A CONTEXT AWARE FRAMEWORK FOR PRODUCT BASED SOFTWARE CERTIFICATION

A CONTEXT AWARE FRAMEWORK FOR PRODUCT BASED SOFTWARE CERTIFICATION

By
VOLODYMYR BABIY, H.B.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Science

McMaster University

MASTER OF SCIENCE (2010)
COMPUTING AND SOFTWARE

McMaster University
Hamilton, Ontario

TITLE: A context aware framework for product based software certification

AUTHOR: Volodymyr Babiy, H.B.Sc. (Laurentian University)

SUPERVISOR: Prof. Ryszard Janicki

SUPERVISOR: Prof. Alan Wassyng

NUMBER OF PAGERS: xv, 137

Abstract

Software certification is becoming a reasonable expectation from the ever growing number of software users. The process of software certification could be described as a process in which an auditing body ensures that the product conforms to certain requirements. The certification models which were analyzed included product based, component based and model based. With our results and findings we have developed a proof of concept context aware framework for product based software certification. The proposed product based software certification process is structured on component based certification principals, while the general core of the certification process is composed from known software certification models.

The framework was developed within an OSGi (Open Service Gateway initiative) environment which is being managed by a collection of automation scripts. The certification models which we reviewed did not represent their knowledge formally and did not have any mechanisms to derive indirect knowledge. To address this issue, we have developed an upper ontology to formally model higher level concepts for the certification, and described a general metric for assigning consistent weights to ontology classes. The framework provides a dynamic environment for the certification process by integrating development and certification domains with the help of ontology. Its main objective is to allow the certification process to be able to adjust to ever changing certification demands and extend more easily into different domains.

. The developed ontology can maintain many properties and attributes, but for some of these the measuring mechanism are unknown. Therefore, we have described the process on how to derive software metrics for measurable and subjective attributes which can be used to evaluate product, processes and resources. In conclusion, we have outlined some areas for future research.

Acknowledgments

I would like to thank my supervisors professor Ryszard Janicki and professor Alan Wassyng for their valuable teaching, support, and help. Working on this thesis research has benefited me greatly. They gave me an opportunity to gain experience through their mentoring and taught me how to conduct proper academic research.

I would like to thank all my colleagues who I have met during my time at McMaster University. There feedback and support has been invaluable during my research. They have not only helped me become better in academia but in life.

Also, I would like to thank everyone else who had either a direct or indirect influence on the research. Specifically, I would like to thank my family and friends.

Hamilton, Ontario, Canada — Volodymyr Babiy

May, 2010

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Overview

The process of software certification is time consuming and expensive. Therefore, some smaller software systems which are not critical may never go through a certification process. The majority of software is still being developed without any consideration for software certification. Validating these types of software is a difficult task and undoubtedly error prone. Due to substantial certification costs, it is desirable to develop an independent tool for the certification process. This approach would not be easily implemented because the tool would have to be modified for almost every project [61]. In a computer system, software is considered as one of the most complex components. At the same time it is considered as one of the most error prone, despite the increasing demand for reliable software. As a result, there is an apparent need for a viable dynamic software certification process [58] which can adjust rapidly to the dynamic demands of the rapidly evolving software industry.

The desired situation for certification would be one in which a third party would certify the software. This approach is preferred because it could eliminate biased opinions which could be present in in-house evaluations. This approach may also help to spread liability which could have a negative effect on the customers

and developers. Sometimes the third party is not able to thoroughly evaluate every component of the system, therefore they may use a spot check technique where the auditor issues a certificate based on a thorough evaluation of a few components of the system. It is impossible to achieve an accurate understanding of the product from only being able to thoroughly evaluate a few components of the system. Evaluation of only a few components of the system may lead to an opinion which could be misleading. By allowing biased opinions we create a risk of faking the software certification process. As stated by Voas et al. [85], the certification process should include an evaluation of resources, processes and final product. This is also known as 'the software quality certification triangle' [85]. From the certification point of view, the product is the most important entity. Well established software development practices and methods do not guarantee a quality product, while reviewing processes and resources could add extra assurance that the product was developed in an industry standard environment [89].

Software systems are developed for different purposes. Properties such as safety, reliability and modularity could have different priorities for different projects. Certification providers are obliged to provide a guarantee that their software will operate reliably, but a certification process can not imply nor guaranty that the software will not fail in all unexpected situations [86]. For example, some factors which could affect software performance and lead to failure could be unexpected hardware failures, or any other factors which can not be controlled or imitated during the evaluation process. It is difficult to certify products by taking into account unexpected behaviors. In certain environments, where hardware failures or other failures are unpredictable, there should be a claim stating that everything possible was accomplished to achieve a desirable behavior of the product. Unexpected situations often could occur in situations where the software is integrated in a very dynamic environment. It is therefore unrealistic to expect that the software will perform exceptionally well in all scenarios.

In some domains, such as nuclear, failure of the software is not accepted. This is because software failure, in a very critical domain, could lead to a serious consequences. Software certifiers usually state conditions under which the software was certified, and the final output of the certification process clearly outlines what has been and has not been evaluated during the certification process [65], [58], [6].

The current proposed approaches to software certification, which were reviewed during the research, outline what must be evaluated, but they do not outline how to evaluate the product. They also lack an applied infrastructure which could support certification of the larger projects. This could be considered as one of the main reasons why companies choose not to certify their products. The main goal of software certification is to provide stakeholders and users with reassurance that the product possesses low risks of failure and conforms to requirements. The entities which could be evaluated during the certification process could consist of two domains. The first domain is composed of entities which are measurable by some accepted methods. The second domain is composed of entities which are subjective and require direct expert evaluation. We should mention that the majority of automated measuring methods heavily rely on an expert's input [84]. The following software certification hypotheses were deduced by Keith and Vertinsky et al. [40] during their research.

- Hypothesis-1: *"Companies that are in a competitive market will be more likely to choose to certify than those companies that have relatively little competition."*
- Hypothesis-2: *"Companies that have a higher exposure to risk in their projects will be more likely to choose to certify than those companies that face relatively lower risk."*
- Hypothesis-3: *"Companies that produce products are more likely to choose to certify than are companies that provide services."*
- Hypothesis-4: *"Companies that have larger project teams will be more likely to choose to certify than those companies that have relatively smaller project teams."*
- Hypothesis-5: *"Companies that are more methodologically rigorous are more*

likely to choose to certify than companies that are relatively less methodologically rigorous."

- Hypothesis-6: "6-1: *Larger companies are more likely to choose to certify than are smaller companies.*", "6-2: *Smaller companies are more likely to choose to certify than are larger companies.*"
- Hypothesis-7: *"Companies with a corporate culture that values quality are more likely to choose to certify than companies that value quality less."*

The case study included twenty nine questions which were related to the probability of a company to certify their software. In addition, another twenty five questions were asked in order to test the validity of the hypotheses. The actual survey occurred in the fall of 2005 and in the spring of 2005. It is very unlikely that views on software certification have changed since 2005. The participants for the survey were selected from the Canadian software industry [40]. The hypotheses above may not apply to companies which certify their products due to customer requests or government regulations. Companies are faced with two choices which are either to certify their products or not, and risk a chance of not being considered as a provider of credible products. The above hypotheses were tested on 235 participants. Out of these, 79 were not contacted successfully and of the remaining 156 participants who were contacted successfully, only 100 agreed to participate. The remaining candidates were either not willing to participate, or considered software certification as an unrelated subject.

A competitive environment is the most important driving factor for companies to certify their software. Some of the most competitive markets are medical, civil aviation, automotive, military, etc. The desire for companies to certify their software may be driven by their ability to increase sales and to maintain a competitive advantage in the industry. This comparative advantage could be achieved if companies would develop their products while conforming to the product's requirements and industry regulations. Another factor which may influence software certification is team

size. This comes from the fact that larger projects may require rigorous maintenance compared with smaller projects [40], [14].

1.2 Scope

Objective one was to analyze software certification methodologies, such as product based, component based and model based. Objective two was to evaluate current issues with software certification and approach it from a practical point of view. Therefore, not only to outline which properties should be evaluated during the certification process, but develop a product based certification process which would be supported by an applied proof of concept context aware framework. The framework had to support certification of the intermediate or final product and its development was driven by hurdle eight of software certification which was defined by Hatcliff, Heimdahl, Lawford, Maibaum, Wassyng and Wurden et al. [31]. The description of hurdle eight is given below.

'Lack of interoperable tools to manage, reason, and provide traceability - The result is that small change often requires a large effort. We need tools that scale.'

In addition, the *Hypothesis-5* that was described by Keith and Vertinsky et al. [40], and listed earlier, was considered during research. In order for the software certification process to be successful it should be integrated with current methodologies and aligned with standards from the ISO and IEEE domains. Specifically, we focused on software which cannot be fully verified with formal verification due to its size and complexity.

1.3 Contribution

Based on our results and findings we have developed a proof of concept context aware framework for product based software certification. The framework provides an applied environment for the software certification by integrating tools such as Eclipse, Jena, OntoStudio, Protégé, Equinox, EMMA, log4j, RCP, JUnit, Jfeature and Apache Ant. It was developed within the Open Service Gateway initiative (OSGi) environment and is managed by a collection of automation scripts. The framework is intended for the certification of software which are developed with high level languages.

We also proposed a product based software certification process that is structured on component based certification principles and integrates current methodologies such as Integrated Component Maturity Model (ICMM), PECA framework (Plan the evaluation, Establish criteria, Collect data and Analyze data) and Goal Question Metric (GQM). The process also tries to be aligned with ISO JTC1 SC7, ISO IEC 25000, ISO 15939, ISO IEC 14598 and ISO 9126 standards. The general core of the certification process is composed from a variety of known software certification models. Therefore, we gathered the benefits of other software certification models into a single model. At first every component is evaluated independently and then a global certificate level is computed which depends on the evaluation status of every component.

We have developed a higher-level ontology in OWL (Web Ontology Language) to formally model knowledge for the product based certification process. The objective of the upper ontology is to represent knowledge formally and consistently throughout the certification process and provide an environment for reasoning. Some of the developed upper ontologies are: DOLCE, BFO, Cyc, GFO, Sowa's ontology, PROTON and SUMO. Suggested Upper Merged Ontology (SUMO) is one of the biggest upper ontologies. With its domain ontologies it has about 60,000 axioms and

20,000 terms. Initially, it was designed by Teknowledge Corporation. Currently, it is owned by IEEE and is available for public usage under GNU General Public License. Upper ontologies are often used in domains of reasoning, linguistics and search applications [54]. Kluge et al. [43] defined ontology as a 'formal specification of a conceptualization', or it can be defined in detail as:

'An ontology specification is a formally described, machine-readable collection of terms and their relationship expressed with a language in a document file. A conceptualization refers to an abstract model of a domain that identifies concepts.'

An upper ontology and ontology are almost identical in their definitions, except upper ontology may model more general concepts of the domain while ontology may model more specific concepts of the domain. We argue that there should be a consistent understanding of the certification process in order for it to become a widespread practice for both industry and academia. The consistent understanding of the certification process, knowledge capturing and knowledge sharing can be achieved with the help of an ontology. With its help, a dynamic certification environment can occur by bringing together the development and certification processes, because ontologies allow for the knowledge to be freely shared between all the participants in a consistent and formal manner. This dynamic environment can expose every participant to the certification process, because the development and certification environments are integrated. The approach addresses issues which are currently present in the certification processes, where the developers and the certifiers maintain different perspectives and understanding of the certification process. Everyone should have a clear understanding of the certification process in order for it to be predictable and consistent. The framework allows for the components to be developed, maintained and certified in parallel that is achieved with the help of integrated tools.

During our research we did not come across an acceptable measuring mechanisms for many attributes. For example, attributes such as quality and complexity.

Therefore, we have described a process of how to derive software metrics for measurable and subjective attributes which can be used to evaluate product.

There could be a large number of attributes which can be considered during the certification process and in situations where it is difficult to use an algorithm, we revert to the use of heuristics. There should be a process to assign consistent priorities to attributes and software metrics as projects evolve and grow in complexity. The pairwise comparison method is ideal for this task, because it can reduce inconsistencies while still maintaining some acceptable margin of error. In addition to PC method, the Multiple Attribute Decision Making (MADM) method could be used to model the scenario for ranking alternative plans in situation where one ore more experts are present. Every expert would provide one or more alternative plans. The consistency of every alternative would be achieved by the pairwise comparison (PC) method. The actual ranking of alternative plans would be accomplished by the Simple Additive Weighting (SAW) and the Weighted Product (WP) methods. An example showing how these methods could be applied to assign consistent priorities and to rank alternative plans is given in appendix A. The consistent priorities are also known as consistent weights.

1.4 Declaration

Some content from this thesis has been published. A simplified least squares method (LSM) algorithm, as an alternative algorithm to the proposed algorithm by Bozóki et al. [11], was accepted by the Central European Journal of Operations Research (CEJOR) as a follow up paper [4]. One of the co-authors of the submitted follow up paper was Bozóki. The general description of the algorithm is given in appendix A. The remaining content from the appendix A was submitted for review to the C^3S^2E (C^* Conference on Computer Science & Software Engineering) [7].

1.5 Thesis outline

The remaining chapters are organized as follows:

Chapter 2 describes the general process and algorithms for product based software certification and why this method was selected over other certification methods, such as process based and model based. In addition, a brief background and comparison is given for some known process based and product based certification models.

Chapter 3 describes a context aware framework for product based software certification. The proof of concept framework was developed within an OSGi (Open Service Gateway initiative) environment which addresses key limitations which are present in some software certification models.

Chapter 4 describes the process of how to derive software metrics for measurable and subjective attributes which can be used to evaluate product. In addition, this chapter discusses the history, purpose and motivation behind software metrics.

Chapter 5 summarizes the work which was accomplished in this thesis and suggests areas for potential future research.

Appendix A describes an approach on how to assign consistent weights to ontological classes and ranking of alternative plans. The consistent weight assignment can be achieved through the Pairwise Comparison (PC) method. Multiple Attribute Decision Making (MADM) method could be used to model the scenario for ranking alternative plans.

Appendix B maintains a collection of an automation scripts and their output.

Appendix C provides instructions on how to obtain required software, checkout framework from the SourceForge repository and configure it on the local system.

Appendix D contains an upper ontology in OWL (Web Ontology Language) for the product based software certification.

Chapter 2

Product based certification process

This chapter describes the process and algorithms for product based software certification, and why this method was selected over others, such as process based and model based methods. The ontology was used to formally model knowledge for the software certification process. A brief background is also given on process based and product based software certification methods.

2.1 Product based certification

The objective of product based certification is to deduce whether the product conforms to requirements and provide an evaluation of the developers abilities to produce new products while conforming to requirements [76]. ISO IEC 14598 provides instructions on how to evaluate a software product. It uses ISO IEC 9126 standard which describes how general attributes could be subdivided into less general attributes. In practice, both standards are often applied in parallel. ISO IEC 14598 has four phases: defining evaluation requirements, identifying evaluation, building evaluation schedule and executing evaluation schedule. In the defining evaluation requirements phase, attributes and subattributes for the product evaluation are defined. These attributes

and subattributes could be taken from the McCall's and Blundell's quality models [26], [10]. In the identifying evaluation phase, a collection of metrics are defined for the evaluation of all attributes and subattributes. In addition, metrics which will evaluate relationships between a product and its environment are also defined. In the building evaluation schedule phase, a detailed evaluation plan is constructed. Finally, in the executing evaluation schedule phase, the evaluation schedule is executed [51].

2.2 Process based certification

IEC 61508 and DO-178B standards describe the software certification process by following a process based methodology. These standards describe a collection of practices which should be followed during the software development. They claim that it would be easier to achieve validation and verification of the software by following the proposed practices. IEC 61508 and DO-178B standards should be followed in correlation with other regulations where they outline significance of the software failure. The Development Assurance Levels (DALs), from the domain of civil aerospace, are an example of this correlation which dictate critical levels of rigorous. The automotive and European rail industries use Safety Integrity Levels (SILs). (DALs) and (SILs) are not similar in their applications, despite their strong tendency to focus on the risk reduction. The more critical the software, the greater the need for risk reduction to be an essential attribute of the software. Greater demands upon (SILs) and (DALs) lead to stricter demands from the process of software development. DO-178B standard argues that the verification of a system should be accomplished through extensive testing, while highly emphasizing the need for a good traceability process and a manual review of the components [41]. The verification process supported by DO-178B standard is subdivided into four levels. There are twenty eight properties at the lowest level, D. They validate tools, high level requirements, and configuration

of the development process. The next level, C, deals with twenty nine properties. They validate low level requirements, testing and code coverage. The next level, B, deals only with eight properties and logic. The highest level, A, is responsible for the sixty six attributes. At this level, while the overall quality of the product is being evaluated, a significant attention is being allocated to traceability [41].

2.3 Hurdles with process based certification

There is a collection of disapprovals related to the process based certification. The most important claim, which disproves credibility of the process based certification, claims that it is very difficult to maintain evidence which would claim that specific processes are able to achieve and maintain a certain level of reliability and integrity. It is also difficult to relate failures with processes.

Certification type	**Pros**	**Cons**
Product based	This approach could be applicable to almost all software products and this method is free of software development processes.	Most of the time, testing methods that are being developed cannot keep up with the demands that are needed to test new products.
	The software attributes are all known during the evaluation and are evaluated independently. The test results, documentation, formal proofs etc. are all directly related to the quality of the software.	Large companies are less willing to release all their documents related to the software.
	It is possible to eliminate some redundancy in the evaluation if properties overlap in its descriptions and requirements.	Software testing and formal verification is costly and could be time consuming. These extra costs and possible delays may lead to shorter lifecycle of the software.
Process based	The same process could apply to a range of software that could belong to different groups.	Software is very diverse and innovative, therefore some processes may by inappropriate and infeasible for the development.
	The overall cost of certifying software by following a process based certification method could be lower compared to product based certification method.	There is no evidence to support a claim that good software development processes will lead to the development of good quality software.

The time it takes to evaluate software processes may not heavily influence the lifecycle of the software.	Small companies may not have sufficient resources to implement and maintain complex processes.

Table 2.1: Pros and cons of product and process based certification

The strong theory and regulations of the process based methodology can prevent other processes from being implemented. The superior flexibility of the system could be achieved with the Model Driven Development (MDD). If the methodology does not allow for the new processes to be introduced, then some components of the system could be limited to inadequate development processes. It is also could be difficult to transfer software certification judgments from one domain to another. The software certification judgments which must be achieved in order to satisfy the IEC 61508 SIL 4 standard cannot be easily correlated with judgments which must satisfy level A of the DO-178B standard [41]. Potentially one of the most crucial hurdles associated with the process based certification, is that the arguments and evidence which support the product do not provide a sufficient guarantee concerning quality. This is because they could be indirect. This strong argument raises an important question, whether it is even worth considering process based certification as one of the valid methods for the software certification. David Lorge Parnas mentioned that instances of the processes are not always perfect. Due to nature of the software development, processes are not always perfect and may contain work arounds. Therefore, they maintain some form of backtracking, or other imperfections which may not be a part of a formal process description. At the end of the project, it is possible to present software development processes without any imperfections [52]. This is why merely looking at processes of the software development is not sufficient, because it will not provide sufficient amounts of information which is needed for the software certification [53], [52], [77], [38], [63]. Table 2.1 describes some pros and cons of the

product based and process based certification methods.

