Evaluation of an Application Ontology

He TAN [a], Anders ADLEMO [a], Vladimir TARASOV [a] and Mats E. JOHANSSON [b]

[a] *Department of Computer Science and Informatics, School of Engineering, Jönköping University, Sweden*
[b] *Saab AB, Jönköping, Sweden*

Abstract. The work presented in this paper demonstrates an evaluation procedure for a real-life application ontology, coming from the avionics domain. The focus of the evaluation has specifically been on three ontology quality features, namely usability, correctness and applicability. In the paper, the properties of the three features are explained in the context of the application domain, the methods and tools used for the evaluation of the features are presented, and the evaluation results are presented and discussed. The results indicate that the three quality features are significant in the evaluation of our application ontology, that the proposed methods and tools allow for the evaluation of the three quality features and that the inherent quality of the application ontology can be confirmed.

Keywords. application ontologies, ontology evaluation, ontology quality features, ontology verbalization

1. Introduction

Ontologies as a modelling approach have long been recognized by the software engineering community (e.g. [1,2]). One example where ontologies have been introduced in software development is requirements engineering (e.g. [3,4,5]). Requirements engineering (RE), is an area concerned with the elicitation, specification and validation of software systems requirements [6]. The use of ontologies in RE dates back to the 1990s, e.g. [7,8]. More recently, the interest in utilizing ontologies in RE, as well as software engineering in general, has been renewed due to the emergence of semantic web technologies [1,9]. Much of the research on the use of ontologies in RE has focused on inconsistency and incompleteness problems in requirements specifications. For example, in [3], an ontology provides a formal representation of the engineering design knowledge that can be shared among engineers, such that ambiguity, inconsistency, incompleteness and redundancy can be reduced to a minimum when engineers concurrently develop requirements for sub-systems of a complex artefact. In [7,10], ontologies have been used to represent requirements in formal ontology languages, such that the analysis of consistency, completeness and correctness of requirements can be performed through reasoning over requirements ontologies. In the research project carried out by the authors of this paper [11,12], ontologies have also been developed to represent

software requirements. The developed ontologies have mainly been employed to support the automation of the software testing process and, more specifically, the creation of software test cases.

The ontologies developed in the research project presented in this paper can be considered to be application ontologies. They contain the knowledge of a particular domain, that is the requirements specification for software, to be able to achieve a specific task or support a particular application, that is the generation of software test cases. An ontology, like all engineering artefacts, needs a thorough evaluation to be able to put confidence in it and its intrinsic quality features. Not only will the quality of an ontology have a direct impact on the quality of the results originating from it, the ontology's quality during the ontology development should also be measured in a quantitative way, to decide whether the ontology will be able to meet its assigned goals or not. In this paper we demonstrate an ontology evaluation carried out in the aforementioned research project. We have focused on the evaluation of three specific quality features, i.e. usability, correctness and applicability, and proposed methods and tools for the evaluation of the quality features.

The remainder of the paper is organised as follows. Section 2 presents one of the application ontologies developed in our research project. In section 3, we discuss the quality features used to evaluate the ontology and the methods for evaluating the features. Section 4 presents the evaluation tools and the ontology evaluation with the results. Section 5, finally, presents our conclusions and discussions on future work.

2. An Application Ontology

In this section, we first present one of the application ontologies that were developed in our research project, along with the task it supports. The task of the ontology is to capture the knowledge contained in a set of given software requirements in an effort to automate the creation of software test cases. The ontology represents the requirements of a communication software component pertaining to an embedded system situated in an avionics system. The software component is fully developed in compliance with the DO-178B standard [13] which means that the requirements of the software component have been prepared, reviewed and validated by experts at the avionics partner company. Based on these written requirements, the application ontology for the software component, or requirements ontology as we also call it in this paper, was developed to support the creation of software test cases. Due to confidentiality reason the ontology is not published on the internet.

The ontology was created using Protégé. Beside the ontology itself, documentation was created to give an overview of the represented knowledge and describe the design of the ontology. The current version of the ontology contains 42 classes, 34 object properties, 13 datatype properties, and 147 instances in total. Figure 1 shows the ontology fragment for one particular functional requirement, in this case the SRSRS4YY-431, a requirement that has a focus on error handling. Ontology fragments for the remaining individual requirements related to the com-

