the art (Section 2.1). Two characteristics that were not

obtained from the SEALS evaluation campaigns appear in the analysis, and were also included in the model. Those are:

- *Semantic web service time behaviour.* The capability of the software product to provide appropriate response and processing times when performing semantic web service discovery tasks.
- *Matching time behaviour.* The capability of the software product to provide appropriate response and processing times when performing matching tasks.

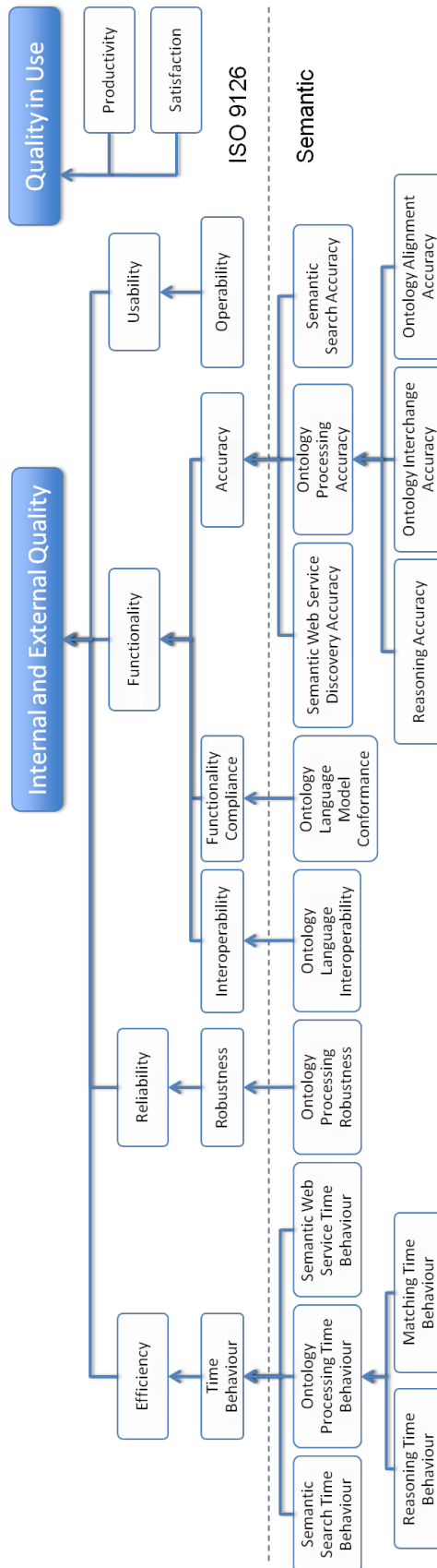


Figure 4.4: External and internal quality characteristics of semantic technologies.

Chapter 5

Detailed Description of the Quality Model

This chapter presents in detail the quality model obtained for all the evaluation campaigns performed in the SEALS project which covered: ontology engineering tools (Section 5.1), reasoning systems (Section 5.2), ontology matching tools (Section 5.3), semantic search tools (Section 5.4), and semantic web service tools (Section 5.5).

5.1 Ontology Engineering Tools

The evaluation campaign for ontology engineering tools contained three evaluation scenarios to evaluate the conformance (Fig 5.1), interoperability (Fig 5.2), and scalability (Fig 5.3) of these tools.

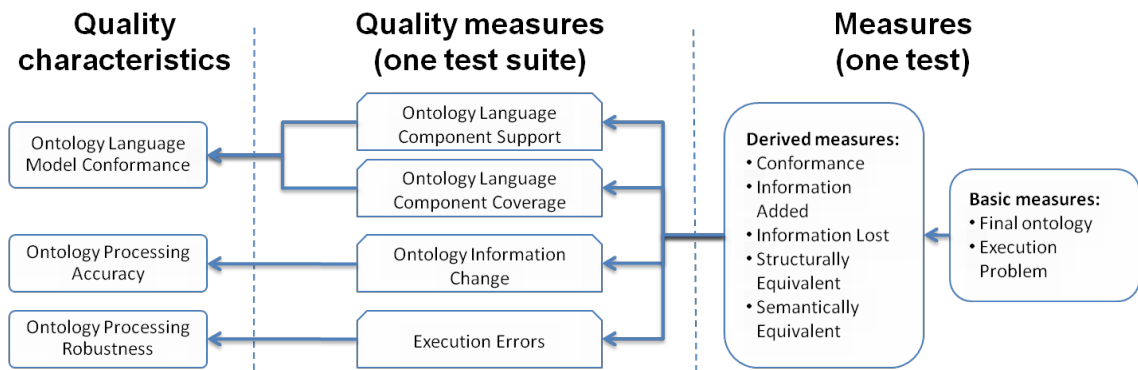


Figure 5.1: Conformance scenario.

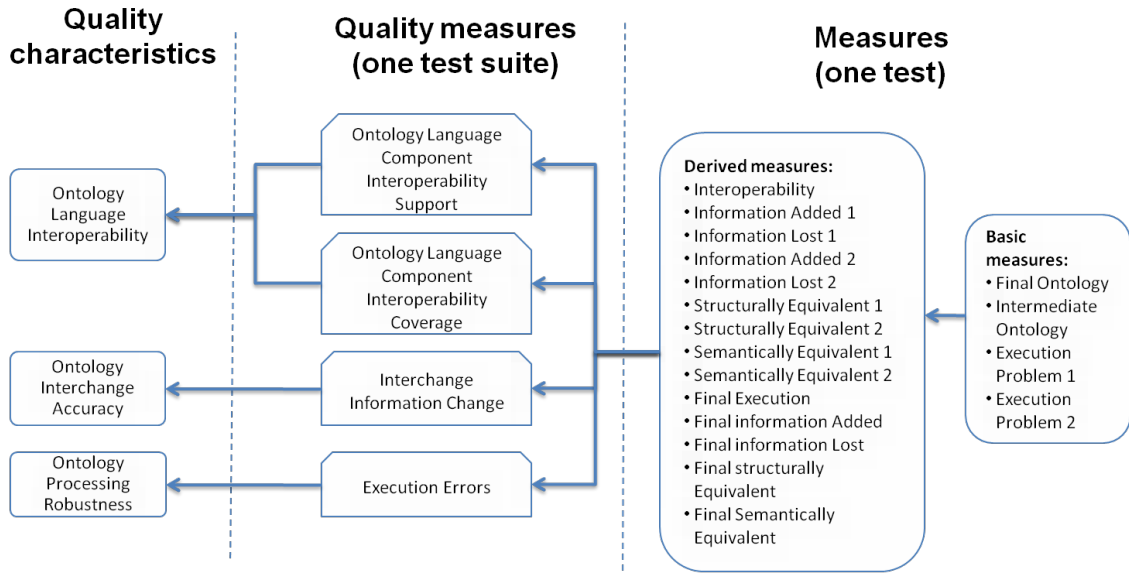


Figure 5.2: Interoperability scenario.

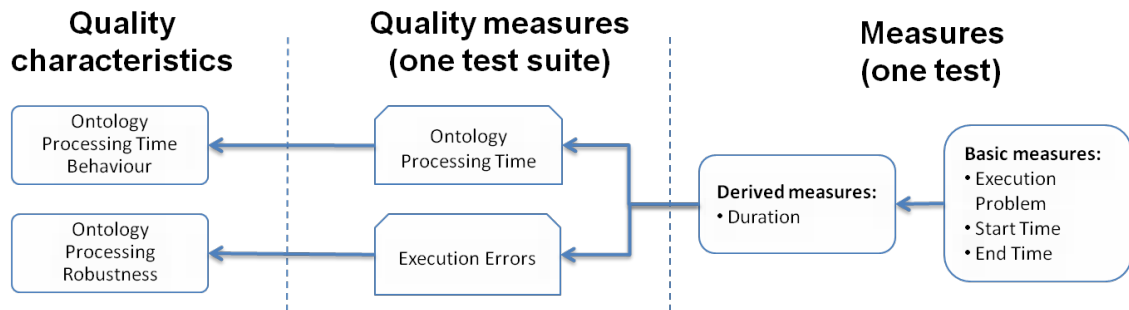


Figure 5.3: Scalability scenario.

5.1.1 Test Data

The test data used in the evaluation scenarios had the same structure. Different test suites were used, each containing several tests with one origin ontology each.

Origin Ontology - The ontology to be used as input

5.1.2 Basic Measures

Conformance Scenario

Final Ontology - The ontology that is produced by the tool when importing and exporting the origin ontology

Execution Problem - Whether there was any execution problem in the tool when importing and exporting the origin ontology. Possible values are *true*, and *false*

Interoperability Scenario

Intermediate Ontology - The ontology that is produced by the origin tool when importing and exporting the origin ontology

Final Ontology - The ontology that is produced by the tool when importing and exporting the intermediate ontology

Execution Problem 1 - Whether there was any execution problem in the tool when importing and exporting the origin ontology. Possible values are *true*, and *false*

Execution Problem 2 - Whether there was any execution problem in the tool when importing and exporting the intermediate ontology. Possible values are *true*, and *false*

Scalability Scenario

Execution Problem - Whether there was any execution problem in the tool when importing and exporting the origin ontology. Possible values are *true*, and *false*

Start Time - The time when the operation of importing and exporting the origin ontology starts

End Time - The time when the operation of importing and exporting the origin ontology ends

5.1.3 Derived Measures

Conformance Scenario

Information Added - The information added to the origin ontology (e.g., triples, axioms) after importing and exporting it

$$\text{final ontology} - \text{origin ontology}$$

Information Lost - The information lost from the origin ontology (e.g., triples, axioms) after importing and exporting it

$$\text{origin ontology} - \text{final ontology}$$

Structurally Equivalent - Whether the origin ontology and the final one are structurally equivalent. Possible values are *true*, and *false*

$$(\text{information added} = \text{null}) \wedge (\text{information lost} = \text{null})$$

Semantically Equivalent - Whether the origin ontology and the final one are semantically equivalent. Possible values are *true*, and *false*

$$\text{final ontology} \equiv \text{origin ontology}$$

Conformance - Whether the origin ontology has been imported and exported correctly with no addition or loss of information. Possible values are *true*, and *false*

$$\text{semantically equivalent} \wedge \neg(\text{execution problem})$$

Interoperability Scenario

Information Added 1 - The information added to the origin ontology after importing and exporting it

$$\text{intermediate ontology} - \text{origin ontology}$$

Information Lost 1 - The information lost from the origin ontology after importing and exporting it

$$\text{origin ontology} - \text{intermediate ontology}$$

Information Added 2 - The information added to the intermediate ontology after importing and exporting it

$$\text{final ontology} - \text{intermediate ontology}$$

Information Lost 2 - The information lost from the intermediate ontology after importing and exporting it

intermediate ontology – final ontology

Semantically Equivalent 1 - Whether the origin ontology and the intermediate one are semantically equivalent. Possible values are *true*, and *false*

intermediate ontology \equiv origin ontology

Semantically Equivalent 2 - Whether the intermediate ontology and the final one are semantically equivalent. Possible values are *true*, and *false*

final ontology \equiv intermediate ontology

Structurally Equivalent 1 - Whether the origin ontology and the intermediate one are structurally equivalent. Possible values are *true*, and *false*

(information added 1 = null) \wedge (information lost 1 = null)

Structurally Equivalent 2 - Whether the intermediate ontology and the final one are structurally equivalent. Possible values are *true*, and *false*

(information added 2 = null) \wedge (information lost 2 = null)

Final Information Added - The information added to the origin ontology after importing and exporting it by the first tool and then importing and exporting the intermediate ontology by the second tool

final ontology – origin ontology

Final Information Lost - The information lost from the origin ontology after importing and exporting it by the first tool and then importing and exporting the intermediate ontology by the second tool

origin ontology – final ontology

Final Structurally Equivalent - Whether the origin ontology and the final one are structurally equivalent. Possible values are *true*, and *false*

(final information added = null) \wedge (final information lost = null)

Final Semantically Equivalent - Whether the origin ontology and the final one are semantically equivalent. Possible values are *true*, and *false*

$$(\text{semantically equivalent 1}) \wedge (\text{semantically equivalent 2})$$

Final Execution - Whether there was any execution problem in the tool when importing and exporting the origin and intermediate ontology. Possible values are *true*, and *false*

$$(\text{execution problem 1}) \vee (\text{execution problem 2})$$

Interoperability - Whether the origin ontology has been interchanged correctly with no addition or loss of information. Possible values are *true*, and *false*

$$(\text{final semantically equivalent}) \wedge \neg(\text{execution final})$$

Scalability Scenario

Duration - The amount of time needed for importing and exporting the origin ontology

$$\text{end time} - \text{start time}$$

5.1.4 Quality Measures

Conformance Scenario

Ontology Language Component Support - Whether the tool fully supports an ontology language component

$$\frac{\# \text{ tests that contain the component where conformance} = \text{true}}{\# \text{ tests that contain the component}} = 1$$

Ontology Language Component Coverage - The ratio of ontology components that are shared by a tool internal model and an ontology language model

$$\frac{\# \text{ components in the ontology language where component support} = \text{true}}{\# \text{ components in the ontology language}} \times 100$$

Ontology Information Change - The ratio of information additions or losses when importing and exporting ontologies

$$\frac{\# \text{ tests where (information added} \neq \text{null or information lost} \neq \text{null)}}{\# \text{ tests}} \times 100$$

Execution Errors - The ratio of tool execution errors when importing and exporting ontologies

$$\frac{\# \text{ tests where execution problem} = \text{true}}{\# \text{ tests}} \times 100$$

Interoperability Scenario

Ontology Language Component Interoperability Support - Whether the tool fully supports an ontology language component interchange

$$\frac{\# \text{ tests that contain the component where interoperability} = \text{true}}{\# \text{ tests that contain the component}} = 1$$

Ontology Language Component Interoperability Coverage - The ratio of ontology components that can be interchanged with other tools

$$\frac{\# \text{ components in the ontology language where component interoperability support} = \text{true}}{\# \text{ components in the ontology language}} \times 100$$

Interchange Information Change - The ratio of information additions or losses when interchanging ontologies

$$\frac{\# \text{ tests where (final information added} \neq \text{null or final information lost} \neq \text{null)}}{\# \text{ tests}} \times 100$$

Execution Errors - The ratio of tool execution errors when interchanging ontologies

$$\frac{\# \text{ tests where final executions} = \text{true}}{\# \text{ tests}} \times 100$$

Scalability Scenario

Ontology Processing Time - The average amount of time needed for importing and exporting ontologies

$$\frac{\sum_n \text{duration in the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Execution Errors - The ratio of tool execution errors when importing and exporting ontologies

$$\frac{\# \text{ tests where execution problem} = \text{true}}{\# \text{ tests}} \times 100$$

5.1.5 ISO 9126 Quality Characteristics

Functionality - The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions

Functionality Compliance - The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality. *Functionality Compliance* is a sub-characteristic of *Functionality*

Interoperability - The capability of the software product to interact with one or more specified systems. *Interoperability* is a sub-characteristic of *Functionality*

Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision. *Accuracy* is a sub-characteristic of *Functionality*

Efficiency - The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions

Time Behaviour - The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions. *Time behaviour* is a sub-characteristic of *Efficiency*

Reliability - The capability of the software product to maintain a specified level of performance when used under specified conditions

Robustness - The ability of the software product to function correctly in the presence of invalid inputs or stressful environmental conditions. *Robustness* is a sub-characteristic of *Reliability*

5.1.6 Semantic Quality Characteristics

Ontology Language Model Conformance - The degree to which the knowledge representation model of the software product adheres to the knowledge representation model of an ontology language. *Ontology language model conformance* is a sub-characteristic of *Functionality Compliance* and can be measured using:

- Ontology Language Component Coverage
- Ontology Language Component Support

Ontology Processing Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when processing ontologies. *Ontology Processing Accuracy* is a sub-characteristic of *Accuracy* and can be measured using:

- Ontology Information Change

Ontology Language Interoperability - The degree to which the software product can interchange ontologies and use the ontologies that have been interchanged.

