Ontology-based software reliability modelling

Conference Paper · April 2007

CITATIONS
5

READS
158

3 authors:

Jiehan Zhou
University of Oulu
116 PUBLICATIONS **875** CITATIONS
SEE PROFILE

Eila Ovaska
University of Oulu
167 PUBLICATIONS **1,972** CITATIONS
SEE PROFILE

Antti Evesti
VTT Technical Research Centre of Finland
40 PUBLICATIONS **327** CITATIONS
SEE PROFILE

Ontology-Based Software Reliability Modelling

Jiehan Zhou, Eila Niemelä, Antti Evesti

VTT Technical Research Centre of Finland
Kaitoväylä 1, 90571 Oulu, Finland
Email:{firstname.surname}@vtt.fi

Abstract. Reliability has become a major concern for software-intensive systems. This paper proposes a novel ontology-based method for software reliability modelling, including a software reliability ontology and an ontology-based software reliability modelling system. The software reliability ontology is analysed and developed for software reliability engineering with respect to domains of reliability measurement processes, methods, models, organization, and tools. The ontology-based software reliability modelling system validates the ontology-based method with presentations of the system infrastructure, the system implementation techniques, and the system application cases.

1. Introduction

Modern society is permeated by software-intensive systems, such as consumer electronics, buildings, automobiles, and aircraft. The size and complexity of software-intensive systems have grown dramatically during the past decade, and the trend is certain to continue with the emergence of service-oriented architectures and Web services [1, 2]. Meanwhile, software reliability has become a major concern for software-intensive systems.

'Software reliability' refers to the probability of failure-free operation of a computer program in a specified environment over a specified period of time [3-5]. In the interest of increasing software reliability, a number of studies have been carried out on reliability modelling [6-8], reliability estimation, and prediction tools development [9, 10]. At the same time, several books [4, 5, 11] have been published for the purposes of reliability education and training. For the sake of brevity, when speaking of 'reliability' below, we are referring to software reliability.

In what follows, we will explore a novel ontology-based method for reliability modelling, including a reliability ontology and an ontology-based reliability modelling system. To the best of our knowledge, there are only few reports available on the topic. The remainder of the paper is organized as follows: Section 2 outlines a set of key concepts related to ontology-based reliability modelling. Section 3 defines the objectives of building a reliability ontology. In Section 4, different ontology engineering methods are compared and a guideline is presented for reliability ontology modelling. Section 5 examines reliability domains and develops the reliability ontology. The reliability modelling system is presented in Section 6, along with the system infrastructure, system implementation techniques, and some system

application cases. Section 7 draws the conclusions from our discussion and anticipates directions for future research.

2. Concepts

Ontology is a shared knowledge standard or knowledge model explicitly defining primitive concepts, relations, rules, and their instances. Ontology can be used for capturing, structuring, and enlarging explicit and tacit knowledge across people, organizations, and computer and software systems [12].

Reliability ontology consists of concepts and their relationships related to the topic of reliability engineering and aimed at facilitating the work of reliability experts in managing and developing reliability knowledge.

Ontology-based reliability design is an ontology-based method that provides reliability experts with the reliability ontology and associated management tools for facilitating software reliability definition and measurement.

3. Objectives of the reliability ontology development

The intended uses of the reliability ontology include the following application contexts:

- Management and development of reliability knowledge. By making use of the reliability ontology, the software customer can understand reliability terminologies explicitly, assess the reliability of the provided software or service, and fluently communicate with reliability experts. The reliability experts can adopt, adjust, and choose reliability models based on the reliability ontology.
- Support for computer-aided reliability estimation and prediction. Once the reliability ontology is built into software programs (so-called ontology-based programs), these programs can be enhanced by customizing reliability measurement procedures and by providing a common reliability development and management framework.
- Support for reliability knowledge management in software engineering. Component and service reliability becomes a critical factor influencing component-based and service-oriented software development. The reliability ontology promises to provide software community with a common communication platform for addressing reliability knowledge, and to facilitate reliability-based service discoveries and compositions.

4. Ontology engineering and reliability ontology modelling

4.1 Ontology engineering methodologies

The ontology engineering methods are summarised in Table 1. An ontology engineering method is a series of interrelated activities and techniques used for creating ontology. Thus, the traditional ontology development methods given in Table 1 are presented in terms of involved activities, created ontologies, and unique features. A more detailed comparison between the different ontology engineering methodologies is presented in [13].