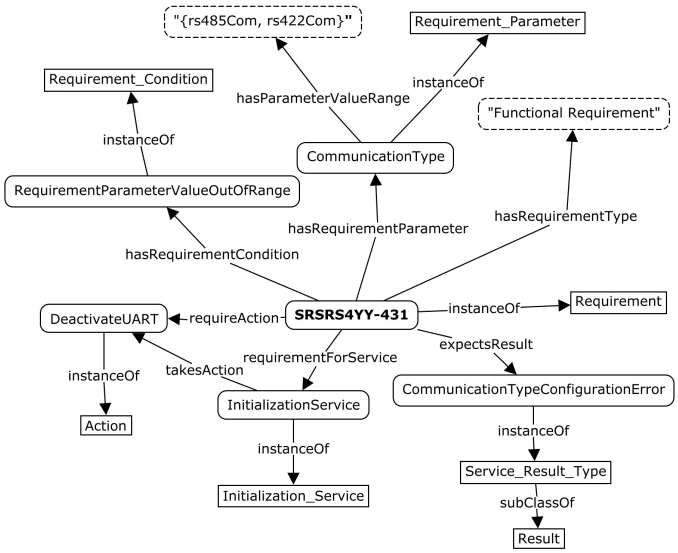

Figure 1. Ontology fragment for the requirement SRSRS4YY-431

munication software component can be represented in a way similar to the one in Figure 1. The SRSRS4YY-431 defines that if the communication type is out of its valid range, the initialization service shall deactivate the UART (Universal Asynchronous Receiver/Transmitter), and return the result "comTypeCfgError". In the figure, the rectangle boxes represent the concepts of the ontology; the rounded rectangle boxes represent the instances; and the dashed line rounded rectangle boxes provide the data values of the datatype property for instances.

The ontology has been designed to support inference rules to generate the software test cases [12]. The inference rules, that represent the expertise of an expert software tester, are coded in Prolog and make use of the ontology entities to generate the test cases. The Prolog inference engine controls the process of selecting and invoking the inference rules. To be able to implement this process, the ontology first needs to be translated into Prolog syntax by applying a set of predefined translation rules, and thus become a part of the Prolog program.

3. Quality Features and Evaluation Methods

The challenge in ontology evaluation is to determine the quality features that are to be evaluated, along with the evaluation method. Many different evaluation features have been discussed in literature (e.g. [14,15,16,17]). Which quality features to evaluate depend on various factors, such as the type of ontology, the focus of the evaluation and the person who is performing the evaluation. Burton-Jones et al. argue that different types of ontologies require different evaluation features [18]. For example, for an application or domain ontology it is enough to evaluate the features important to the application or domain. In our research project, the ontology evaluation has focused on three specific quality features: usability, correctness and applicability, as these features are estimated to be the

most critical and valuable when deciding whether our specific application ontology has met its assigned task of capturing the knowledge contained in a set of given software requirements to support the creation of software test cases.

In the context of this paper, we consider *usability* of an application ontology to be a set of attributes that describe the effort needed by a human to make use of the ontology. The usability feature provides a measure of the quality a user experiences when interacting with an ontology. A user of an application ontology is normally not the creator of the ontology, and he or she may not even be knowledgeable about ontologies or ontology engineering. This observation was also true for the domain experts participating in the evaluation of the application ontology presented in this paper. Usability, as defined and applied in this paper, is a necessary condition for an application ontology to be used on a regular basis. Users should not feel frustrated when attempting to understand the ontology. Instead, users should be able to put trust in the ontology and be confident that they can carry out their tasks effectively and efficiently while using the ontology.

The evaluation of the usability of a product or system has a long history and back in 1986, Brooke developed a questionnaire, so called System Usability Scale (SUS) [19], with exactly this purpose. Ever since then, the SUS has demonstrated its excellence in thousands of applications, proving that it can be used for a wide range of systems and types of technology. It has been shown that the SUS produces comparable results to those of more extensive attitude scales that intend to provide a deeper insight into a user's attitude towards the usability of a specific system. The SUS also possess the ability to discriminate and identify systems that demonstrate good and poor usability tendencies. An example of a special application where the SUS has been applied, an application that is relevant to the work presented in this paper, is the evaluation of the usability of an ontology, as described in [20]. Yet another example has been presented in [**?**]. In both these papers the usability of an ontology was evaluated by other people than ontology experts.

In our project, the correctness of an ontology is probably the most important quality feature that needs to be evaluated. Different definitions of correctness can be found in different articles, e.g. [14,17]. This divergence in the definition leads to confusion when trying to specify evaluation goals for an ontology. In the context of this paper, we define *correctness* of an application ontology as the degree to which the information asserted in the ontology conforms to the information that need be represented in the ontology. It is about being able to extract the accurate information from an ontology and to accurately document the information gathered from the same ontology. Validation methods, like reasoning, can find logical errors, such as inconsistency, but give no indication as to the correctness of the content. Correctness is therefore probably the most important quality feature, but at the same time one of the most difficult to measure.