Ontology language interoperability is a sub-characteristic of *Interoperability* and can be measured using:

- Ontology Language Component Interoperability Coverage
- Ontology Language Component Interoperability Support

Ontology Interchange Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when interchanging the ontologies. *Ontology Interchange Accuracy* is the sub-characteristic of *Ontology Processing Accuracy* and can be measured using:

- Interchange Information Change

Ontology Processing Time Behaviour - The capability of the software product to provide appropriate response and processing times when working with ontologies. *Ontology Processing Time Behaviour* is a sub-characteristic of *Time Behaviour* and can be measured using:

- Ontology Processing Time

Ontology Processing Robustness - The ability of the software product to process ontologies correctly in the presence of invalid inputs or stressful environmental conditions. *Ontology Processing Robustness* is a sub-characteristic of *Robustness* and it can be measured using:

- Execution Errors

Figure 5.4 shows the part of the quality model obtained from the evaluation campaign for ontology engineering tools.

5.2 Reasoning Systems

The evaluation campaign for reasoning systems contained evaluation scenarios to evaluate the accuracy, scalability, and robustness of these tools. Those scenarios include: classification scenario (Fig 5.5), class satisfiability scenario (Fig 5.6), ontology satisfiability scenario (Fig 5.7), entailment scenario (Fig 5.8), and non-entailment scenario (Fig 5.9).

5.2.1 Test Data

Classification Scenario

Ontology - The ontology to be classified

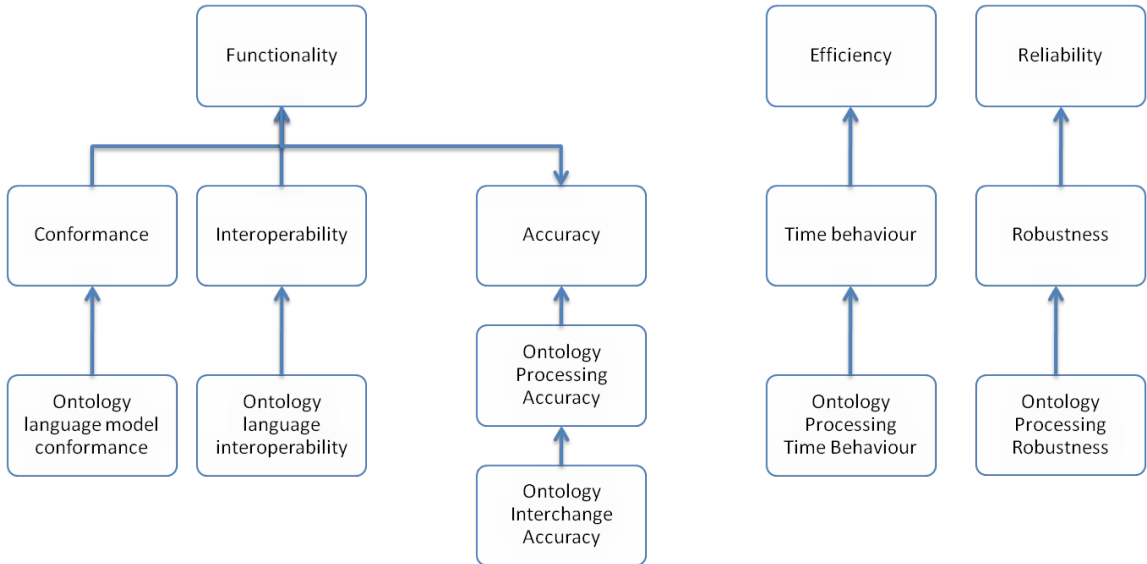


Figure 5.4: Quality characteristics of ontology engineering tools.

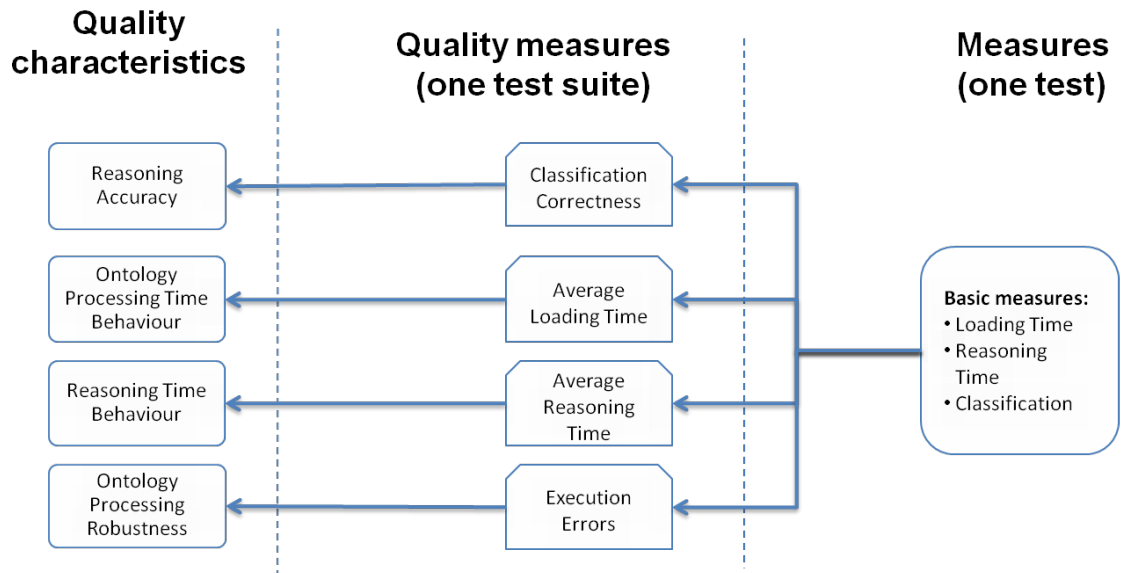


Figure 5.5: Classification scenario.

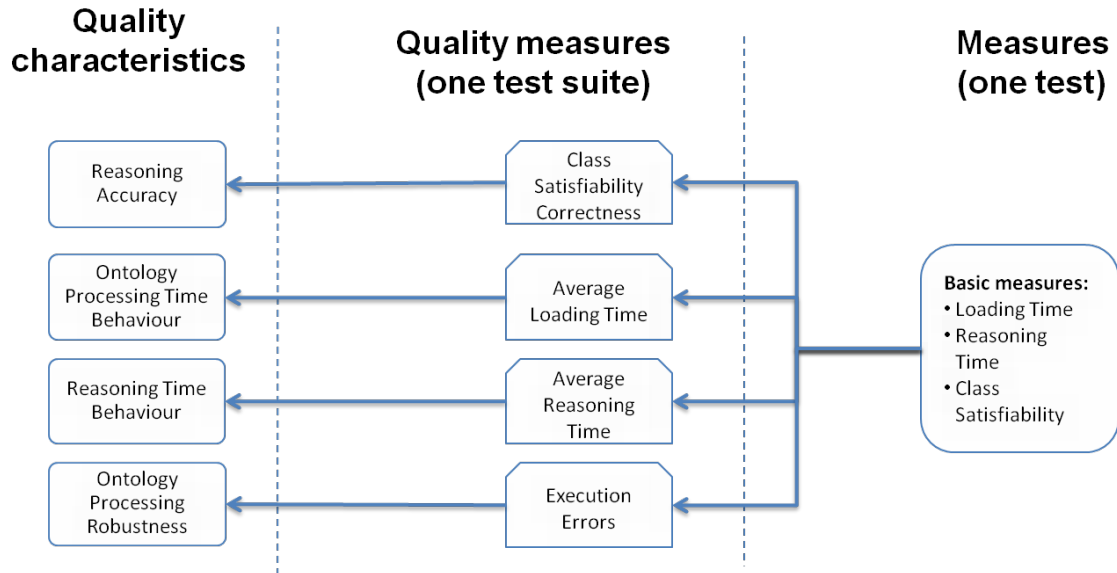


Figure 5.6: Class satisfiability scenario.

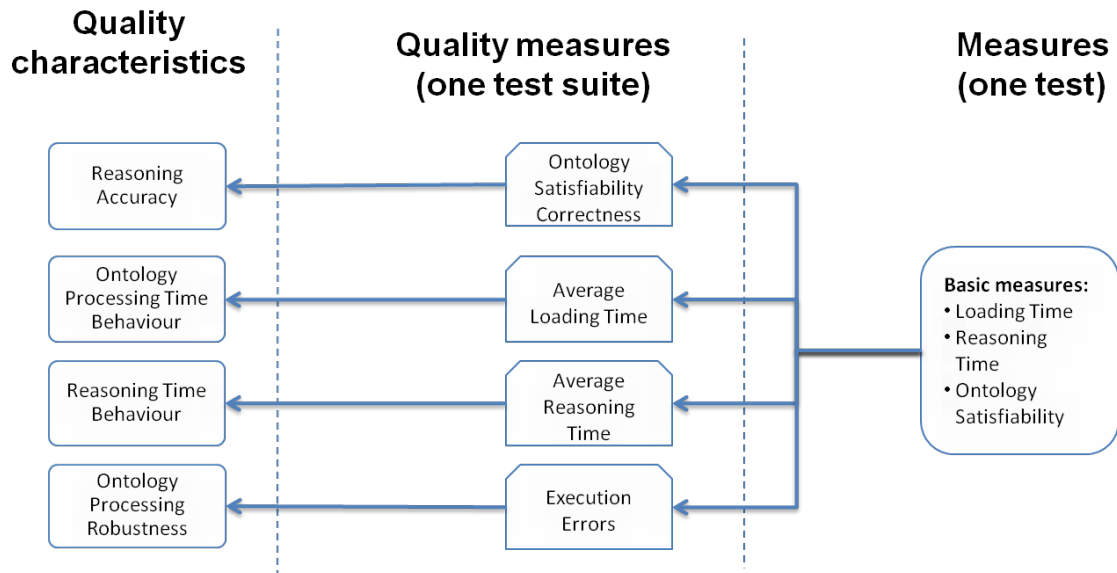


Figure 5.7: Ontology satisfiability scenario.

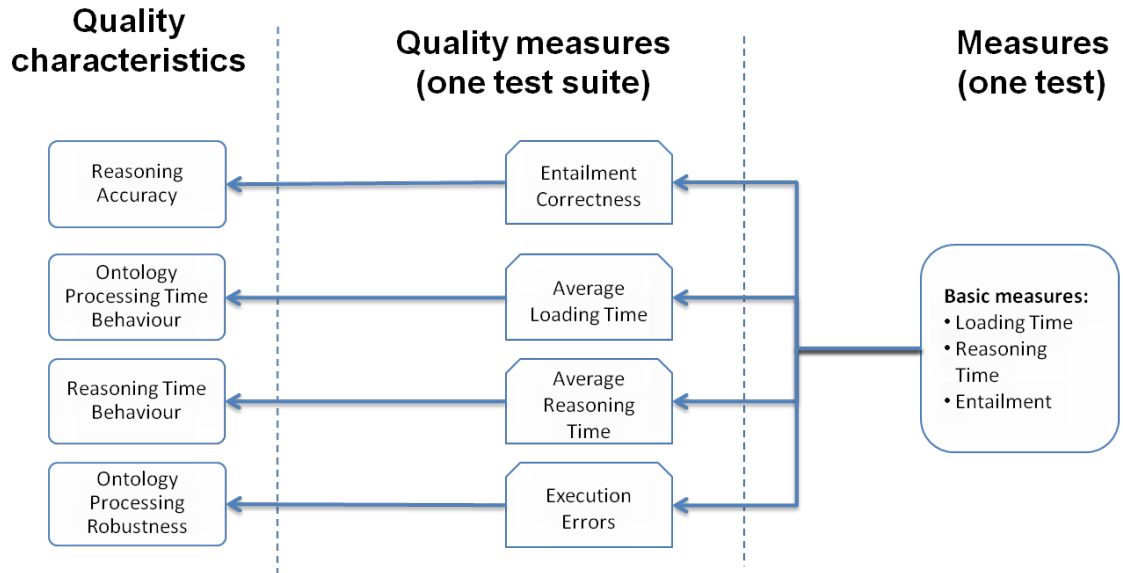


Figure 5.8: Entailment scenario.

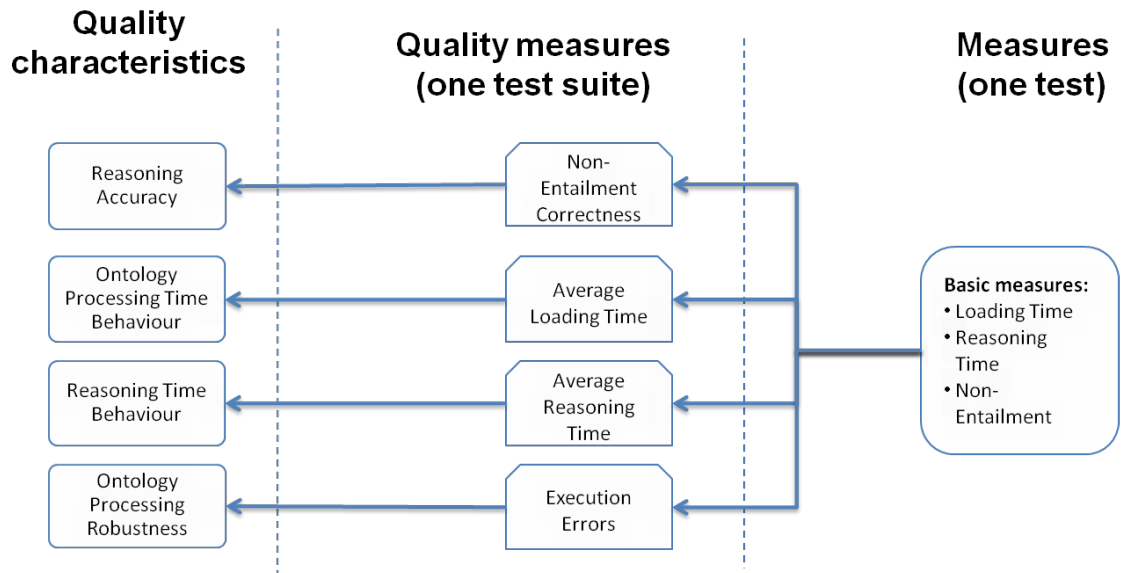


Figure 5.9: Non-entailment scenario.