Table 1. Summary of traditional ontology development methods

Method	Process	Ontologies created	Notes
Cyc [14]	Manual coding, computer coding, computer managing	Cyc	Unanimous knowledge capture
Uschold and King [15]	Purpose identification, ontology building, evaluation, and documentation	Enterprise ontology	Relevant to business enterprise
Gruninger and Fox [16]	Scenario identification, informal competency question formulation, terminology specification, formal competency question formulation, axiom specification, and completeness theorem specification	TOVE ontologies including enterprise design, project, scheduling, and service ontology	High degree of formality
KACTUS [17]	Application specification, preliminary design, refinement, and structuring	Fault and service recovery planning ontology	Conditioned by application development
Methontology [5, 18][8]	Roots in IEEE1074-1995	Chemicals, reference, KM ontology, etc.	Most mature, the method proposed by FIPA
SENSUS-based [19]	Identify seed terms, link the seed terms to SENSUS, add paths to the root, add new domain terms, and add complete sub-trees	Military air campaign planning	Easy generation of skeleton ontologies from huge ontologies
On-To-Knowledge [20, 21]	Feasibility study, kick-off, refinement, evaluation, and maintenance	Skills management, virtual enterprise, OntoWeb, etc.	Created ontologies highly dependent on the application, ontology learning

4.2 Reliability ontology modelling

Based on the summary of ontology engineering methods and a typical ontology engineering process given in [8], we can propose a guideline for creating the reliability ontology:

Step 1. Determine the domain of the reliability ontology. The reliability ontology covers reliability process, method, model, specification, tool, and organization domains.

Step 2. Consider reusing existing reliability ontologies. Unfortunately, no reliability ontologies exist as of yet. Information on reliability is distributed though books and research publications.

Step 3. Enumerate important reliability concepts. It is useful to start with classical works on reliability engineering, creating a list of all the concepts that reliability experts would prefer to use.

Step 4. Define the reliability concept hierarchy. A top-down development process may be used in this step. First we define the most general reliability concepts (process, method, and specification) and the subsequent specialisation of these concepts (reliability definition and operational profile development).

Step 5. Define the reliability properties. In general, there are two types of reliability properties: internal properties and external properties. The internal properties indicate properties belonging to the concept itself, such as the name of a process. The external properties indicate relationships between concepts, such as the method of a process.

Step 6. Create reliability instances. We define an individual reliability instance in terms of a concept and its properties.

5. Reliability ontology design

5.1. Reliability ontology domains

We describe reliability engineering as a series of interrelated processes by which reliability knowledge is reorganized with the support of methods, tools, models, organization, and the specifications of input and output (Fig. 1).

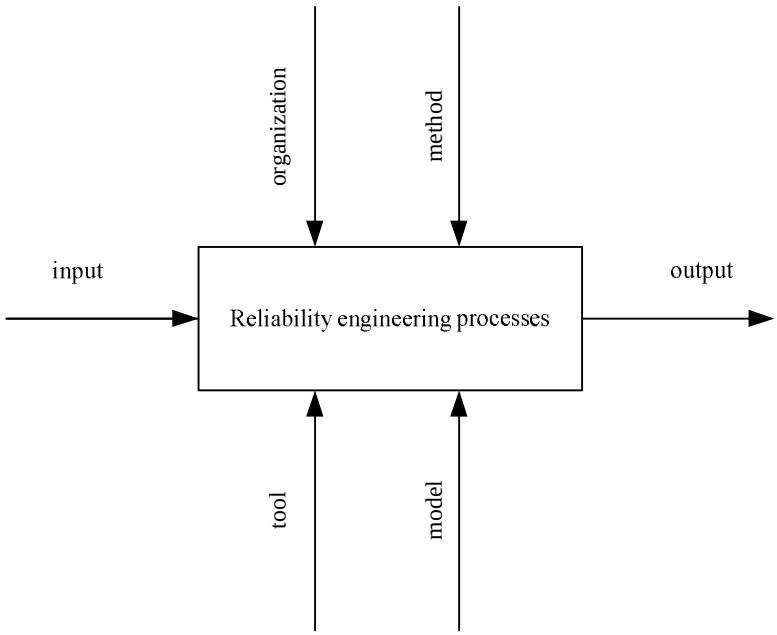


Fig. 1. Reliability ontology domains

The reliability process refers to reliability engineering processes, consisting of five activities: a) definition of necessary reliability, b) development of operational profiles, c) preparation for test, d) execution of test, and e) application of failure data to guide decision-making.

The reliability organization is in charge of executing the processes. Reliability measurement is usually carried out in an operation organization. The commonly involved roles are the end-user for executing the software, the manager for resource allocation, the system engineers for tailoring reliability procedures, and the quality assurance engineers for running tools and collecting failure data.

The reliability method identifies a way for the organization to undertake reliability processes in terms of cost-efficiency and purpose-specification.