2.4 General hurdles with certification

To the best of our knowledge, there is no well known and generally accepted scientific method for the software certification. However, software certification is on the path towards becoming a widely used practice, as consumers are starting to prefer software which has been certified. In particular, important government projects may soon be mandated to have software certification. Software certification can be described as a process in which an auditing body ensures that the product conforms to a given set of requirements. There seems to be no alternative to software certification. In the absence of software certification, it comes to a *'trust me'* stated by the software developer, which goes against a well known doctrine in the justice system: *'no one can be his own judge'*. In many situations, any failure in a system could endanger human well being; therefore potential failures must be eliminated. This creates the need for a requirement where the system must be evaluated through testing, formal verification and manual review, before it gets delivered to the customer [32]. Most software certification methods require risk to be evaluated rigorously. Risk represents the combination of undesirable outcomes and their probability. In the aviation industry, any type of failure *'those which would prevent continued safe flight and landing'* must be very unlikely. This statement could be translated to a requirement where any type of software failure should not occur, not only during the flight, but throughout the entire life of a particular aircraft. The uncertainties of the software certification could be subdivided into two groups: the possibility of undesirable outcomes and the effectiveness of arguments that guarantee the claimed likelihood of failure. If human judgments are present, then they must be evaluated and analyzed with some mathematical framework to reason about the uncertainty of the judgments. Several

mathematical frameworks such as fuzzy logic, possibility theory and Dempster Shafer theory have been proposed to deal with this issue [40].

Companies could gain financial value from products which are certified. However, software certification process could be very expensive and time consuming. It may also require specialists, which may not be available. These, and other issues, could prohibit companies from certifying their products. Importantly, there are a number of benefits which companies could gain by certifying their products. Companies could gain confidence in their products and maintain a competitive edge in the industry. Certification could also increase consumers' confidence in the product. With the certification system in place, companies could prove that their products conform to the regulations and requirements. Especially, in situations where some development is being outsourced. Overall, software certification could prevent poor quality software from being developed [32], [76].

2.5 Capturing and sharing knowledge with a formal model

Ontology could be used to formally model knowledge for the product based certification. This approach was selected because it adds flexibility to the software certification process. Participants can add and remove facts freely, while contributing new knowledge. The intention was to develop an upper ontology where knowledge from different experts could be captured and utilized throughout the certification process.

2.5.1 The motivation behind ontology use

The software certification process possesses a vast amount of knowledge, facts, regulations, standards, etc. Therefore, an ontology is an ideal formal mechanism to capture

all that knowledge. Other industries, such as the medical and chemistry, have already utilized the benefits of ontologies. Ontologies have an ability to support equivalent classes which allow for the construction of different certification models for different domains. With an upper ontology it is possible to achieve the following goals:

- Commonalities in current proposed certification models can be identified.
- Certification gaps can be addressed.
- Acceptable and non-contradicting approaches for the evaluation of components and attributes can be maintained.
- A formal model where facts, goals and measuring processes could be collected.

The upper ontology provides a mechanism where criteria can be captured and identified with a complete metadata. Most importantly, it is possible to identify and model relationship between different criteria. It serves as a framework for capturing knowledge from standards and known certification methods. The objective was to build an upper ontology which would allow for situations to occur where criteria could conform to terminology and descriptions which are supported by the industry and academia. This also includes measurement areas which are very large in terminology and practices. The other objective was to be consistent with ISO/IEC and IEEE organizations, which in 2002 made a decision to have their terminology as consistent as possible and aligned with accepted international standards. ISO JTC1 SC7 standard tries to follow this objective [27].

2.5.2 An approach to build ontology

There are few different ontology building methodologies. Ontology Engineering (OE) is still a fairly young discipline, despite the existence of only a few ontology building

methodologies. Almost every proposed building method utilizes its own formalism and methods. The steps to build ontology could be subdivided into four components:

- **Specification:** All goals are gathered and documented. The document outlines the main objectives of the ontology, scope and abstraction level. Overall, the main objective is to identify classes and their metadata.
- **Conceptualisation:** After some knowledge is gathered, it is in unstructured format which must be structured. During this stage, knowledge is structured with the help of the external representation language. This language is independent from the implementation language. Knowledge takes a semiformal structure where the domain exports and ontology builders can start discussing future modifications.
- **Implementation:** During this stage conceptual models are implemented with the formal languages, such as RDF/S (Resource Description Framework / Schema), Ontolingua or OWL.
- **Evaluation:** During this stage the ontology goes through the rigorous technical evaluation. This evaluation is gathered from the domain experts [21].

The Web Ontology Language is one of the preferred and dominant standard in the industry. It was built based on the Description Logic (DL) model. OWL utilizes Vocabulary Description Language (VDL) for its syntax. There are three different types of OWL: OWL-Lite, OWL-DL and OWL-Full. Protégé software was selected to develop an upper ontology in OWL-DL. OWL-DL supports all the constructs which are required to express ontologies. The most expressive language is OWL-Full. It is used only in situations where elaborate expressiveness of the language is desired. Therefore, it is very difficult to guarantee language completeness and practically impossible to carry out an automated reasoning [34], [82].

2.5.3 Ontology structure

The basic building blocks of the ontology are classes (D), concepts, axioms, instances, retaliations (R) and functions. The Description Logic (DL) is one of the best languages to describe knowledge representation. This language is utilized in many fields and has been used to describe knowledge in fields such as medicine and nuclear engineering [60]. Classes represent concepts, which are objects and instances of entities. They are abstract sets or collections of objects which may contain both individuals and other classes. The axioms are known as a first order logic sentences, which do not require formal proofs because they are believed to be true. An instance is a representation of a specific entity which belongs to some specific domain or domains. The instances can be concrete entities, such as animals, tables, planets, or abstract entities such as words and numbers. The relations are utilized to model the relationships between different terms, instances and classes. The relationship between the two different terms is called a binary, and in situations where it is used among n terms it is called a function. It is a special relation where certain terms are related precisely to one another term. For example, the binary relation could be connectedTo, subclassOf, partOf, etc. It is apparent that even with these basic constructs, it is possible to build a knowledge representation framework where manual management is not feasible due to its complexity [74]. Ontologies are ideal to capture and represent domain knowledge. Their main purpose is to maintain and describe relationships between concepts. OWL language is one of the recent standards which immerged in the industry for the purpose of building formal ontologies. This standard is supported and recommended by the World Wide Web Consortium (W3C). It supports construction of the complex concepts where they could have more then a single parent.

The knowledge base, which is based on the description logic, is separated into two parts: Terminology Box (TBox) and Assertion Box (ABox). The TBox maintains

axioms such as $D_1 \sqsubseteq D_2$, and the ABox stores facts and knowledge about the entities. For example, the role assertion $R(x, y)$ means that x and y are related over a role R. The entity assertion is known as concept assertion $D(x)$ which asserts if entity x belongs to class D. Roles and concepts can be referred to automatically by using their global description names. It is possible to construct complex statements out of roles and concepts with the help of constructors. This is why description logic is so powerful in its expressiveness. As mentioned, Protégé software was used to develop an upper ontology for the product based software certification. This software was selected because it has a built in reasoner, FaCT++, which can validate definitions and statements for the consistency. It can also determine to which definition every concept should belong [34]. Table 2.2 describes some constructors and axioms which are used by the upper ontology [29].

Constructor	**Description in DL**	**Axiom**	**Description in DL**
Union	$D_1 \sqcup D_2 \sqcup ... \sqcup D_n$	Subclass	$D_1 \sqsubseteq D_2$
Intersection	$D_1 \sqcap D_2 \sqcap ... \sqcap D_n$	Equality	$D_1 \equiv D_2$
Negation	$\neg D$	Subproperty	$P_1 \sqsubseteq P_2$
One from	$\{x_1, x_2, ...x_n\}$	Same property	$P_1 \equiv P_2$
Existential restriction	$\exists R.D$	Disjoint classes	$D_1 \sqsubseteq \neg D_2$
Value restriction	$\forall R.D$	Same entity	$\{x_1\} \equiv \{x_2\}$
Has property	$\exists R.\{x_i\}$	Different entity	$\{x_1\} \sqsubseteq \neg\{x_2\}$
Max restrictions	$\leq nP.D$	Inverse property	$P_1 \equiv P_2^-$
Min restrictions	$\geq nP.D$	Transitive	$P^+ \sqsubseteq P$
Restriction	$= nP.D$	x type of D	$x : D$

Table 2.2: Constructors and axioms [29]

The partial visualization of the ontology is shown in Figure 2.1 and it was generated with the OntoStudio software [72]. The complete upper ontology is given in Appendix A, and is described with Web Ontology Language (OWL). Some of the data properties, which are utilized in the upper ontology, are given in Table 2.3. They are also known as relations. The entities which are described in Tables 2.4, 2.5 and 2.6 were modeled in the upper ontology as classes.

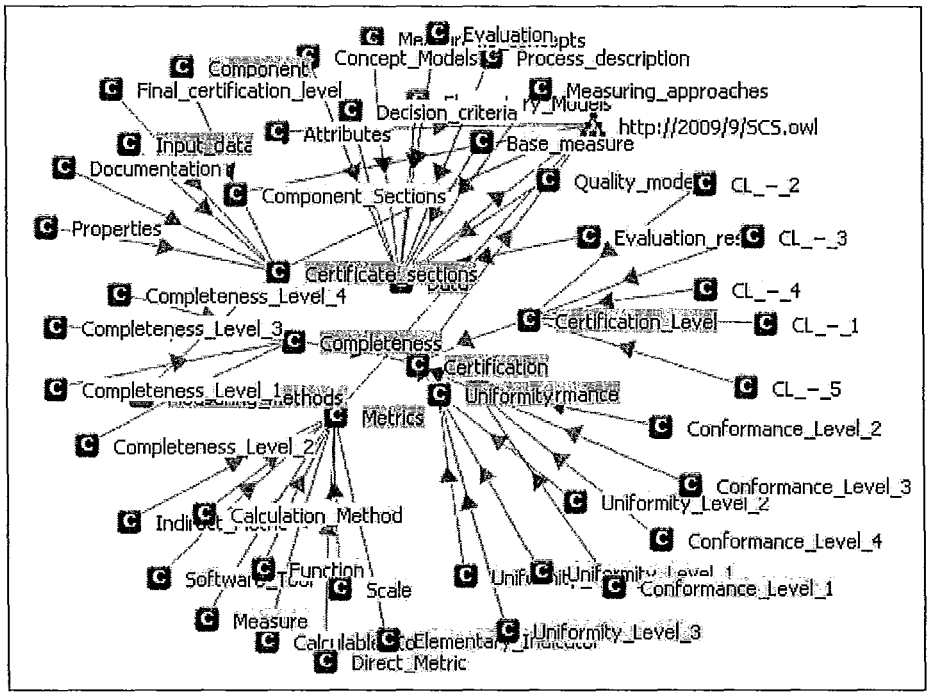

Figure 2.1: The partial visualization of the ontology

2.6 Certification process

The described certification process is based on the product based methodology and incorporates a collaborative approach between developers, owners, stakeholders and certifiers. There are a few advantages from having third party certifiers, for example; biased, unfair and accelerated evaluations could be eliminated. It is believed that third party evaluations are the only way to certify software in which customers should trust [89]. General definition of third party certification was defined by Councill et al. [18].

'Third-party certification is a method to ensure that software components conform to well-defined standards; based on this certification, trusted assemblies of components can be constructed.'

Product based certification may possess a list of challenges. These challenges could be influenced by the same issues which occur in the software development

Ontology relation	Notion	Description
belongsTo	Scale or collection of scales	Every scale or collection of scalers is associated with a scale type. A single scale type can be associated with a collection of classes.
conversion	Conversion function	It is possible for two or more measurement functions to be associated with each other.
articulatedWith	Unit of measure	Every measurement is associated with a unit of measure. Unit of measure is used to communicate the objective of measurement.
maintainsScale	Scale of measure	Every measure must have a scale associated with it.
measuredWith	Measurement function	Every derived measurement value is achieved by applied one or more measurement functions.
composedOf	Collection of sections	Entity within the ontology can be composed out of other entities.
has	Entities have properties	Entities can maintain properties which contain values.

Table 2.3: Description of the ontology relations [27]

domain. They could be related to economical, organizational and technical areas. One of the most crucial issue with product based certification is how the developers of any system could obtain a desirable level of trust from the customers. The issue is driven by the fact that the behavior of the system, as different components are interrelated, is not always being considered in the certification process [49]. To address this issue and others which are mentioned below, we have described a process for the product based certification which is based on the component based principels. At first every component is evaluated independently and then a global certificate is being computed which depends on the status of the components. This approach was selected because industry supports a collection of component development practices, including CORBA, COM, JavaBeans, COM+, .NET and OpenDoc which can be integrated into the certification process. This could allow for the certification process to span more easily into different domains [39].

The research driven by component based principles could be subdivided into two streams. The first stream deals with formalism, and addresses questions on how

to construct formal methodologies in order to foresee component characteristics. The other stream deals with an actual quality of the component, and attempts to answer questions on how to construct quality models which deduce quality of the attributes [2]. In 1999 and 2000 the Software Engineering Institute (SEI) from the Carnegie Mellon University conducted a study, from the business and technological point of view, regarding the use of components in the software development process. The following concerns were the most widely spread:

- The desirable components are not always available.
- The components are being developed without following industry standards.
- The certified components are not always available.

With the help of the internet, the first two issues are of lesser concern, but the third issue has not been addressed properly [2]. Our described methodology addresses areas which are known in software certification as magic steps (ref. Prof. Wassyng). With component based software development it is usually the case that the user of a component and its creator are the different parties. Therefore, users have to determine from the collection of off-the-shelf (COTS) components which component will fit into their requirements. This situation almost always requires some evaluation by the user. The quality of the component is usually determined though a series of tests and manual checks and only if the source code is provided. This process is expensive and any average size project could require many different components. There could be a situation where validation of the component is being accomplished by every user. This collaborative cost could be eliminated if a third party would certify components and customers would utilize components without implementing in-house evaluation process. Many companies experience difficulties while selecting appropriate components for their systems [17]. It is apparent that component based certification could benefit not only developers and users, but the industry as well [59]

[78]. Therefore, we believe that in order for the product based certification process to be successful, it should be driven by component based certification principles.

More and more systems are being developed by utilizing COTS components, which means that consumers will start expecting components to be certified. The current methods of validating COTS components through statistical analysis and fault injection are not sufficient to offer reassurance that components are of a good quality [58]. The components should be classified prior to any evaluation, and based on the classification, different certification techniques should be applied. There is no well known method or standard to classify components or software. ISO IEC 12182 standard describes a general framework which outlines concepts of classification, but the framework is not very specific. It would be very difficult to group components of the software into groups based on their permissible uses. Despite this difficulty, some classification groups have been proposed. There are four major classification groups: attribute value classification, free text keyword indexing, faceted classification and enumerated classification [61].

The idea of reuse is the other compelling reason for executing product based certification by following component based certification objectives. Currently, some sections of the software are being developed by using existing components which can be interoperated. The basic idea of reduced development time and a possibility to make a good quality products from a good quality components makes this method of certification very attractive. Current literature is not very rich on results related to practical component based certification methods, but there has been some work done on this method of certification in the academic area. The research on component based certification could be divided into two time lines. The first between 1993 and 2001, where mainly all certification models were focusing on mathematical and test based approaches, and the other from after 2001 where the focus shifted to predicting

quality of the product and its behavior. There is always some uncertainty with quality prediction, and it is difficult to predict how products would behave in unforeseen situations [3].

2.6.1 Structure of the certification process

The product based certification process is subdivided into four stages. They are the data gathering stage, component identification stage, metrics construction stage and component evaluation stage. It is apparent that in order to achieve valuable evaluation results, it is necessary to follow a well outlined evaluation process. This does not necessarily indicate that the evaluation process has to be very complex, but if the evaluation process varies than it is possible that the results will vary as well. The collaboration between the National Research Council of Canada (NRC) and the Software Engineering Institute (SEI) from the Carnegie Mellon University have developed a PECA framework which stands for Plan the evaluation, Establish criteria, Collect data and Analyze data. We have incorporated this evaluation framework into the product based certification process. The PECA framework consists of an ongoing process where the evaluation and data gathering takes on a spiral effect. PECA framework is demonstrated in Figure 2.2 [17].

2.6.2 Stage 1: data gathering

Collection of data requires a knowledge of the evaluation processes in order for the meaningful content to be collect. Under different circumstances, different collecting methods will have to be applied. For example, collecting methods for criteria of significant value would be more rigorous, while it could not be the same for criteria which are of a smaller value. The data gathering methods must reflect the degree of confidence that is planned to be achieved in the final evaluation [17].

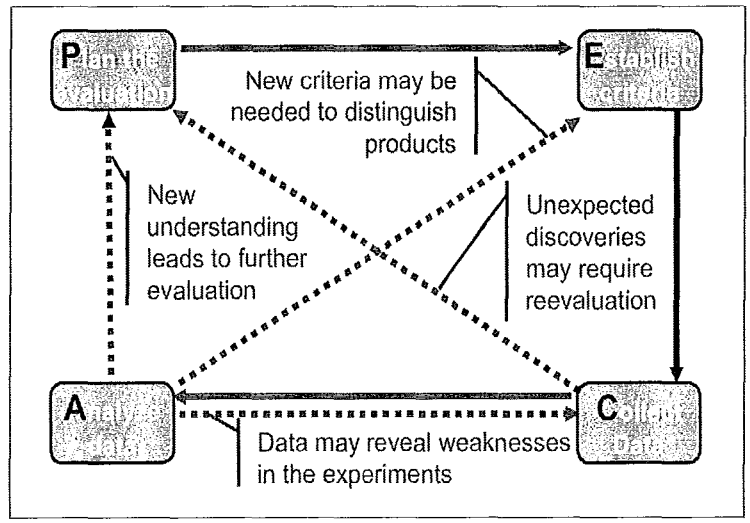

Figure 2.2: PECA framework [17]

The data for the certification could be separated into two parts. The first part could contain the sections of the component, and the second could contain attributes which are considered in the certification process. Every component is associated with six sections which are described in Table 2.4. The attributes could be either dedicated or general. The dedicated attributes are often referred to as conformance properties, and the general attributes are usually independent of the product. The conformance measures are selected based on the product which is being evaluated [56], [13]. Every attribute can not always be evaluated without some transformation. For example, a general attribute could be subdivided into less general attributes. The same could be said about the sours code. It could to be transformed into a form where automatic theorem provers could be applied. Some examples are given in 'Static verification' section which is in chapter three. For the certification purposes not only the final product could be evaluated, but all the other intermediate deliverables which may include design and requirement documents [32], [2], [88].