Class Satisfiability Scenario

Ontology - The ontology to be used in the class satisfiability task

Class URIs - URIs of the classes to be checked for satisfiability

Ontology Satisfiability Scenario

Ontology - The ontology to be checked for satisfiability

Entailment Scenario

Premise Ontology - The ontology to be used in the entailment scenario

Conclusion Ontology - The ontology to be checked for being logically entailed by the premise ontology

Non-Entailment Scenario

Premise Ontology - The ontology to be used in the non-entailment scenario

Conclusion Ontology - The ontology to be checked for not being logically entailed by the premise ontology

5.2.2 Basic Measures**Classification Scenario**

Loading time - The amount of time needed to load the ontology

Reasoning Time - The amount of time needed to perform a classification operation

Classification - Whether the classification of the ontology is performed correctly. Possible values are *true*, *false*, *unknown*, and *error*

Class Satisfiability Scenario

Loading time - The amount of time needed to load the ontology

Reasoning Time - The amount of time needed to perform a class satisfiability operation

Class Satisfiability - Whether a specific class from the ontology is satisfiable. Possible values are *true*, *false*, *unknown*, and *error*

Ontology Satisfiability Scenario

Loading time - The amount of time needed to load the ontology

Reasoning Time - The amount of time needed to perform an ontology satisfiability operation

Ontology Satisfiability - Whether the origin ontology is satisfiable. Possible values are *true*, *false*, *unknown*, and *error*

Entailment Scenario

Loading time - The amount of time needed to load the ontology

Reasoning Time - The amount of time needed to perform entailment operation

Entailment - Whether the conclusion ontology is being logically entailed by the premise ontology. Possible values are *true*, *false*, *unknown*, and *error*

Non-Entailment Scenario

Loading time - The amount of time needed to load the ontology

Reasoning Time - The amount of time needed to perform non-entailment operation

Non-Entailment - Whether the conclusion ontology is not being logically entailed by the premise ontology. Possible values are *true*, *false*, *unknown*, and *error*

5.2.3 Quality Measures

Classification Scenario

Average Loading Time - The average time needed for the tool to load an ontology

$$\frac{\sum_n \text{loading time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Reasoning Time - The average time needed for the tool to perform a classification task

$$\frac{\sum_n \text{reasoning time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Execution Errors - The ratio of tool execution errors when performing a classification task

$$\frac{\# \text{ tests where classification} = \text{error}}{\# \text{ tests}} \times 100$$

Classification Correctness The ratio of correctly performed classification tasks

$$\frac{\# \text{ tests where classification} = \text{true}}{\# \text{ tests}} \times 100$$

Class Satisfiability Scenario

Average Loading Time - The average time needed for the tool to load an ontology

$$\frac{\sum_n \text{loading time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Reasoning Time - The average time needed for the tool to perform a class satisfiability task

$$\frac{\sum_n \text{reasoning time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Execution Errors - The ratio of tool execution errors when performing class satisfiability tasks

$$\frac{\# \text{ tests where class satisfiability} = \text{error}}{\# \text{ tests}} \times 100$$

Class Satisfiability Correctness The ratio of correctly performed class satisfiability tasks

$$\frac{\# \text{ tests where class satisfiability} = \text{true}}{\# \text{ tests}} \times 100$$

Ontology Satisfiability Scenario

Average Loading Time - The average time needed for the tool to load an ontology

$$\frac{\sum_n \text{loading time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Reasoning Time - The average time needed for the tool to perform an ontology satisfiability task

$$\frac{\sum_n \text{reasoning time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Execution Errors - The ratio of tool execution errors when performing an ontology satisfiability task

$$\frac{\# \text{ tests where ontology satisfiability} = \text{error}}{\# \text{ tests}} \times 100$$

Ontology Satisfiability Correctness - The ratio of correctly performed ontology satisfiability tasks

$$\frac{\# \text{ tests where ontology satisfiability} = \text{true}}{\# \text{ tests}} \times 100$$

Entailment Scenario

Average Loading Time - The average time needed for the tool to load an ontology

$$\frac{\sum_n \text{ loading time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Reasoning Time - The average time needed for the tool to perform an entailment task

$$\frac{\sum_n \text{ reasoning time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Execution Errors - The ratio of tool execution errors when performing an entailment task

$$\frac{\# \text{ tests where entailment} = \text{error}}{\# \text{ tests}} \times 100$$

Entailment Correctness - The ratio of correctly performed entailment tasks

$$\frac{\# \text{ tests where entailment} = \text{true}}{\# \text{ tests}} \times 100$$

Non-Entailment Scenario

Average Loading Time - The average time needed for the tool to load an ontology

$$\frac{\sum_n \text{ loading time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Reasoning Time - The average time needed for the tool to perform a non-entailment task

$$\frac{\sum_n \text{ reasoning time for the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Execution Errors - The ratio of tool execution errors when performing a non-entailment task

$$\frac{\# \text{ tests where non-entailment} = \text{error}}{\# \text{ tests}} \times 100$$

Non-Entailment Correctness The ratio of correctly performed non-entailment tasks

$$\frac{\# \text{ tests where non-entailment} = \text{true}}{\# \text{ tests}} \times 100$$

5.2.4 ISO 9126 Quality Characteristics

Functionality - The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions

Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision. *Accuracy* is a sub-characteristic of *Functionality*

Efficiency - The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions

Time Behaviour - The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions. *Time behaviour* is a sub-characteristic of *Efficiency*

Reliability - The capability of the software product to maintain a specified level of performance when used under specified conditions

Robustness - The ability of the software product to function correctly in the presence of invalid inputs or stressful environmental conditions. *Robustness* is a sub-characteristic of *Reliability*

5.2.5 Semantic Quality Characteristics

Ontology Processing Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when processing ontologies. *Ontology Processing Accuracy* is a sub-characteristic of *Accuracy*

Reasoning Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when performing the reasoning task. *Reasoning Accuracy* is a sub-characteristic of *Ontology Processing Accuracy* and can be measured using:

- Class Satisfiability Correctness

- Ontology Satisfiability Correctness
- Classification Correctness
- Entailment Correctness
- Non-Entailment Correctness

Ontology Processing Time Behaviour - The capability of the software product to provide appropriate response and processing times when working with ontologies. *Ontology Processing Time Behaviour* is a sub-characteristic of *Time Behaviour* and can be measured using:

- Average Loading time

Reasoning Time Behaviour - The capability of the software product to provide appropriate response and processing times when performing reasoning tasks. *Reasoning Time Behaviour* is a sub-characteristic of *Ontology Processing Time Behaviour* and can be measured using:

- Average Reasoning Time

Ontology Processing Robustness - The ability of the software product to process ontologies correctly in the presence of invalid inputs or stressful environmental conditions. *Ontology Processing Robustness* is a sub-characteristic of *Robustness* and it can be measured using:

- Execution Errors

Figure 5.10 shows the part of the quality model obtained from the evaluation campaign for reasoning tools.

5.3 Matching Tools

The evaluation campaign for matching tools contained one evaluation scenario to evaluate accuracy (Fig 5.11).

5.3.1 Test Data

Ontology 1 - One ontology to be used as input for an alignment task

Ontology 2 - Another ontology to be used as input for an alignment task

Expected Alignment - The expected alignment to be obtained when aligning the two input ontologies

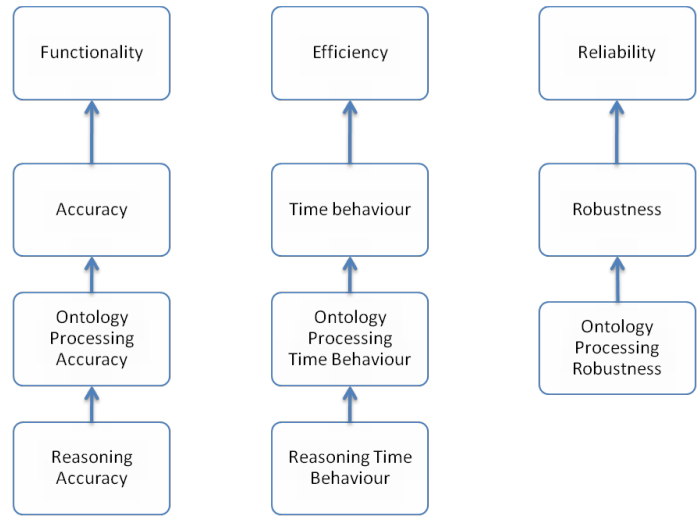


Figure 5.10: Quality characteristics of reasoning systems.

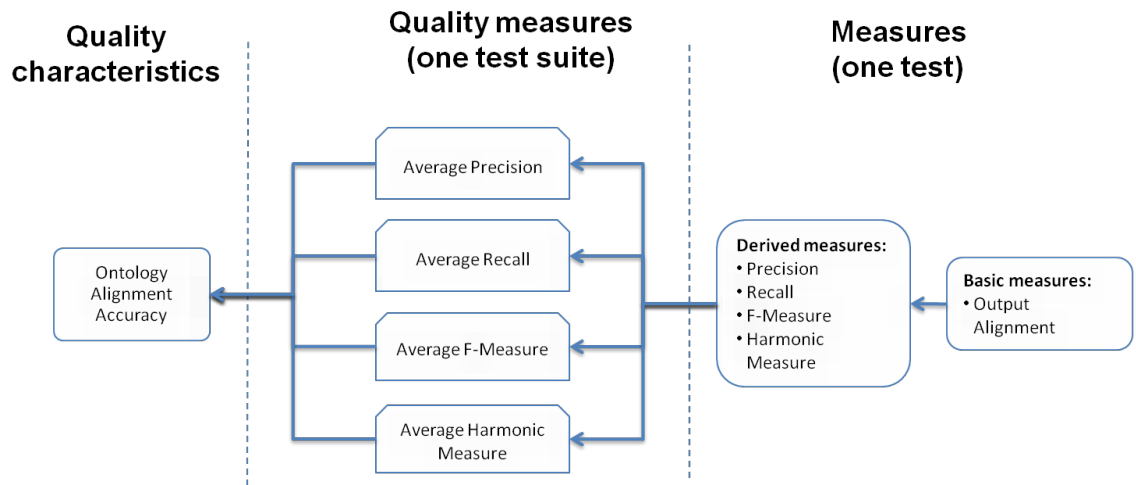


Figure 5.11: Accuracy scenario.

5.3.2 Basic Measures

Output Alignment - The alignment that is produced by the tool when aligning ontology 1 and ontology 2

5.3.3 Derived Measures

Precision - The precision of the alignment task

$$\frac{\# \text{ output alignments that match expected alignments}}{\# \text{ output alignments}}$$

Recall - The recall of the alignment task

$$\frac{\# \text{ output alignments that match expected alignments}}{\# \text{ expected alignments}}$$

F-Measure - The aggregation measure of precision and recall

$$\frac{\text{precision} \times \text{recall}}{(1-\alpha) \times \text{precision} + \alpha \times \text{recall}}, \alpha = [0..1]$$

Harmonic Measure - The aggregation measure of precision and recall (value of F-Measure when $\alpha = 0.5$)

$$\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 2$$

5.3.4 Quality Measures

Average Precision - The average precision of the ontology matching tool

$$\frac{\sum_n \text{precision of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Recall - The average recall of the ontology matching tool

$$\frac{\sum_n \text{recall of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average F-Measure - The average F-Measure of the ontology matching tool

$$\frac{\sum_n \text{F-Measure of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Harmonic Measure - The average harmonic measure of the ontology matching tool

$$\frac{\sum_n \text{harmonic measure of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

5.3.5 ISO 9126 Quality Characteristics

Functionality - The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions

Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision. *Accuracy* is a sub-characteristic of *Functionality*

5.3.6 Semantic Quality Characteristics

Ontology Processing Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when processing ontologies. *Ontology Processing Accuracy* is a sub-characteristic of *Accuracy*

Ontology Alignment Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision when performing the alignment task. *Ontology alignment accuracy* is a sub-characteristic of *Ontology Processing Accuracy* and can be measured using:

- Average Precision
- Average Recall
- Average F-measure
- Harmonic Mean

Figure 5.12 shows the part of the quality model obtained from the evaluation campaign for matching tools.

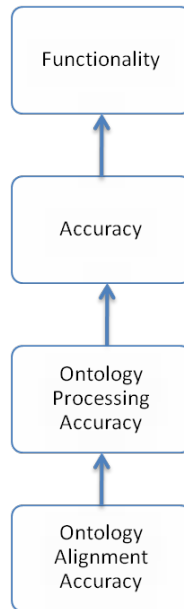


Figure 5.12: Quality characteristics of ontology matching tools.

5.4 Semantic Search Tools

The evaluation campaign for semantic search tools contained evaluation scenarios to evaluate accuracy, performance and usability. The evaluation was divided into two separate phases, an automated phase (Fig 5.13 and Fig 5.14) and a user-in-the-loop phase (Fig 5.15 and Fig 5.16).