The reliability models are chosen and used in any given applications depending on the purpose of the application. A software reliability model usually has the form of a random process that describes the behaviour of failures over time.

Reliability tools are computer programs or simulations used in software reliability measurement.

Reliability specification of input and output refers to the input and output data for the reliability process. This data can take the form of tables, files, and graphics.

5.2. Concepts hierarchy

A reliability concepts hierarchy is a hierarchical concept classification. Fig. 2 describes these concepts in detail.

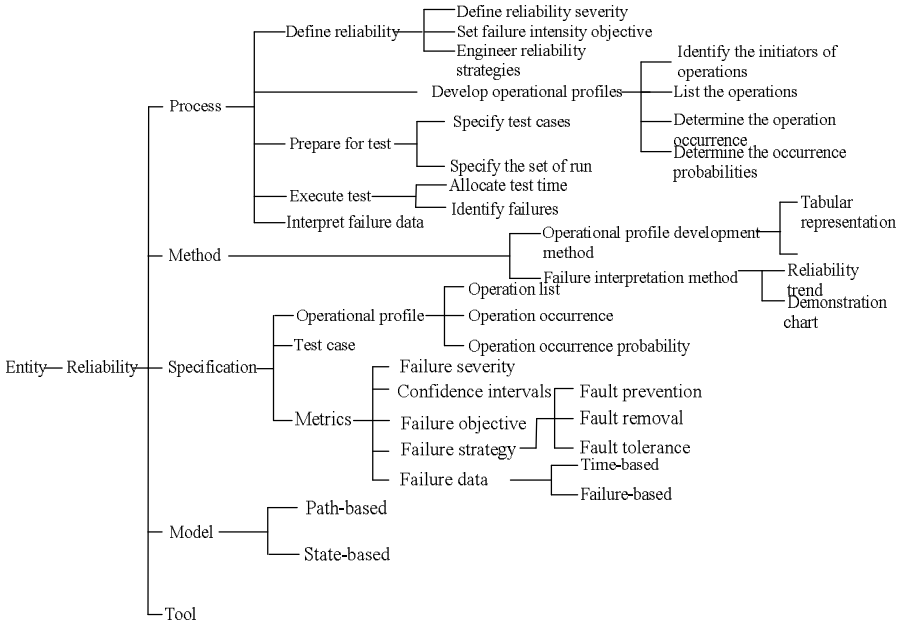


Fig. 2. Reliability concept hierarchy

Reliability process concepts

Defining reliability means quantitative reliability definition of a software system, which in turn makes it possible for reliability experts to balance customer needs for reliability, delivery date, and cost. The reliability definition mainly consists of the sub-processes of reliability severity definition, failure intensity objective setting, and reliability strategies engineering.

The operational profile development mainly consists of the sub-processes of identifying the initiator of operations, listing the operations, determining the operation occurrence, and determining the occurrence probabilities.

In preparing for test, the operational profile information is applied to planning efficient testing, including preparation of test cases and test procedures.

In executing the test, the test cases and test procedures are used in such a way that the desired goal of the efficient test can be achieved. The execution of tests involves three main activities: allocating test time, invoking the tests, and identifying failures that occur.

In failure data interpretation, the predicted reliability is compared with the set reliability objectives for guiding system-reliability decision making, e.g., accepting or rejecting an acquired component or a software system.

Reliability method concepts

A method describes how to conduct a process efficiently and effectively. The reliability method mainly consists of the methods used in the reliability processes.

The following lists the methods that are presented in [5], including operational profile development methods and failure interpretation methods.

Operational profile development methods include tabular representation and graphical representation. Tabular representation generally works better for systems whose operations have few (often only one) attributes. Graphical representation is generally better suited for software the operations of which have multiple attributes.

Failure interpretation methods. There are two approaches for interpreting software system reliability: failure intensity trend and reliability chart demonstration. The failure intensity trend method estimates the failure intensity over all severity classes and across all operational modes against time. The reliability chart method is used in certification tests in which each failure is plotted and labelled with its severity class.

Specification concepts

Specifications are the input or output of the reliability measurement processes, such as operational profiles, test cases, and reliability metrics.

An operational profile is an operation scenario that describes typical uses of the system, consisting of an operation list, the operation occurrence, and the operation occurrence probability. The operation list is a tabular or graphical representation of the operations each initiator produces. The operation occurrence refers to the occurrence rates of the operation. The operation occurrence is commonly measured with respect to time. The operation occurrence probability is the ratio of the operation compared to the total occurrence rates.

A test case is the partial specification of a run through the naming of its direct input variables and the values of these variables during the preparations for the test process.