Component	Description
Context Description	Describes main objectives and environment of the component
User Requirements	List of expectations from the component
High Level Design	Represents mapping between customer's requirements and system's design
Detail Design	Could be a collection of designs that demonstrate every aspect of the component
Implementation	Represents relationship between component's code and its documentation
Testing	Description of the testing architecture
Correctness proofs	Formal proofs for some sections of the component

Table 2.4: Sections of the component [32], [52]

2.6.3 Stage 2: component identification

Every component within the product is associated with an achievement level. For this task we selected a subsection of the Integrated Component Maturity Model (ICMM) [81]. The complete model focuses on the evaluation of the final product and the software development processes. We are only interested in the section of the ICMM model which focuses on evaluation of the final product. Other standards such as ISO IEC 9126 and ISO IEC 14598, which are very similar to the ICMM model, could be integrated into this stage. The component identification levels are: preliminarily, component reuse, quality orientation and quantitative analysis. In the preliminarily level, the component is presented without any formal documents. In the component reuse level, the component is presented with a goal of reusing it in a similar applications and domains. The proper automated testing framework was utilized to test the component. The requirement document is up to date. In the quality orientation level, the component has an acceptable quality for both internal and commercial purposes. Some instances of the component were verified formally. In the quantitative analysis level, the various component metrics have being utilized to evaluate the component. The certification results of the component are consistent.

Entity	Description
Process Description	Information which describes objectives, risks and goals.
Measurable Concepts	Abstract description of components that could be measured.
Quality Models	Specification for the quality requirements and description of entity class relationships.
Concept Models	The collection of sub-concepts and associations between them.
Elementary Models	A collection of models with are based on on some known algorithms which can evaluate known criteria.
Base Measure	A basic measurement that could be applied individually without any external input.
Decision Criteria	Description on how to achieve certain level on confidence in a particular result.
Measuring Approaches	The measurement approach could be measurement function, analysis model which is interrelated with quality model or particular measuring approach.
Evaluation	A collection of evaluation which produces measuring results for a single attribute by applying one or more measuring approaches.
Evaluation Result	A number or an abstract value which indicates some level of achievement.

Table 2.5: Data entities [21], [27]

2.6.4 Stage 3: metrics construction

The measurement is a crucial part of the software certification process. It can provide data through which it would be possible to answer questions related to the product. Only a few approaches which deal with the measurable criteria have been developed. Some of them are: the Quality Functional Deployment (QFD), the Software Quality Metrics (SQM) and the Goal Question Metric (GQM). For the stage of the metrics construction the Goal Question Metric method was selected, because we think it is the most sound and applicable for the product based certification process. We are following the ISO IEC 25000 standard, which uses the same model, by selecting the GQM model. The individual metrics can be constructed with the help of GQM approach [81]. The key attributes can be subdivided, as described by the ISO 9126 standard, into the following groups: general external, dedicated external, general internal and dedicated internal. Some of the internal attributes are usability, efficiency, functionality, maintainability and portability. Other attributes, which are related to

software quality, also can belong to this group. The software quality attributes can be subdivided into many different weighted groups. All groups should be separated into three domains such as high, moderate, and low. Every attribute should have a priority weight assigned to it by the expert(s). The priority weight of the attribute should be dictated by the expectations and requirements. The external attributes are strictly subjective. They are responsible for the user satisfaction, and conformance to the requirements [22], [89], [61].

Entity	Description
Attribute	Description about what abstract or physical property should be measured
Calculable Concepts	The relationship between attributes and calculable Concepts
Calculation Method	A sequence of logical steps where a formula or indicator could be applied in order to obtain a concept of measure
Direct Metric	An independent metric that can be applied individually and does not depend on other metrics
Elementary Indicator	An indicator that is independent and does not depend on other indicators to deduce calculable concept
Function	Can be a formula or an algorithm that associates two or more metrics
Indirect Metric	A metric that is constructed from other metric or metrics that are being utilized for other attributes
Measure	A value that is associated with an attribute after evaluation process
Scale	A collection of values that have specific meaning associated with them
Software Tool	A tool or set of tools that is used during the evaluation

Table 2.6: Software metric entities [21]

2.6.5 Stage 4: component evaluation

Every component of the product is associated with an achievement level between 1 and 4 for the completeness, uniformity and conformance. At the same time, every level is associated with data which has to be delivered and verified for every component. The algorithms 4, 7 and 2 will compute levels for completeness, uniformity and conformance. The algorithm 1 will compute the overall certification level of the product. It is computed after all the components of a product have been evaluated.

The final certification level indicates the maturity of the product. This level is deduced from the achievements which are obtained by the components. The described methodology allows for the partial certification as well. For example, under different circumstances some companies may certify only specific sections of the product. The higher the level of certification, the more rigorous and formal verification must be applied. The final deliverable of the certification would include all the computed results, and would provide data with detailed descriptions of the components. This data would include a list of properties and attributes which were evaluated, objectives of the certification, description of the test/real data, and precise outline of the measuring methods which were utilized [32].

Algorithm 1: Algorithm to compute Product Certification Level

```
   Data: Collection of components which are associated with the product
   Result: Product Certification Level
 1 begin
 2     int score ← 0
 3     Iterator<Component> componentItr ← product.getComponentsList().iterator
 4     Component component ← null
 5     Level 5: same as Level 4 except real data and real environment should be used for
       evaluation  Level 4: all relationships between elements have been evaluated with formal
       mathematical ( in situations where can be applied and with test data and test
       environment)
 6     some elements of the product and their properties have been evaluated with formal
       mathematical verification
 7     if product.formalEvalTestData == true then
 8         while componentItr.hasNext do
 9             component ← componentItr.next
10             if component.getCompltLevel == 4 AND component.getUnifmLevel == 4 AND
               component.getConfmLevel == 4 then
11                 if product.formalEvalRealData == true then
12                     score ← score + 5
13                 end
14                 else
15                     score ← score + 4
16                 end
17             end
18         end
19         return (int) (score / product.getComponentsList().size)
20     end
21     Level 3: all relationships between elements have been evaluated with automated tools
22     elements of the product and their properties have evaluated with automated tools
23     if product.allElemAutoValidated == true then
24         while componentItr.hasNext do
25             component ← componentItr.next
26             if component.getCompltLevel >= 3 AND component.getUnifmLevel >= 3 AND
               component.getConfmLevel >= 3 then
27                 score ← score + 3
28             end
29         end
30         return (int) (score / product.getComponentsList().size)
31     end
32     Level 2: all relationships between elements have been manually evaluated
33     elements of the product and their properties have been manually validated
34     if product.allElementsManuallyValidated == true then
35         while componentItr.hasNext do
36             component ← componentItr.next
37             if component.getCompltLevel >= 2 AND component.getUnifmLevel >= 3 AND
               component.getConfmLevel >= 2 then
38                 score ← score + 2
39             end
40         end
41         return (int) (score / product.getComponentsList().size)
42     end
43     Level 1: all required elements of the product have been delivered
44     if product.allElementsDelivered == true then
45         while componentItr.hasNext do
46             component ← componentItr.next
47             if component.getCompltLevel >= 2 AND component.getUnifmLevel >= 2 AND
               component.getConfmLevel >= 1 then
48                 score ← score + 1
49             end
50         end
51         return (int) (score / product.getComponentsList().size)
52     end
53     return score
54 end
```

Algorithm 2: Algorithm to compute Conformance Level Section Two

```
Data: Inferred ontology model inferredModel, ontology's local name space nameSpaceInp, list of
      automated testing properties auotoTestingPrpt, list of manual evaluation properties
      manualEvalPrpt, list of rigorous evaluation properties rigorousEvalPrpt
Result: Conformance Level, component does not qualify for any Conformance Level if final computed
        Conformance Level == 0
1  begin
2      int conformanceLevel ← 0
3      OntProperty ontPropertyLocal ← null
4      Level 1: errors were found during regular automated testing
5      boolean errorsFoundAutoTest ← false
6      Individual auotoTesting ← inferredModel.getIndividual(nameSpaceInp +
       Status.REGULAR_AUTOMATED_TESTING)
7      Iterator<OntProperty> autoTestItr ← auotoTestingPrpt.iterator
8      Statement dataStm ← null
9      while autoTestItr.hasNext do
10         ontPropertyLocal ← autoTestItr.next
11         dataStm ← auotoTesting.getProperty(ontPropertyLocal)
12         if dataStm.getString().equals(Status.ERRORS_FOUND.toString) then
13             conformanceLevel ← 1
14             errorsFoundAutoTest ← true; break
15         end
16     end
17     Level 2: no errors were found with manual spot evaluation and regular automated testing
18     Individual spotEvaluation ← inferredModel.getIndividual(nameSpaceInp +
       Status.MANUAL_SPOT_EVAL)
19     boolean errorsFoundManualEval ← false
20     Iterator<OntProperty> spotEvalItr ← manualEvalPrpt.iterator
21     while spotEvalItr.hasNext do
22         ontPropertyLocal ← spotEvalItr.next
23         dataStm ← spotEvaluation.getProperty(ontPropertyLocal)
24         if dataStm.getString().equals(Status.ERRORS_FOUND.toString) then
25             errorsFoundManualEval ← true; break
26         end
27     end
28     if errorsFoundAutoTest == false AND errorsFoundManualEval == false then
29         conformanceLevel ← 2
30     end
31     Level 3: rigorous automatic testing did not detect any errors (includes stress testing)
32     Individual rigorousAutoTest ← inferredModel.getIndividual(nameSpaceInp +
       Status.RIGOROUS_AUTOMATED_TESTING)
33     boolean rigorousTestingPass ← true
34     Iterator<OntProperty> rigorousEvalItr ← rigorousEvalPrpt.iterator
35     while rigorousEvalItr.hasNext do
36         ontPropertyLocal ← rigorousEvalItr.next
37         dataStm ← rigorousAutoTest.getProperty(ontPropertyLocal)
38         if dataStm.getString().equals(Status.ERRORS_FOUND.toString) then
39             rigorousTestingPass ← false; break
40         end
41     end
42     if rigorousTestingPass == true then
43         conformanceLevel ← 3
44     end
45     Level 4: all formal verification of the component pass
46     Individual individualProofs ← inferredModel.getIndividual(nameSpaceInp +
       Status.CORRECTNESS_PRF_DELIVERY)
47     Property formalProofStatus ← inferredModel.getProperty(nameSpaceInp + Status.FORMAL_PRFS)
48     Statement formalVerificationStm ← individualProofs.getProperty(formalProofStatus)
49     if formalVerificationStm.getString().equals(Status.PASS.toString) then
50         conformanceLevel ← 4
51     end
52     return conformanceLevel
53 end
```

Algorithm 3: Algorithm to compute Uniformity Level Section Two

```
Data: Inferred ontology model inferredModel, ontology's local name space nameSpaceInp, list of general
      standardization properties generalStadrdPrpt, list of company standardization properties
      compStandardization, list of industry standardization properties industStandardization
Result: Uniformity Level
begin
    int uniformityLevel ← 0
    Level 1: general uniformity and standardization for all properties is average
    Individual generalUnifIndv ← inferredModel.getIndividual(nameSpaceInp +
    Status.GENERAL_STANR_DELIVERY)
    Iterator<OntProperty> genStadrdPrptItr ← generalStadrdPrpt.iterator
    Statement dataStm ← null
    OntProperty ontPropertyLocal ← null
    uniformityLevel ← 1
    boolean allPrpAboveAvg ← false
    while genStadrdPrptItr.hasNext do
        ontPropertyLocal ← genStadrdPrptItr.next
        dataStm ← generalUnifIndv.getProperty(ontPropertyLocal)
        if !dataStm.getString().equals(Status.AVERAGE.toString) AND
        !dataStm.getString().equals(Status.ABOVE_AVERAGE.toString) then
            uniformityLevel ← 0
            break
        end
        if dataStm.getString().equals(Status.ABOVE_AVERAGE.toString) then
            allPrpAboveAvg ← true
        end
    end
    Level 2: general uniformity and standardization for all properties is above average
    if allPrpAboveAvg == true then
        uniformityLevel ← 2
    end
    Level 3: component conforms to uniformity and standardization based on companies
    expectations
    Individual compSndrIndv ← inferredModel.getIndividual(nameSpaceInp +
    Status.COMPANY_STANDR_DELIVERY)
    Iterator<OntProperty> compSndrIndvItr ← compStandardization.iterator
    uniformityLevel ++
    while compSndrIndvItr.hasNext do
        ontPropertyLocal ← compSndrIndvItr.next
        dataStm ← compSndrIndv.getProperty(ontPropertyLocal)
        if !dataStm.getString().equals(Status.CONFORMS.toString) then
            uniformityLevel - -
            break
        end
    end
    Level 4: component conforms to uniformity and standardization based on industry
    expectations
    Individual industSndrIndv ← inferredModel.getIndividual(nameSpaceInp +
    Status.INDUSTRY_STANDR_DELIVERY)
    Iterator<OntProperty> industSndrIndvItr ← industStandardization.iterator
    uniformityLevel ++
    while industSndrIndvItr.hasNext do
        ontPropertyLocal ← industSndrIndvItr.next
        dataStm ← industSndrIndv.getProperty(ontPropertyLocal)
        if !dataStm.getString().equals(Status.CONFORMS.toString) then
            uniformityLevel - -
            break
        end
    end
    return uniformityLevel
end
```

Algorithm 4: Algorithm to compute Completeness Level

```
Data: Data about component: global name space globalNameSpaceInp, name space used by ontolgy
      nameSpaceInp, location of the ontology onologyLocationInp
Result: Completeness Level, component does not qualify for any Completeness Level if final computer
        Completeness Level == 0
1  begin
2      int completenessLevel ← 0; OntModel inferredModel ← computeInferredOntology(
       globalNameSpaceInp, nameSpaceInp, onologyLocationInp)
3      OntClass componentSectionsClass ← inferredModel.getOntClass(nameSpaceInp +
       Status.COMP_SECTIONS)
4      ExtendedIterator<OntClass> subClasses ← componentSectionsClass.listSubClasses
5      Property globalProperty ← null ; Individual individual ← null
6      Statement deliveryProperty ← null
7      while subClasses.hasNext do
8          OntClass subClass ← subClasses.next; ExtendedIterator<OntResource> allInstances ←
           (ExtendedIterator<OntResource>) subClass.listInstances
9          while allInstances.hasNext do
10             individual ← (Individual) allInstances.next; globalProperty ←
               inferredModel.getProperty(nameSpaceInp + Status.DELIVERY_STATUS); deliveryProperty
               ← individual.getProperty(globalProperty)
11             if deliveryProperty.getString.equals(Status.NOT_COMPLETE.toString) then
12                 Level 1: only some sections of the component are not complete
13                 completenessLevel ← 1 ; break
14             end
15         end
16     end
17     Level 2: all section of the component have been delivered
18     if completenessLevel == 0 then
19         completenessLevel ← 2
20     end
21     Level 3: some formal and informal proofs have been delivered
22     individual ← inferredModel.getIndividual(nameSpaceInp + Status.CORRECTNESS_PRF_DELIVERY)
23     Property infPrfsPrty ← inferredModel.getProperty(nameSpaceInp +
       Status.INFORMAL_PRFS_STATUS)
24     Property formPrfsPrty ← inferredModel.getProperty(nameSpaceInp +
       Status.FORMAL_PRFS_STATUS)
25     Statement infPrfsPrtyStm ← individual.getProperty(infPrfsPrty)
26     Statement formPrfsPrtyStm ← individual.getProperty(formPrfsPrty)
27     if infPrfsPrtyStm.getString.equals(Status.SOME.toString) AND
       formPrfsPrtyStm.getString.equals(Status.SOME.toString) then
28         completenessLevel ← 3
29     end
30     Level 4: all formal proofs have been delivered
31     if infPrfsPrtyStm.getString.equals(Status.COMPLETE.toString) AND
       formPrfsPrtyStm.getString.equals(Status.COMPLETE.toString) then
32         completenessLevel ← 4
33     end
34     return completenessLevel
35 end
```

Algorithm 5: Algorithm to compute inferred ontology model

```
Data: Global name space globalNameSpaceInp, local name space nameSpaceInp, ontology location
      onologyLocationInp
Result: Inferred ontology
1  begin
2      OntModel model ← ModelFactory.createOntologyModel; OntDocumentManager documentManager ←
       model.getDocumentManager
3      documentManager.addAltEntry(globalNameSpaceInp, file: + onologyLocationInp)
4      model.read(globalNameSpaceInp); OntModel inferredModel ←
       ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM_MICRO_RULE_INF, model)
5      return inferredModel
6  end
```

Algorithm 6: Algorithm to compute Conformance Level

```
   Data: Global name space globalNameSpaceInp, local name space nameSpaceInp, ontology location
         onologyLocationInp
   Result: input to Conformance Level Section Two algorithm, and final Conformance Level
 1 begin
 2 |  OntModel inferredModel ← computeInferredOntology( globalNameSpaceInp, nameSpaceInp,
   |  onologyLocationInp)
 3 |  List<OntProperty> auotoTestingPrpt ← new ArrayList<OntProperty>
 4 |  List<OntProperty> manualEvalPrpt ← new ArrayList<OntProperty>
 5 |  List<OntProperty> rigorousEvalPrpt ← new ArrayList<OntProperty>
 6 |  ExtendedIterator<OntProperty> allProperties ← inferredModel.listAllOntProperties
 7 |  OntProperty ← null
 8 |  while allProperties.hasNext do
 9 |  |  ontPropertyLocal ← allProperties.next
10 |  |  if ontPropertyLo-
   |  |  cal.getLocalName().startsWith(Status.REGULAR_AUTOMATED_TESTING_DATA.toString)
   |  |  then
11 |  |  |  auotoTestingPrpt.add(ontPropertyLocal)
12 |  |  end
13 |  |  if ontPropertyLocal.getLocalName().startsWith(Status.MANUAL_SPOT_EVAL_DATA.toString)
   |  |  then
14 |  |  |  manualEvalPrpt.add(ontPropertyLocal)
15 |  |  end
16 |  |  if ontPropertyLocal.getLocalName().startsWith(Status.RIGOROUS_AUT_TEST_DATA.toString)
   |  |  then
17 |  |  |  rigorousEvalPrpt.add(ontPropertyLocal)
18 |  |  end
19 |  end
20 |  int result ← ConformanceLevelSectionTwo(inferredModel, nameSpaceInp, auotoTestingPrpt,
   |  manualEvalPrpt, rigorousEvalPrpt)
21 |  return result
22 end
```

Algorithm 7: Algorithm to compute Uniformity Level

```
   Data: Global name space globalNameSpaceInp, local name space nameSpaceInp, ontology location
         onologyLocationInp
   Result: input to Uniformity Level Section Two algorithm, and final Uniformity Level
 1 begin
 2 |  OntModel inferredModel ← computeInferredOntology( globalNameSpaceInp, nameSpaceInp,
   |  onologyLocationInp)
 3 |  List<OntProperty> generalStadrdPrpt ← new ArrayList<OntProperty>
 4 |  List<OntProperty> compStandardization ← new ArrayList<OntProperty>
 5 |  List<OntProperty> industStandardization ← new ArrayList<OntProperty>
 6 |  ExtendedIterator<OntProperty> allProperties ← inferredModel.listAllOntProperties
 7 |  OntProperty ontPropertyLocal ← null
 8 |  while allProperties.hasNext do
 9 |  |  ontPropertyLocal ← allProperties.next
10 |  |  if ontPropertyLocal.getLocalName().startsWith(Status.GENERAL_STANR_DATA.toString) then
11 |  |  |  generalStadrdPrpt.add(ontPropertyLocal)
12 |  |  end
13 |  |  if ontPropertyLocal.getLocalName().startsWith(Status.COMPANY_STANR_DATA.toString) then
14 |  |  |  compStandardization.add(ontPropertyLocal)
15 |  |  end
16 |  |  if ontPropertyLocal.getLocalName().startsWith(Status.INDUSTRY_STANR_DATA.toString) then
17 |  |  |  industStandardization.add(ontPropertyLocal)
18 |  |  end
19 |  end
20 |  int result ← UniformityLevelSecondPart(inferredModel, nameSpaceInp, generalStadrdPrpt,
   |  compStandardization, industStandardization )
21 |  return result
22 end
```

Chapter 3

A context aware framework

A context aware framework offers a mechanism which allows for a certification process to adapt to ever changing expectations. This is because the framework is aware of its context and is able to adjust seamlessly to its evolved context. The context aware applications, which could be developed within the framework, are capable of adjusting to the evolved context in order to support new certification demands. Such flexibility allows for the certification process to expand more easily into different domains. The framework uses context, which is provided by the context providers, to construct an intelligent environment of the software certification processes. This intelligent environment is achieved by incorporating different context providers within a single framework in order achieve a common goals. The context providers are not limited to only specific components within the product, but other entities such as specification documents, industry standards and company standards could be context providers. Importantly, the framework could be located on a single machine or span over a collection of physical machines.

