Extending Software Quality Models - A Sample In The Domain of Semantic Technologies

Filip Radulovic
Ontology Engineering Group
Departamento de Inteligencia Artificial
Facultad de Informática, Universidad Politécnica de Madrid
Madrid, Spain
fradulovic@delicias.dia.fi.upm.es

Raúl García-Castro
Ontology Engineering Group
Departamento de Lenguajes y Sistemas Informáticos
e Ingeniería Software
Facultad de Informática, Universidad Politécnica de Madrid
Madrid, Spain
rgarcia@fi.upm.es

***Abstract*—In order to correctly evaluate semantic technologies, which have become widely adopted in recent years, we need to put evaluations under the scope of a unique software quality model. This paper presents a quality model for semantic technologies. First, some well-known software quality models are described, together with methods for extending them. Afterwards, a new method for extending quality models is proposed and it is then used to define a quality model for semantic technologies by extending the ISO 9126 quality model. Finally, the proposed model is validated by analyzing existing semantic technology evaluations.**

I. Introduction

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 [1].

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. Nevertheless, 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.

An example of this occurs in the semantic technology field. Semantic technologies provide new ways to express in machine processable formats knowledge and data that can be exploited by software agents. We have seen an exponential growth of semantic technologies in recent years and multiple evaluations of such technologies have been proposed, from general evaluation frameworks [2] to tool-specific evaluations [3], [4] and even characteristic-specific evaluations [5].

However, it is very difficult to compare semantic technologies because of the different evaluation characteristics used. Furthermore, there is no consistent terminology for describing the quality of semantic technologies, and available software quality models do not specify the quality characteristics specific to them.

This paper describes a bottom-up approach for specifying a software quality model by extending an existing one. Using this approach, we have defined a quality model for semantic technologies, starting from real semantic technology evaluations and extending the ISO 9126 quality model.

Clearly, during the definition of the quality model not every available evaluation can be taken into account. To validate and complete the quality model, we have performed a literature review over those semantic technology evaluations presented in the most relevant conferences in the semantic research field.

The reminder of this paper is organized as follows. Section II gives an overview of existing software quality models. Section III describes top-down methods for extending software quality models, while Section IV presents the bottom-up method that we have defined. Section V describes how we have applied such method to define a quality model for semantic technologies. Section VI presents the validation of the quality model and, finally, Section VII draws some conclusions and includes ideas for future work.

II. Review of Software Quality Models

Various software quality models have been described in the literature: McCall's model, Boehm's model, ISO 9126 model, and SQuaRE model. This section describes the models most used and identifies their main elements.

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 [6]. 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, all of which are further decomposed into sub-characteristics that are manifested externally when the software is used and are the result of internal software attributes [6]. The standard also provides the internal and external measures for sub-characteristics.

Regarding quality in use, the model proposes four characteristics: effectiveness, productivity, safety, and satisfaction.

The ISO 9126 standard gives a complete view of software quality with evaluation criteria and definitions of every software characteristic and sub-characteristic. Some authors also suggest that according to the nature of the product itself some new sub-characteristics can be added, the definitions of existing ones changed, or some sub-characteristics can be eliminated from the model [7].

However, as pointed out in [8], 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.

SQuaRE's Model. Because of advances in technologies and changes of users' needs over time, some problems and issues have arisen with the ISO 9126 standard. Therefore, it is currently being redesigned and has been renamed SQuaRE (System and software Quality Requirements and Evaluation). At the time of writing this paper, the parts of the SQuaRE standard related to the quality model and evaluations are still under development (ISO 25010 and ISO 25040 respectively) and their final versions will be published during 2011.

As a summary of this section, Table I presents the elements of the quality models mentioned. In our work, and in the rest of the paper, we have adopted the terminology of the ISO 9126 standard.

TABLE I: Elements of mentioned 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

III. Approaches for Extending Software Quality Models

Existing software quality models (e.g., the ISO 9126 one) provide insight into characteristics that are general and common for almost every type of software. However, different types of software products have characteristics which are specific to them and, therefore, the actual application of software quality models usually requires reusing an existing quality model and extending it for a specific software product or domain.