5.4.1 Test Data

Automated Scenario

Origin Ontology - The ontology to be used as input for the search task

Input Question - The question to be used as input for the search task

Expected Answer - The expected answer to be obtained from the origin ontology for the input question

User-In-The-Loop Scenario

Origin Ontology - The ontology to be used as input for the search task

Input Question - The question to be used as input for the search task

Expected Answer - The expected answer to be obtained from the origin ontology for the input question

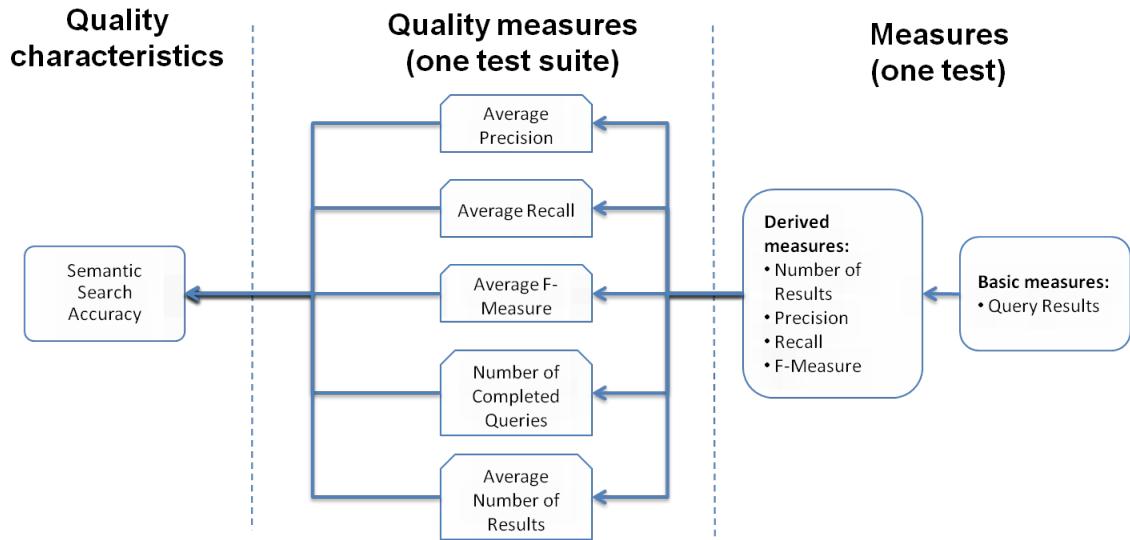


Figure 5.13: Automated scenario (I).

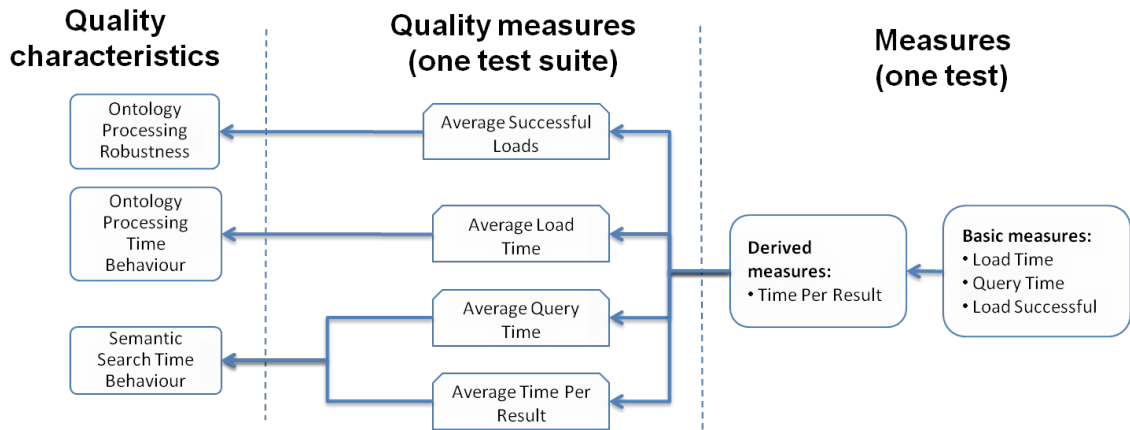


Figure 5.14: Automated scenario (II).

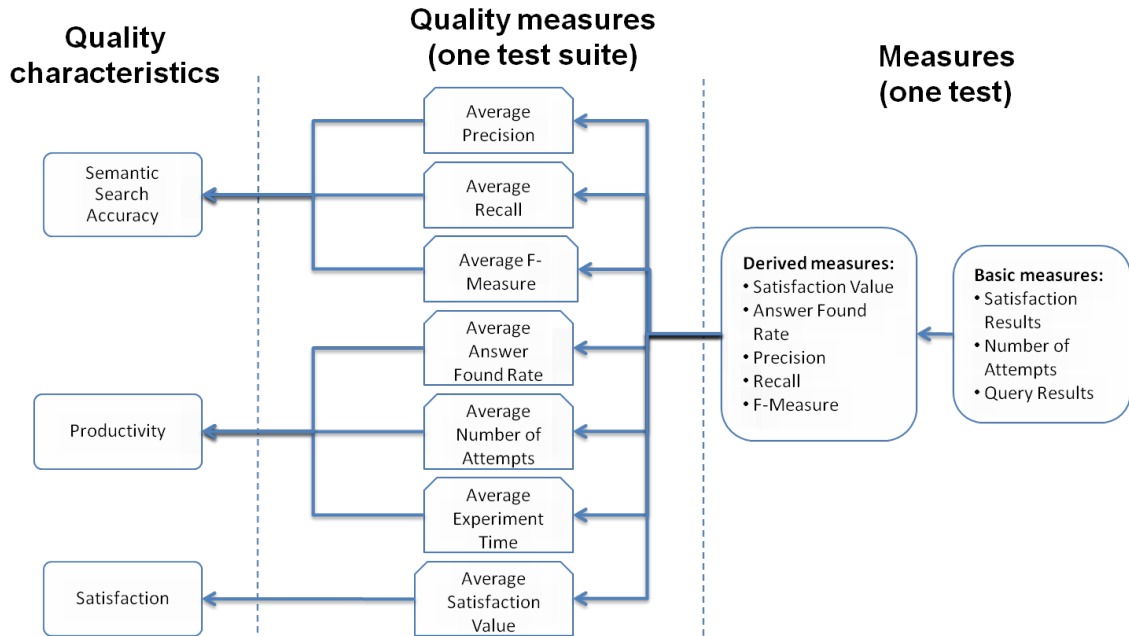


Figure 5.15: User-in-the-loop scenario (I).

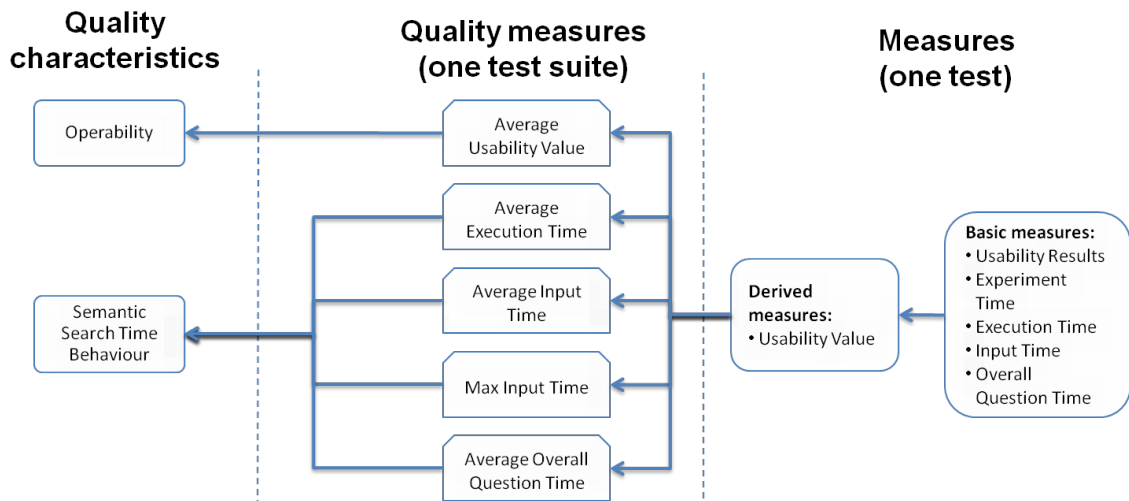


Figure 5.16: User-in-the-loop scenario (II).

Usability Questionnaire - Standardized Usability Test by Brooke [4] containing ten standardized questions to be given to the users

Satisfaction Questionnaire - User satisfaction questionnaire to be given to the users

Demographics Questionnaire - Questionnaire about the characteristics of a population to be given to the users

5.4.2 Basic Measures

Automated Scenario

Query Result - The results returned by the tool when executing the query

Load Successful - Whether the ontology has been successfully loaded. Possible values are *true* or *false*

Load Time - The amount of time needed to load the ontology

Query Time - The amount of time it takes for the query to execute

User-In-The-Loop Scenario

Usability Result - The results of the usability questionnaire filled by the user

Satisfaction Result - The results of the satisfaction questionnaire filled by the user

Number of Attempts - The number of times the user had to reformulate the query using the tool interface in order to obtain satisfactory answers

Query Result - The result of the executed query

Experiment Time - The amount of time needed for one user to complete the whole experiment

Input Time - The amount of time the user spends for formulating a query using the tool's interface

Execution time - The amount of time needed to execute the query

Overall Question Time - The amount of time needed for answering one question

5.4.3 Derived Measures

Automated Scenario

Number of Results - The number of results obtained for the query

query result

Time Per Result - The amount of time needed to obtain one result

$$\frac{\text{query time}}{\# \text{ results}}$$

Precision - The ratio between the relevant answers and all answers returned by the tool

$$\frac{\# \text{ relevant answers returned by the tool}}{\# \text{ answers returned by the tool}}$$

Recall - The ration between the relevant answers returned by the tool and answers in groundtruth

$$\frac{\# \text{ relevant answers returned by the tool}}{\# \text{ answers in groundtruth}}$$

F-Measure - The aggregation measure of precision and recall

$$\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 2$$

User-In-The-Loop Scenario

Usability Value - The value obtained from the usability questionnaire

Satisfaction Value - The value obtained from satisfaction questionnaire

Answer Found Rate - The ratio of finding the appropriate answer after a number of attempts and the user giving up after a number of attempts

$$\frac{\text{number of attempts}}{\# \text{ of attempts before giving up}}$$

Precision - The ratio between the relevant answers and all answers returned by the tool

$$\frac{\# \text{ relevant answers returned by the tool}}{\# \text{ answers returned by the tool}}$$

Recall - The ration between the relevant answers returned by the tool and answers in groundtruth

$$\frac{\# \text{ relevant answers returned by the tool}}{\# \text{ answers in groundtruth}}$$

F-Measure - The aggregation measure of precision and recall

$$\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 2$$

5.4.4 Quality Measures

Automated Scenario

Average Successful Loads - The ratio of successful ontology loads

$$\frac{\# \text{ tests where load successful} = \text{true}}{\# \text{ tests}} \times 100$$

Average Load Time - The average amount of time it takes for the ontology to load

$$\frac{\sum_n \text{load time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Query Time - The average amount of time it takes for a query to execute

$$\frac{\sum_n \text{query time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Number of Results - The average number of results obtained for the query

$$\frac{\sum_n \text{number of results of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Number of Completed Queries - The number of queries that produced expected answer

$$\# \text{ tests where expected answer} = \text{query result}$$

Average Time Per Result - The average amount of time needed to obtain one result

$$\frac{\sum_n \text{time per result of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Precision - The average precision

$$\frac{\sum_n \text{precision of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Recall - The average recall

$$\frac{\sum_n \text{recall of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average F-Measure - The average F-Measure

$$\frac{\sum_n \text{F-Measure of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

User-In-The-Loop Scenario

Average Experiment Time - The average time needed for the experiment to complete

$$\frac{\sum_n \text{experiment time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Usability Value - The average value of results obtained from usability questionnaire

$$\frac{\sum_n \text{usability value of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Satisfaction Value - The average value obtained from satisfaction questionnaire

$$\frac{\sum_n \text{satisfaction value of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Number of Attempts - The average number of attempts before the user is happy with the results

$$\frac{\sum_n \text{number of attempts of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Answer Found Rate - The ratio of finding the appropriate answer after a number of attempts and the user giving up after a number of attempts

$$\frac{\sum_n \text{answer found rate of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Execution Time - The average time needed to execute the question

$$\frac{\sum_n \text{execution time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Input Time - The average time needed for the query to be formulated

$$\frac{\sum_n \text{input time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Max Input Time - The maximum time needed for the query to be inputted

$$\max(\text{input time of the } n^{\text{th}} \text{ test})$$

Average Overall Question Time - The average amount of time needed for answering one question

$$\frac{\sum_n \text{overall question time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Precision - The average precision of the search tool

$$\frac{\sum_n \text{precision of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Recall - The average recall of the search tool

$$\frac{\sum_n \text{recall of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average F-Measure - The average F-Measure of the search tool

$$\frac{\sum_n \text{F-Measure of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

5.4.5 ISO 9126 Quality Characteristics

Functionality - The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions

Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision. *Accuracy* is a sub-characteristic of *Functionality*

Efficiency - The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions

Time Behaviour - The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions. *Time behaviour* is a sub-characteristic of *Efficiency*

Reliability - The capability of the software product to maintain a specified level of performance when used under specified conditions

Robustness - The ability of the software product to function correctly in the presence of invalid inputs or stressful environmental conditions. *Robustness* is a sub-characteristic of *Reliability*

Usability - The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions

Operability - The capability of the software product to enable the user to operate and control it. *Operability* is a sub-characteristic of *Usability*, and can be measured using:

- Average Usability Value

Productivity - The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use. *Productivity* is a characteristic of *quality in use*, and can be measured using

- Average Number of Attempts
- Average Answer Found Rate
- Average Experiment Time

Satisfaction - The degree to which users are satisfied in a specified context of use. *Satisfaction* is a characteristic of *quality in use*, and can be measured using:

- Average Satisfaction Value

5.4.6 Semantic Quality Characteristics

Semantic Search Accuracy - The capability of the software product to provide the right or agreed results with the needed degree of precision when performing the search task. *Semantic search accuracy* is a sub-characteristic of *Accuracy* and can be measured using

- Average Precision
- Average Recall
- Average F-measure
- Number of Completed Queries
- Average Number of Results

Ontology Processing Time Behaviour - The capability of the software product to provide appropriate response and processing times when working with ontologies. *Ontology Processing Time Behaviour* is a sub-characteristic of *Time Behaviour* and can be measured using:

- Average Load Time

Ontology Processing Robustness - The ability of the software product to process ontologies correctly in the presence of invalid inputs or stressful environmental conditions. *Ontology Processing Robustness* is a sub-characteristic of *Robustness* and it can be measured using:

- Average Successful Loads

Semantic Search Time Behaviour - The capability of the software product to provide appropriate response and processing times when performing search tasks. *Semantic Search Time Behaviour* is a sub-characteristic of *Time Behaviour* and can be measured using:

- Average Query Time
- Average Time Per Result
- Average Execution Time
- Average Input Time

- Max Input Time
- Average Overall Question Time

Figure 5.17 shows the part of the quality model internal and external characteristics obtained from the evaluation campaign for semantic search tools. Fig 5.18 shows the quality in use.