Within the last decade OSGi (Open Service Gateway initiative) has gained popularity in the industry. It has been successfully applied in the development of software for smart homes and mobile devices. OSGi has been proven to work well in

these areas and has performed as expected [23], [68]. The definition of context awareness, in both literature and industry, is not precisely defined because this approach in computing is still emerging. The wider spread of this type of computing approach can be seen in mobile devices and service oriented systems for smart homes. Mokhtar et al. [57] defined context awareness as follows:

'Context awareness is a property of a system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.'

There are three main categories by which context aware systems can be classified; these are device context, user context and physical context. The user context category deals with user driven actions and is the most appropriate type of context awareness for the proposed framework. The efficient management of the context is supported and driven by the context model and its structure. The philosophy behind modeling context models follows two main objectives; namely, the ownership of a flexible structure in which knowledge sharing is enabled, and logical reasoning in which reasoning over static data can occur. The success of the context aware systems directly depends on their ability to maintain these key objectives. The multi level ontological approach was selected to model the context for the framework. The upper layer within the ontological hierarchy, which is given in appendix A, models generic concepts and relations for the product based software certification. The lover levels within the ontology are used for the modeling of domain specific concepts and relations. This allows for the criteria, which occur commonly in lower levels, to be gathered in one location which would often be moved to upper levels within the ontological hierarchy without being redefined multiple times. This approach eliminates issues in which concepts or properties could be defined or evaluated differently in different domains [57].

3.1 Overview of a context aware framework

The framework is structured as both a contextually aware and service oriented entity. Within the framework service discovery and binding is accomplished independently from the developed component. A high level design of the framework is given in Figure 3.10. The context discovery and interpretation is handled by a context reasoner layer.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY SCS "http://2009/9/SCS.owl#" >
    <!ENTITY CL_ "http://2009/9/SCS.owl#CL_&#8722;" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://2009/9/SCS.owl#"
     xml:base="http://2009/9/SCS.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:SCS="http://2009/9/SCS.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:CL_="&SCS;CL_&#8722;">
    <owl:Ontology rdf:about=""/>
    <Correctness_proofs rdf:about="#correctness_proofs_delivery">
        <rdf:type rdf:resource="&owl;Thing"/>
        <formalProofsStatus rdf:datatype="&xsd;string">complete</formalProofsStatus>
        <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
        <informalProofsStatus rdf:datatype="&xsd;string">complete</informalProofsStatus>
        <formalProofs rdf:datatype="&xsd;string">pass</formalProofs>
    </Correctness_proofs>
    <owl:Thing rdf:about="#industry_standardization_delivery">
        <rdf:type rdf:resource="#Implementation"/>
        <deliveryStatus rdf:datatype="&xsd;string">NOT complete</deliveryStatus>
        <industryStandardization rdf:datatype="&xsd;string">does conforms</industryStandardization>
    </owl:Thing>
</rdf:RDF>
```

Figure 3.1: The partial context which may be submitted by the components

This layer reasons over the various direct contexts and derives indirect context, it can also be described as a reasoner over a high level context to derive a low level context. Every context aware component maintains specific roles within the framework. The main objective of the framework is to construct a contextually aware

environment in which the development and certification processes are integrated and managed in parallel. Importantly, a complex certification processes could be managed by the framework because its context is formally modeled with an upper ontology [83]. An example of partial context which can be submitted by the components is given in Figure 3.1. A complete context of the component would describe all its certification achievements. For example, what verification techniques have been applied and their results. The context itself is a partial ontology which is a subsection of the upper ontology. The developed proof of concept framework for product based software certification has a context providers layer, a context interfaces layer, a context reasoners layer, a remote layer and an automation layer. The remote layer is responsible for the secure communication between the remote components of the framework. This would only be required in situations where the framework would have to span over multiple physical machines. Remote communication is managed by the Remote-Open Service Gateway initiative (R-OSGi) [68].

3.2 Constructing independent context providers

The context aware framework is designed to work completely within the Open Service Gateway initiative (OSGi) environment. The OSGi alliance maintains and supports the OSGi environment. This alliance includes vendors from both the industry and academia. The environment has been successfully utilized in a number of systems, including Eclipse, Knopflerfish, Concierge and Apache Felix. Its specifications are currently on the third version, but the core implementation has not been changed since the first version. Only the service binding feature has become more efficient in the later versions. The advanced service binding capabilities, which are described in the upcoming sections, offer many features which could be beneficial to the product based certification process.

3.2.1 Dynamic and static sections of the bundle

The communication between components within the OSGi environment, which are also known as bundles, is accomplished through declarative services. The bundles are often referred to as agents in the Artificial Intelligence (AI) domain. An important benefit of the environment is its capability to add, remove, stop, and reactivate components without the need of restarting. It is apparent how the properties of plug and play capability, a well known feature in the hardware domain, has spread into the domain of software development [68]. The architecture is implemented in a service oriented framework where services are loosely coupled. Any class or package of the components can be published as a service to be utilized by other components. Therefore, the environment imposes smaller restrictions on the development domain. Within the OSGi environment, the interfaces are often used to serve as services to other components. This practice allows for the implementation details of the components to be hidden. The OSGi environment has a registry which maintains all of the services which are offered and registered within the framework. An individual component can search the registry to find specific services with the help of service binders, or define the services which are necessary for its successful activation within the XML definition file. Figure 3.2 demonstrates an example of the XML definition file which has to be maintained by every component in the framework. In the definition file, components identify which services will be required and which services will be offered within the framework for other components to use. Figure 3.2 demonstrates that the component needs two services from the framework. They are LOG-SERVICE and CONTEXT-INTERFACE-SERVICE. These services are offered within the OSGi environment through the interface reference. The services which are offered by the component are specified within the <service> ... </service> tag. For example, <provide interface = "ContextReasoner.IContextReasoner" /> deceleration will offer an inter-

face as a service to the framework. Many other services could be offered within this tag and not necessarily just interface. They could be regular Java classes. The services which are being imported from the framework are listed within the <reference> ... </reference> tag. For example, <reference name="LOG- SERVICE" interface="org.osgi.service.log.LogService" cardinality="1..1" policy="static" bind= "bindLogService" unbind="unbindLogService"/> declaration will import LOG-SERVICE service from the framework by referring to an appropriate interface, set its cardinality, and correlate an appropriate methods within the activator class with bind and unbind statements. The "1..1" cardinality means one to one relationship between provider of the service and subscriber of the service. The other possibilities to define cardinality are "1..n", "n..1" and "n..n". The policy type can by static or dynamic and the preferred type is static. [68], [70].

```
1  <?xml version="1.0" encoding="UTF-8"?>
2      <component name="ContextReasoner.IContextReasoner" immediate="true">
3      <implementation class="ContextReasoner.Internal.ContextReasoner"/>
4      <reference name="LOG-SERVICE"
5              interface="org.osgi.service.log.LogService"
6              cardinality="1..1"
7              policy="static"
8              bind="bindLogService"
9              unbind="unbindLogService"/>
10         <reference name="CONTEXT-INTERFACE-SERVICE"
11             interface="ContextInterface.IContextInterface"
12             cardinality="1..1"
13             policy="static"
14             bind="bindContextInterface"
15             unbind="unbindContextInterface"/>
16                     .....
17      <service>
18             <provide interface="ContextReasoner.IContextReasoner"/>
19                     .....
20     </service>
21 </component>
```

Figure 3.2: XML definition file which describes dynamic section of the component

Every component has an activator class which binds and unbinds the required services. An example of the activator class is given in Figure 3.3. The activator class

```
1  public class ContextReasoner implements IContextReasoner {
2      private LogService logService = null;
3      private IContextInterface contextInterface = null;
4          .....
5      protected void activate(ComponentContext context) {
6          .....
7      }
8      protected void deactivate(ComponentContext context) {
9          logService = null;
10         contextInterface = null;
11         .....
12     }
13     protected void bindLogService(LogService logService) {
14         this.logService = logService;
15         .....
16     }
17     protected void unbindLogService(LogService logService) {
18         this.logService = null;
19         .....
20     }
21     protected void bindContextInterface
22         (IContextInterface contextInterface) {
23         this.contextInterface = contextInterface;
24         .....
25     }
26     protected void unbindContextInterface
27         (IContextInterface contextInterface) {
28         this.contextInterface = null;
29         .....
30     }
31         .....
32 }
```

Figure 3.3: Shell of the activator class

is associated with, and driven by, the MANIFEST file. The MANIFEST file contains meta data which describes the properties and requirements of the component, such as which packages are imported from the framework, and which packages are exported to the framework for other components to use. An example of the MANIFEST file is given in Figure 3.4. This file manages the static section of the component while the XML definition file manages the dynamic section of the component.

3.2.2 Component/bundle state

The OSGi environment implements a white board pattern instead of the publisher subscriber pattern. When using a publisher subscriber pattern, requesters have to

```
1  Manifest-Version: 1.0
2  Bundle-ManifestVersion: 2
3  Bundle-Name: Context Reasoner
4  Bundle-SymbolicName: Context Reasoner; singleton:=true
5  Bundle-Version: 0.0.1
6  Bundle-Vendor: project
7  Bundle-ClassPath:.
8       ......
9  Service-Component:
10 OSGI-INF/ContextReasoner.xml
11      ......
12 Import-Package:
13 ContextInterface,
14      org.osgi.service.log
15      ......
16 Export-Package:
17      ContextReasoner
18      ......
```

Figure 3.4: MANIFEST.MF file which describes static section of the component

subscribe to every single service they need while providers have to maintain subscriptions and usually with imposed cardinality restrictions. OSGi uses a service registry where requesters register themselves by means of a listener component, while subscribers register through the provider component. Therefore, there is no direct dependency between components because the service binding and releasing is handled seamlessly by the OSGi environment. The component can not become fully active unless all of its required services are present and offered by the environment. The environment is interactive and supports components which can be in one of six different states: starting, active, stopping, installed, resolved, and uninstalled. Figure 3.5 demonstrates dependency between states and the virtual binding of services between the different components. Every component can change its state dynamically while the OSGi environment is in active state. The same features and capabilities apply to remote components as well [68], [70].

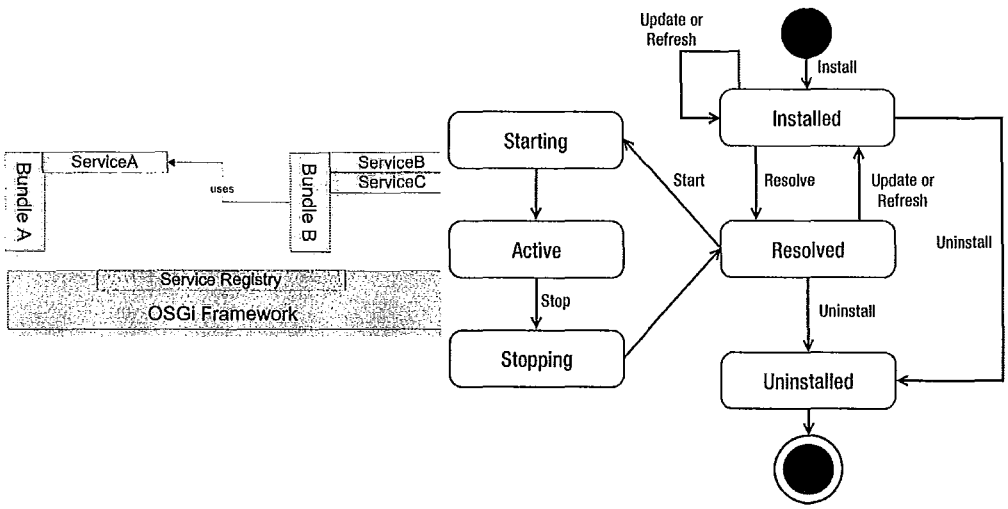

Figure 3.5: Bundle/component states and service binding [70], [68]

3.2.3 Verifying state of the framework

After the framework is activated under the OSGi console a user can check for all active bundles. The listing of all bundles can be achieved with the *'ss'* command. All of the framework's required components/bundles have to be in the active state in order for the framework to be fully operational. Figure 3.6 shows a partial snapshot of the bundles and their states. The snapshot shows bundles which were prototyped for the context aware framework and some other additional bundles which are not immediate components of the framework. These additional bundles are third party bundles. The list of third party bundles could be extensive. The framework requires 154 external bundles. The external bundles usually come with the redistribution of Eclipse, while others have to be downloaded from different vendors. We gathered all the required third party bundles which are needed for the context aware framework and made them part of the prototype. For example, the org.eclipse.equinox.ds_1.0.0.v20070226 bundle is responsible for handling declarative services within the framework. Appendix C provides instructions on how to set up the framework starting from the installation of the required software.

```
1  osgi> ss
2  Framework is launched.
3  id      State       Bundle
4  0       ACTIVE      org.eclipse.osgi_3.3.2.R33x_v20080105
5  1       ACTIVE      org.eclipse.core.net_1.0.1.r33x_20070709
6  2       ACTIVE      org.eclipse.ui.cheatsheets_3.3.1.v20080125_33x
7          .....
8  94      ACTIVE      org.eclipse.jdt.core.manipulation_1.1.0.v20070606-0010
9  95      ACTIVE      Pairwise Comparison_0.0.1
10 96      ACTIVE      org.eclipse.pde.build_3.3.2.v20071019
11         .....
12 615     ACTIVE      Context Interface_0.0.1
13 616     ACTIVE      Context Reasoner_0.0.1
14 osgi>
```

Figure 3.6: Partial output produced by OSGi console by using *'ss'* command

The OSGi console feature supports a collection of helpful commands. Figure 3.7 shows a small list of active services which are offered by the framework. The complete listing of all the services can be achieved with a *'status'* command. The components within the framework do not import each other, but communicate via published services. Therefore, if any component fails, the remaining components would still be operational. Their state would change however, from active to resolved as is indicated in Figure 3.5.

3.3 Description of a context aware framework

Figure 3.10 demonstrates a high level design of the context aware framework. The design has three main layers: ontology building, context aware and automation. In the ontology building layer, information could be gathered from experts into upper ontology which overtime may contain domain specific sub-ontologies. The context aware layer is subdivided further into context providers layer, context interfaces layer, context reasoners layer and remote context providers layer. The sections below describe in detail the automation layer and sublayers of the context aware layer.

```
1  osgi> status
2  Framework is launched.
3       .....
4  {ContextReasoner.IContextReasoner}={component.name=ContextReasoner.
5       IContextReasoner,component.id=1, service.id=51}
6  {ch.ethz.iks.r_osgi.RemoteOSGiService, ch.ethz.iks.r_osgi.Remoting}=
7       {service.id=52}
8  {ch.ethz.iks.r_osgi.channels.NetworkChannelFactory}={protocol=r-osgi,
9       service.id=53}
10 {ContextInterface.IContextInterface}={component.name=ContextInterface.
11      IContextInterface, component.id=3, service.id=54}
12 {PairwiseComparison.IPairwiseComparison}={component.name=PairwiseComparison.
13           IPairwiseComparison, component.id=2, service.id=55}
14 {r_osgi_client.IRemoteService}={service.id=56}
15 {ch.ethz.iks.r_osgi.SurrogateRegistration}={service.remote.registration=true,
16           service.remote.smartproxy=r_osgi_service.internal.SmartService,
17           legacy.service.reference={r_osgi_client.IRemoteService}={service.id=56},
18           service.id=57}
19 {r_osgi_service.ServiceHandler}={service.id=58}
20      .....
21 osgi>
```

Figure 3.7: Partial output produced by OSGi console by using *'status'* command

3.3.1 Context providers layer

In this layer, all components provide context information which is used for reasoning. Context providers can obtain information from a variety of sources. They transform the collected data through the context interface layer to the context reasoner layer. Not only the context providers are able to provide valuable context, but other entities such as calendar, schedule, or specification documents. Each context provider may use a collection of interfaces in the context interfaces layer to submit direct context to the context reasoner layer. This communication through the interfaces is an essential part of the framework because it hides implementation details of the reasoner layer. This feature is crucial for the network components of the framework. All components within the context aware framework should communicate via the interfaces. Larger applications may also maintain a large number of components in the context providers layer.

The semantics of the OWL language allows for explicit knowledge to be de-

duced from implicit knowledge with the help of concepts. This ability makes OWL language a powerful language for knowledge reasoning. The information becomes useful and can be manipulated freely if it is presented in a higher level context. The lower level context is not very useful because, in most situations, it is a single instance of data. Every context provider, within the context provider layer, has to bind with the services necessary for allowing them to submit context. The framework supports remote context providers in addition to the local context providers. Therefore, context providers do not have to be located on a single physical machine.

3.3.2 Context interfaces layer

This layer serves as a bridge between the components and the context reasoners layer. It was added to incorporate the network capabilities of the framework. Currently, interfaces support Jena selectors and are used to query OWL ontologies. Another option for querying OWL ontologies would be by using RDQL, which a specific query language used to query Jena's RDF ontology models. The query declarations which follow the RDQL pattern are data oriented, while the Jena selectors follow the procedural pattern. The major difference between these two approaches is that the data oriented approach operates over static, not inferred, ontologies, while Jena selectors can operate with inferred ontologies. This is why the Jena selectors method was selected for communication and the other approach was left as an option [73].

3.3.3 Context reasoners layer

This layer automatically reasons over the context which is submitted by the context providers. The main responsibility of this layer is to interpret the collected context and evaluate it against the context which is saved in the ontology. Components within the framework are responsible for submitting only higher level context. This high level

context represents the factual state of the component and its current achievement in the certification process. This layer is also responsible for maintaining and updating knowledge within the ontology. The factual information could be direct, which does not require reasoning, or indirect, which requires reasoning. Bayesian networks can be applied in situations where uncertainty is present and where certain reasoning cannot be successfully applied.

Algorithm 8: Algorithm to publish remote services