To this end, starting from a certain quality model, its quality characteristics and sub-characteristics should be adapted according to the nature of the domain by excluding those out of scope, redefining others, and introducing new ones.

Some authors have proposed software quality models for various types of applications: B2B [1], mail servers [9], web-based applications [10], e-learning systems [11], and ERP systems [7]. 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 [12]:

- 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.

Franch and Carvallo proposed a method based on a top-down approach for customizing the ISO 9126 quality model [13]. After defining and analyzing the domain, their method proposes six steps:

1) *To determine quality sub-characteristics*. In this first step, according to the domain, some new quality sub-characteristics are added while other are excluded, or their definitions are changed.
2) *To define a hierarchy of sub-characteristics*. If needed, sub-characteristics are further decomposed according to some criteria.
3) *To decompose sub-characteristics into attributes*. In this step, abstract sub-characteristics are decomposed into more concrete concepts that refer to some particular software attribute (i.e., observable feature).
4) *Decomposing derived attributes into basic ones*. Attributes not directly measurable are further decomposed into basic ones.
5) *To state relationships between quality entities*. Relationships between quality entities are explicitly defined. Three possible types of relationships are identified: a) *collaboration* means that increasing the value of one entity implies increasing the value of another entity; b) *damage* means that increasing the value of one entity implies decreasing the value of another entity; and c) *dependency* implies that some values of one entity require that another entity fulfills some conditions.
6) *To determine 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 [1]. The main difference with the previous method is that in Behkamal's approach the quality characteristics are ranked by experts; thus, the experts should provide weights for all quality characteristics and sub-characteristics and these weights are later used to establish their importance, which can be time consuming and resource demanding. Besides, Behkamal's approach does not contemplate defining relationships between quality entities.

IV. A Bottom-Up Approach For Extending Software Quality Models

The approaches presented in the previous section follow a top-down approach and, at the time of writing this paper, we have not found any example of a bottom-up approach in the literature. However, there are scenarios where it would be convenient to base on real practices the extension of the quality model because of the existence of a significant body of software evaluations and evaluation results (as in the case of the semantic technology field).

In our approach, 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:

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., 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 [13]) or more formally (e.g., with the formulas used for obtaining the measures, as proposed in [14]). 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 recommendable to identify the function that defines it based on other measures.
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 with 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.

V. Defining A Quality Model For Semantic Technologies

This section describes the definition of a software quality model in the domain of semantic technologies by following the bottom-up method presented in the previous section.

A. Identifying Basic Measures

The starting point for defining software quality measures has been the set of evaluation results obtained in the SEALS project[1], which has produced evaluation results for different types of semantic technologies (ontology engineering tools [15], reasoning systems [16], ontology matching tools [17], semantic search tools [18], and semantic web service tools [19]).

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).

Due to space reasons, we cannot present details about all evaluation scenarios. Therefore, we just present the outcomes of each step for one concrete scenario, that of evaluating the conformance of ontology engineering tools.

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 in 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 1.

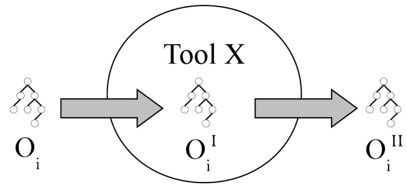

Fig. 1: Steps of a conformance test execution.

The basic measures obtained after a test case execution are

- *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.

B. Identifying Derived Measures

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.

[1]http://www.seals-project.eu

- *Structurally equivalent*. Whether the origin ontology and the final one are structurally equivalent.
- *Semantically equivalent*. Whether the origin ontology and the final one are semantically equivalent.
- *Conformance*. Whether the ontology has been imported and exported correctly with no addition or loss of information.

C. Identifying Quality 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 persistence*. The ratio of information additions or losses 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 II summarizes the results obtained.

TABLE II: Number of measures obtained for semantic technologies.