Reliability metrics are measurable, quantitative software attributes, consisting mainly of the following items:

- A failure is the departure of program behaviour from user requirements during execution. A failure confidence interval represents a range of values within which a parameter is expected to lie with a certain statistical degree of probability.
- A failure severity class is a set of failures that affect users to the same degree or level. The severity is often related to the criticality of the operation that fails [5, 11].
- A failure objective is set for the software system.
- Failure strategies include fault prevention, fault removal, and fault tolerance. Fault prevention uses requirements, design, and coding technologies and processes to reduce the number of faults. Fault removal uses code inspection and development testing to remove faults in the code once it is written. Fault tolerance reduces the number of failures that occur, by detecting and countering deviations in program execution that may lead to failures [5].
- The failure data refers to the metric of representing failure occurrence, including the time-based class and the failure-based class. The failure-based class represents failure occurrence by indicating the frequency of the failures experienced within a time interval. The time-based class represents failure occurrence by determining the time interval between failures.

Model concepts

A reliability model usually has the form of a failure process that describes the behaviour of failures with time, also called failure random process. The possibilities for different mathematical forms to describe the failure process are almost limitless [5]. The following lists two classes of reliability models used in software architecture reliability evaluation.

The state-based models use a control flow graph to represent the architecture of the system. It is assumed that the transfer of control between components has a Markov property, which means that, given what is known of the component in control at any given time, the future behaviour of the system is conditionally independent of its past behaviour [22]. State-based models consist of the states, or externally visible modes of operation, that must be maintained, and the state transitions labelled with system inputs and transition probabilities. State-based models can be used even if the source code of the component is not available. State-based model instances include: the Littlewood model [23], the Cheung model [24], the Laprie method [25], the Kubat method [26], the Gokhale et al. method [27], and the Ledoux method [28].

The path-based models are based on the same common steps as the state-based models [22]. In addition, the path-based models enable one to specify, with the help of the simulation, component reliability estimations [29]. Path-based model instances include: the Shooman model [30], the Krishnamurthy and Mathur model [31], and the Yacoub et al. model [32].

Tool concepts

This section presents some reliability evaluation tool instances, such as SMERFS [4], CASRE[9], SoRel [33], and AgenaRisk [34].

SMERFS (Statistical Modelling and Estimation of Reliability Functions for Systems) [4] allows the end user to enter data, to edit and/or transform the data if necessary, to plot the data, to select an appropriate model to fit the data, to determine the fit of the model using both statistical and graphical techniques, to make various reliability predictions based upon the fitted model, and to try different models if the initial model proves inadequate.

CASRE (Computer Aided Software Reliability Estimation) [9, 35] is an extension of SMERFS. Users are guided through the selection of a set of failure data and the execution of a model with the assistance of selectively enabling pull-down menu options.

SoRel [33] is a tool for software (and hardware) reliability analysis and prediction that provides qualitative and quantitative elements concerning, for instance, a) the evolution of the reliability in response to the debugging effort; b) the estimation of the number of failures for the subsequent time periods to allow test effort planning and the assignment of the numerical importance by the test and/or maintenance team; and c) the prediction of reliability such as the mean time to failure, the failure rate, and the failure intensity.

AgenaRisk [34] is a risk assessment and risk analysis tool. It arms users with the latest algorithms that allow quantification of uncertainty and offer models for prediction, estimation, and diagnosis, all made accessible via a sophisticated graphical user interface.

Additional software reliability modelling tools and programs are surveyed in [36] and listed on the Web [37].

5.3. Properties definition

We categorize the reliability property into internal property and external property. The internal property describes the internal structure and attributes of concepts, and the external property describes the relationships between concepts. For an internal property, we must determine which concept it describes; for instance, specificationName is one of the internal properties of the concept Specification. For an external property, we must determine the class(es) of which the values of the property will be members, and the class(es) that will have the property as a member; for instance, hasMethod is an internal property between concepts of Method and Process. The initial reliability properties are defined in Fig. 3.