```
   Data: bundleContext an instance of the bundle which is going to publish remote services and events
   Result: publish services and post events for the remote users
 1 begin
 2 |  ServiceRegistration serviceRegistration ← null
 3 |  Hashtable localPropert ← new Hashtable
 4 |  localPropert.put(RemoteOSGiService.R_OSGi_REGISTRATION, Boolean.TRUE)
 5 |  localPropert.put(RemoteOSGiService.SMART_PROXY, SmartService.class.getName)
 6 |  ServiceRegistration localServiceRegist ←
   |  bundleContext.registerService(RemoteServiceInterface.class.getName, new ServiceImpl, null)
 7 |  localPropert.put(SurrogateRegistration.SERVICE_REFERENCE, localServiceRegist.getReference)
 8 |  serviceRegistration ← bundleContext.registerService(SurrogateRegistration.class.getName, this,
   |  localPropert)
 9 |  ServiceReference serviceReference ← bundleContext.getServiceReference(EventAdmin.class.getName)
10 |  if serviceReference ! = null then
11 |  |  final EventAdmin eventAdmin ← (EventAdmin) bundleContext.getService(serviceReference)
12 |  |  activateThread(eventAdmin)
13 |  |  bundleContext.registerService(ServiceHandler.class.getName, new ServiceHandler {
14 |  |   public Object validateService(Object service) {
15 |  |    validate the service
16 |  |    return service
17 |  |   }
18 |  |   public void handleService(Object service, String[] args) throws Exception {
19 |  |    handle the service
20 |  |   }
21 |  |  }, new Hashtable)
22 |  |  add properties
23 |  |  Dictionary newProperties ← new Hashtable
24 |  |  newProperties.put(ServerStatus.PROP_NAME, ServerStatus.PROP_VALUE)
25 |  |  localServiceRegist.setProperties(newProperties)
26 |  end
27 |  else
28 |  |  return error ← ServerStatus.ERROR_NULL_EVENT_ADMIN
29 |  end
30 end
```

Another task of the reasoning layer is to validate and inspect the consistency between classes and relationships. This task is required because the majority of relationships could be implied, especially in situations where relationships may spread between ontologies which could integrate different domains. This layer may support

two types of reasoning which are supported by the OWL language. They are RDQL reasoning and Jena selectors reasoning. The RDQL reasoning supports all the constructs mandated by the RDF Core Working Group, making it a powerful reasoning engine. However, as mentioned above, it operates only on non-inferred ontologies. The Jena selectors support primitive constructs when compared to RDQL, but they allows developers to construct a complex reasoning algorithms which may not be accomplished so easily with the RDFS library. In order to allow for flexibility to occur, it is recommended to use both methods as viable ways for reasoning over the context. In both cases, all the rules have to be predefined within the ontology for reasoning purposes. For example, the rules which are specified below will work in both cases.

- disjointWith: $((?A\ owl:disjointWith\ ?B) \wedge (?x\ rdf:type\ ?A) \wedge (?y\ rdf:type\ ?B) \Rightarrow (?x\ owl:differentFrom\ ?y))$
- subClassOf: $((?X\ rdfs:subClassOf\ ?Y) \wedge (?Y\ rdfs:subClassOf\ ?Z) \Rightarrow (?X\ rdfs:subClassOf\ ?Z))$,
- subPropertyOf: $((?X\ rdfs:subPropertyOf\ ?Y) \wedge (?Y\ rdfs:subPropertyOf\ ?Z) \Rightarrow (?X\ rdfs:subPropertyOf\ ?Z))$
- transitiveProperty: $((?X\ rdf:type\ owl:transitiveProperty) \wedge (?P\ ?X\ ?Q) \wedge (?Q\ ?X\ ?O) \Rightarrow (?P\ ?X\ ?O))$
- inverseOf: $((?X\ owl:inverseOf\ ?Y) \wedge (?A\ ?X\ ?B) \Rightarrow (?B\ ?Y\ ?A))$

3.3.4 Remote context providers layer

With the help of R-OSGi it is possible to create a secure communication channel between remote Java Virtual Machines (Java VM). A secure section of communication can be obtained by using a Virtual Private Network (VPN). It is more advantageous

to rely on secure communication channels which are already maintained and offered by the majority of both public and private networks. If we look at the Open System Interconnection Reference Model (OSI model), the framework only deals with lower levels, such as Physical and Data Link. The rest of the layers of the OSI model are handled by the network. Information is sent between remote parties in a serialized form. Every remote component is associated with a unique serial number which is used for the identification of remote conversations. Local components do not require serial numbers because their data is not being serialized during their transfer. Algorithm 8 can be used locally within the framework to publish services for the remote components. The remote components can use algorithm 11 to search and bind with services which are offered by the remote framework. In order for communication to be established, the remote components require a Uniform Resource Locator (URL) of the framework and the port number on which communication is allowed to occur. The default port for this type of communications is 9278. This information is the minimum of what is required for remote communication to be established between two different context aware frameworks. Two frameworks, which reside on different physical machines, become one virtual framework after connection is established between their remote components. This allows for declarative services to span over many different physical frameworks. Developers may add many other layers of security and authentication to the communication channel, but it is not necessary, because the security of the communication is already handled in the upper layers of the OSI model [68], [70], [23].

Algorithm 9: Algorithm to save the context within the inferred ontology

```
   Data: Global name space globalNameSpaceInp, local name space nameSpaceInp, ontology location
         onologyLocationInp, new context which is stored within the ontology contextOntInput
   Result: the inferred ontology will be updated with new context
 1 begin
 2     OntModel contextInferredModel ← computeInferredOntology(globalNameSpaceInp, nameSpaceInp,
       contextOntInput)
 3     OntModel ontologyInferredModel ← computeInferredOntology(globalNameSpaceInp, nameSpaceInp,
       onologyLocationInp)
 4     Iterator<Individual> ontPropertyItr ← contextInferredModel.listIndividuals
 5     Individual contextIndividual ← null
 6     Individual ontologyIndividual ← null
 7     while ontPropertyItr.hasNext do
 8         contextIndividual ← ontPropertyItr.next
 9         StmtIterator contextStmtIterator ← contextIndividual.listProperties
10         ontologyIndividual ← ontologyInferredModel.getIndividual(contextIndividual.getURI)
11         Statement ontologyStatament ← null
12         Statement contextStatement ← null
13         while contextStmtIterator.hasNext do
14             contextStatement ← contextStmtIterator.next
15             if contextStatement.getObject().isLiteral then
16                 StmtIterator ontologyStmtIterator ← ontologyIndividual.listProperties
17                 while ontologyStmtIterator.hasNext do
18                     ontologyStatament ← ontologyStmtIterator.next
19                     if ontologyStata-
                       ment.getPredicate().toString.equals(contextStatement.getPredicate().toString)
                       then
20                         break
21                     end
22                 end
23                 ontologyInferredModel.remove(ontologyStatament)
24                 ontologyInferredModel.add(contextStatement)
25             end
26         end
27     end
28     writeOntology(ontologyInferredModel)
29 end
```

3.4 Description of the automated section

An automation within the OSGi environment is accomplished by scripts which are written with Apache Ant software, and they are responsible for different tasks. Some of them will clean components from unnecessary entities, compile components for the generation of the code coverage report and the requirements coverage report, package components into redistributable builds, monitor the testing of the system, and will automatically notify the use via email about the status of the certification process. Automation scripts can be developed completely within the Apache Ant developing environment without the need of integration with any other programming

Algorithm 10: Algorithm to deduce predicate implication

```
Data: Global name space globalNameSpaceInp, local name space nameSpaceInp, ontology location
      onologyLocationInp, new context which is stored within the ontology contextOntInput
Result: implication for every predicate in ontology
1  begin
2  |  OntModel ontologyInferredModel ← computeInferredOntology(globalNameSpace, nameSpace,
   |  onologyLocation)
3  |  Iterator<Individual> ontIndividualsItr ← ontologyInferredModel.listIndividuals
4  |  Individual ontologyIndividual ← null
5  |  while ontIndividualsItr.hasNext do
6  |  |  ontologyIndividual ← ontIndividualsItr.next
7  |  |  StmtIterator contextStmtIterator ← ontologyIndividual.listProperties
8  |  |  Statement ontologyStatament ← null
9  |  |  collectInfo("Ont Class: "+ontologyIndividual.getOntClass)
10 |  |  collectInfo("Individual: "+ontologyIndividual.getLocalName)
11 |  |  while contextStmtIterator.hasNext do
12 |  |  |  ontologyStatament ← contextStmtIterator.next
13 |  |  |  if ontologyStatament.getObject().isLiteral then
14 |  |  |  |  collectInfo("Predicate: "+ontologyStatament.getPredicate().getLocalName)
15 |  |  |  |  collectInfo("Value: "+ontologyStatament.getString)
16 |  |  |  |  collectInfo(computeIndividualImplication(ontologyStatament))
17 |  |  |  end
18 |  |  end
19 |  end
20 |  return collected implication info
21 end
```

language. They can also invoke specific libraries to execute larger tasks which can be written with programming languages such as Java and C++. The prototype automation scripts, which we have developed, utilized both methods. The submission of the context is handled entirely by the components and is accomplished through declarative services. Algorithm 11, which uses the Jena library to communicate with OWL ontologies, will save the context into the main ontology.

3.4.1 Automation scripts

Script B.5 will execute targets which will invoke algorithms to compute the certification level of the product. It also has targets to compute the completeness, conformance, and uniformity for every component of the product. Another small target within the B.5 script will invoke an algorithm to compute the implication of every concept which is referred to as an individual in the ontology domain. In addition to the individual scripts, we have developed an automation script which will evaluate

the current status of the certification process and will create a comprehensive report. This comprehensive report it is being automatically emailed to a single individual or can be emailed to a group of individuals. An example of the comprehensive report is given in Appendix C. This report could be enhanced further to produce desirable reports for evaluations. The automatic notification via email will work in both outcomes, the system will either pass or fail during the verification. Script B.4 maintains the appropriate targets for the email management. The user may take appropriate action if failure does occur. There are two other reports which are being generated, they are the code coverage report and the requirements coverage report. The automation script B.1, also known as an upper automation script, is responsible for executing all other lower level automation scripts. The general features from all of the automation scripts have been extracted and gathered into one global common automation script, B.2, which can be used by all other scripts. For example, the script B.3 maintains all the targets of the automation system, but it relies on the general components in the B.2 script by inheriting all of its properties and references.

3.4.2 Coverage reports

In our opinion, the requirements tracking could be considered as one of the most important properties of the software certification process. The majority of software developing companies utilize some form of requirements tracking. It provides a mechanism which is able to trace requirements to their implementation. From the point of view of software certification, requirements tracking is a property which any framework should have. It would be very difficult to verify implementation against a set of requirements without a proper tracking system.

The main goal of the requirements coverage report is to provide a mechanism in which links between stakeholder requirements and the system's components could

be apparent. In order for the trace to be useful, it must follow some structured plan. A collection of sound frameworks for traceability have been proposed. Although, they are not easy to use or maintain because they were developed through theoretical work and literature analysis [8]. Since *'requirements traceability'*, which is also known as requirements coverage, was first mentioned in 1970, it has been described in a variety of ways. Some examples are listed below as they were described by their authors. [5].

- 'a characteristics of a system in which the requirements are clearly linked to their sources and to the artifacts created during the system development life cycle based on these requirements' [8], [5]
- 'the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origin, through its development and specification, to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)' [28], [5]
- 'IEEE standard for software maintenance: the ability of a software to provide a thread from the requirements to the implementation, with respect to the specific development and operational environment' [35], [5]
- Solution driven: 'the ability of tracing from one entity to another based on given semantic relations'
- Information driven: 'the ability to link between functions, data, requirements and any text in the statement of requirements that refers to them'
- Direction driven: 'the ability to follow a specific item at input of a phase of the software life cycle to a specific item at the output of that phase' [28]

The automation scripts that were developed integrate Jfeature software which has ability to trace specific requirements within the framework. The reports, which

can be generated with the Jfeature software, support forward and reverse requirements tracking. They can also be used to check if there is a relationship between different components of the system, primarily the predecessor to successor or successor to predecessor association. It is apparent that automation scripts integrated with tracking software may belong to the family of traceability metrics and can be associated with documentation and test metrics. Documentation metrics can manage the relationships between specification requirements and would deal with the low level design of a system. Test metrics can manage the relationships between sections of the system and the system's tests. They would also deal with the validation of developed software [5].

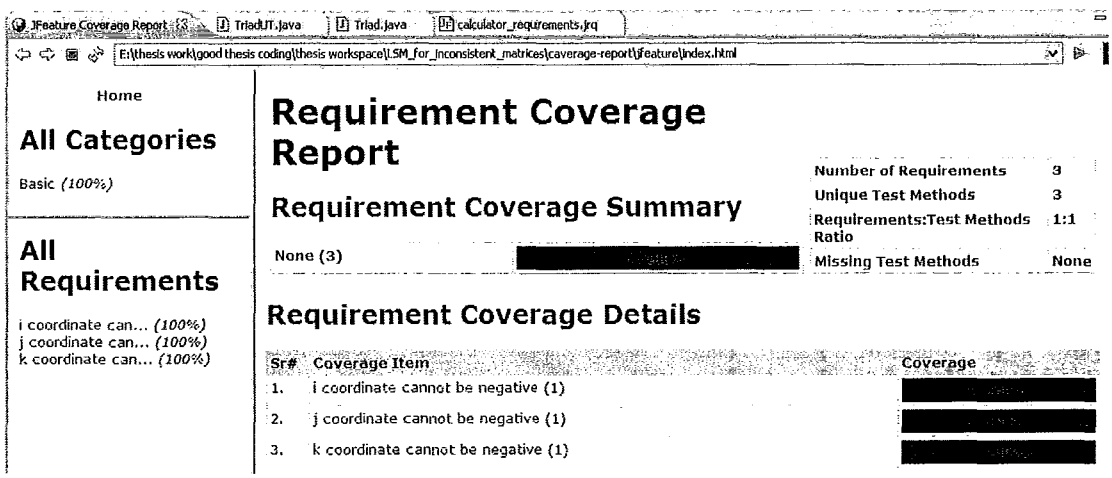

Figure 3.8: An example of the Requirements Coverage Report

Some projects could have a large number of experts specifying the requirements of the software. For such large projects, inconsistent and imprecise methods for tracing requirements should not be used. The use of automation scripts allows to overcome this limitation and facilitates the team with the ability to specify the requirements and track them within the software. An additional benefit of utilizing requirements coverage process includes a potential consistent software development in relation to the specification documents [64], [62]. The snapshots of the Requirements

Coverage Report can be seen in Figure 3.8.

One of the purposes of software testing is to provide a guarantee that the software conforms to the requirements. It is very difficult to prove, through manual testing and automated testing, that the software conforms to the requirements. It is possible to provide reesuarnce that the software conforms to the requirements with formal verification, but we have to assume that sections of the product which are not part of the formal verification are also correct. The code coverage reports which could be generated by EMMA software can serve as an intermediate step prior to formal verification. Emma is a Java code coverage software [69]. EMMA's coverage reports are very helpful for eliminating unreachable and dead code, because it provides a graphical representation of the code base by highlighting tested code as green and, untested code as red.

EMMA Coverage Report (generated Sun Mar 21 00:24:18 EDT 2010)
[all classes]

OVERALL COVERAGE SUMMARY

name	class, %	method, %	block, %	line, %
all classes	50% (7/14)	26% (21/80)	7% (179/2432)	12% (56.3/488)

OVERALL STATS SUMMARY

total packages:	2
total executable files:	14
total classes:	14
total methods:	80
total executable lines:	488

COVERAGE BREAKDOWN BY PACKAGE

name	class, %	method, %	block, %	line, %
PairwiseComparison.Internal	14% (1/7)	11% (7/64)	3% (63/2195)	5% (21/427)
PairwiseComparison.Internal.Test	86% (6/7)	88% (14/16)	49% (116/237)	58% (35.3/61)

[all classes]
EMMA 2.0.5312 (C) Vlad Roubtsov

Figure 3.9: An example of the Code Coverage Report

3.4.3 Static verification

Verification by an automatic theorem provers is very appealing and could be attractive to the average programmer, but it is very unlikely that they would be compelled to verify developed code with an interactive prover such as PVS or Isabelle. Currently,

it is almost impossible to verify large code bases with interactive theorem provers. Some smaller Java code bases can be annotated with the Java Modeling Language (JML), and its correctness can be validated by the Extended Static Checker for Java version 2 (ESC/Java2 software). This software can be incorporated into the Eclipse environment and become a part of the context aware framework to verify Java code.

The SoS organization, which belongs to the Radboud University Nijmegen, has developed a Loop tool. This tool is able to prove the correctness of the JavaCard, which is a subset of the Java language. It extracts poof obligations from the Java code that is annotated with JML which can then be verified interactively by the Prototype Verification System (PVS). This approach is preferred only for verifying a subset of the Java, and it can not be applied to verify applications written with all features of Java [42].

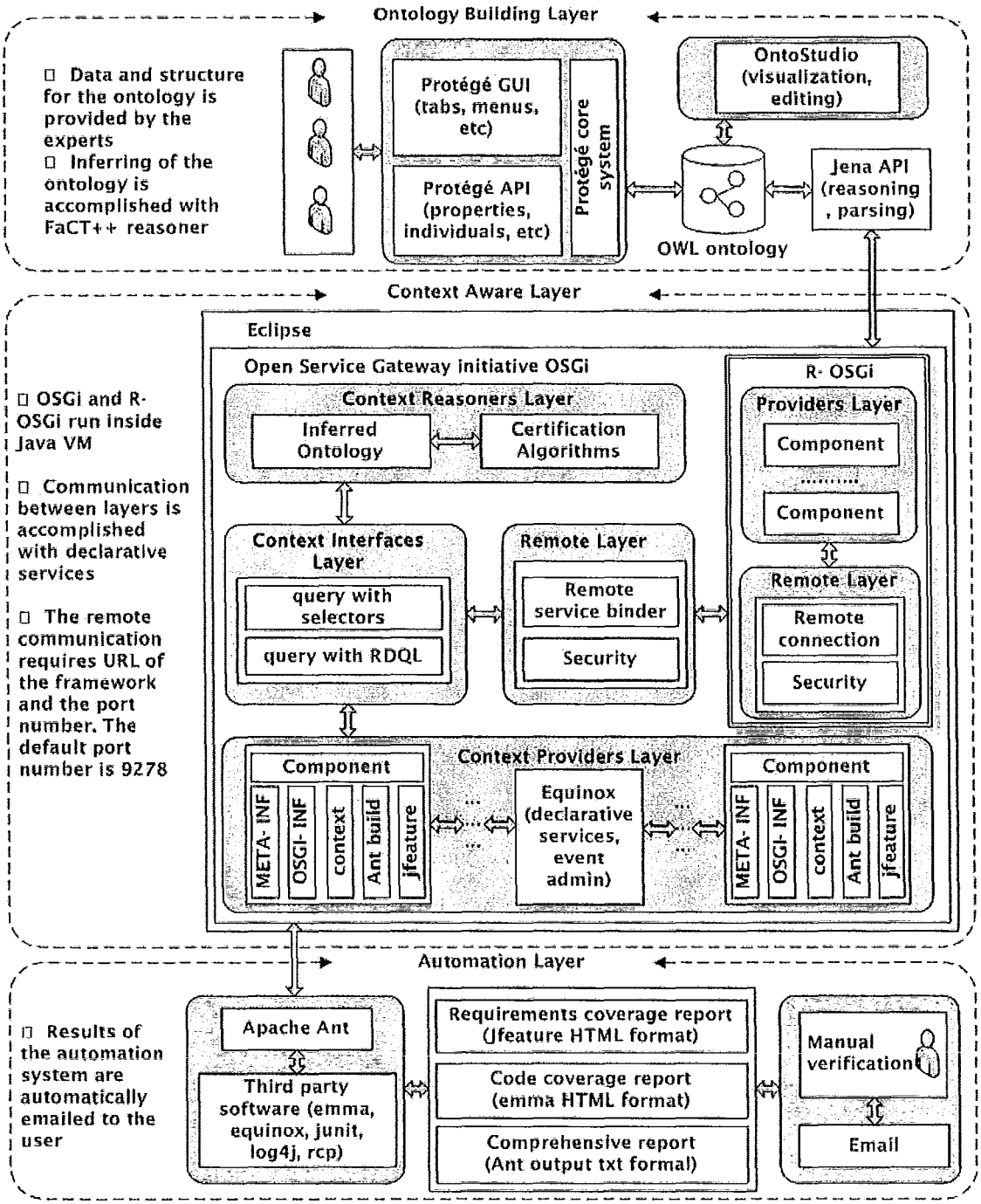

Figure 3.10: High level design of a context aware framework

Algorithm 11: Algorithm to bind with remote service