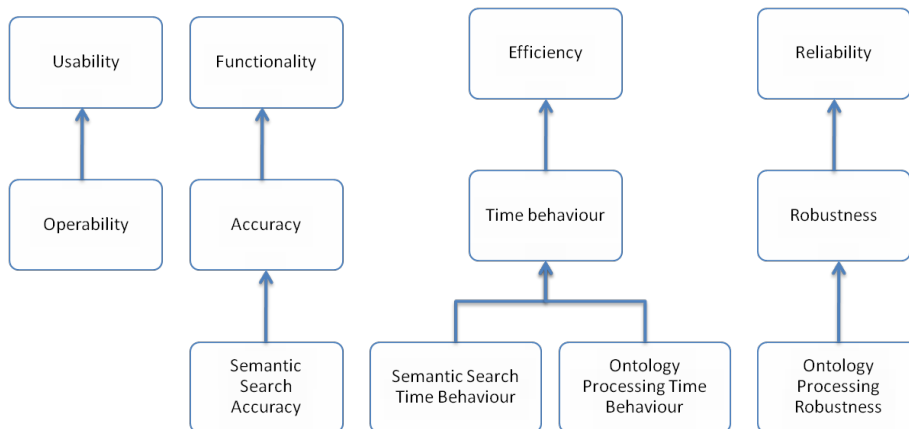


Figure 5.17: Internal and external quality characteristics of semantic search tools.



Figure 5.18: Quality in use of semantic search tools.

5.5 Semantic Web Service Tools

The evaluation campaign for semantic web services contained one evaluation scenario to evaluate discovery (Fig 5.19).

5.5.1 Test Data

Goal Document - The semantic web service document containing users' request description to be used in discovery test

Service Document - The semantic web service document containing service offer to be used in discovery test

Queries - The query to be used in discovery test

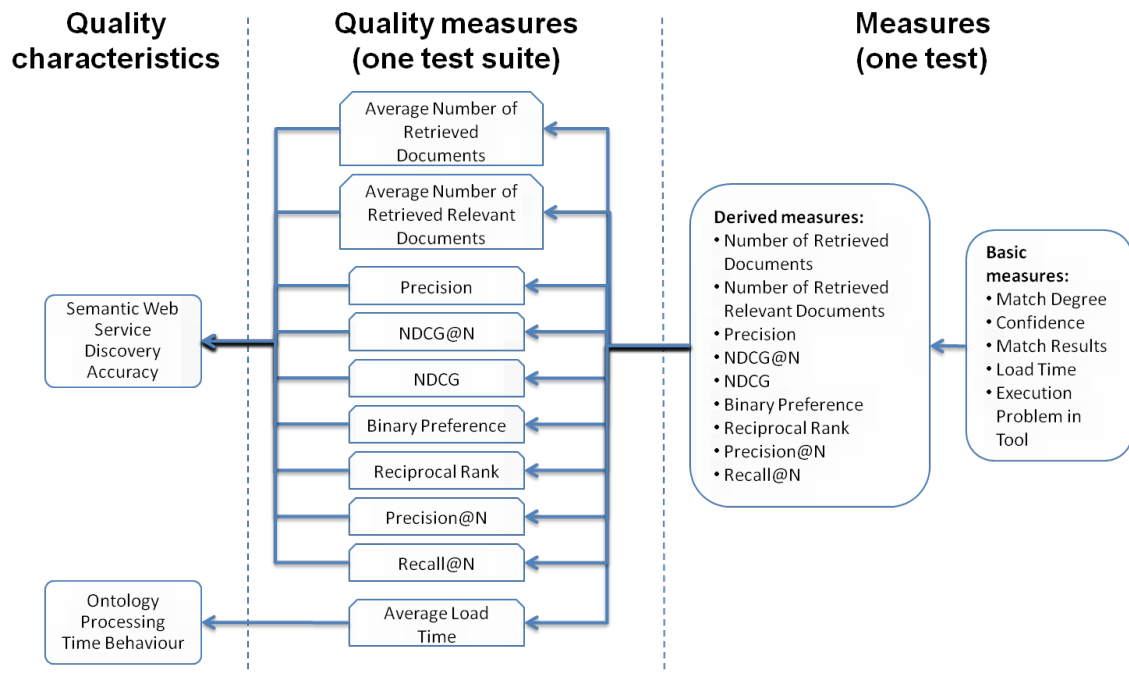


Figure 5.19: Semantic web service discovery scenario.

5.5.2 Basic Measures

Match Degree - Information about the match degree. Possible values are *Non*, *Plugin*, *Match*, *Exact*, *Subsumption*

Confidence - The confidence value of the produced match

Match Results - The results that are produced by the tool in the process of discovery

Load Time - The amount of time needed to load the query

Execution Problem in Tool - Whether there were any execution problems in the tool while performing the discovery process. Possible values are *true*, and *false*

5.5.3 Derived Measures

Number of Retrieved Documents - The number of retrieved documents

retrieved documents

Number of Retrieved Relevant Documents - Number of retrieved documents that were judged relevant

retrieved documents where (match degree = match \vee exact)

Precision - Precision of the semantic web service discovery

$$\frac{\# \text{ relevant documents retrieved}}{\# \text{ retrieved documents}}$$

Normalized Discounted Cumulative Gain (NDCG) - A normalized measure of the effectiveness of the search algorithm according to the relevance of a search result

$$rel_1 + \sum_{i=2} \frac{rel_i}{\log_2 i}$$

Normalized Discounted Cumulative Gain @N - Normalized discounted cumulative gain at a given number of documents retrieved

$$N \frac{\sum_i (2^{r(i)} - 1)}{\log(1 + i)}$$

Binary Preference - Measures the relevant documents retrieved when dealing with incomplete information

$$\frac{1}{R} \sum_r 1 - \frac{|nrankedgreaterthanr|}{R}$$

Reciprocal Rank - Returns the reciprocal of the rank of the first relevant document retrieved, or zero if no relevant documents were retrieved

$$\frac{1}{\text{rank of the first relevant document retrieved}}$$

Precision@N - The precision of the retrieval at a given number of documents retrieved

$$\frac{\# \text{ relevant documents retrieved}}{\# \text{ retrieved documents}}$$

Recall@N - The recall of the retrieval at a given number of documents retrieved

$$\frac{\# \text{ relevant documents retrieved}}{\# \text{ relevant documents in groundtruth}}$$

5.5.4 Quality Measures

Average Number Retrieved Documents - The average number of retrieved documents for all tests

$$\frac{\sum_n \text{average number of retrieved documents of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Relevant Documents Retrieved - The average number of retrieved documents that were judged relevant

$$\frac{\sum_n \text{number of retrieved relevant documents of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Precision - The average precision of semantic web service discovery

$$\frac{\sum_n \text{precision of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average NDCG@N - The average NDCG@N for all tests

$$\frac{\sum_n \text{NDCG@N of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average NDCG - The average NDCG for all tests

$$\frac{\sum_n \text{NDCG of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Binary Preference - The average binary preference for all tests

$$\frac{\sum_n \text{binary preference of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Reciprocal Rank - The average reciprocal rank for all tests

$$\frac{\sum_n \text{reciprocal rank of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Precision@N - The average Precision@N for all tests

$$\frac{\sum_n \text{precision@N of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Recall@N - The average Recall@N for all tests

$$\frac{\sum_n \text{recall@N of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

Average Load Time - The average amount of time needed to load the service

$$\frac{\sum_n \text{load time of the } n^{\text{th}} \text{ test}}{\# \text{ tests}}$$

5.5.5 ISO 9126 Quality Characteristics

Functionality - The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions

Accuracy - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision. *Accuracy* is a sub-characteristic of *Functionality*

Efficiency - The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions

Time Behaviour - The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions. *Time behaviour* is a sub-characteristic of *Efficiency*

5.5.6 Semantic Quality Characteristics

Ontology Processing Time Behaviour - The capability of the software product to provide appropriate response and processing times when working with ontologies.

Ontology Processing Time Behaviour is a sub-characteristic of *Time Behaviour* and can be measured using:

- Average Load time

Semantic Web Service Discovery Accuracy - The accuracy of the process of finding services that can be used to fulfil a given requirement from the service re-

quester. *Semantic web service discovery accuracy* is a sub-characteristic of *Accuracy* and can be measured using:

- Average Number Retrieved Documents
- Average Relevant Documents Retrieved
- Average Precision
- Average NDCG@N
- Average NDCG
- Average Binary Preference
- Average Reciprocal Rank
- Average Precision@N
- Average Recall@N
- Average Load Time

Figure 5.20 shows the part of the quality model obtained from the evaluation campaign for semantic web services.

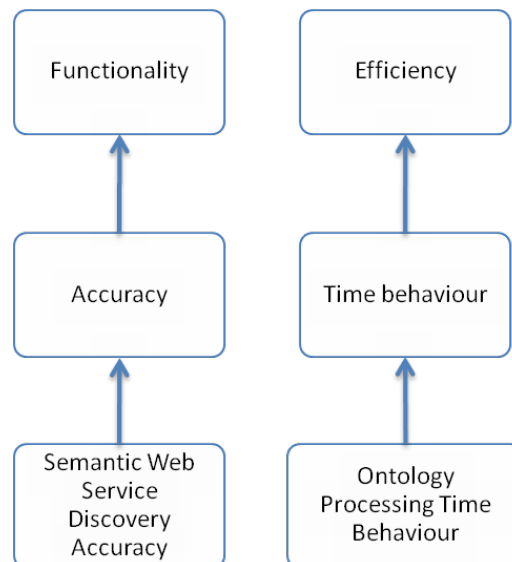


Figure 5.20: Quality characteristics of semantic web service tools.

Chapter 6

Application of the Semantic Technology Quality Model

In this chapter, we present two web applications that are based on the quality model described in Chapter 5: for visualization of evaluation results, and for the recommendation of semantic technologies. These two applications use the results that are produced in the conformance and interoperability evaluation scenarios for the ontology engineering tools defined in the SEALS project.

Section 6.1 describes the architecture, which is common for both applications and Section 6.2 describes the open-source libraries that were used in their implementation, while Section 6.3 gives instruction for the installation. Section 6.4 presents the application for the visualization of the evaluation results. Finally, Section 6.5 presents the application for semantic technology recommendation according to users' needs.

6.1 Architecture

The architecture of the two applications describes five separate components (Fig. 6.1):

1. Quality model - quality model for semantic technologies that helps us to put evaluations of semantic technologies under a common ground and to identify all the elements in the evaluation results on which the two applications are based.
2. User Interface - the user interface is the bridge between the user and the application; the user interacts with the application through the user interface.
3. Application logic - the application logic is the key part of the system and is in charge of processing user' requests and present the results to them.

4. Ontologies - These include the SEALS ontologies, which are used to describe evaluation results; those ontologies used to represent basic and derived measures, described in [15]; and a *QualityModel* ontology, which extends the SEALS ontologies and is described in Appendix A.
5. Data storage - a set of data that includes test data used in the process of evaluation, the results of the evaluation, and quality measures results obtained for the available tools. Those data are stored in the RDF form and used by the applications.

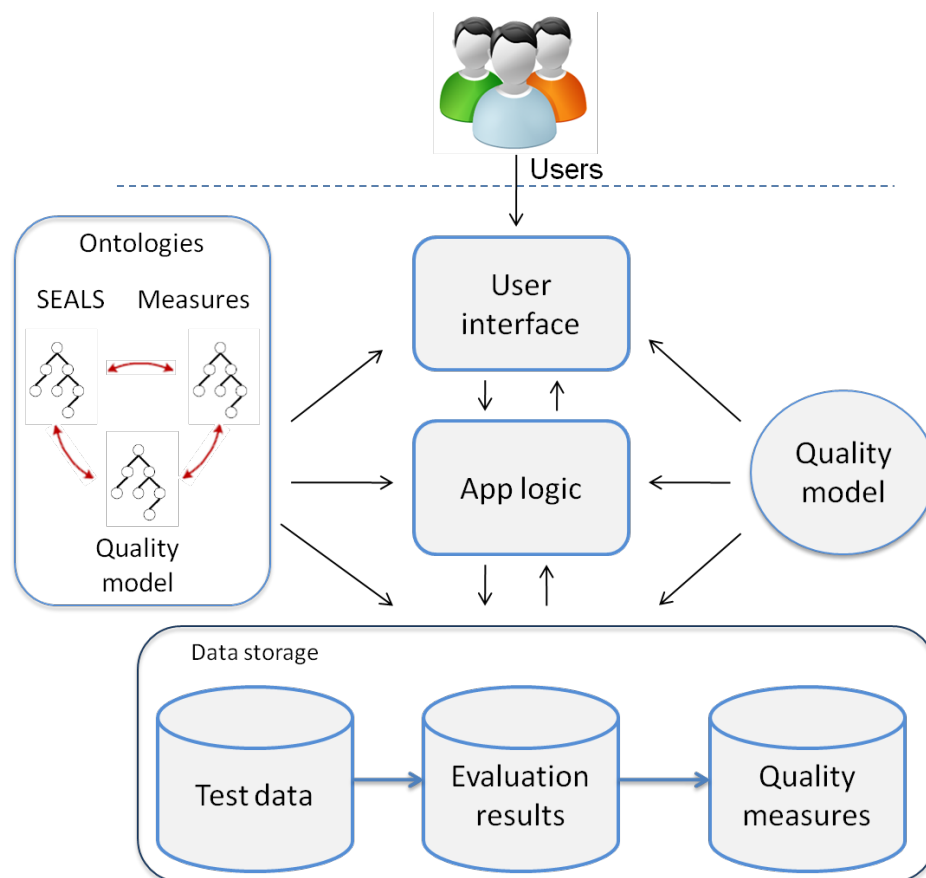


Figure 6.1: Application architecture.

6.2 Dependencies

In the implementation of the visualization and recommendation applications, Maven¹ is used for project and dependency management. It has become practically a standard in the software development process and provides easy maintenance and reusability.

Tapestry² is an open-source framework for building dynamic and robust web applications in Java. It complements and builds upon the standard Java Servlet API, and so it works in any servlet container or application server. Developing Tapestry applications involves creating HTML templates using plain HTML and combining the templates with small amounts of Java code. In Tapestry, applications are created in terms of objects and the methods and properties of those objects – and specifically not in terms of URLs and query parameters.

Spring³ is an open-source application framework for the Java platform, and it is used as Inversion of Control container. The container is responsible for managing object lifecycles. Objects are obtained by means of Dependency Injection where container provides objects by name and which allows for high flexibility in the implementation.

JenaBean⁴ is an annotation-based framework built on top of the Jena⁵ RDF/OWL API. Using JenaBean, the object-oriented Java model that is in the core of both applications is bound to the ontologies that are used for representing evaluation results. That way, RDF data can be easily managed using objects in Java.

6.3 Instalation

Maven provides an easy way for collecting all dependencies, classes and resources of the web application and packaging them into a web application archive. Using the console, once in the root folder of the application, the command

```
mvn package
```

will create a WAR file in the *target* folder of the project. The created WAR file can be used in any web container (e.g., Tomcat or Jetty).

Evaluation results that are used are stored in the RDF files inside the WAR file, under *WEB-INF/classes/results/* location.