Tool\Measures	**Basic**	**Derived**	**Quality**
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

D. 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, 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:

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

E. Defining Domain-Specific Quality Sub-characteristics

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 evaluation 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 a software product adheres to the knowledge representation model of an ontology language. It can be measured with two different quality measures: *Ontology language component coverage*, and *Ontology language component support*.
- *Ontology processing accuracy*. The accuracy of the process of importing and exporting ontologies. It can be measured with *Ontology information persistence*.
- *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 with *Execution errors*.

Fig. 2 presents the basic measures, derived measures, quality measures, and quality characteristics of the conformance evaluation for ontology engineering tools.

In total, we have identified twelve semantic quality sub-characteristics. Three of them are those described for the conformance evaluation and the others are the following:

- *Ontology language interoperability*. The degree to which the software product can interchange ontologies and use the ontologies that have been interchanged.
- *Reasoning accuracy*. The accuracy of the reasoning process.
- *Ontology alignment accuracy*. The accuracy of the matching process.
- *Semantic search accuracy*. The accuracy of the semantic search process.
- *Semantic web service discovery accuracy*. The accuracy of the process of finding services that can be used to fulfill a given requirement from the service requester.
- *Ontology interchange accuracy*. The accuracy of the interchange of ontologies between tools.
- *Ontology processing time behaviour*. The capability of the software product to provide appropriate response and processing times when working with ontologies.
- *Reasoning time behaviour*. The capability of the software product to provide appropriate response and processing times when performing reasoning tasks.
- *Semantic search time behaviour*. The capability of the software product to provide appropriate response and processing times when performing search tasks.

Apart from these domain-specific quality sub-characteristics, we have identified the following general ones: *Operability*, *Productivity*, and *Satisfaction*.

Finally, we have also identified those sub-characteristics that are contained into others (e.g., *Reasoning time behaviour* is a sub-characteristic of *Ontology processing time behaviour*).

F. Aligning Quality Sub-Characteristics with a Quality Model

As ISO 9126 has been used by a number of authors, as mentioned in Section III, we have also adopted it for constructing the quality model for semantic technologies.

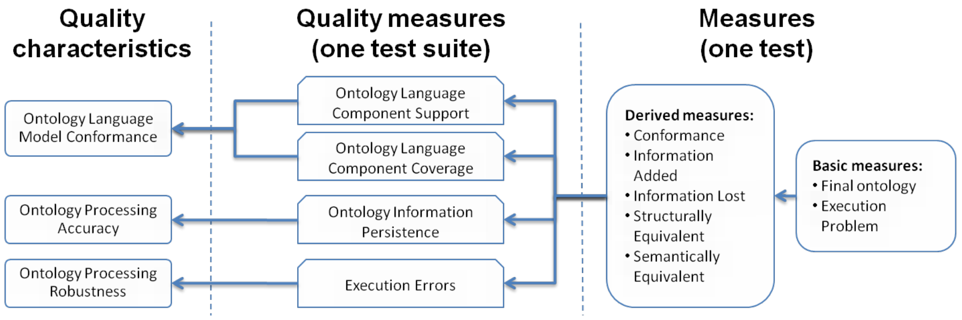
Quality characteristics
Quality measures (one test suite)
Measures (one test)
Ontology Language Model Conformance
Ontology Processing Accuracy
Ontology Processing Robustness
Ontology Language Component Support
Ontology Language Component Coverage
Ontology Information Persistence
Execution Errors
Derived measures:
• Conformance
• Information Added
• Information Lost
• Structurally Equivalent
• Semantically Equivalent
Basic measures:
• Final ontology
• Execution Problem

Fig. 2: Entities in the conformance scenario for ontology engineering tools.

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

For instance, *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).

VI. Quality Model Validation

Since we started from a specific set of evaluations in order to define the quality model for semantic technologies, we performed a literature review to validate the quality model and to complete it if needed. The review was performed according to the procedures described in [20] and, due to space reasons, we will only present the final results.

We have analysed 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. We focused on publications that describe evaluation methods or suggest measures for evaluation as well as on publications that suggest new algorithms (e.g., for reasoning or semantic web service discovery) that are also evaluated.

In total, we have analysed fifty seven publications. Table III shows an overview of this analysis including, for each type of semantic technology, the evaluation measures used. Every evaluation measure is classified according to the quality sub-characteristics that our model describes and the number of occurrences is shown in brackets.