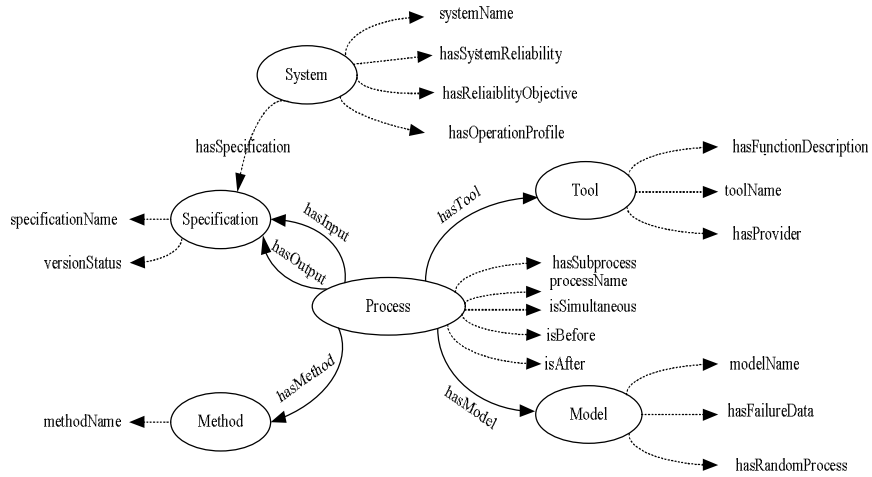


Fig. 3. Reliability properties definition

System properties

systemName: a system has a name.

hasSystemReliability: a system has a reliability objective.

hasOperationProfile: a system has an operational profile.

hasReliabilityObjective: a system has a failure intensity objective or reliability objective.

hasSpecification: a system has at least one specification.

Process properties

processName: a process has a name.

hasSubprocess: a process could have two or more sub-processes.

isBefore/isAfter/isSimutaneous: a process can occur before/after/simultaneously with at least one other process.

hasMethod: a process could have a method.

hasTool: a process could have a supporting tool.

hasModel: a process could have a model.

hasInput: a process could have an input specification.

hasOutput: a process could have an output specification.

Tool Properties

toolName: a tool must have a name.
hasFunctionalDescription: a tool must have a functional description.
hasProvider: a tool must have a provider.

Model properties

modelName: a model must have a name.
hasFailureData: a model must have failure data for input.
hasRandomProcess: a model must have a random process.

Specification properties

specificationName: a specification must have a name.
versionStatus: a specification has a version status.

Method Properties

methodName: a method must have a name.

6. Ontology-based reliability modelling system

6.1. System infrastructure

Suppose a company is planning to introduce a new software product of HomeSecurity for monitoring home environment information automatically. The system consists of a set of home automation service components (e.g. a video capture service, a light switch service, and an alarming service). Those components must satisfy quality requirements to some degree. For example, the alarm service needs a high reliability in the HomeSecurity system. In order to facilitate product reliability measurement, we have designed an ontology-based reliability modelling system, which has the primary functions of managing the software reliability ontology and designing software system reliability (Fig. 4).

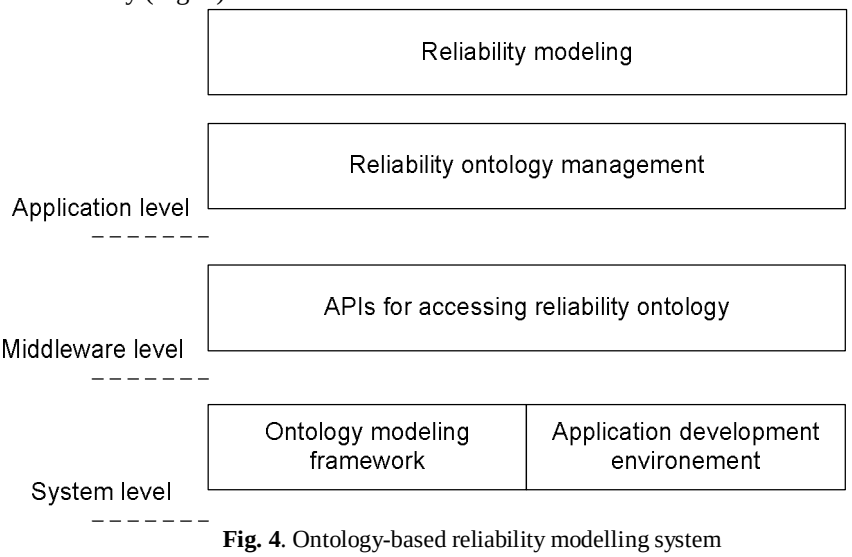


Fig. 4. Ontology-based reliability modelling system

The system level consists of an ontology modelling framework and an application development environment. The ontology modelling framework is used for reliability ontology modelling by allowing reliability experts to create reliability ontology and load reliability ontology files. The application development environment is an application development platform used for building, deploying, and managing application software across the lifecycle.

The middleware level consists of APIs (Application Programming Interfaces) responsible for handling reliability ontology, including build, access, display, and update reliability ontology.

The application level includes reliability ontology management and reliability modelling. The reliability ontology management application is responsible for reading, writing, and visualizing reliability ontology documents written in Web ontology languages (e.g., RDF [38], and OWL [39]). The reliability ontology management further enables reliability experts to make semantic knowledge queries about reliability. The reliability modelling application supports software system reliability measurement processes, including definition of reliability, development of operation profiles, preparation for and execution of reliability test, and interpretation of failure data.