```
   Data: bundleContext an instance of the bundle
   Result: bind with remote service and test its validity
 1 begin
 2   ServiceReference serviceReference ← null
 3   RemoteOSGiService remoteOSGiService ← null
 4   RemoteServiceInterface remoteServIntrf ← null
 5   serviceReference ← bundleContext.getServiceReference(RemoteOSGiService.class.getName)
 6   if serviceReference ! = null then
 7     remoteOSGiService ← (RemoteOSGiService) bundleContext.getService(serviceReference)
 8   end
 9   else
10     return error ← ClientStatus.ERROR_REMOTE_SERVICE.toString
11   end
12   Hashtable localProperties ← new Hashtable
13   localProperties.put(EventConstants.EVENT_TOPIC, new String[]
     ClientStatus.REMOTE_SERVICE_NAME.toString )
14   bundleContext.registerService(EventHandler.class.getName(), this, localProperties)
15   if Boolean.getBoolean(ClientStatus.R_OSGi_SERVICE_DISCV.toString) then
16     bundleContext.registerService(ServiceDiscoveryListener.class.getName(), new
       ServiceDiscoveryListener {
17       public void announceService(String remoteServiceIntr, URI uri) {
18         remoteOSGiService.connect(uri)
19         RemoteServiceReference ref ← remoteOSGiService.getRemoteServiceReference(uri)
20         remoteServIntrf ← (RemoteServiceInterface) remoteOSGiService.getRemoteService(ref)
21         validateRemoteService
22       }
23       public void discardService(String remoteServiceIntr, URI uri) {
24         report(ClientStatus.SERVICE_NOT_BOUNDED.toString + uri)
25       }
26     } , null)
27   end
28   else
29     URI uri ← new URI(System.getProperty(ClientStatus.R_OSGi_URI.toString,
30     ClientStatus.DL+ClientStatus.LOCALHOST+ClientStatus.LOCAL_PORT))
31     remoteOSGiService.connect(uri)
32     RemoteServiceReference[] refs ← remoteOSGiService.getRemoteServiceReferences(uri,
       RemoteServiceInterface.class.getName(), null)
33     Iterator servicesItr ← Arrays.asList(refs).iterator
34     RemoteServiceReference remoteService ← null
35     while servicesItr.hasNext do
36       remoteService ← (RemoteServiceReference) servicesItr.next
37       if remoteService.getClass.getName.endsWith(ClientStatus.REMOTE_SERV_IMPL.toString)
         then
38         if REMOTE_CLIENT == REMOTE_PROXY then
39           remoteServIntrf ← (RemoteServiceInterface)
             remoteOSGiService.getRemoteService(remoteService)
40         end
41         else if REMOTE_CLIENT == BUNDLE_NUMBER then
42           remoteServIntrf ← (RemoteServiceInterface)
             remoteOSGiService.getRemoteServiceBundle(remoteService, 0)
43         end
44         break
45       end
46     end
47     validateRemoteService
48   end
49 end
```

Chapter 4

Process to derive software metrics

This chapter describes the process of deriving software metrics for measurable and subjective attributes which can be used to evaluate product. This chapter will also discuss history, purpose, types, motivation and views on software metrics.

4.1 Quality as a driving factor for software metrics construction

The quality of the product is one of the driving factors behind software certification. Within the last few years not only software developers, but software users have begun allocating greater attention to software quality [66], [14]. These expectations inspired many organizations to implement practices which would enable them to provide evidence of the quality of their products. It is possible to consider quality as one of the primary properties for any product. Software has a tendency to evolve at a rapid pace. Therefore, measuring mechanisms which are available must change rapidly in order to adjust to the ever changing quality demands while results must be repeatable.

In general, almost all software certification models adopt a series of standards which are well known and supported by the industry. For example, the ISO/IEC ISO 9126 outlines six main attributes of the software such as maintainability, reliability, efficiency, functionality, usability and portability. A general description of these at-

tributes is given in Table 4.5. These six attributes are subdivided into sub-attributes and the possible subdivisions are shown in Table 4.1. Constructing an acceptable measure for a top general attribute could be infeasible and unpractical. Therefore, the subdivision of the attributes is required in order to allow for the criteria at lower levels to be evaluated by software metrics more accurately. The actual subdivision of the top attributes is not a part of the official standard, but is applied in almost all software certification models.

Attribute	**Sub-attribute**
reliability	fault tolerance, maturity, recoverability
efficiency	resource performance, time performance
functionality	accuracy, compliance, interoperability, security, suitability
maintainability	analyzability, changeability, stability, testability
usability	learn ability, operability, understandability
portability	adaptability, insatiability, replacing ability

Table 4.1: Attribute breakdown as supported by ISO 9126 [14], [61]

Determining the value of some of the attributes could be one of the most subjective tasks in the evaluation process. It is important to mention that subjective evaluations are not fully supported by the software industry. It is possible that one evaluator may evaluate attributes in a positive way, while another might evaluate them negatively. Valid evaluations should only be based on objective measurements instead of individual preference. Software metric could produce objective measurements. It also has an ability to provide indirect evaluation of the attributes. Users should provide data based on the system's features, and that data should be used for the evaluation of specific attributes. At the earliest stages of metric usage only low level design and code were considered in the evaluation process. For example, McCabe's Metric focused only on the low-level design and source code [87].

Over time, and with the introduction of third party evaluators, the need to look at software attributes rather than just the initial design and source code has

begun to increase. Third party evaluators are often overwhelmed with claims and statistics about specific products. Different companies also have a tendency to use different reporting techniques. Therefore, it is difficult sometimes to reuse evaluation methods. Everyone involved in the certification process should agree on a set of software metrics. The agreement should involve regulations on how the evaluation metrics will be implemented. The P1061 (Standard for a Software Quality Metrics Methodology) standard attempted to standardize software metrics for similar usage, and ISO/IEC 9126 dictates principals for software quality evaluation [66], [14]. The quality of the software could be directly derived from its characteristics. ISO 9000 (Quality Management and Quality-Assurance Standards), which was released in 1987, references a collection of international standards which deal with processes as a means of deducing software quality. This standard is already a few decades old and does not properly reflect current scenarios of software quality evaluations. Standards, such as ISO JTCl SC7, often update the principles of the development process. In 1992 the Computer Society released the P1209 standard (A Recommended Practice for the Evaluation and Selection of CASE Tools). This standard outlines a general software evaluation framework. In addition, it has almost a complete collection of evaluation characteristics based upon ISO 9126. SC7 WG4 is an international standard which can help with the evaluation of CASE tools [87]. The computing society has done some work on the development of standards for safety critical systems such as IEC 1508 [26].

4.2 Software metrics

One of the known ways to collect information about software is by using software metrics. Within the last few decades a collection of software metrics have been proposed, but some of them are unpractical because it is difficult to interpret their

final results [20]. This difficulty is a result of the paramount values of the measuring criteria often being unknown. Some attributes are very subjective and very difficult to measure, but it is possible to construct a measuring mechanism for the attributes which are less subjective such as traceability and portability.

Software metrics evolved during three periods as their principles were exposed to the software engineering community. These three periods are the introductory period (1971-1985), growth period (1985-1997) and current period (1997-present time). During the introductory period the theory of metrics was just beginning to be experimented with. During the growth period the development of software which utilized software metrics began, and at the same time, the acceptance of metrics also increased. There is a noticeable spread of the metrics in the software industry. The main use of metrics has changed during these three periods, and in particular the views and acceptance of metrics have changed [56].

4.2.1 Purpose of the metrics

Benefits which could be obtained by using software metrics, as outlined by Goldensen from the Software Engineering Institute at Carnegie Melon, are listed below [13].

- Will it create a universal understanding of the project status
- Will it deduce required processes and information necessities
- Will it deduce acceptable methods of measure according to the expectations and requirements
- Will it identify entities that should be measured and also store, analyze and collect information after the measurement procedure.
- Will it provide measurement results that will inspire discussions
- Will it provide some sense of measure and understanding to the customer [13]

Metrics are intended for measuring products and processes. We think, in order for the software certification process to be successful it should be based on the analysis

of the product and its supporting documents. As mentioned above the product is the most important entity. The modified Process Capability Metric (PCM) can be used for the analysis of the process [13]. The product, on the other hand, should be evaluated with a collection of metrics. In some situations different collections of metrics must be used and it will depend on the product. For example, if the product was developed with a language that supports object orientation, then the software metrics which were developed by Chidamber and Kemerer could be used in the analysis process [16], [10], [55]. Table 4.2 lists software metrics that were described by Chidamber and Kemerer et al. [16]. Hitz and Montazeri proposed a more detailed implementation of metrics for object oriented languages as compared with the metrics proposed by Chidamber and Kemerer [33].

Metric	**Measurement criteria**
Weighted methods per class - *WMC*	Calculates sum of the weights of methods in every class
Depth of inheritance tree - *DIT*	Calculates the maximum distance object can achieve in the inheritance tree, the distance is considered from the root of the tree
Number of children - *NOC*	The number of classes inheriting attributes from the parent class
Coupling between object classes - *CBO*	The count represents the number of coupling with other classes
Response for a class - *RFC*	Responsiveness of the class that is based on private and public methods

Table 4.2: Chidamber and Kemerer OO metrics [16]

4.2.2 Motivation, views and types of software metric

"No single metric can provide wisdom!"[13] Therefore, a collection of metrics must be used in order to gain a clear understanding of the project. The following questions could be asked during the construction of the metric.

- What is impossible to manage or measure?

- What degree of criticism is acceptable?
- What degree of expert opinion should we consider?
- Are we looking for an indicator or measurement? [13]

This list of questions could be very large. The lack of widely accepted software certification standards leads to conflicts and wide debate. The low success of software metrics programs could be because software metrics programs can be viewed in many different ways [13]. Table 4.2.2 describes some of the different views on metrics which different individuals may have. This table could be a subject for the first conversation between the certifier and developer during the initial stage of the software certification process. It is very apparent that the need for software measurement and validation has grown and there was a drift in the sixties and seventies where the primary concern of measurement was the product. In the eighties and nineties the concern of measurement focused upon the process and quality scheme, and after the nineties measurement concern shifted towards process incorporation. In order for any measurement technique to be successful it must provide *'a positive return on investment'*. The return must provide a noticeable benefit to the entire business and not only the developing team [13]. Table 4.2.2 examines types of software metrics, all of which could be used in the certification process.

Participant	**Steps to derive measurement objectives**		
	Interests	**Goal**	**Metric**
manager	economic	costs, dates	effort, quality
developer	technical	development environment	size, complexity
end user	social	usability	functionality
estimator	economic	costs, effort, dates	effort, budget, project size, duration
project manager	technical	effort, dates, size, complexity	earned value, progress to date, impact of change

Table 4.3: Point of view for the metrics [13]

Metric	Description	Forms	Examples
absolute metrics	fundamental measures are manipulated to collect new information	single data sums, averages, differences	start and end dates, software size, effort in person hours
relative metrics	relative data, structuring data, relational data, relating several absolute measures together	factorial figures, relationships, derived data	percentages of function points attributable to EIs compared to the total size of the software
coefficients	indicators, maximum, average and Minimum, calculated from other metrics on a time series basis and used for comparison	measured data chronicled over time, metrics	the relation of IT effort to total business
index figures	figures for general presentation of many changes of organizational data	percentages, single figures indexed, basic values	annual increase in productivity

Table 4.4: Types of software metrics [13]

4.2.3 Metric construction

In 1984 Weiss and Basili created the concept known as *'Goal-Question-Metric'* (*GQM*). This concept describes steps on how to build a measurement process. Building the measurement process could be subdivided into two approaches, such as bottom-up and top-down. The *'Goal-Question-Metric'* is an example of a bottom-up approach, while the Capability Maturity Model (CMM) and Capability Maturity Model Integrated (CMMI) are examples of top-down approach [24]. The top-down methodology focuses on benchmarking and evaluation, while the bottom-up methodology focuses on implementing measurements with the intention for improvement. The CMM model was developed by Software Engineering Institute (SEI) at Carnegie Mellon between 1987 and 1997. The CMMI model was also developed by SEI and was introduced in 2002 [81]. The main goal of the measurement process is to provide feedback about the product to the developer. Measuring methods could often require dedication and

a vast amount of data. This may contribute to the implementation of a complex measuring process. Consequently, some sections of the product may never get properly evaluated. We think GQM is a preferred method, because it is able to focus on specific areas of the product which need evaluation. There always should be a purpose and judgment as to why certain attributes or sections are being evaluated [9].

GQM is probably one of the most practical ways to develop measuring metrics. Since it was first introduced, organizations such as NASA have used this method in their evaluation process. The strongest feature of GQM is its ability to transform business goals into a collection of characteristics which can be measured. The objective of GQM concept, which is shown in Figure 4.1, could be summarized into three steps. Step one: team members and certifiers outline business goals. Step two: for every business goal a set of questions is constructed and answered in order to determine whether or not business goals were achieved. Step three: for every business goal a metric or a collection of metrics are defined in order to provide feedback on them [71]. Figure 4.1 also demonstrates the relationships between phases of the GQM concept and it tries to incorporate objectives of the ISO 15939 standard [24].

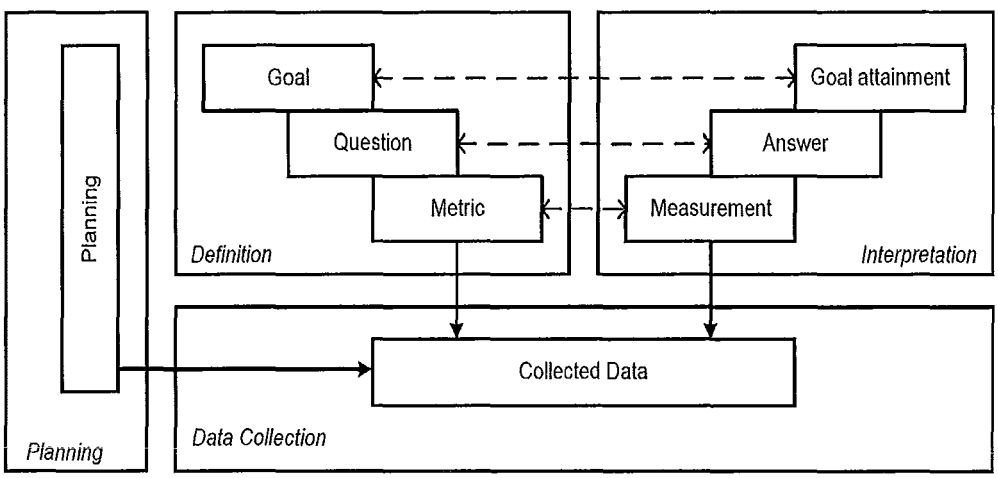

Figure 4.1: Goal Question Metric methodology [48], [24]

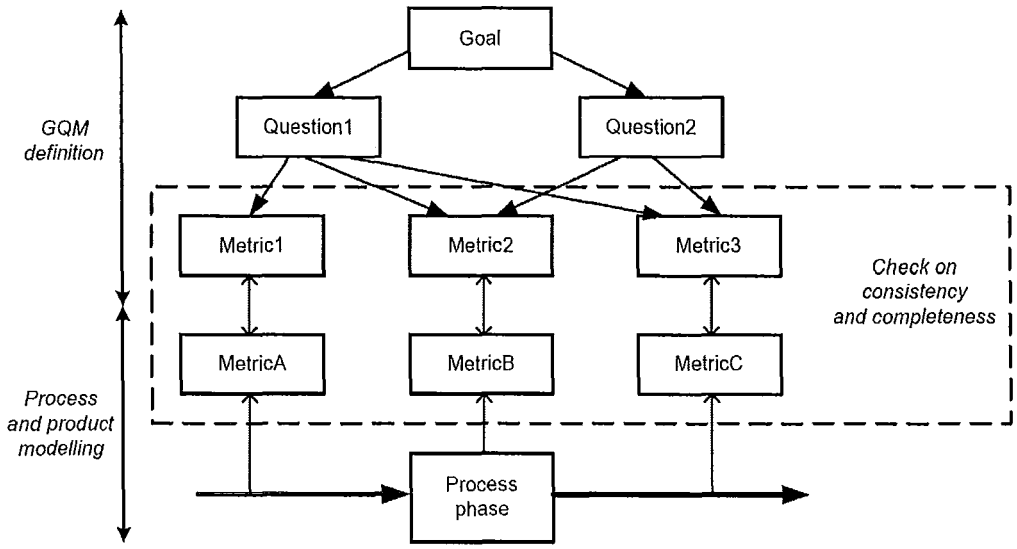

Figure 4.2: Goal Question Metric approach [48], [24]

The topology of the GQM concept has four phases: planning, definition, data collection and interpretation. In the planning phase, the measurement applications are selected and defined. In the definition phase, the measurement plan is defined and documented. The goals, questions, hypotheses and metrics are also defined. In the data collection phase, all the data is being collected. In the interpretation phase, the collected data is being evaluated with defined metrics which produce measurements. These measurements are used to deduce answers for the identified questions. In the end, a goal attainment could be assessed. The interpretation phase is the most important because during this phase the results are the most criticized. Figure 4.2 demonstrates more detailed steps which could be taken during the GQM process to construct a specific metric. A large number of attributes could be considered during the evaluation of a given product. Therefore, a large number of software metrics could also be required. Table 4.5 lists some of attributes which could be considered for the evaluation [10], [26].

Attribute	**Some description of what to measure**
Accuracy	a possibility for an error to occur, accuracy in control
Adaptability	new components and features could be added easily
Auditability	document can be easily verified against documentation
Availability	percentage that represents the systems' availability for the use
Changability	how easily the program could be changed
Completeness	how complete is the implementation against the requirements
Conciseness	how precise and brief are the lines of code
Consistency	how consistent the code is against design and documentation
Correctness	all specifications are satisfied, user expectations are satisfied, does not contain any known errors
Data commonality	use of well know types and data structures
Dependability	the same as reliability
Efficiency	complex or simple arithmetic operations, number and complexity of nested loops, complex arrays
Error tolerance	how damaging the occurrence of an error to the system
Expandability	design of the system is able to support expansion
Flexibility	resources that have to be invested in order to modify or change the system
Functionality	security, generality and capability of the system
Generality	extensiveness of the system
Hardware independence	to what degree the system is dependent on some type of hardware
Human factors	how usable is the software, friendliness of the user interface
Integrity	how the unauthorized access is managed
Interoperability	what resources are needed for the system to get integrated with other systems
Maintainability	how long does it take to recognize and analyze the problem, what time is needed to change the system and requirements
Modifiability	how easy it is to change the system
Modularity	information hiding, functional independence
Operability	how easy it is for the user to operate the system
Portability	resources that are required in order for the system into be transfered to a different unknown environment
Reliability	performance of the function is accurate
Reusability	the infrastructure of the system is standardized
Robustness	ability to operate during invalid input
Security	a mechanism that protects the system
Self-documentation	how descriptive is the code
Simplicity	how easy it is to understand the system
Supportability	how easy it is to support the system
Testability	resources that are needed to accurately test the system
Traceability	ability to trace requirement to the code
Transportability	the same as portability
Understandability	how easy it is to understand the program for the novice user and programmer
Usability	what resources are required in order to learn how to use the system
Utility	to what degree the system satisfies its intended purpose

Table 4.5: Software Quality Attributes [10], [26]

Chapter 5

Conclusion and future research

5.1 Conclusion

Software certification is becoming an important area of research as more software is being developed which not only controls hardware, but quite literally our daily lives. Our research focused on addressing hurdle eight of the software certification process which was defined by Hatcliff, Heimdahl, Lawford, Maibaum, Wassyng and Wurden et al. [31]. It stated: *Lack of interoperable tools to manage, reason, and provide traceability - The result is that small change often requires a large effort. We need tools that scale.*

We have developed a proof of concept context aware framework which provides a dynamic environment for the software certification process by integrating development and certification domains. The integration of domains is achieved with the help of upper ontology which supports formal information exchange and reasoning. The upper ontology was developed in Web Ontology Language (OWL). The framework could adapt to the changing certification demands by being able to adapt seamlessly to the evolved context. Such flexibility allows for the certification process to expand more easily into different domains. It is utilizing a collection of tools such as Eclipse, Jena, OntoStudio, Protégé, Equinox, EMMA, log4j, RCP, JUnit, Jfeature and Apache

Ant and was developed within an Open Service Gateway initiative (OSGi) environment. We selected OSGi environment because it allows for the framework to span over many physical machines. The communication between physical machines can be accomplished with Remote Open Service Gateway initiative (R-OSGi). With the help of OSGi and R-OSGi we can create a virtual framework by incorporating many physical frameworks. The framework also has a collection of automation scripts which manage generation of the requirements coverage and code coverage reports. Importantly, automation scripts could be used to manage many other tasks of the software certification process. We expect that the certification models which were utilized for past projects could be incorporated with possible modifications into our context aware software certification framework. This could allow for a more accurate and consistent software certification process. The framework also permits for the software certification to occur at intermediate stages of product development. This allows for intermediate releases of a product while maintaining historical records of the certification. Overall, the software certification process could be a very large and complex task. We believe that our context aware framework could make the process of software certification more manageable and applicable in academia and industry.

We also described a product based software certification process which is structured on component based certification principals. The core of the certification process is composed from a variety of known software certification models. Therefore, we consolidated beneficial features of other certification models into a single certification model. The process incorporates methodologies such as Integrated Component Maturity Model (ICMM), Plan the evaluation, Establish criteria, Collect data and Analyze data (PECA) framework and Goal Question Metric (GQM). It also tries to be aligned with ISO JTC1 SC7, ISO IEC 25000, ISO 15939, ISO IEC 14598 and ISO 9126 standards. Incorporating different methodologies into the certification process