Competency questions (CQ) are among the available methods of gathering information. They are natural language sentences that express patterns that apply to a type of questions one would expect an ontology to be able to answer. CQ have been suggested to serve as functional requirements of an ontology [21]. The functional correctness of an ontology can consequently be validated against the CQ. For a large or complex ontology it is a challenging task to prepare CQ.

The correctness of the CQ should also be verified. In the case of utilizing an application ontology, the information contained in the ontology may already exist and be documented in other formats. For the software requirements ontology developed in our research project, the information is written in plain English and documented in software requirements documents. Furthermore, the functionality of an application ontology may not be clearly defined before or during the development of the ontology. The functionality can only be fully understood and clearly defined during or after the application of the ontology. This observation is also valid for the ontology developed during our research project. It is probably also true for other kinds of ontologies, e.g. those within the context of the Semantic Web, which are often used in ways not anticipated by the creators of the ontology. Hence, the functional requirements cannot always be fully identified for future applications.

Ideally, correctness verification is an activity realised by a domain expert who not only grasps the details of the application itself but also has sufficient knowledge of ontologies to be able to perform the evaluation. In many situations, this kind of human evaluator, with knowledge of ontologies, is not available. Hence, we suggest that a method, which is based on the verbalization of an ontology, provides support to the correctness evaluation process. In this process, the ontology is first verbalized into a natural language text. Thereafter, the text can be read and understood by non-ontology experts, i.e. application domain experts, and compared with the information contained in the source information that was used to construct the ontology.

Applicability of an application ontology, in the context of this paper, is defined as the quality of the ontology regarding its appropriateness for a particular application, task or purpose. The applicability concept includes the appropriateness of the structure or semantics (e.g. the depth of the class hierarchy and the number of instances), the appropriateness of the ontology language (i.e. which language constructs that can be used in an ontology constructed with that language), and the appropriateness of the ontology document (i.e. a particular serialization of an ontology, such as RDF/XML [22] and OWL Functional Syntax [23]). The selection of these conditions should support the application in such a way that the application could be implemented and the task could be performed effectively and efficiently. The applicability of an ontology can be evaluated by the person who designs or develops the application and who uses the ontology to implement the functionality of the application, or it can be evaluated through the application itself. The evaluation considers the quality features of the application or the performance of the task, which are decided or affected by the use of the ontology.

4. The Evaluation

In this section, we demonstrate the specific details of the evaluation with regard to the three quality features, applying them on the software requirements ontology described in Section 2. We also describe how the methods and tools were applied during the evaluation, and discuss the evaluation results. Four industry domain experts (DE1-DE4) from the avionics domain (and specifically from the

Statements to evaluate the previous knowledge of ontologies, Protégé and SRS		DE1	DE2	DE3	DE4	OE
1	To what extent do you know about ontologies, what they represent, how they can be used, etc.?	2	4	2	1	6
2	To what extent do you know about the ontology tool, Protégé, and how to use it?	1	3	2	1	6
3	To what extent are you familiar with SRS?	6	4	3	2	2
4	To what extent do have you been directly involved in developing different SRSs?	6	3	1	2	1
5	How familiar are you with the content of the SRS used in this project?	5	4	1	2	5
6	To what extent have you been involved in the development of the software component defined in the SRS?	4	1	1	1	1

Table 1. Validation of the evaluators previous knowledge in the ontology and SRS domains

software domain), and one ontology expert (OE) (not being the ontology developer), participated in the evaluation of the usability and the correctness of the application ontology. The applicability was evaluated by the ontology expert only, as he was the person who developed the ontology-based program for the test case generation.

To validate the evaluators' background competence in the field of ontologies and software requirements specifications (SRS), a quick questionnaire was applied. The results are presented in Table 1, where the grades range from 1, indicating "not at all", to 6, indicating "to a great extent". As can be observed in the table, none of the industry experts had a deeper knowledge of working with ontologies, apart from an overall understanding of the specific application ontology as a result of the many presentations provided during the recurrent project meetings. Additionally, they did not have any previous experience of using ontology editors, such as Protégé. However, these deficiencies were made up for by their experiences with SRS, both as designers and users. As a comparison, the ontology expert has a deep knowledge of ontologies, ontology tools and also the content of the SRS used in the project. This difference in knowledge and experience is the reason why the results presented in the following two subsections look like they do. In spite of these differences, by applying an appropriate preparation before the evaluation of an application ontology using adequate tools, such as Protégé and verbalization, the negative effects of such differences can be reduced to a minimum.