6.2. System implementation

In the implementation of the ontology-based reliability modelling system, we have explored and adopted the following technonologies: Eclipse for the application development environment, OWL for the reliability ontology modelling, and Jena for the reliability ontology file management.

- Eclipse for the ontology-based system development environment [40]. Eclipse is an open source development platform comprised of extensible frameworks and tools for building, deploying, and managing software across the lifecycle. Eclipse possesses a powerful modelling framework and code generation facility (Eclipse Modelling Framework, or EMF). The EMF provides the foundation for interoperability with other EMF-based tools and applications.
- OWL (Web Ontology Language) [41] for the reliability ontology modelling. OWL enables greater machine interpretability of Web content than XML, RDF, or RDF Schema. OWL further provides additional vocabulary along with formal semantics, and adds qualifiers for describing properties and classes, such as disjointness, cardinality, and symmetry. The reliability OWL documents refer to files written with the OWL language.
- Jena API [42] for accessing reliability ontology OWL documents. Jena framework provides rich OWL APIs for reading and writing reliability ontology OWL documents. These can be used to save and read the reliability ontology in the form of files.

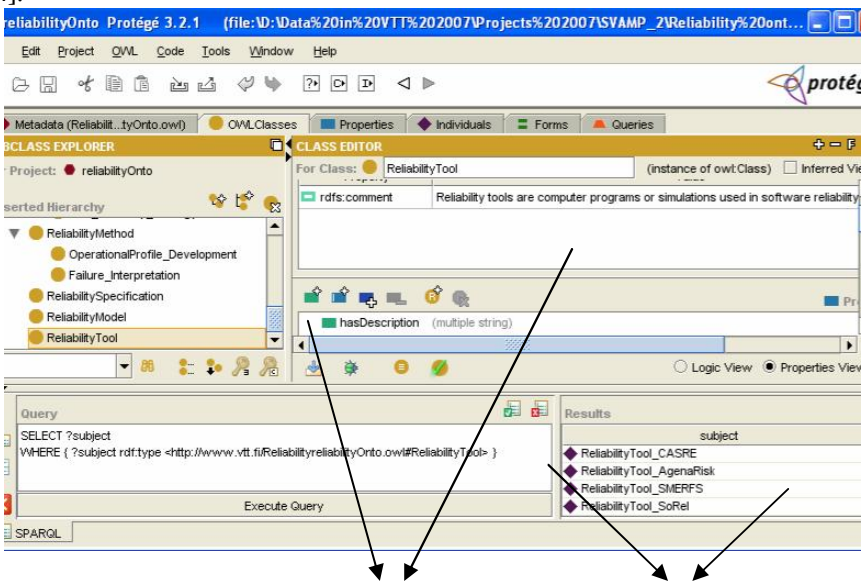
6.3. Application cases

To demonstrate the usefulness of the ontology-based reliability modelling system, including reliability ontology document management, reliability ontology query and reliability modelling, we would like to briefly mention some application cases supported by the system.

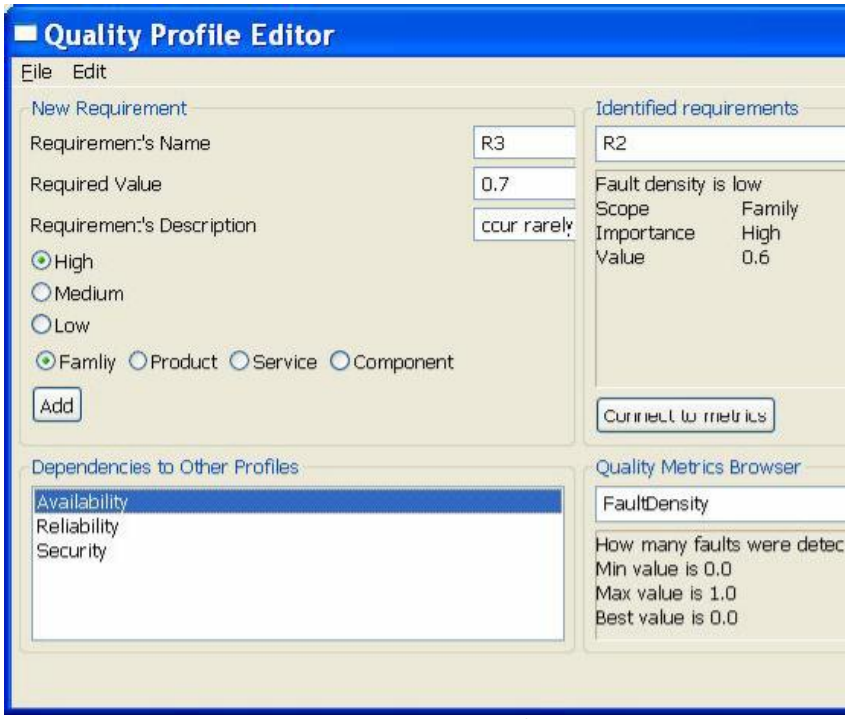
§ Reliability ontology document management. In this case, the ontology-based design environment enables reliability experts to load reliability ontologies from and to OWL files and graphically create, modify, and store the reliability ontology. At the moment, we adopt Protégé [43] to fulfill reliability ontology management (See Figure 5 (a)).