¹<http://maven.apache.org/>

²<http://tapestry.apache.org/>

³<http://www.springsource.org/>

⁴<http://code.google.com/p/jenabean/>

⁵<http://jena.sourceforge.net/>

In order to update the results or insert additional data, it is necessary just to put new RDF files in the appropriate folder (or to replace existing ones), and restart the application on the server.

The source code for the applications can be found at <https://svn.seals-project.eu/seals-dev/oet/seals-visualization/trunk>, and at <https://svn.seals-project.eu/seals-dev/oet/seals-recommendation/trunk>, for the visualization and recommendation applications, respectively.

6.4 Evaluation Results Visualization

In this section, we present the web application for the dynamic visualization of evaluation results that allows to browse the results of the different tools and compare them. It is based on the semantic technologies quality model, in a way that it visualizes any evaluation results that are obtained following the quality model and described according to the provided ontologies.

6.4.1 Requirements

In order to develop, implement, and use the visualization application, the following requirements are needed:

- *Req01.* The semantic technology quality model which will serve as a basis to identify entities in the results to be visualized.
- *Req02.* Evaluation results, which have to be obtained and interpreted with respect to the quality model.
- *Req03.* For describing and storing the evaluation results, ontologies are needed, which provide properties and classes for describing evaluation results. Such ontologies are already developed in the SEALS project [15].

6.4.2 Use

This section describes the usage of the visualization application, from the user perspective, for the conformance and interoperability scenarios for ontology engineering tools.

Conformance Results Visualization

In particular, the application contains the following functionalities for the conformance scenario to browse for:

- Results of a tool according to a certain ontology language (test suite execution).
- Results of a tool according to ontology language features.
- Results of several tools according to a certain ontology language.
- Statistics of results for a particular tool according to a certain ontology language.
- Details of the results of a test execution.

Next, we present an overview of the current version of this web application.

The homepage of the application shows an option for a user to choose which results to browse (i.e., conformance or interoperability results). After selecting the desired scenario, the page listing the executions of all the tools with all the test suites, sorted by tool, will appear (figure 6.2).


Tool	Name 	Version	Date
Jena Version1	OWLDLImportTestSuite	1.3	07.10.10 10:41:51
Jena Version1	OWLLiteImportTestSuite	1	07.10.10 10:41:24
Jena Version1	RDFSImportTestSuite	1	07.10.10 10:40:48
Jena Version1	OWLFullImportTestSuite	1	23.02.11 13:30:34
NeOn Toolkit Version1	OWLLiteImportTestSuite	1	06.10.10 15:58:37
NeOn Toolkit Version1	OWLDLImportTestSuite	1.3	06.10.10 16:08:08
NeOn Toolkit Version1	RDFSImportTestSuite	1	06.10.10 16:04:26
NeOn Toolkit Version1	OWLFullImportTestSuite	1	23.02.11 13:27:51

Figure 6.2: Part of the table showing available conformance executions.

By clicking on a certain execution (which are distinguished by the execution date), all the results for a particular tool according to a certain test suite are shown. For each test, results include the name of the ontology file that is used as input in the conformance test, conformance and execution status, structural and semantical equivalence, as well as information that has been added or lost during the test execution. Figure 6.3 shows an example of conformance results. Furthermore, results can be filtered according to specific criteria (e.g., if a user wants to see only test executions where there is no addition of information, or where imported and exported ontologies are not semantically equivalent).

In order to have better visibility and due to a space reasons on web pages, all additions and losses of information are initially hidden. Therefore, a user can click on

Test Id	Results	Equivalence	Information Added	Information Lost
DLTestOWLDLImportTest001 ontology001.owl	Conformance= SAME Execution= OK	Semantic= true Structural= true		
DLTestOWLDLImportTest002 ontology002.owl	Conformance= SAME Execution= OK	Semantic= true Structural= true		
DLTestOWLDLImportTest003 ontology003.owl	Conformance= SAME Execution= OK	Semantic= true Structural= true		
DLTestOWLDLImportTest004 ontology004.owl	Conformance= SAME Execution= OK	Semantic= true Structural= true		

Figure 6.3: Part of the table showing the conformance results of a particular execution.

Test Id	Results	Equivalence	Information Added	Information Lost
DLTestOWLDLImportTest403 ontology403.owl	Conformance= SAME Execution= OK	Semantic= true Structural= false	<input type="button" value="+"/> ClassAssertion(owl:Datatype _:#genid1)	<input type="button" value="+"/> ClassAssertion(owl:Datatype _:#genid-A0)
DLTestOWLDLImportTest404 ontology404.owl	Conformance= SAME Execution= OK	Semantic= true Structural= false	<input type="button" value="+"/> ClassAssertion(owl:Datatype _:#genid1)	<input type="button" value="+"/> ClassAssertion(owl:Datatype _:#genid-A0)
DLTestOWLDLImportTest405 ontology405.owl	Conformance= SAME Execution= OK	Semantic= true Structural= false	<input type="button" value="+"/>	<input type="button" value="+"/>

Figure 6.4: Information added and lost in a test execution.

a small button that shows information added or information lost, if such information exist (figure 6.4).

Through the *execution* field in the shown results, a user can get an insight into the robustness of a tool. If there were no problems in a test execution, the field shows the *OK* value. Otherwise, the *FAIL* value indicates that there were some execution problems, and it is then shown in red colour.

Results for a particular tool according to a particular test suite can be also browsed regarding the ontology language features. In that scenario, test execution results are classified according to each ontology language feature that is covered in a test (figure 6.5).

[+] Named individual

[+] Object property

[+] Ontology annotation

Test Id	Results ⊖	Equivalence ⊖	Information Added ⊖	Information Lost ⊖
DLTestOWLDDLImportTest558 ontology558.owl	Conformance= N.E.	Semantic= false		
	Execution= P.E.	Structural= false		
DLTestOWLDDLImportTest559 ontology559.owl	Conformance= SAME	Semantic= true		
	Execution= OK	Structural= true		
DLTestOWLDDLImportTest560 ontology560.owl	Conformance= SAME	Semantic= true		
	Execution= OK	Structural= true		
DLTestOWLDDLImportTest561 ontology561.owl	Conformance= SAME	Semantic= true		
	Execution= OK	Structural= true		

Figure 6.5: Test results according to ontology language features.

Furthermore, a user can observe the results according to a certain ontology language, that are obtained for several tools (figure 6.6).

Additionally, following the link of the ontology file name that is used in a test execution, it is possible to get an insight into the ontology files that have been used (imported and exported ontologies), as shown in figure 6.7. This page also separately shows information added and information lost, so that it is easier for users to observe it.

At the end, the statistic analysis is available for a particular tool, and according to a particular test suite. It shows sums and percentages for all the results in a single test execution (e.g., percentage of tests where there is addition of information, or where there was a problem during test execution). Also, the page shows statistics for

OWLDLImportTestSuite Version1.3

Category	Number	Protege4 Version1	OWLAPI Version1	NeOnToolkit Version1	Sesame Version1	Jena Version1	ProtegeOWL Version1
Same Ontologies	561	557	557	555	558	559	527
Different Ontologies	561	2	2	2	1	0	32
Execution Problem in Tool	561	0	0	3	0	0	0
Execution Problem in Platform	561	2	2	1	2	2	2
TOTAL	561	561	561	561	561	561	561

Figure 6.6: Results for tools according to OWL DL ontology language.

Original ontology	Exported ontology
<pre><rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" > <rdf:Description rdf:about="http://www.seals-project.eu/Conformance/ontology001"> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl:Ontology"/> </rdf:Description> <rdf:Description rdf:about="http://www.seals-project.eu/Conformance/ontology001#class01"> <rdf:label xml:lang="en">"class01"</rdfs:label> <rdf:type rdf:resource="http://www.w3.org/2002/07/owl:Class"/> </rdf:Description> </rdf:RDF></pre>	<pre><rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:owl="http://www.w3.org/2002/07/owl#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" > <owl:Ontology rdf:about="http://www.seals-project.eu/Conformance/ontology001"/> <owl:Class rdf:about="http://www.seals-project.eu/Conformance/ontology001#class01"> <rdf:label xml:lang="en">"class01"</rdfs:label> </owl:Class> </rdf:RDF></pre>

Figure 6.7: Ontologies used in one test execution.

every ontology language feature, i.e., the number of tests where a feature is included, as well as the number of tests that include a feature and in which the conformance is *SAME*. Figure 6.8 shows an example of these statistic and a part of the statistics for ontology language features.

Category	No	%	Information Added/Lost	No	%	Category	No	Same
Same ontologies	559	99	Only information added	0	0	Annotation property	40	40
Different ontologies	0	0	Only information lost	11	1	Annotation with URI	48	48
Execution problem in tool	0	0	Information added and lost	30	5	Annotation with individual	53	53
Execution problem in platform	2	0	No information added or lost	520	92	Annotation with literal	354	353
TOTAL	561	100%	TOTAL	561	100%	Anonymous individual	26	25
						Cardinality restriction	31	31

Figure 6.8: Statistics for a particular tool according to a particular test suite.

Interoperability Results Visualization

In particular, the application contains the following functionalities for the interoperability scenario to browse for:

- Results of two tools according to a certain ontology language (test suite execution).
- Results of two tools according to ontology language features.
- Results of several tools according to a certain ontology language.
- Statistics of results for two particular tools according to a certain ontology language.
- Details of the results of a test execution.

Next, we present an overview of the current version of this web application.

The homepage of the application shows an option for a user to choose which results to browse (i.e., conformance or interoperability results). After selecting the desired scenario, the page listing all the executions of every tool with every test suite, ordered by tool, will appear (figure 6.9).


Origin Tool	Destination Tool	Name 	Version	Date
Jena Version1	Jena Version1	OWLFullImportTestSuite	1	23.02.11 15:30:45
Jena Version1	Jena Version1	OWLDLImportTestSuite	1.3	07.10.10 11:18:29
Jena Version1	Jena Version1	OWLLiteImportTestSuite	1	07.10.10 11:18:20
Jena Version1	Jena Version1	RDFSImportTestSuite	1	07.10.10 11:18:14
Jena Version1	NeOn Toolkit Version1	RDFSImportTestSuite	1	07.10.10 11:20:45
Jena Version1	NeOn Toolkit Version1	OWLFullImportTestSuite	1	23.02.11 15:31:16
Jena Version1	NeOn Toolkit Version1	OWLLiteImportTestSuite	1	07.10.10 11:22:11
Jena Version1	NeOn Toolkit Version1	OWLDLImportTestSuite	1.3	07.10.10 11:23:35

Figure 6.9: Part of the table showing available interoperability results.

By clicking on a certain execution (which are distinguished by the execution date), all the results for a particular pair of tools according to a certain test suite are shown. Each test result includes the name of the ontology file that is used as the input in the interoperability test, interoperability and execution status, structural and semantical equivalence for both steps, as well as information that has been added or lost during the test execution. Figure 6.10 shows an example of the interoperability results. Furthermore, results can be filtered according to specific criteria (e.g., if a user wants to see only test executions where there is no addition of information in either of the steps, or where the imported and exported ontologies are not semantically equivalent).

In order to accomplish better visibility and due to space reasons on web pages, all the additions and losses of information are initially hidden. Therefore, a user can

Test Id	Final Results	Final Equivalence	Final Information Added	Final Information Lost	Step1 Results	Step1 Equivalence
DLTestOWLDLImportTest001_ontology001.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=true			Interoperability=SAME Execution=OK	Semantic=true Structural=true
DLTestOWLDLImportTest002_ontology002.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=true			Interoperability=SAME Execution=OK	Semantic=true Structural=true
DLTestOWLDLImportTest003_ontology003.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=true			Interoperability=SAME Execution=OK	Semantic=true Structural=true
DLTestOWLDLImportTest004_ontology004.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=true			Interoperability=SAME Execution=OK	Semantic=true Structural=true

Figure 6.10: Part of the table showing interoperability results for two particular tools according to a particular test suite.

click on a small button that shows the information added or the information lost, if such information exist (figure 6.11).

DLTestOWLDLImportTest406_ontology406.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=false	<input type="button" value="+"/> ClassAssertion(owl:Datatype _:#genid1)	<input type="button" value="+"/> ClassAssertion(owl:Datatype _:#genid-A2)
DLTestOWLDLImportTest407_ontology407.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=false	<input type="button" value="+"/>	<input type="button" value="+"/>
DLTestOWLDLImportTest408_ontology408.owl	Interoperability=SAME Execution=OK	Semantic=true Structural=false	<input type="button" value="+"/>	<input type="button" value="+"/>

Figure 6.11: Information added and lost in a test execution.

Through the *execution* field in the final results column, a user can get an insight into the robustness of a tool. If there were no problems in a test execution, the field takes the *OK* value. Otherwise, the *FAIL* value indicates that there were some execution problems, and it is then shown in red colour. Also, execution results for both steps in the interoperability test are shown.

Results for a particular tool according to a particular test suite can be also browsed regarding the ontology language features. In that scenario, test execution results are classified according to each ontology language feature that is covered in a test (figure 6.12).

Furthermore, a user can observe the results according to a certain test suite, which are obtained for several tools (figure 6.13).

Additionally, following the link of the ontology file name that is used in a test execution, it is possible to get an insight into the ontology files that have been used (imported, intermediate, and final ontologies), as shown in figure 6.14. This page also separately shows information added and information lost, so that it is easier for users to observe it.

At the end, a statistic analysis is available for the two tools in the interoperability execution, and according to a particular test suite. It shows sums and percentages

for all the results in a single test execution (e.g., the percentage of tests where there is addition of information, or where there was a problem during test execution). Also, the page shows statistics for every ontology language feature, i.e., the number of tests where a feature is included, as well as the number of tests that include a feature and in which the conformance is *SAME*. Figure 6.15 shows an example of the statistics and a part of the statistics for ontology language features.

Category	No	%	Information Added/Lost	No	%	Category \ominus	No \ominus	Same \ominus
Same ontologies	559	99	Only information added	0	0	Annotation with literal	354	353
Different ontologies	0	0	Only information lost	11	1	Class union	26	26
Execution problem in tool	0	0	Information added and lost	30	5	Class disjointness	36	36
Execution problem in platform	2	0	No information added or lost	520	92	Classes related by a property	110	110
TOTAL	561	100%	TOTAL	561	100%	Named class	359	358
						Named individual	146	146

Figure 6.15: Statistics for two particular tools according to a particular test suite.

6.5 Semantic Technology Recommendation

As the software started to become an important part for business in many companies, and as the number of available software solutions started to grow, the problem of selecting the appropriate software system emerged. The process of software selection has become a very complex task due to the availability of large number of software products, but also because of the improvements in information technology and incompatibilities between hardware and software systems [29].

In the informatics era, the success of business depends on the availability of appropriate software systems [31] and the selection of an inappropriate system can lead to a significant financial loss and disruption of the project [19]. Therefore, when dealing with the number of available software solutions, the selection of the appropriate software system is of crucial importance for the user.