4.1. Evaluation of Usability

In the evaluation of the usability of the application ontology, we applied a version of the SUS that was introduced by Casellas in [20]. The result from the evaluation is presented in Table 2, including the ten questions, as required for a SUS evaluation. The intricate details of how to use the SUS can be found in [19], for example, why the odd numbered statements are all in positive form while the even numbered statements are all in negative form. The formulations of the texts in the ten questions have been slightly modified from the work by Casellas, to better

Statements to evaluate the usability of the requirement ontology		DE1	DE2	DE3	DE4	OE
1	I think that I could contribute to this ontology	5	2	2	2	5
2	I found the ontology unnecessarily complex	2	3	2	4	2
3	I find the ontology easy to understand	3	2	4	2	4
4	I think that I would need further theoretical support to be able to understand this ontology	4	3	4	5	1
5	I found that various concepts in this system were well integrated	4	4	4	3	5
6	I thought there was too much inconsistency in this ontology	4	3	3	4	2
7	I would imagine that most domain experts would understand this ontology very quickly	3	3	5	3	2
8	I find the ontology very cumbersome to understand	3	2	2	4	2
9	I am confident I understand the conceptualization of the ontology	4	3	3	2	5
10	I needed to ask a lot of questions before I could understand the conceptualization of the ontology	3	3	3	5	1

Table 2. Ontology usability evaluation

adjust to the ontology domain. The evaluation of the usability of an ontology is especially important and relevant when the ontology is going to be used by application domain experts who are normally not ontology experts, as stated before. Hence, the ontology was evaluated by four experts in the avionics domain, plus one ontology expert. The grades in the table range from 1 to 5, where 1=strongly disagree, 2=disagree, 3=no preference, 4=agree, 5=strongly agree.

A sample of only five evaluators could by some be considered as being too small. Tullis and Stetson [24] have shown that it is possible to get reliable results from a sample of only 8-12 users. Hence, we argue that our sample of five persons provides a positive indication of the usability of the ontology, as far as the application domain experts are concerned (not to mention the ontology expert, but for obvious reasons). Nonetheless, in the future a more extensive usability evaluation will be undertaken, to prove our statements regarding the usability of the ontology.

Since it was difficult for the industry experts to comprehend the ontology only by reading the raw ontology document (i.e. the ontology documented in formal ontology language), Protégé was used as the tool to visualize the application ontology in the evaluation. Before the evaluation, a 40-minute tutorial on the application ontology and the Protégé tool was provied to the 4 industry experts. During the evaluation itself, the questions in Table 2 were answered by all of the 5 experts. After having answered the questions, the score for each question was calculated (where the score can range from 0 to 4). For items 1, 3, 5, 7, and 9 (the positive statements) the score contribution was calculated as the scale position value minus 1. For items 2, 4, 6, 8, and 10 (the negative statements), the score contribution was calculated as 5 minus the scale position value. After having done

Evaluator	DE1	DE2	DE3	DE4	OE
# Requirements Evaluated	3 (of 9)	5 (of 9)	8 (of 10)	5 (of 10)	10 (of 10)
Evaluation Time (in minutes)	73	65	60	75	20
SUS Scores	57,5	50,0	60,0	25,0	82,5
Grades	poor	poor	poor	poor	excellent

Table 3. The ontology usability results, including the number of requirements evaluated by the experts, the time spent on the evaluations, the overall SUS scores from the evaluators, and grades given based on the normalised scores indicating percentile ranks

this simple exercise for the 10 statements, the sum of the scores was multiplied with 2.5 to obtain the overall SUS scores.

The final SUS scores are presented in Table 3 together with the grades, accordingly to Bangor et al. [25]. The results could be considered discouraging, but a closer perusal of the written comments by the evaluators indicated that the "poor" results were not caused by a "poor" ontology, but rather the lack of experience using Protégé. The comments from the industry domain experts (DE) were positive regarding the ontology itself, or as one of them stated: "Due to inexperience in reading and using ontologies in testing activities, at first it was not obvious how to apply the requirements ontology. But after some time, the almost one-to-one correspondence between the requirements identified in the SRS document and the corresponding parts found in the ontology, made it quite straightforward to understand. To overcome the initial problems of understanding some of the technicalities within the ontology, several questions had to be asked to the developers of the ontology. Consequently, some extra training in the field of ontologies would have been helpful." Not surprisingly, the comments from the ontology expert (OE) were more positive: "In general, it was easy to understand the ontology. Most concepts were well integrated but some of them needed an extra integration, mainly through object properties. No inconsistencies were found, just some entities missing from the application viewpoint. Most ontology engineers would quickly understand the ontology but, still, they would need some extra time because the domain represented by the application ontology is quite complex." The ontology expert did not have to ask a lot of questions because the documentation was available in addition to the ontology itself. However, some very concrete questions were needed to understand how the ontology could be improved from an application point-of-view. Negative comments from the industry application domain experts were mainly about the tool. For these experts it was difficult to get an overview of the complete solution and, hence, they got stuck in details.