§ Reliability ontology query. In this case, the ontology-based modelling system enables experts to make semantic reliability knowledge queries through a friendly user interface, for example, Q1 (internal reliability property query): What tool instances does the class of 'reliability tool' have? Q2 (external reliability properties query): What is the output for the process of 'operational profile development' when choosing the 'tabular representation' method? (See Figure 5 (a))

§ Reliability modelling. In this case, the system enables reliability experts to conduct system reliability measurements, to define software system reliability objectives, to develop system operational profiles, to prepare and execute system reliability tests, and to interpret system reliability. Figure 5 (b) presents the user interface of the Quality Profiler. The Quality Profiler utilizes the reliability ontology and allows the end user to specify quality requirements for a given software component. The detail about the Quality Profiler module is given in [44].



a. Reliability ontology management and ontology-based reliability knowledge query



b. Ontology-based reliability-aware software modelling

Fig.5. Reliability ontology-based applications

7. Conclusions and future work

Growing attention has been given to the quality driven software design, increasing software complexity and emerging service-oriented architectures. At present, only few studies exist on ontology-based software reliability design. In the foregoing we have explored and proposed a novel ontology-based method for designing software reliability. The ontology-based method aims to provide reliability experts with a reliability ontology and related computer-aided tools for facilitating reliability engineering. First, the concepts related to reliability were specified. Next, studies associated with ontology engineering were discussed, including the objectives of creating a reliability ontology, ontology engineering methods, and guidelines for creating a reliability ontology. Further, by identifying the knowledge scopes of reliability-aware software design, the reliability ontology was designed primarily with respect to reliability concepts and properties. The experiences gained in developing the ontology-based reliability design tool were also presented. Future work will focus on elaborating the ontology-based method in the following aspects:

- Continuing the development of reliability ontology, along with the mining and refining of the reliability concepts and properties.

- Applying the method to software architecture design. This work will extend the existing reliability ontology by developing and merging a software architecture ontology and developing associated applications supporting reliability-aware software architecture design.
- Elaborating the implementation of the ontology-based reliability modelling system for specified technical features.

References

1. Kreger, H.: Web Service Conceptual Architecture (WSCA 1.0). www-3.ibm.com/software/solutions/webservices/pdf/WSCA.pdf, 2001.
2. WSA/W3C.: Web Services Architecture. http://www.w3.org/TR/ws-arch/#whatis, 2004.
3. Standard Glossary of Software Engineering Terminology. ANSI/IEEE 1991.
4. Lyu, M. R.: Handbook of software reliability engineering. McGraw-Hill, 1995.
5. Musa, J.: Software reliability engineering, more reliable software faster development and testing: McGraw-Hill, 1998.
6. Wallace, D. R.: Practical software reliability modeling. 26th Annual NASA, Software Engineering Workshop, 2001.
7. Wang, W.L., Chen, M.H.: Heterogeneous software reliability modeling. 13th International Symposium on Software Reliability Engineering, 2002.
8. Noy, N. F., McGuinness, D. L.: Ontology Development 101: A Guide to Creating Your First Ontology. http://ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html, 2006.
9. Lyu, M. R., Nikora, A.: CASRE: a computer-aided software reliability estimation tool. Fifth International Workshop on Computer-Aided Software Engineering, 1992.
10. Ramani, S., Gokhale, S. S., Trivedi, K. S.: SREPT: software reliability estimation and prediction tool. Performance Evaluation, vol. 39, pp. 37 - 60, 2000.
11. Musa, J. D., Iannino, A., Okumoto, K.: software reliability: measurement, prediction, application: McGraw-Hill Book Company, 1987.
12. Zhou, J.: Knowledge Dichotomy and Semantic Knowledge Management. In the proceedings of the 1st IFIP WG 12.5 working conference on Industrial Applications of Semantic Web, Jyvaskyla, Finland, 2005.
13. Zhou, J., Niemela, E.: State of the Art on Metamodel-Driven Multimedia over Mobile Ubiquitous Computing Environments. 4th IASTED International Conference on Communications, Internet and Information Technology CIIT2005, Cambridge, USA, 2005.
14. Lenat, D. B., Guha, R.V.: Building large knowledge-based systems: representation and inference in the Cyc project. Boston, Massachusetts: Addison-Wesley, 1990.
15. Uschold, M., King, M.: Towards a methodology for building ontologies. IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.
16. Gruninger, M., Fox, M. S.: Methodology for the design and evaluation of Ontologies. IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada, 1995.
17. Bernaras, A., Laresgoiti, I., Corera, J.: Building and reusing ontologies for electrical network applications. European Conference on Artificial Interlligence (ECAI'96), Budapest, Hungary, 1996.
18. Computer Society. IEEE standard for developing software life cycle processes. IEEE std 1074. New York, 1995.