allowed us to address *Hypothesis-5* that was described by Keith and Vertinsky et al. [40]. It stated: *"Companies that are more methodologically rigorous are more likely to choose to certify than companies that are relatively less methodologically rigorous."*

As projects evolve and grow in complexity, different properties and attributes which were not part of the certification process could become an important component of the certification. During the research, we did not come across an acceptable measuring technique for many attributes. For example, the attributes such as complexity and interoperability. In order to overcome this obstacle, we have described a process on how to derive software metrics for measurable and subjective attributes. The other important issue which we addressed is the maintenance of consistent priorities (weights) for the attributes and properties. We demonstrated the applicability and benefits of the Pairwise Comparison (PC) method in the software certification process and how it can be used to assign consistent priorities (weights) to the attributes and properties. In addition, we demonstrated how the PC method could be used in correlation with Simple Additive Weighting (SAW) and Weighted Product (WP) methods to rank alternative plans.

5.2 Potential areas for the future research

The future goal of the research is to enhance the context aware framework by enhancing an automated section of the framework. The other goal is to implement an extensive verification of the messaging system and the frameworks' ability to function in situations in which some key components may fail. The beneficial feature that could be added is the integration of the framework's self analytical feature with the software certification process. The self analytical feature analyzes the status of the component within the OSGi environment. This integration would provide extra technical information which could be used in the certification process.

The other potential research goal is to contribute to the development of a general framework for the Software Knowledge Repository which would operate on a collection of ontologies. The achievements of the Software Engineering Institute (SEI) at Carnegie Melon are to be carefully rescarched in even greater detail to determine how they could be incorporated to further enhance the proposed framework.

Bibliography

[1] P. Adamic, V. Babiy, R. Janicki, T. Kakiashvili, W. W. Koczkodaj, and R. Tadeusiewicz. Pairwise comparisons and visual perceptions of equal area polygons. *Perceptual and Motor Skills*, 108(1):37–42, 2009.

[2] A. Alvaro, E. S. de Almeida, and S. R. de Lemos Meira. Software component certification: a survey. In *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference*, pages 106–113, Aug.-3 Sept. 2005.

[3] A. Alvaro, E. S. de Almeida, and de Lemos M. S. R. A component quality assurance process. In *SOQUA '07: Fourth international workshop on Software quality assurance*, pages 94–101, New York, NY, USA, 2007. ACM.

[4] M. Anholcer, V. Babiy, S. Bozki, and W. W. Koczkodaj. A simplified implementation of the least squares solution for pairwise comparisons matrices. *Central European Journal of Operations Research*, 2010.

[5] G. Arbi. A matrix-less model for tracing software requirements to source code. *International journal of computers*, 2(3):301–309, 2008.

[6] D. Aziz, J. Yahaya, and A. R. Hamdan. Software product certification: A continuous improvement. *International Journal of Knowledge, Culture and Change Management*, pages 125–134, 2008.

[7] V. Babiy, A. D. Bogobowicz, R. Janicki, and W. W. Koczkodaj. Selecting the best strategy for the software certification process (under review). In *C* Conference on Computer Science and Software Engineering, 2010 May 19 - 2010 May 21, http://confsys.encs.concordia.ca/c3s2e/*, 2010.

[8] R. Balasubramaniam and J. Matthias. Toward reference models for requirements traceability. *IEEE transaction on software engineering*, 27(1):58–93, 2001.

[9] P. Berander and P. Jönsson. A goal question metric based approach for efficient measurement framework definition. pages 316–325, New York, NY, USA, 2006. ACM.

[10] J. K. Blundell, M. L. Hines, and J. Stach. The measurement of software design quality. *Annals of Software Engineering*, 4(1):235–255, 1997.

[11] S. Bozóki. Solution of the least squares method problem of pairwise comparison matrices. *CEJOR*, 16:345–358, 2008.

[12] S. Bozóki and T. Rapcsák. On saaty's and koczkodaj's inconsistencies of pairwise comparison matrices. *Journal of Global Optimization*, 42(2):157–175, 2007.

[13] M. Bundschuh and C. Dekkers. Software measurement and metrics: Fundamentals. *The IT Measurement Compendium*, pages 179–206, Aug. 15, 2008.

[14] P. Caliman. Software product quality evaluation and certification: the qseal consortium methodology. *http://www.cse.dcu.ie/ essiscope/sm4/qseal.doc.*, Aug. 2009.

[15] M. J. A. N. Caritat. *Marquis de Condorcet: Essai sur l'Application de L'Analyse à la Probabilité des Décisions Rendues à la Pluraliste des Voix.* Imprimerie Royale., 1972.

[16] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, Jun 1994.

[17] S. Comella-Dorda, J. C. Dean, and P. Morris, E. Oberndorf. A process for cots software product evaluation. *COTS-Based Software Systems*, pages 86–96, 2002.

[18] B. Councill. Third-party certification and its required elements. In *Proc. of the 4th Workshop on Component-Based Software Engineering (CBSE), Canada*, May 2001.

[19] N. Cowan. The magical number 4 in short-term memory. a reconsideration of mental storage capacity. *Behavioural and Brain Sciences*, 20:87–185, 2001.

[20] F. H. Damborg and L. Mathiassen. Information-centric assessment of software metrics practices. *IEEE Transactions on engineering management*, 52(3):350–362, Aug. 2005.

[21] M. de los Angeles Mart and L. Olsina. Towards an ontology for software metrics and indicators as the foundation for a cataloging web system. *Web Congress, Latin American*, 2003.

[22] A. Deraman, J. H. Yahaya, F. Baharom, A. F. A. Fadzlah, and A. R. Hamdan. Continuous quality improvement in software certification environment. In *Proceedings of the International Conference on Electrical Engineering and Informatics Institut Teknologi Bandung*, pages 17–19, June 2007.

[23] P. Dobrev, D. Famolari, C. Kurzke, and B. A. Miller. Device and service discovery in home networks with osgi. *Communications Magazine, IEEE*, 40(8):86–92, 2002.

[24] C. Ebert and R. Dumke. Planning the measurement process. *Software Measurement*, pages 73–90, Jul. 25, 2007.

[25] G. T. Fechner. *Elements of Psychophysics.* Rinehart and Winston, New York, 1965.

[26] N. E. Fenton and S. L. Pfleeger. *Software Metrics.* PWS Publishing Company, 20 Park Plaza, Boston, MA, 2st edition, 1997.

[27] F. Garca, M. F. Bertoa, C. Calero, A. Vallecillo, F. Ruz, M. Piattini, and M. Genero. Towards a consistent terminology for software measurement. *Information and Software Technology*, 48(8):631 – 644, 2006.

[28] O. C. Z. Gotel and C. W. Finkelstein. An analysis of the requirements traceability problem. pages 94–101, Apr 1994.

[29] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. pages 48–57, 2003.

[30] W. Hasselbring and R. Reussner. Toward trustworthy software systems. *Computer*, 39(4):91–92, 2006.

[31] J. Hatcliff, M. Heimdahl, M. Lawford, T. Maibaum, A. Wassyng, and F. Wurden. A software certification consortium and its top 9 hurdles. *Electronic Notes in Theoretical Computer Science*, 238:11 – 17, 2009.

[32] P. Heck, M. Klabbers, and M. Eekelen. A software product certification model. *Software Quality Journal*, June 2009.

[33] M. Hitz and B. Montazeri. Chidamber and kemerer's metrics suite: a measurement theory perspective. *Software Engineering, IEEE Transactions on*, 22(4): 267–271, Apr 1996.

[34] M. Horridge, S. Jupp, G. Moulton, A. Rector, R. Stevens, and C. Wroe. A practical guide to building owl ontologies using protege 4 and co-

ode tools. 2009 `http://www.co-ode.org/resources/ tutorials/Protege OWLTutorial.pdf`.

[35] IEEE. Ieee standard for software maintenance. *IEEE Std 1219-1998*, Oct 1998.

[36] R. Janicki. Ranking with partial orders and pairwise comparisons. *Lecture Notes in Computer Science*, 5009:442–451, 2008.

[37] R. Janicki and W. W. Koczkodaj. A weak order solution to a group ranking and consistency-driven pairwise comparisons. *Applied Mathematics and Computation*, 94(2-3):227–241, 1998.

[38] R. Janicki and A. Wassyng. Tabular expressions and their relational semantics. *Fundam. Inform.*, 67(4):343–370, 2005.

[39] P. Kaur and H. Singh. Certification process of software components. *SIGSOFT Softw. Eng. Notes*, 33(4):1–6, 2008.

[40] F. G. Keith and I. Vertinsky. Antecedents to certification of software development processes. In *Standardization and Innovation in Information Technology, 2007. SIIT 2007. 5th International Conference*, pages 81–90, Oct. 2007.

[41] T. P. Kelly. *Improvements in System Safety.* Springer London.

[42] J. R. Kiniry, P. Chalin, and C. Hurlin. Integrating static checking and interactive verification: Supporting multiple theories and provers in verification. *Verified Software: Theories, Tools, Experiments*, 4171:153–160, 2008.

[43] R. Kluge, T. Hering, R. Belter, and B. Franczyk. An approach for matching functional business requirements to standard application software packages via ontology. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 1017 –1022, Aug 2008.

[44] W. W. Koczkodaj. A new definition of consistency of pairwise comparisons. *Mathematical and computer modelling*, 18(7):79–84, 1993.

[45] W. W. Koczkodaj. Statistically accurate evidence of improved error rate by pairwise comparisons. *Percept Mot Skills*, 82:43–48, 1996.

[46] W. W. Koczkodaj and W. O. Mackakey. Mineral-positional assessment by consistency-driven pairwise comparisons. *Explor. Mining Geol.*, 6(1):23–33, 1997.

[47] W. W. Koczkodaj, M. Orlowski, L. Wallenius, and R. M. Wilson. A note on using a consistency-driven approach to cd-rom selection. *Library software review*, 16 (1):4–11, 1997.

[48] H. Koziolek. Goal, question, metric. *Lecture Notes in Computer Science*, 4909: 39–42, May 29, 2008.

[49] C. W. Kurt. Software component certification: 10 useful distinctions,. *CMU/SEI-2004-TN-031, Carnegie Mellon University,*, 2004.

[50] A. M. Lazarevska, N. Fischer, A. Haarstrick, and K. MNnich. A multi-criteria decision making conceptual approach to optimal landfill monitoring. *GeoSpatial Visual Analytics*, pages 85–96, 2009.

[51] K. Lee and S. J. Lee. A quantitative evaluation model using the iso/iec 9126 quality model in the component based development process. *Computational Science and Its Applications*, pages 917–926, 2006.

[52] T. Maibaum. *Challenges in Software Certification*, volume 4789. Springer Berlin / Heidelberg, 2007.

[53] T. Maibaum and A. Wassyng. A product-focused approach to software certification. *Computer*, 41(2):91–93, Feb. 2008.

[54] V. Mascardi, A. Locoro, and P. Rosso. Automatic ontology matching via upper ontologies: A systematic evaluation. *Knowledge and Data Engineering, IEEE Transactions on*, 22(5):609 –623, 2010.

[55] J. A. McQuillan and J. F. Power. On the application of software metrics to uml models. *Lecture Notes in Computer Science*, 4364:217–226, 2007.

[56] E. E. Mills. Metrics in the software engineering curriculum. *Annals of Software Engineering*, 6:181–200, Oct. 28, 2004.

[57] S. B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny. Context-aware service composition in pervasive computing environments. *Lecture Notes in Computer Science*, pages 129–144, 2006.

[58] R. Moraes, J. Dures, E. Martins, and H. Madeira. *Component-Based Software Certification Based on Experimental Risk Assessment.* Springer Berlin / Heidelberg, 2007.

[59] J. Morris, G. Lee, K. Parker, G. A. Bundell, and Chiou Peng Lam. Software component certification. *Computer*, 34(9):30–36, Sep 2001.

[60] Z. Nagy, G. Lukcsy, and P. Szeredi. Translating description logic queries to prolog. *Lecture Notes in Computer Science*, 3819:168–182, 2006.

[61] J. Oh, D. Park, B. Lee, J. Lee, E. Hong, and C. Wu. *Certification of Software Packages Using Hierarchical Classification*, volume 3026. 2004.

[62] G. O'Regan. Mathematical approaches to software quality. 2006.

[63] D. L. Parnas and P. C. Clements. A rational design process: How and why to fake it. *Proc. TAPSOFT Joint Conference on Theory and Practice of Software Development*, pages 25–29, 1985.

[64] D. K. Peters, M. Lawford, and W. B. Trancn. An ide for software development using tabular expressions. *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, pages 248–251, 2007.

[65] A. Rae, P. Robert, and H. Hans-Ludwig. *Software Evaluation for Certification.* McGraw-Hill, Inc., New York, NY, USA, 1st edition, 1994.

[66] K. Rangarajan, N. Swaminathan, V. Hegde, and J. Jacob. Product quality framework: a vehicle for focusing on product quality goals. *SIGSOFT Softw. Eng. Notes*, 26(4):77–82, 2001.

[67] R. Rao. *Decision Making in the Manufacturing Environment.* Springer London, 2008.

[68] J. S. Rellermeyer, G. Alonso, and T. Roscoe. R-osgi: Distributed applications through software modularization. *Lecture Notes in Computer Science*, pages 1–20, 2007.

[69] V. Roubtsov. Emma: a free java code coverage tool, april 2010, `http://emma.sourceforge.net/`.

[70] D. Rubio. *Pro Spring Dynamic Modules for OSGi Service Platforms.* Apress, 2009.

[71] K. Sacha. Evaluation of expected software quality: Acustomers viewpoint. *Fundamental Approaches to Software Engineering*, pages 170–183, 2006.

[72] H. P. Schnurr and J. Angele. ontoprise, nov. 2009, `http://www.ontoprise.de/en/home/ products/ontostudio`.

[73] A. Seaborne. Jena tutorial a programmer's introduction to rdql, january 2010, `http://jena.sourceforge.net/tutorial/ RDQL`.

[74] W. V. Siricharoen. A software engineering approach to comparing ontology modeling with object modeling. In *Computer Science and its Applications, 2008. CSA '08. International Symposium*, pages 320–325, 2008.

[75] I. Sommerville. *Software Engineering.* Pearson Education Limited, Edinburgh Gate, Harlow Essex, CM20 2JE, England, 8st edition, 2007.

[76] J. Souter. Process certification and product testing coming together. In *Software Quality Improvement Through Process Assessment, IEE Colloquium*, pages 5/1–5/6, Mar 1992.

[77] T. W. Swain and S. L. Scott. *Model-Based Statistical Testing of a Cluster Utility*, volume 3514. Springer Berlin / Heidelberg, 2005.

[78] A. Taleghani. Using software model checking for software component certification. In *Software Engineering - Companion, 2007. ICSE 2007 Companion. 29th International Conference*, pages 99–100, May 2007.

[79] L. L. Thurstone. Law of comparative judgements. *Psychological Review*, 34: 273–286.

[80] P. N. Tran and N. Boukhatem. Comparison of madm decision algorithms for interface selection in heterogeneous wireless networks. pages 119–124, Sept. 2008.

[81] A. K. Tripathi and Ratneshwer. Some observations on a maturity model for cbse. In *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference*, pages 273–281, June 2009.

[82] C. Tsinaraki, P. Polydoros, and S. Christodoulakis. Interoperability support between mpeg-7/21 and owl in ds-mirf. *Knowledge and Data Engineering, IEEE Transactions*, 19(2):219–232, 2007.

[83] H. van Kranenburg, M. S. Bargh, S. Iacob, and A. Peddemors. A context management framework for supporting context-aware distributed applications. *Communications Magazine, IEEE*, 44(8):67–74, 2006.

[84] A. I. Vermesan. Software certification for industry-verification and validation issues in expert systems. *Database and Expert Systems Applications*, pages 3–14, 1998.

[85] J. Voas. The software quality certification triangle. *The Journal of Defense Software Engineering*, 11(11):12–14, 1998.

[86] J. Voas and K. Miller. Software certification services: Encouraging trust and reasonable expectations. *IT Professional*, 8(5):39–44, 2006.

[87] T. E. Vollman. Software quality assessment and standards. *Computer*, 26(6): 118–120, Jun 1993.

[88] M. Woodman, O. Benebiktsson, B. Lefever, and F. Stallinger. Issues of cbd product quality and process quality. In *Proc. of the 4th Workshop on Component-Based Software Engineering (CBSE), Canada*, 2001.

[89] J. H. Yahaya, A. Deraman, and A. R. Hamdan. Scfm_prod: A software product certification model. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference*, pages 1–6, April 2008.

Appendix A

Metric for assigning consistent weights

In situations where it is difficult or infeasible to use an algorithm, we revert to the use of heuristics in order to find solutions. There are a large number of attributes which should be considered during the certification process. As projects evolve rapidly and grow in complexity we need mechanisms to assign consistent weights to attributes and properties. The pairwise comparison method is ideal for this task because it can reduce inconsistencies while still maintaining some acceptable margin of error. This chapter describes an approach on how to assign consistent weights to ontology classes which are associated with attributes and properties. Once the inconsistency is minimal, preferably not zero, the developed ontological model can be used as a dynamic entity in the software certification process. It is very difficult and not advisable to achieve zero inconsistency between all ontology classes [44], [46]. In addition, the Multiple Attribute Decision Making (MADM) method could be used to model the scenario for ranking alternative plans in situation where one ore more experts are present. Every expert provides one or more alternatives where the consistency of every alternative is achieved by using the pairwise comparison (PC) method. The actual ranking of alternative plans is accomplished by both the Simple Additive Weighting (SAW) and the Weighted Product (WP) methods.

A.1 Pairwise comparison method

The pairwise comparison method was used for the first time in 1785 by Condorcet. He was using this method in the election process where voters rank candidates based on their preference [15]. The method was a voting system which used matrices for particular pairwise comparisons with rows representing each candidate as a runner and columns representing each candidate as an opponent. It was Fechner who specified pairwise comparisons as a scientific method in 1860, although only from the psychometric perspective [25]. Thurstone, in 1927, provided a mathematical analysis of this method and called it the *law of comparative judgments* [79]. The *law of comparative judgments* can be used to scale a collection of attributes based on simple comparisons between attributes taken two at a time. Although, Thurstone referred to it as a law, it can be more appropriately identified as a measurement model which could be of important use for software certification. This model allows experts to synthesize diverse procedures involved in software certification. The hierarchy reduces the number of comparisons from $O(n^2)$ to approximately $O(n \ln n)$, making it applicable to a wide variety of problems. For example, a moderate case with 49 features would require 1,176 comparisons without a hierarchy and only 168 comparisons of these 49 features are arranged into hierarchy by grouping seven features. Measurements of length such as a meter or foot or by mass and weight are commonly used and accepted. Society has become accustomed to have standards for the majority of tasks, and sometimes it is difficult to understand standards, which often occur in the software industry, without an acceptable universal measuring method. In the case of software certification many models may need to be developed for a single project. It is safe to conclude that developing a single certification model is not feasible and would not work for all types of projects, because some projects have very little in common. In the context of software certification, the introduction of a hierarchical ontological structure can

express the fundamental knowledge that could be used for the software evaluation [46], [47], [37].

A.1.1 Demonstration of the analysis

Every class within the ontology is associated with an attribute and all its data. For the purpose of demonstrating how to assign consistent weights to ontology classes we will refer to an ontology class by the name of the attribute with which it is associated with. The pairwise comparison method does not impose any limit on the number of criteria. Setting the maximum number of entities on one level to seven is accepted as heuristic, because seven items gives 21 distinct pairs to compare. The first step of pairwise comparisons is to establish the relative preference of two criteria for situations in which it is impractical or irrelevant to provide the absolute estimations. The relative comparison coefficients a_{ij} for criteria $C_1, C_2, \ldots, C_n$ are expected to satisfy $a_{ii} = 1$ and $a_{ij} = 1/a_{ji}$. The first constraint is related to comparing a given attribute with itself. The second constraint is a consequence of the obvious fact that $x/y = 1/(y/x)$ for $x, y \neq 0$. A scale from 1 to 5, as demonstrated in Table A.1, is used for expressing the importance of one attribute over others. This is accomplished in a pair. Other scales also exists, but as described by [44] larger values lose meaning in the comparison process.

Code	Definition of intensity or importance
1	Equal or unknown importance
2	Weak importance of one over another
3	Moderate to essential importance
4	Demonstrated importance
5	Absolute importance
3.5 etc	Intermediate importance

Table A.1: Comparison scale

The absolute estimations of the weights defining the importance of analyzed software certification criteria are practically unobtainable through either statistical or formal procedures. It would be beneficial to have experiments which may contribute to the accuracy of the estimates. However, it is unrealistic to expect such experiments to take place. This approach allows us to improve the processing of often subjective expert assessments in the certification process. We propose the use of the following comparison scale, that is demonstrated in Table A.1, for the subjective expression of relative preference.

Reference	Criterion
C_1	Functionality
C_2	Reliability
C_3	Usability
C_4	Efficiency
C_5	Maintainability
C_6	Portability

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	1	1	3.7	3	4	4.3
C_2	1	1	3	2	4	3.7
C_3	0.27	0.33	1	0.6	1	1.7
C_4	0.33	0.5	1.6	1	2	2.1
C_5	0.25	0.25	1	0.5	1	1
C_6	0.23	0.27	0.58	0.47	1	1

Figure A.1: Relative importance of considered software quality attributes

The values of relative importance, which are listed in the above table, have been entered by a single person, solely for demonstration of the method, and deduced from the comparison in pairs. In a real scenario, the values should be reasoned by a team of experts. The attributes have been taken from the ISO/IEC 9126 software standard. It is also known as the top six level attributes, which are considered to be key attributes for the software quality [65].

In the pairwise comparisons method attributes are presented in pairs to one or more experts. It is necessary to evaluate individual attributes, derive weights for the attributes, construct the overall ratings of the alternatives, and to identify the best alternative. Lets denote the attributes by $A_1, A_2, \ldots, A_n$ (n is the number of compared attributes), their actual weights by $\gamma_1, \gamma_2, \ldots, \gamma_n$, and the matrix of the ratios of all weights by $\Gamma = [\gamma_i/\gamma_j]$. The matrix of pairwise comparisons M $= [a_{ij}]$

represents the assessments between individual pairs of alternatives (M_i versus M_j, for all $i, j = 1, 2, ...n$) chosen usually from a given scale. The elements a_{ij} are considered to be estimates of the ratios γ_i/γ_j, where γ is the vector of actual weights of the attributes. All the ratios are positive and satisfy the reciprocity property $a_{ij} = 1/a_{ji}$, $i, j = 1, 2, \ldots, n$. The inconsistency concept was explained in [12]. The distance based inconsistency indicator is defined as the maximum over all triads $\{a_{ik}, a_{kj}, a_{ij}\}$ of elements of M (with all indices i, j, k distinct) of their inconsistency indicators. An implementation is demonstrated in Algorithm 12.

Algorithm 12: Algorithm to compute maximum inconsistency and its position