To sum up, there exist strong indications that the usability of an ontology is not exclusively determined by the ontology complexity, but also by the tool used to read and comprehend the ontology. Before initiating the evaluation of the usability of an ontology, it is therefore recommended to commence with a thorough introduction of the tool that is going to be used. A good ontology visualisation tool, like VOWL, would also be of great help in this endeavour.

4.2. Evaluation of Correctness

In this real application the information to be represented in the ontology is the information written in the SRS document. Furthermore, the information in the SRS document has been reviewed and validated. Therefore, the correctness is evaluated as a whole, by comparing the information asserted in the ontology to the information in SRS document. For this purpose, two different tools were used by the five evaluators. The first tool that was used once again was Protégé. The evaluators used Protégé to access the information represented in the ontology and compared this information with the information written in the SRS document. The second tool that was used was a web-based application, developed within the project, that transformed the information represented in the ontology into a natural language text in English. The web-based application is an ontology verbalization tool with the goal of making ontologies more readable to non-ontology experts [26]. The focus of the research in the ontology verbalization field has so far mainly been on expressing the axioms in an ontology in natural language (e.g. [27,28]), or on generating a summarized report of an ontology (e.g. [29]). The SRS provided in the case study that is presented in this paper was well-structured. Hence, the verbalization process was based on a simple pattern-based algorithm.

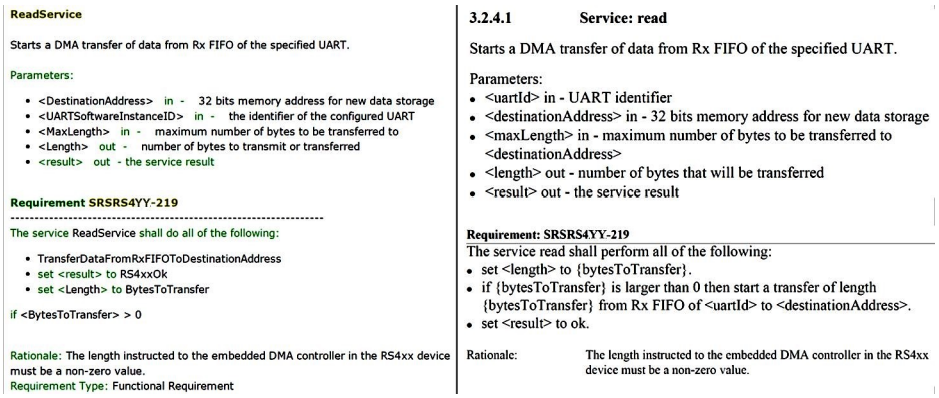

Figure 2. A fragment of the verbalization of the ontology (to the left) and the SRS (to the right)

The evaluation was organized in two parts. In the first part, the ontology was evaluated relying solely on Protégé. In the second part, the ontology was evaluated using only the verbalization tool. However, while using the verbalization tool, the evaluators could make use of Protégé for further validation of details, if required. The evaluation using the verbalization tool proceeded as follows. The evaluators uploaded the ontology (in OWL format) to the tool, retrieved the verbalization of the ontology (in plain English), and compared the verbalization result with the written information in the SRS document. Figure 2 presents the verbalization of a piece of the application ontology (the left part of the figure) and the corresponding text found in the requirements document (the right part of the figure). The green colored texts in the left part of the figure indicate the predefined patterns found in the verbalization algorithm. The remainder of the text (the black text found in the left part of the figure) comes from the ontology

	Tool	DE1	DE2	DE3	DE4	OE
# Req. Evaluated	Protégé	3 (of 9)	1,5 (of 9)	8 (of 10)	4,5 (of 10)	10 (of 10)
	verbalization	9 (of 9)	7,5 (of 9)	8 (of 10)	10 (of 10)	10 (of 10)
Eval. Time (in minutes)	Protégé	73	65	60	75	36
	verbalization	15	40	60	35	16

Table 4. The ontology correctness results, including the number of requirements evaluated by the experts, and the time spent on the evaluations (using Protégé or verbalization)

itself. The fragment of the ontology for requirement SRSRS4YY-219 is similar to requirement SRSRS4YY-431 presented in Figure 1.