19. Swartout, B., Patil, R., Knight, K., Russ, T.: Toward distributed use of large-scale ontologies. AAAI'97 Spring Symposium on Ontological Engineering, Standford University, California, 1997.
20. Staab, S., Schnurr, H. P., Studer, R., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems, vol. 16, pp. 26-34, 2001.
21. Roshandel, R., Medvidovic, N.: Toward Architecture-based Reliability Estimation. The International Conference on Dependable Systems and Networks, Florence, Italy, 2004.
22. Popstojanova, K. G., Trivedi, K. S.: Architecture-based approach to reliability assessment of software systems. Performance Evaluation, vol. 45, pp. 179-204, 2001.
23. Littlewood, B.: Software reliability model for modular program structure. IEEE Trans. Reliability, vol. 28, pp. 241-246, 1979.
24. Cheung, R. C.: A user-oriented software reliability model. IEEE Trans. Software Eng., vol. 6, pp. 118-125, 1980.
25. Laprie, J. C.: Dependability evaluation of software systems in operation. IEEE Trans. Software Eng., vol. 10, pp. 701-714, 1984.
26. Kubat, P.: Assessing reliability of modular software. Operation Research Letters, vol. 8, pp. 35-41, 1989.
27. Gokhale, S., Wong, W. E., Trivedi, K., Horgan, J. R.: An analytical approach to architecture based software reliability prediction. Third International Computer Performance and Dependability Symposium (IPDS'98), 1998.
28. Ledoux, J.: Availability modeling of modular software. IEEE Trans. Reliability, vol. 48, pp. 159-168, 1999.
29. Immonen, A.: A method for predicting reliability and availability at the architectural level. in Research Issues in Software Product-Lines - Engineering and Managemen, T. Kakola and J. C. Duenas, Eds. Berlin Heidelberg: Springer Verlag, 2006, pp. 373-422.
30. Shooman, M.: Structural models for software reliability prediction. Second International Conference on Software Engineering, 1976.
31. Krishnamurthy, S. Mathur, A. P.: On the estimation of reliability of a software system using reliabilities of its components. Eighth International Symposium on Software Reliability Engineering (ISSRE?7), 1997.
32. Yacoub, S., Cukic, B., Ammar, H.: Scenario-based reliability analysis of component-based software. 10th International Symposium on Software Reliability Engineering (ISSRE'99), 1999.
33. Kanoun, K., Kaaniche, M., Laprie, J.-C., Metge, S.: SoRel: A tool for reliability growth analysis and prediction from statistical failure data. The Twenty-Third International Symposium on Fault-Tolerant Computing, 1993.
34. AgenaRisk. http://www.agenarisk.com/, 2006.
35. AT&T SRE Toolkit. http://www.cse.cuhk.edu.hk/~lyu/book/reliability/sretools.html, 2006.
36. Stark, G. E.:A Survey of Software Reliability Measurement Tools. The International Symposium on Software Reliability Engineering, 1991.
37. Reliability Modeling Programs.http://www.enre.umd.edu/tools/rmp.htm
38. RDF/W3C.: Resource Description Framework (RDF). http://www.w3.org/RDF/, 2005.
39. W3C-OWL.: OWL Web Ontology Language Overview. 2005.
40. Eclipse. http://www.eclipse.org/, 2006.
41. OWL Web Ontology Language Overview. http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2, 2005.
42. Jena A Semantic Web Framework for Java. http://jena.sourceforge.net/index.html, 2005.
43. Protégé. http://protege.stanford.edu/.
44. Evesti, A.: Quality oriented software architecture development. Department of Electrical and Information Engineering. Oulu: University of Oulu, 2007, pp. 57.