From the analysis we can observe that the quality model that we have proposed is quite complete regarding current semantic technology evaluations. Almost all the measures described in the publications fit the quality characteristics that our model describes. However, some measures found did not fit our model, and therefore we have defined new quality characteristics for them. These 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.

TABLE III: Measures used in conference publications.

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

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

Fig. 3 shows an overview of the quality model for semantic technologies after completing it during the validation.

VII. Conclusions and Future Work

This paper presents a new method for extending software quality models, which is based on a bottom-up approach. It starts from existing evaluations and continues defining quality measures and quality sub-characteristics, which are aligned with an existing quality model.

In practice, the quality model to be extended is taken into account from the beginning of the method, even if the alignment to such quality model is the last step of the method. Therefore, it seems natural to follow a hybrid approach, which combines the bottom-up and top-down approaches, and a future extension of the method should also cover this approach.

We have used the method for defining a quality model for semantic technologies, which extends the ISO 9126 software quality model. Such quality model can provide a framework for the evaluation and comparison of semantic technologies.

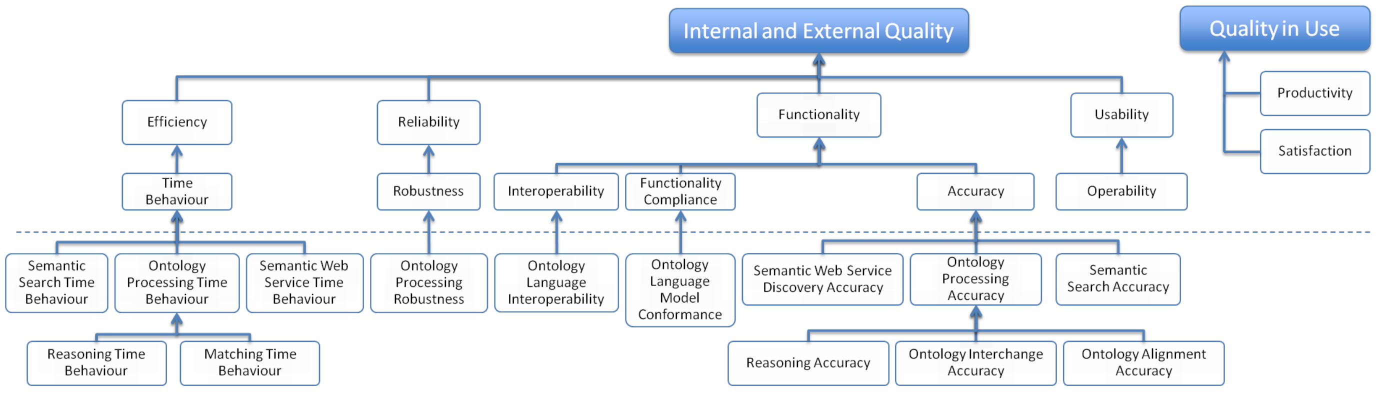
Internal and External Quality
Quality in Use
Productivity
Satisfaction
Efficiency
Reliability
Functionality
Usability
Time Behaviour
Robustness
Interoperability
Functionality Compliance
Accuracy
Operability
Semantic Search Time Behaviour
Ontology Processing Time Behaviour
Semantic Web Service Time Behaviour
Ontology Processing Robustness
Ontology Language Interoperability
Ontology Language Model Conformance
Semantic Web Service Discovery Accuracy
Ontology Processing Accuracy
Semantic Search Accuracy
Reasoning Time Behaviour
Matching Time Behaviour
Reasoning Accuracy
Ontology Interchange Accuracy
Ontology Alignment Accuracy

Fig. 3: External and internal quality characteristics for semantic technologies.

Although some problems with ISO 9126 have been identified (as described in [8]), we have introduced quality measures specific to semantic technologies, and we have also specified functions for all derived and quality measures, which result in reducing ambiguities in our model.

Furthermore, we can note that our quality model is easily extensible and that new quality measures or characteristics can be easily introduced and categorized, as shown during the validation process.