In both parts of the evaluation, i.e. using Protégé or verbalization, the domain experts were asked to validate the correctness of the ontology as compared to the text found in the SRS document. Each of the four industry domain experts were assigned a fixed number of requirements that they were asked to validate (9 or 10 requirements each). Table 4 shows the number of requirements that were verified by each of the evaluators and the evaluation time needed while using either Protégé or the verbalization tool. As can be observed in the table, in general the evaluators used less time but verified more requirements when using the verbalization tool. The reason for this is simple: the difficulty when using Protégé for the first time, together with the staggering amount of information represented in the application ontology, causes a reduced evaluation speed. Some of these observed differences could most likely be reduced by increasing the time spent on training the users in using Protégé. As a secondary result, the evaluations sometimes indicated errors in the application ontology, errors that was subsequently removed by the ontology developers.

So, when should a domain expert rely on Protégé for the evaluation of an ontology and when it is enough to rely on the verbalization of an ontology? Our conclusion is that, in most situations, it is sufficient to verbalize an ontology to be able to detect possible errors or mismatches. And, as shown in Table 4, if time is of the essence, verbalizing an ontology normally beats relying on Protégé, especially if the domain experts are not familiar with Protégé. However, in some situations it is recommended to use both tools in a complementary fashion. Or, as one of the domain experts put it, "Note that 'human understandable' issues that are 'machine impossible' will go undetected [if relying solely on verbalization, authors comments]. In Protégé it was possible to see when a requirement was misinterpreted. Here it is back to textual representations [when using verbalization, authors comments] that may hide the misunderstanding. Easier to read though, but changing format is sometimes good".

4.3. Evaluation of Applicability

According to our definition of applicability, the ontology should exhibit the quality of appropriateness when used for a particular application domain or purpose. The applicability of the ontology is evaluated by looking at its applicability in the task of automatically generating software test cases based on SRS.

In the case study, consisting of the communication software component, software test cases were generated by a knowledge-based system consisting of a knowl-

edge base (i.e. the application ontology), a set of inference rules and an inference engine. The knowledge base had to provide the following: 1) Domain knowledge at the instance level sufficient for the construction of test cases; 2) Domain knowledge at the schema level necessary for domain-specific reasoning; 3) Syntax and semantics allowing for querying the knowledge base during the reasoning; and 4) Documentation that allows a developer to understand the represented knowledge. The inference rules used to create the test cases represent strategies for test case generation that were acquired through an analysis of the case study and discussions with the domain experts. The inference rules were implemented in Prolog because it provided a built-in inference engine, which was used for reasoning. The implementation details can be found in [12].

To be used as part of the knowledge-based system, the serialization of the requirements ontology was suggested to follow the OWL functional-style syntax [23] after a study of the Prolog syntax [30]. As the functional-style syntax is very similar to the Prolog syntax, the translation from OWL to Prolog was straightforward and without any loss of syntactic details (the translation rules can be found in [12]). As a result, the ontology became part of a Prolog program containing the inference rules, which simplified the implementation of the test case generator. The assertions of the translated ontology contained subclass axioms, domain and range axioms, and class assertions. The axioms made it possible to perform domain-specific reasoning based on the class membership and relations between the classes in the domain. More complex axioms, such as restrictions on object properties, turned out to be unused during domain-specific reasoning.

The conditions and actions of the inference rules included patterns that were matched against the statements in the knowledge base. The pattern matching was performed by Prolog, resulting in retrieval of instances from the application ontology, to check conditions of the rules as well as to construct different parts of test cases. The continued experiments showed that the instances contained in the application ontology were sufficient for the execution of the inference rules and the generation of software test cases. During the first experiment, 40 inference rules were used to generate 18 test cases. 66 distinct entities from the ontology were used for the test case construction. In the first attempt, the test cases were generated as plain text in English. The experiment showed an almost one-to-one correspondence between the texts in the generated test cases and the texts provided by one of our industrial partners in the form of a Software Test Description (STD) document. It should be stressed, however, that the test cases could as easily have been generated as executable scripts, if so desired.

The evaluation showed that the developed application ontology fulfilled its purpose. The ontology has been used for the automation of a part of the testing process and allowed for the successful generation of test cases. The ontology documentation contained an overview of the represented knowledge, which helped to bootstrap the development of the knowledge-based system. The syntax and semantics provided by the ontology satisfactorily supported the execution of the inference rules by the inference engine. The domain knowledge represented in the application ontology provided means for domain-specific reasoning and allowed for the construction of test cases that corresponded to the sample test cases provided in the case study. Minor deficiencies in the application ontology, discovered

during the development of the inference rules, were addressed and removed in the following iterations of the ontology development process. One example of a deficiency that was identified during the applicability evaluation, was the lack of distinction between single-value requirement parameters and enumeration parameters. This distinction is important for test case construction during the domain-specific reasoning. The lack of explicit representation of the enumeration type of parameters led to the loss of domain semantics, something that had to be remedied by an extra inference rule checking the type of a requirement parameter. Because this knowledge is declarative rather than procedural, this distinction was introduced in the ontology in the next iteration. The result of this can be observed in Figure 1 as a "list-type" data value (the upper-left green box, "{rs485Com, rs422Com}"). Another example of a modification of the application ontology, as a direct result of the applicability evaluation, was the division of the FIFO class into subclasses of transmission queues and reception queues.