This is also the case when it comes to semantic technologies. A number of tools exist, but on the other hand it is very difficult for users to find the tool that best suits their needs.

Various authors have proposed selection systems for different types of software that rely on different approaches, such as selection of software components [30, 31] (Atana, AHP) or software packages selection [22] (HKBS); and there are some examples of systems that rely on less known and used methods developed by the authors of those systems, such as in COTS selection [25], open source software [28], simulation software [19] and discrete event simulation software [35].

Weighting Scoring Method The idea of the Weighted Scoring Method (WSM) is to aggregate the values of all the characteristics to a single value by using weights for every characteristic. In order to achieve this, all values for characteristics have to be in a common scope and therefore all values are normalized to the scale ranging from 0 to 10 [31]. After this step, the aggregation is performed by multiplying each characteristic value with the characteristic's weight, and summing the results for every characteristic of the candidate product.

Some authors discuss the weaknesses of the mentioned approach [22, 25]. For example, it is pointed out that it can be very difficult to assign the weights if the number of characteristics is high, and that also it is difficult to apply the method for Multiple Criteria Decision Making (MCDM) problem. Furthermore, in the case when some characteristics are estimated using an ordinal scale, the final values for each candidate product will represent only the relative value and the differences between candidates will not be represented objectively.

The Analytic Hierarchy Process The Analytic Hierarchy Process (AHP) is a flexible approach for decision making in complex multicriteria problems [30], in which the selection criteria (characteristics of a software product) and possible candidates are represented in a hierarchical way.

The method first requires the comparison of all criteria that are on a same hierarchy level. In order to obtain weights, criteria are compared to each other by pair wise comparisons using a scale from 1 to 9 [31]. It is important just to note that two compared criteria have reciprocal values, which means that if criterion A has value x , criterion B will have value $1/x$.

After the comparison matrix has been calculated, the value of each criterion is multiplied with the appropriate comparison weight and those results then ascend up the tree to calculate the final value for every alternative [30].

The AHP helps the decision making problem to be more simplified and is a good approach for handling both qualitative and quantitative multicriteria problems [22]. On the other hand, AHP is time consuming, and in the case when a new criterion is introduced or an existing one is deleted, the re-estimation of all candidates is needed.

Hybrid Knowledge Based System Hybrid knowledge based system (HKBS) is an approach that is applied in software selection process and that employs rule based reasoning and case based reasoning [22].

A rule based reasoning component stores information about evaluation criteria and evaluation results. It is used to guide users to specify his requirements that he wants to consider in software selection process. User requirements are collected in the form of feature and feature value [22] and are later submitted to the case based reasoning part of the system.

A case based reasoning component is used in order to compare user requirements with the available candidates and determine how well each candidate software product matches user needs. Furthermore, it is possible to calculate the similarity level of each candidate in order to rank all candidates that satisfy user needs.

In their comparative study of WSM, AHP, and HKBS [23], Jadhav and Sonar showed that HKBS approach is better regarding the computational efficiency, knowledge reuse, consistency of the results, and flexibility in specifying user requirements. This means that e.g., if the number of candidate software or evaluation criteria changes, unlike with AHP, the HKBS approach does not require any additional efforts or calculations in order to adapt to changes.

The Atana Approach Atana [31] is an approach that starts with a user entering his required criteria and desired limitations. Based on that information, in the calculation phase the set of solutions is found that satisfies all given constraints.

The Atana Approach, unlike AHP and WSM, does not aggregate multiple criteria values into a single one, but rather leaves all values separated. Furthermore, the authors argue that WSM and AHP will produce just one solution as a result, which forces the user to accept it, and point out the fact that providing a priori information in WSM or AHP, i.e., weights for criteria, presents a drawback and a problem for end users.

In the implementation of the semantic technologies recommendation application we have decided to use HKBS approach. It gives to users the possibility for specifying the importance for each requirement, and on the other hand it is flexible and easily extensible.

6.5.1 Requirements

When selecting a software product, user needs and product characteristics are the most important concerns. Based upon product characteristics, available candidates are evaluated in an evaluation process, and the results of the evaluation are used as inputs to the selection process [20]. It is necessary that all the candidates are evaluated under a common framework, and therefore a proper software quality model can significantly reduce the amount of time and effort needed for the selection process. Fig 6.16 depicts the software selection process.

In order to develop, implement, and use the recommendation system for semantic technologies, the following requirements are needed:

- *Req01.* The semantic technology quality model will serve as a basis for specifying user criteria, providing the set of already defined quality measures.

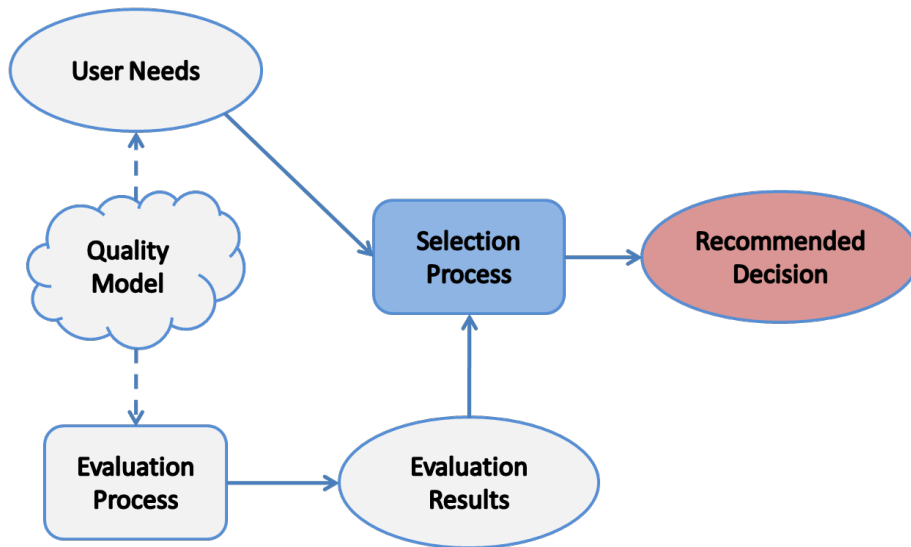


Figure 6.16: Software selection process.

- *Req02*. Each candidate tool in the recommendation process has to be evaluated with respect to the quality model, providing evaluation results.
- *Req03*. For describing and storing the quality measures, an ontology is needed, which provides properties and classes for describing the quality measures and results obtained in the evaluation process.
- *Req04*. With respect to a set of predefined characteristics of semantic technologies to be recommended, the user has to provide to the system input quality requirements upon which the recommendation algorithm is run. Those requirements should be gathered using a user interface.
- *Req05*. A recommendation algorithm that takes user quality requirements and evaluation results as inputs has to be implemented.

6.5.2 Use

The first version of the semantic technology recommendation system works only for each type of the tool separately. Therefore, since there are different types of tools with different purposes and characteristics, the homepage of the system shows the menu for selecting the type of tool that the user wants to use (Figure 6.17).

After selecting a tool type, a page for specifying quality requirements will appear. Requirements are consistent with the quality measures described in the quality model. The user can choose which ones he wants to take into account and for every

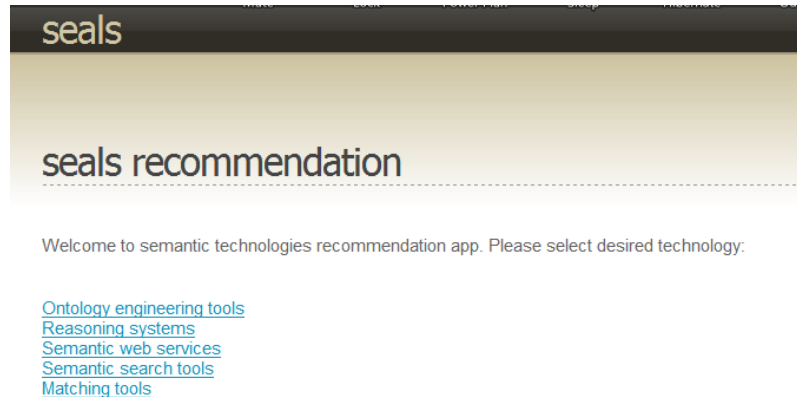


Figure 6.17: Homepage of the semantic technologies recommendation system.

desired quality measure it is possible to specify its importance in form of a weight, and a desired level (threshold) (Figure 6.18). Also, by pointing the mouse on an *info* icon, additional information about the requirement is shown.

Welcome to the recommendation of **Ontology Engineering Tools**.

You can choose which of the available criteria you want to include into the recommendation process. You can specify the importance of each criterion (Weight) and the desired level (Threshold) with possible values range 1-100. Field *Threshold* denotes desired criteria value.

Measure	Weight	Threshold	
Ontology information change	<input type="text" value="45"/>	<input type="text" value="40"/>	
Ontology language component coverage	<input type="text" value="35"/>	<input type="text" value="65"/>	
Interchange information change	<input type="text"/>	<input type="text"/>	
Ontology language component interoperability coverage	<input type="text" value="45"/>	<input type="text" value="60"/>	
Execution errors	<input type="text"/>	<input type="text"/>	

The ratio of ontology components that are shared by a tool internal model and an ontology language model. The result is in range 0-100, where higher value denotes better result.

Figure 6.18: Requirements specification form.

Once the requirements are filled and the form is submitted, the recommendation results will be shown, ordered by how well each tool meets user requirements (Figure 6.19). For each tool, overall similarity with specified requirements is shown, as well as similarities according to each requirement separately. Also, a separate table shows values for every quality measure that was considered in the recommendation process, according to each tool. Measures that do not satisfy given requirements are shown in red color.

Furthermore, by clicking on a particular tool, a page with the tool description appears, listing all the quality measures and values obtained (Figure 6.20).

Tool	Similarity	Ontology information change	Ontology language component coverage	Ontology language component interoperability coverage
Sesame	84	100	100	58
Jena	84	100	100	58
Protege OWL	71	85	72	58
OWL API	61	56	72	58
NeOn Toolkit	61	57	72	58
Protege 4	61	56	72	58

Tool	Ontology information change	Ontology language component coverage	Ontology language component interoperability coverage
Sesame	4	93	35
Jena	5	93	35
Protege OWL	47	47	35
OWL API	71	47	35
NeOn Toolkit	70	47	35
Protege 4	71	47	35

Figure 6.19: Results for the recommendation.

Evaluation results obtained for **Sesame**, Version 1

[back](#)

Measure Name	Value ↕
Ontology language component interoperability coverage	35
Execution errors	0
Interchange information change	4
Ontology language component coverage	93
Ontology information change	4

Figure 6.20: Results for a particular tool.

Chapter 7

Conclusions and Future Work

This thesis aims to provide basis for an evaluation framework for semantic technologies. In summary, the contributions of this thesis are the following:

1. A new method for extending software quality models based on a bottom-up approach, which can be useful in cases where there are already plenty of existing evaluations.
2. A quality model for semantic technologies, which extends the ISO 9126 software quality model, and can be used for the evaluation and comparison of semantic technologies.
3. Two web applications that allow the visualization of evaluation results and provide semantic technology recommendations. Both applications are highly flexible and can be used with any evaluation results that are based on the presented quality model.

The bottom-up method for extending software quality models that is presented in this thesis can be very useful in the cases when there are plenty of evaluations to extract the quality model. Furthermore, the method can be applied together with some other top-down based methods in order to get a more complete quality model.

The quality model for semantic technologies presented in this thesis is very flexible and new elements can be easily introduced and classified. Also, the quality model is complete regarding the current state in the semantic technology evaluation, as presented in Section 2.1.

The benefits of such quality model are broad. By referring to it, all evaluations can be put under a common ground thus enabling the comparison of their results. Besides, it gives a detailed specification and terminology of semantic technology quality, and provides precise guidance for its measurement.

This quality model covers five types of semantic tools, and in analyzed conferences most of the papers also cover the same tool types. Other types of tools (e.g., annotation tools) are not currently covered, which will be addressed in future work.

Future work will also consist in the validation of the quality model, in which we plan to extend our analysis to other conferences, as well as to relevant journals. We plan to apply one of the methods based on the top-down approach in order to extend our quality model.

The semantic technology recommendation system currently provides recommendations only for ontology engineering tools. In the future, when all the evaluation results become available, we plan to extend the application and cover all the types of tools that are being evaluated in SEALS project. One benefit of this system is that all evaluation results that are based on our quality model can be used for the recommendation application without any modifications.

Furthermore, when starting a selection process, it is required to specifically choose the type of tools to be recommended, which will be changed in the next version, allowing users to directly proceed to the requirements specification.

The current versions of the two web applications do not use results from the SEALS repositories. Instead, evaluation results are incorporated into the data storage system of these applications. In the next version, we will enable the direct use of the SEALS repository. Also, the user interface will be incorporated into the SEALS portal.

Finally, some more advanced scenarios in the recommendation system are planned for the future work. For example, one scenario could provide feedback in the process of requirements specification, such that if a user chooses a requirement according to which all the tools are highly similar, a suggestion for omitting such requirement from the recommendation process can be given.

The *QualityModel* ontology developed in this work is the first version of the ontology. It is based on the quality model for semantic technologies and linked with the SEALS ontologies. In the future, this ontology will be further developed, with the accent on reusing existing vocabularies (e.g., the ontology for software measurement [12]) and on compliance to standards for software quality and measurements.

Publications

Part of the work presented in this Master thesis was published in the following scientific publications belonging to conference ranked as CORE B¹, and a workshop within a CORE C conference:

1. F. Radulovic and R. García-Castro. *Extending Software Quality Models - A Sample In The Domain of Semantic Technologies*. In Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE2011). Miami, USA. July, 2011
2. F. Radulovic and R. García-Castro. *Towards A Quality Model For Semantic Technologies*. Software Quality Workshop 2011, within the International Conference on Computational Sciences and Its Applications (ICCSA2011). Santander, Spain. June, 2011

¹[http://core.edu.au/index.php/categories/conference 20rankings/1](http://core.edu.au/index.php/categories/conference%20rankings/1)

Bibliography

- [1] H. Al-Kilidar, K. Cox, and B. Kitchenham. The use and usefulness of the ISO/IEC 9126 quality standard. In *2005 International Symposium on Empirical Software Engineering, 2005*, page 7. IEEE, 2005.
- [2] M. Azuma. SQuaRE: the next generation of the ISO/IEC 9126 and 14598 international standards series on software product quality. In *European Software Control and Metrics Conference (ESCOM), London, UK*, pages 337–346, 2001.
- [3] B. Behkamal, M. Kahani, and M.K. Akbari. Customizing ISO 9126 quality model for evaluation of B2B applications. *Information and software technology*, 51(3):599–609, 2009. ISSN 0950-5849.
- [4] B.W. Boehm, J.R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd International Conference on Software Engineering*, pages 592–605. IEEE Computer Society Press, 1976.
- [5] M. Bombardieri and F.A. Fontana. A specialisation of the SQuaRE quality model for the evaluation of the software evolution and maintenance activity. In *Automated Software Engineering-Workshops, 2008. (ASE Workshops 2008). 23rd IEEE/ACM International Conference*, pages 110–113. IEEE.
- [6] P. Botella, X. Burgués, J. Carvallo, X. Franch, J. Pastor, and C. Quer. Towards a quality model for the selection of ERP systems. *Component-Based Software Quality*, pages 225–245, 2003. ISSN 0302-9743.
- [7] J.P. Carvallo, X. Franch, and C. Quer. Defining a quality model for mail servers. In *Second International Conference on COTS-based Software Systems, (IC-CBSS 2003), Ottawa*, page 51. Springer-Verlag New York Inc, February 2003.
- [8] J.P. Cavano and J.A. McCall. A framework for the measurement of software quality. *ACM SIGMETRICS Performance Evaluation Review*, 7(3-4):133–139, 1978. ISSN 0163-5999.