During such validation, we have concentrated only on the most relevant conferences in the semantic field. In order to get more complete results, we plan to extend our analysis to other conferences, as well as to relevant journals. This will help us to further extend and validate the model.

The ISO 9126 standard is being replaced by the SQuaRE standard; when the SQuaRE software quality model becomes available, the proposed quality model for semantic technologies will be adapted to it.

A future use of the proposed quality model, based on the evaluation results that are being obtained in the SEALS project, is to build a recommendation system for semantic technologies that will allow extracting semantic technology roadmaps. Such a system will provide users with guidance and recommendation of the semantic technologies that better suit their needs.

Acknowledgments

This work is supported by the SEALS European project (FP7-238975) and by the EspOnt project (CCG10-UPM/TIC-5794) co-funded by the Universidad Politécnica de Madrid and the Comunidad de Madrid.

Thanks to Rosario Plaza for reviewing the grammar of this paper.

References

[1] B. Behkamal, M. Kahani, and M. Akbari, “Customizing ISO 9126 quality model for evaluation of B2B applications,” *Information and software technology*, vol. 51, no. 3, pp. 599–609, 2009.
[2] OntoWeb, “Ontoweb deliverable 1.3: A survey on ontology tools,” IST OntoWeb Thematic Network, Tech. Rep., May 2002.
[3] 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*, vol. 3, no. 2-3, pp. 158–182, 2005.
[4] P. Lambrix, M. Habbouche, and M. Perez, “Evaluation of ontology development tools for bioinformatics,” *Bioinformatics*, vol. 19, no. 12, p. 1564, 2003.
[5] R. García-Castro and A. 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*, vol. 8, pp. 278–291, November 2010.
[6] ISO, “ISO/IEC 9126-1:2001, Software engineering – Product quality – Part 1: Quality model,” International Organization for Standardization, Tech. Rep., 2001.
[7] 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*, pp. 225–245, 2003.
[8] 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.* IEEE, 2005, p. 7.
[9] J. Carvallo, X. Franch, and C. Quer, “Defining a quality model for mail servers,” in *COTS-based software systems: second international conference, ICCBSS 2003, Ottawa, Ont., February 10-13, 2003: proceedings.* Springer-Verlag New York Inc, 2003, p. 51.
[10] H. Zulzalil, A. Ghani, M. Selamat, and R. Mahmod, “A Case Study to Identify Quality Attributes Relationships for Web-based Applications,” *IJCSNS*, vol. 8, no. 11, p. 215, 2008.
[11] 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.
[12] R. Dromey, “Software Product Quality: Theory, Model, and Practice,” *Software Quality Institute, Brisbane, Australia*, 1998.
[13] X. Franch and J. Carvallo, “Using quality models in software package selection,” *Software, IEEE*, vol. 20, no. 1, pp. 34–41, 2003.
[14] M. Bombardieri and F. 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 on.* IEEE, pp. 110–113.
[15] R. García-Castro, S. Grimm, I. Toma, M. Schneider, A. Marte, and S. Tymaniuk, “D10.3 Results of the first evaluation of ontology engineering tools,” SEALS Consortium, Tech. Rep., 2010.
[16] M. Yatskevich and A. Marte, “D11.3 Results of the first evaluation of advanced reasoning systems,” SEALS Consortium, Tech. Rep., 2010.
[17] J. Euzenat, C. Meilicke, C. Trojahn, and O. Šváb Zamazal, “D12.3 Results of the first evaluation of matching tools,” SEALS Consortium, Tech. Rep., 2010.
[18] S. N. Wrigley, K. Elbedweihy, D. Reinhard, A. Bernstein, and F. Ciravegna, “D13.3 Results of the first evaluation of semantic search tools,” SEALS Consortium, Tech. Rep., 2010.
[19] S. Tymaniuk, L. Cabral, D. Winkler, and I. Toma, “D14.3 Results of the first evaluation of Semantic Web Service tools,” SEALS Consortium, Tech. Rep., 2010.
[20] B. Kitchenham, “Procedures for performing systematic reviews,” *Joint Technical Report NICTA Technical Report 0400011T1*, vol. 33, 2004.