5. Conclusions and Future Work

Within the vast research field that encompasses all aspects of ontology development is situated the crucial activity of building an ontology of "high quality". The challenge is the difficulty in defining quality, and decide on the methods/tools for the evaluation of quality. Currently, there exist no common definition of quality and well-established evaluation methods/tools. In the work presented in this paper, we have evaluated three quality features, *usability*, *correctness*, and *applicability*, features that we consider critical for the evaluation of an application ontology, which includes knowledge from software requirements in support of the creation of software test cases. We have also presented the methods and tools requested for the evaluation of the three quality features, along with the results from these evaluations.

Correctness is probably the most important quality feature, and also the most difficult to evaluate. Characteristics, such as completeness, conciseness and consistency, can be considered as sub-features of correctness. In the real application study that has been presented in this paper, correctness of an ontology is evaluated as a whole. Correctness of the ontology is measured by comparing the information represented in the ontology to the information found in a requirements document. The evaluation was performed by four industry domain experts and one ontology expert. The results from the evaluation performed by the industry domain experts indicate that the use of a verbalization tool can drastically improve the productivity during the evaluation of an ontology. Nonetheless, the use of a tool like Protégé is still needed to verify details in an application ontology. As the evaluations in this paper have indicated, extra time needs to be invested in the training of domain experts, for them to make full use of Protégé.

Evaluation can be related to different attributes of an ontology, such as its structure and semantics, ontology language and ontology documentation. These attributes of an application ontology are very relevant to the task or application the ontology supports. As such, they can be covered in the evaluation of the *applicability*. It is likely that the requirements on the applicability might not be

well-defined when developing an ontology. Hence, the evaluation of the applicability has to be coordinated with the development of the application that uses the ontology. But, at the same time, the evaluation of the applicability helps to make fine-grained improvements of the ontology. These improvements make the ontology better adjusted to the application.

The *usability* of an application ontology is of utmost importance from an application experts' point of view. If one really wants the application experts to fully embrace ontologies and use them, the ontologies must be "user-friendly". In this paper we have shown that the System Usability Scale is useful when evaluating the usability of an ontology.

To provide good tools for the ontology quality evaluation is paramount for the effectiveness and efficiency of ontology development. In the long term, the focus of our future work will be on investigating more examples of practical methods and tools for ontology quality evaluation. For the near future, we foresee a generalization of the verbalization process, such that the tool can be used for evaluating other types of ontologies.

Acknowledgment

The work presented in this paper was financed by the Knowledge Foundation in Sweden, grant KKS-20140170.

References

[1] H.-J. Happel and S. Seedorf, Applications of ontologies in software engineering, in: *Proceedings of Workshop on Semantic Web Enabled Software Engineering (SWESE) on the ISWC*, Citeseer, 2006, pp. 5–9.

[2] B. Henderson-Sellers, Bridging metamodels and ontologies in software engineering, *Journal of Systems and Software* **84**(2) (2011), 301–313.

[3] J. Lin, M.S. Fox and T. Bilgic, A requirement ontology for engineering design, *Concurrent Engineering* **4**(3) (1996), 279–291.

[4] V. Mayank, N. Kositsyna and M. Austin, Requirements engineering and the semantic Web, Part II. Representation, management, and validation of requirements and system-level architectures, Technical Report, University of Maryland, 2004.

[5] K. Siegemund, E.J. Thomas, Y. Zhao, J. Pan and U. Assmann, Towards ontology-driven requirements engineering, in: *Proceedings of Workshop semantic web enabled software engineering at 10th international semantic web conference (ISWC), Bonn*, 2011, 14 pages.

[6] B. Nuseibeh and S. Easterbrook, Requirements engineering: a roadmap, in: *Proceedings of the Conference on the Future of Software Engineering*, ACM, 2000, pp. 35–46.

[7] J. Mylopoulos, A. Borgida, M. Jarke and M. Koubarakis, Telos: representing knowledge about information systems, *ACM Transactions on Information Systems (TOIS)* **8**(4) (1990), 325–362.