-
- [9] R.G. Dromey. Software product quality: theory, model, and practice. *Software Quality Institute, Brisbane, Australia*, 1998.
- [10] J. Euzenat, C. Meilicke, C. Trojahn, and O. Šváb Zamazal. D12.3 Results of the first evaluation of matching tools. Technical report, SEALS Consortium, November 2010.
- [11] X. Franch and J.P. Carvallo. Using quality models in software package selection. *Software, IEEE*, 20(1):34–41, 2003. ISSN 0740-7459.
- [12] Félix García, Francisco Ruiz, Coral Calero, Manuel f. Bertoa, Antonio Vallecillo, Beatriz Mora, and Mario Piattini. Effective use of ontologies in software measurement. *Knowledge Engineering Review*, 24:23–40, December 2009. ISSN 0269-8889.
- [13] Raúl García-Castro and Asunción Gómez-Pérez. Interoperability results for Semantic Web technologies using OWL as the interchange language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8:278–291, November 2010. ISSN 1570-8268.
- [14] Raúl García-Castro, Asunción Gómez-Pérez, Óscar. Muñoz-García, and L. Nixon. Towards a component-based framework for developing semantic web applications. *The Semantic Web*, pages 197–211, 2008.
- [15] Raúl García-Castro, Stephan Grimm, Michael Schneider, Mick Kerrigan, and Giorgos Stoilos. D10.1 Evaluation Design and Collection of Test Data for Evaluating Ontology Engineering Tools v1. Technical report, SEALS Project, February 2009.
- [16] Raúl García-Castro, S. Grimm, I. Toma, M. Schneider, A. Marte, and S. Tymaniuk. D10.3 Results of the first evaluation of ontology engineering tools. Technical report, SEALS Consortium, November 2010.
- [17] D.A. Garvin. What does product quality really mean? *Sloan management review*, 26(1):25–43, 1984. ISSN 0019-848X.
- [18] Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, 2005. ISSN 1570-8268.
- [19] V. Hlupic and A.S. Mann. SimSelect: a system for simulation software selection. In *Proceedings of the 27 th Conference on Winter Simulation*, pages 720–727. IEEE, 1995.

-
- [20] IEEE. IEEE std 1209-1992. IEEE recommended practice for the evaluation and selection of CASE tools. 1993.
- [21] ISO. ISO/IEC 9126-1:2001, Software engineering – Product quality – Part 1: Quality model. Technical report, International Organization for Standardization, 2001.
- [22] A. Jadhav and R. Sonar. A hybrid system for selection of the software packages. In *First International Conference on Emerging Trends in Engineering and Technology*, pages 337–342. IEEE, 2008.
- [23] A. Jadhav and R. Sonar. Analytic Hierarchy Process (AHP), Weighted Scoring Method (WSM), and Hybrid Knowledge Based System (HKBS) for Software Selection: A Comparative Study. In *Proceedings of the 2009 Second International Conference on Emerging Trends in Engineering & Technology*, pages 991–997. IEEE Computer Society, 2009.
- [24] B. Kitchenham and S.L. Pfleeger. Software quality: the elusive target. *Software, IEEE*, 13(1):12–21, 1996. ISSN 0740-7459.
- [25] J. Kontio. A case study in applying a systematic method for COTS selection. In *Proceedings of the 18th International Conference on Software Engineering, 1996.*, pages 201–209, 1996.
- [26] P. Lambrix, M. Habbouche, and M. Perez. Evaluation of ontology development tools for bioinformatics. *Bioinformatics*, 19(12):1564, 2003. ISSN 1367-4803.
- [27] T.B. Lee, J. Hendler, O. Lassila, et al. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
- [28] Y.M. Lee, J.B. Kim, I.W. Choi, and S.Y. Rhew. A Study on Selection Process of Open Source Software. In *Sixth International Conference on Advanced Language Processing and Web Information Technology, 2007. (ALPIT 2007)*, pages 568–571. IEEE.
- [29] H.Y. Lin, P.Y. Hsu, and G.J. Sheen. A fuzzy-based decision-making procedure for data warehouse system selection. *Expert systems with applications*, 32(3): 939–953, 2007. ISSN 0957-4174.
- [30] A. Lozano-Tello and A. Gómez-Pérez. BAREMO: how to choose the appropriate software component using the Analytic Hierarchy Process. In *Proceedings of the 14th international conference on software engineering and knowledge engineering (SEKE2002)*, page 788. ACM, 2002.

-
- [31] T. Neubauer, J. Pichler, and C. Stummer. A Case Study on the Multicriteria Selection of Software Components. In *2008 IEEE Asia-Pacific Services Computing Conference*, pages 1005–1012. IEEE, 2008.
- [32] OntoWeb. OntoWeb Deliverable 1.3: A survey on ontology tools. Technical report, IST OntoWeb Thematic Network, May 2002.
- [33] I. Padayachee, P. Kotze, and A. van Der Merwe. ISO 9126 external systems quality characteristics, sub-characteristics and domain specific criteria for evaluating e-Learning systems. In *The Southern African Computer Lecturers' Association, University of Pretoria, South Africa*, 2010.
- [34] S.S. Stevens. On the theory of scales of measurement. *Science*, 103(2684): 677–680, 1946.
- [35] T.W. Teweldeberhan, A. Verbraeck, E. Valentin, and G. Bardonnnet. Software evaluation and selection: an evaluation and selection methodology for discrete-event simulation software. In *Proceedings of the 34th conference on Winter simulation: exploring new frontiers*. ACM, New York, USA, 2002.
- [36] S. Tymaniuk, L. Cabral, D. Winkler, and I. Toma. D14.3 Results of the first evaluation of Semantic Web Service tools. Technical report, SEALS Consortium, November 2010.
- [37] S. N. Wrigley, K. Elbedweihy, D. Reinhard, A. Bernstein, and F. Ciravegna. D13.3 Results of the first evaluation of semantic search tools. Technical report, SEALS Consortium, November 2010.
- [38] M. Yatskevich and A. Marte. D11.3 Results of the first evaluation of advanced reasoning systems. Technical report, SEALS Consortium, November 2010.
- [39] H. Zulzalil, A.A.A. Ghani, M.H. Selamat, and R. Mahmud. A Case Study to identify quality attributes relationships for Web-based applications. *IJCSNS*, 8 (11):215, 2008. ISSN 1738-7906.

Appendix A

Quality Model Ontology

The *QualityModel* ontology is used to represent quality characteristic and results for quality measures, and defines the vocabulary for their representation. Fig A.1 shows the graphical representation of the ontology.

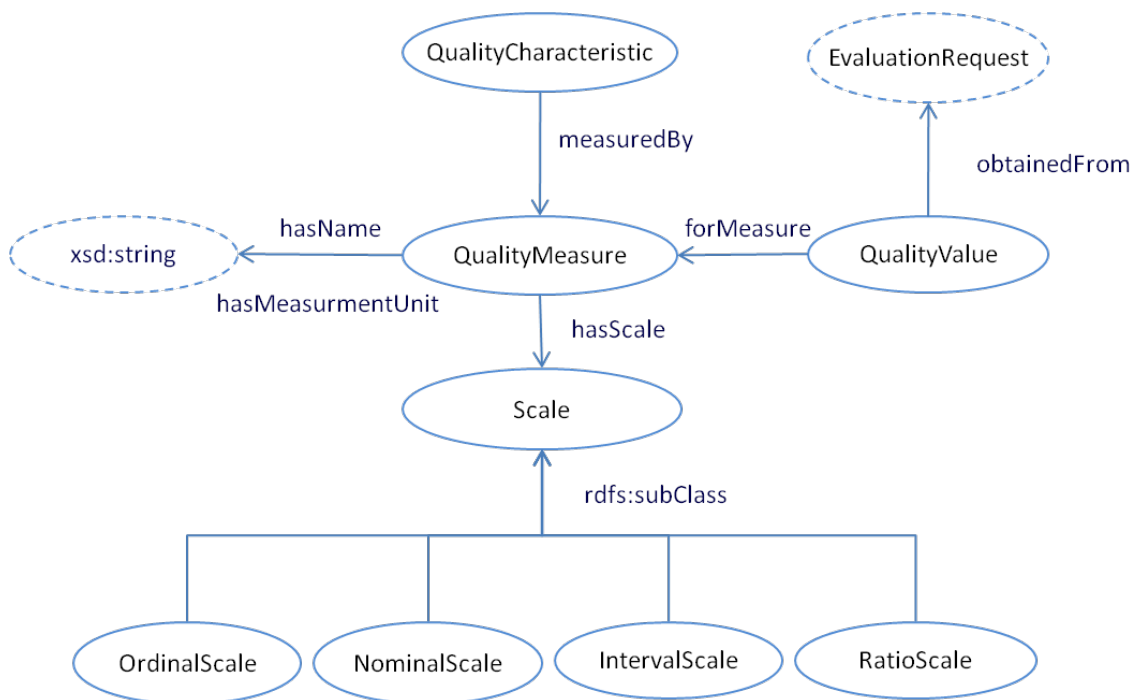


Figure A.1: Graphical representation of the *QualityModel* ontology.

The *QualityMeasure* class represents the quality measure, which is related to a quality characteristic as the range of the property *measuredBy*. A quality measure can be described by properties that have *QualityMeasure* as domain. Such proper-

ties are: *hasName* (the quality measure name), *hasMeasurementUnit* (the unit of measure), and *hasScale* (the scale of quality measure).

There are several possible types of scales, each represented by a class in the ontology: ratio scale, interval scale, nominal scale, and ordinal scale [34].

In the evaluation process, values for quality measures are captured and represented by the *QualityValue* class, which is further described by two properties: *forMeasure* (denotes the quality measure that this value relates to), and *obtained-From* (evaluation request from which the value is obtained, which further provides information about the tool that has been evaluated in this request).

The RDF/XML code is shown below.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <!ENTITY QualityModel
    "http://www.seals-project.eu/ontologies/QualityModel.owl#">
]>

<rdf:RDF xmlns="http://www.seals-project.eu/ontologies/QualityModel.owl#"
  xml:base="http://www.seals-project.eu/ontologies/QualityModel.owl"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:QualityModel=
    "http://www.seals-project.eu/ontologies/QualityModel.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <owl:Ontology rdf:about=""/>

  <!--
  //////////////////////////////////////
  //
  // Object Properties
  //
  //////////////////////////////////////
  -->
```



```
<owl:ObjectProperty rdf:about="#hasOrdinalScaleItem">
  <rdfs:domain rdf:resource="#OrdinalScale"/>
  <rdfs:range rdf:resource="#OrdinalScaleItem"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasRankingFunction">
  <rdfs:domain rdf:resource="#IntervalScale"/>
  <rdfs:range rdf:resource="#RankingFunction"/>
  <rdfs:domain rdf:resource="#RatioScale"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#hasScale">
  <rdfs:domain rdf:resource="#QualityMeasure"/>
  <rdfs:range rdf:resource="#Scale"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#obtainedFrom">
  <rdfs:domain rdf:resource="#QualityMeasure"/>
  <rdfs:range rdf:resource=
    "http://www.seals-project.eu/ontologies/SEALSMetadata.owl
      #EvaluationRequest"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#measuredBy">
  <rdfs:domain rdf:resource="#QualityCharacteristic"/>
  <rdfs:range rdf:resource="#QualityMeasure"/>
</owl:ObjectProperty>

<!--
////////////////////////////////////
//
// Data properties
//
////////////////////////////////////
-->
```

```
<owl:DatatypeProperty rdf:about="#hasLabel">
  <rdfs:domain rdf:resource="#NominalScale"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#hasMeasurementUnit">
  <rdfs:domain rdf:resource="#QualityMeasure"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#hasName">
  <rdfs:domain rdf:resource="#OrdinalScaleItem"/>
  <rdfs:domain rdf:resource="#QualityMeasure"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#hasRanking">
  <rdfs:domain rdf:resource="#OrdinalScaleItem"/>
  <rdfs:range rdf:resource="&xsd:int"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#hasValue">
  <rdfs:domain rdf:resource="#QualityMeasure"/>
  <rdfs:range rdf:resource="&xsd:string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#lowerBoundary">
  <rdfs:domain rdf:resource="#IntervalScale"/>
  <rdfs:range rdf:resource="&xsd:int"/>
</owl:DatatypeProperty>
```

```
<owl:DatatypeProperty rdf:about="#upperBoundary">
  <rdfs:domain rdf:resource="#IntervalScale"/>
  <rdfs:range rdf:resource="&xsd:int"/>
</owl:DatatypeProperty>
```

```
<!--
////////////////////////////////////
//
// Classes
//
////////////////////////////////////
-->
```

```
<owl:Class rdf:about="#IntervalScale">
  <rdfs:subClassOf rdf:resource="#Scale"/>
</owl:Class>
```

```
<owl:Class rdf:about="#NominalScale">
  <rdfs:subClassOf rdf:resource="#Scale"/>
</owl:Class>
```

```
<owl:Class rdf:about="#OrdinalScale">
  <rdfs:subClassOf rdf:resource="#Scale"/>
</owl:Class>
```

```
<owl:Class rdf:about="#OrdinalScaleItem"/>
```

```
<owl:Class rdf:about="#QualityCharacteristic"/>
```

```
<owl:Class rdf:about="#QualityMeasure"/>
```

```
<owl:Class rdf:about="#RankingFunction"/>
```

```
<owl:Class rdf:about="#RatioScale">
  <rdfs:subClassOf rdf:resource="#Scale"/>
</owl:Class>

<owl:Class rdf:about="#Scale"/>

<!--
////////////////////////////////////
//
// Individuals
//
////////////////////////////////////
-->

<RankingFunction rdf:about="#HigherBest"/>

<RankingFunction rdf:about="#LowerBest"/>

</rdf:RDF>
```