[8] S. Greenspan, J. Mylopoulos and A. Borgida, On formal requirements modeling languages: RML revisited, in: *Proceedings of 16th international conference on Software engineering*, IEEE Computer Society Press, 1994, pp. 135–147.

[9] G. Dobson and P. Sawyer, Revisiting ontology-based requirements engineering in the age of the semantic Web, in: *Proceedings of International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs*, 2006, pp. 27–29.

[10] T. Moroi, N. Yoshiura and S. Suzuki, Conversion of software specifications in natural languages into ontologies for reasoning, in: *Proceedings of 8th International Workshop on Semantic Web Enabled Software Engineering (SWESE'2012)*, 2012.

[11] H. Tan, I. Muhammad, V. Tarasov, A. Adlemo and M. Johansson, Development and evaluation of a software requirements ontology, in: *Proceedings of 7th International Workshop on Software Knowledge-SKY 2016, in conjunction with the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management-IC3K 2016*, SCITEPRESS, 2016, pp. 11–18.

[12] V. Tarasov, H. Tan, M. Ismail, A. Adlemo and M. Johansson, Application of inference rules to a software requirements ontology to generate software test cases, in: *OWL: Experiences and Directions–Reasoner Evaluation: 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016, Revised Selected Papers*, Vol. 10161, Springer, 2017, p. 82.

[13] L.A. Johnson et al., DO-178B, Software considerations in airborne systems and equipment certification, *Crosstalk, October* **199** (1998).

[14] A. Gómez-Pérez, Ontology evaluation, in: *Handbook on Ontologies*, Springer, 2004, pp. 251–273.

[15] A. Gangemi, C. Catenacci, M. Ciaramita and J. Lehmann, Modelling ontology evaluation and validation, in: *Proceedings of European Semantic Web Conference*, Springer, 2006, pp. 140–154.

[16] F. Neuhaus, A. Vizedom, K. Baclawski, M. Bennett, M. Dean, M. Denny, M. Grüninger, A. Hashemi, T. Longstreth, L. Obrst et al., Towards ontology evaluation across the life cycle, *Applied Ontology* **8**(3) (2013), 179–194.

[17] H. Hlomani and D. Stacey, Approaches, methods, metrics, measures, and subjectivity in ontology evaluation: a survey, *Semantic Web Journal* (2014), 1–5.

[18] A. Burton-Jones, V.C. Storey, V. Sugumaran and P. Ahluwalia, A semiotic metrics suite for assessing the quality of ontologies, *Data & Knowledge Engineering* **55**(1) (2005), 84–102.

[19] J. Brooke, SUS-a quick and dirty usability scale, *Usability Evaluation in Industry* **189**(194) (1996), 4–7.

[20] N. Casellas, Ontology evaluation through usability measures, in: *Proceedings of OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, Springer, 2009, pp. 594–603.

[21] Y. Ren, A. Parvizi, C. Mellish, J.Z. Pan, K. Van Deemter and R. Stevens, Towards competency question-driven ontology authoring, in: *Proceedings of European Semantic Web Conference (ESWC)*, Springer, 2014, pp. 752–767.

[22] D. Beckett and B. McBride, RDF/XML syntax specification (revised), *W3C recommendation* **10** (2004).

[23] B. Motik, P. Patel-Schneider and B. Parsia, OWL 2 Web ontology language: structural specification and functional-style syntax, 2nd edn.(2012), 2015.

[24] T.S. Tullis and J.N. Stetson, A comparison of questionnaires for assessing website usability, in: *Proceedings of Usability Professional Association Conference*, Citeseer, 2004, pp. 1–12.

[25] A. Bangor, P. Kortum and J. Miller, Determining what individual SUS scores mean: adding an adjective rating scale, *Journal of usability studies* **4**(3) (2009), 114–123.

[26] M. Jarrar, C.M. Keet and P. Dongilli, Multilingual verbalization of ORM conceptual models and axiomatized ontologies, Technical Report, Vrije Universiteit Brussel, 2006.

[27] K. Kaljurand and N.E. Fuchs, Verbalizing OWL in Attempto controlled English, in: *Proceedings of OWLED'07*, Vol. 258, 2007, 10 pages.

[28] S.F. Liang, R. Stevens, D. Scott and A. Rector, Automatic verbalisation of SNOMED classes using OntoVerbal, in: *Proceedings of Conference on Artificial Intelligence in Medicine in Europe*, Springer, 2011, pp. 338–342.

[29] C. Kop, How to summarize an OWL domain ontology, in: *Proceedings of Fourth International Conference on Digital Society (ICDS'10)*, IEEE, 2010, pp. 106–111.

[30] I. Bratko, *Prolog Programming for Artificial Intelligence*, 4th ed. edn, Pearson Education, 2001.