```
   Data: square inconsistent decision matrix MatrixA
   Result: maxInconsistency and max inconsistency positions: iPosition, jPosition and kPosition
 1 begin
 2     initialization
 3     double maxInconsistency ← 0
 4     double inconsistancyLocal ← 0
 5     int iPosition ← 0, jPosition ← 0, kPosition ← 0
 6     double[] result ← new double[4]
 7     computation
 8     for int i ← 1 to i ≤ MatrixA.length i++ do
 9         for int j ← i + 1 to j ≤ MatrixA.length j++ do
10             for int k ← j + 1 to k ≤ MatrixA.length k++ do
11                 inconsistancyLocal ← Math.max(
12                 (1 - (MatrixA[i - 1][k - 1] / (MatrixA[i - 1][j - 1]
13                 * MatrixA[j - 1][k - 1]))),
14                 (1 - ((MatrixA[i - 1][j - 1] * MatrixA[j - 1][k - 1])
15                 / MatrixA[i - 1][k - 1])))
16                 if inconsistancyLocal > maxInconsistency then
17                     maxInconsistency ← inconsistancyLocal
18                     iPosition ← i-1, jPosition ← j-1, kPosition ← k-1
19                 end
20             end
21         end
22     end
23     result[0] ← maxInconsistency, result[1] ← iPosition,
24     result[2] ← jPosition, result[3] ← kPosition
25     return result
26 end
```

Three is the minimal number of attributes which may cause inconsistency. Comparing two attributes will often lead to inaccuracy. The distance based inconsistency is the minimum distance from three ideal triads with no inconsistency when the third value is substituted using the consistency condition $a_{ij} \times a_{jk} = a_{ik}$. Since we are

not in a position to determine which ratio is incorrect, all three assessments must be reconsidered before we attempt finding a consistent approximation for a given pairwise comparisons matrix. The stress on localizing the most inconsistent assessments is expressed by adding the *consistency-driven* to the name of the method since it is easier to remedy implications of an error when we are able to localize it. There is no practical reason to continue decreasing the inconsistency indicator to zero. Only the high values of the inconsistency indicator are considered as unacceptable and harmful. A very small value, or zero, may indicate a faked result rather than a true estimate. The practical challenge in working with the pairwise comparison method comes from the lack of consistency of the pairwise comparisons matrices. Depending on the model it may take some time to get the matrix consistent [46], [47], [37], [36].

A.1.2 Demonstration of the matrix adjustment

Assume the following attributes are considered for evaluation: safety, security, reliability, resilience, robustness, understand ability, testability, adaptability, modularity, complexity, portability, usability, reusability, efficiency and learn ability. They are considered by [75] as a general group of attributes for any software. All the entities are subdivided into into two main categories, such as development and maintenance. These groups are subdivided further and weights are assigned as demonstrated in the Table A.2. It is safe to assume that some areas of software evolution are based on intuition and experience. In situations where there is more than just one person making decisions there is a greater possibility for inconsistency to occur. Industry must rely on the subjective judgments of experts in situations where the practical methods of measure are unknown [46], [47].

From Table A.2 lets evaluate the complexity, portability and reusability triad where C_1 = complexity, C_2 = portability and C_3 = reusability. This triad has an

root: 100%			
development: 80%			maintenance: 20%
efficiency: 15.18%	**modularity: 43.78%**	**reliability: 21.05%**	usability: 10.56%
resilience: 7.85%	complexity: 25.17%	understand ability: 11.75%	learn ability: 6.65%
robustness: 5.44%	portability: 13.32%	safety: 6.73%	testability: 2.79%
adaptability: 1.89%	reusability: 5.29%	security: 2.57%	

Table A.2: Redistribution before adjustment

$$\begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array}\begin{array}{c} \begin{array}{ccc} C_1 & C_2 & C_3 \end{array} \\ \begin{bmatrix} 1 & 3 & 3 \\ 0.33 & 1 & 4 \\ 0.33 & 0.25 & 1 \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \end{array}\begin{array}{c} \begin{array}{ccc} C_1 & C_2 & C_3 \end{array} \\ \begin{bmatrix} 1 & 1.25 & 1.75 \\ 0.33 & 1 & 2 \\ 0.33 & 0.25 & 1 \end{bmatrix} \end{array}$$

inconsistency = 0.75 inconsistency = 0.30

Figure A.2: Inconsistency analysis for a group with three attributes

inconsistency of 0.75. As described in [44] it is not recommended. According to [44] the acceptable inconsistency is around 0.33. We have to adjust the values in order to bring the inconsistency down. After the adjustment, and as it is demonstrated in Figure A.2, the inconsistency has decreased to 0.30 which is more acceptable.

root: 100%			
development: 80%			maintenance: 20%
efficiency: 15.18%	**modularity: 43.78%**	**reliability: 21.05%**	usability: 10.56%
resilience: 7.85%	complexity: 18.18%	understand ability: 11.75%	learn ability: 6.65%
robustness: 5.44%	portability: 16.38%	safety: 6.73%	testability: 2.79%
adaptability: 1.89%	reusability: 9.22%	security: 2.57%	

Table A.3: Redistribution after adjustment

In Table A.2 the weight of the attributes are allocated based on the significance of the attribute, the most important criteria is complexity. After the correction we can see a new percentage redistribution, which is shown in Table A.3. The redistribution could be evaluated and adjusted by many experts in order to achieve a situation in which the redistribution is accepted by all experts. Compared to other attributes complexity could be considered the most important criteria in the evalu-

ation process. The percentage redistribution is needed in order to dictate the work load redistribution [30].

The consistency method is a preferred choice for the construction of the software certification models, because it eliminates a substantial amount of time which could be allocated for discussions. Meetings are very expensive, an acceptable and consistent model is the desired outcome after almost any meeting. The statistical evidence of the accuracy improvement with pairwise comparisons from approximately 15% to 5% for one dimensional case (randomly generated bars) in [45], and from approximately 25% to 15% for randomly generated 2D shapes [1] support our expectations of improvement. However, it is not easy to collect data for statistical analysis. A national repository (or a knowledge base) would help to do it. Different ontology models could be developed for a single project, however it all depends on the complexity of the project. It is vital to remove inconsistency from the ontology model while the judging committee is working on it. The pairwise comparison method also can be easily adjusted to new environments and requirements.

A.2 Multiple attribute decision making

Multiple Attribute Decision Making (MADM) method could be used to model the scenario for ranking alternative plans in situation where one ore more experts are present. Every expert provides one or more alternative plans where the consistency of every alternative is achieved through the use of the pairwise comparison (PC) method. The actual ranking of alternative plans is accomplished by the Simple Additive Weighting (SAW) and the Weighted Product (WP) methods. The Multiple Attribute Decision Making problem is formulated as follows: P = $\{P_1, P_2, \ldots P_n\}$ is a finite collection of alternatives and C = $\{C_1, C_1, \ldots C_m\}$ is a finite collection of criteria. The following steps could be taken during the construction of MADM problem:

research the project and identify areas of measure, refine areas of measure, identify potential criteria, agree only on relevant criteria, discuss potential alternatives, make judgments, eliminate infeasible and overlapping criteria, rank criteria and defined appropriate metric or metrics for measurement [50]. The Simple Additive Weighting (SAW) and Weighted Product (WP) methods are applied to rank alternatives. SAW method is one of the well know methods for solving Multiple Attribute Decision Making Problems. This method uses the weighted sum of all criteria to compute the overall score of an alternative. This is achieved by summing the normalized values of every x_{ij} i=1,2,..n j=1,2,..m and multiplying by the computed weight w_j j=1,2,..m. The weight w_j is also known as importance weight and can be computed with Algorithm 15 [80], [67]. The normalization of every x_{ij} is achieved through the pairwise comparison (PC) method.

$$X = \begin{array}{c} \\ P_1 \\ P_2 \\ \vdots \\ P_n \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_m \end{array} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix} \end{array}$$

Figure A.3: Matrix for MADM problems [80]

A.2.1 Demonstration of the analysis

The following section describes how to assign consistent weights for the functionality attribute. Based on the ISO 9126 standard, and as demonstrated in Table 4.1, this attribute can be subdivided further into accuracy, compliance, interoperability, security and suitability attributes. Assume there are two experts which provide their judgments and assign preferences to attributes with the pairwise comparison method.

Two judgments are demonstrated in Figure A.4. Let C_1 = accuracy, C_2 = compliance, C_3 = interoperability, C_4 = security and C_5 = suitability. For all decision matrices the independent vector w_j is computed where j = 1,2,..m.

$$E_1 = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{bmatrix} 1 & 2 & 2 & 2 & 2 \\ 0.5 & 1 & 1.5 & 1.5 & 1.5 \\ 0.5 & 0.6 & 1 & 1.5 & 1.5 \\ 0.5 & 0.6 & 0.6 & 1 & 1.5 \\ 0.5 & 0.6 & 0.6 & 0.6 & 1 \end{bmatrix} \end{array} \qquad E_2 = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{bmatrix} 1 & 0.5 & 1.5 & 2 & 1.5 \\ 2 & 1 & 2 & 3 & 2.5 \\ 0.6 & 0.5 & 1 & 1.5 & 1.5 \\ 0.5 & 0.33 & 0.6 & 1 & 1 \\ 0.6 & 0.4 & 0.6 & 1 & 1 \end{bmatrix} \end{array}$$

$$E_1 = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{bmatrix} 5 & 3 & 5 & 2 & 2 \\ 3 & 2 & 3 & 3 & 2 \\ 4 & 5 & 4 & 2 & 5 \end{bmatrix} \end{array} \qquad E_2 = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \end{array} \begin{array}{c} \begin{array}{ccccc} C_1 & C_2 & C_3 & C_4 & C_5 \end{array} \\ \begin{bmatrix} 2 & 5 & 3 & 3 & 5 \\ 4 & 1 & 2 & 2 & 3 \\ 2 & 5 & 1 & 1 & 2 \end{bmatrix} \end{array}$$

Figure A.4: Values assigned by two experts

The $w = \{w_1, w_2,.. \ w_j \}$ vector represents subjective weights for the attributes where $\sum_{j=1}^{m} w_j = 1$ and every $w_j \geq 0 \ \forall j$. The individual decision matrices which are shown in Figure A.4, and individual weight vectors will, serve as inputs to Algorithm 13 in order to calculate the group decision matrix P_g and the group weight vector w_g.

The next step, after computing the group weight vector w_g and group decision matrix P_g, is to rank the alternatives with the Simple Additive Weighting (SAW) and the Weighted Product (WP) method. Algorithm 14 demonstrates the implementation of the SAW and WP methods. The final result after the ranking is shown in the Table A.4. These methods are known to produce rankings which are not consistent. Both methods are able to find a preferred alternative, but the rankings may not be the same, other then the top alternative.

Algorithm 13: Algorithm to compute group weight vector and group decision matrix

```
Data: number of alternatives: numberOfAlternatives, number of criteria numberOfCriteria, individual
      weight vectors: w1, w2, wm and individual decision matrices: P1, P2, Pm where m is the number of
      experts
Result: group weight vector wg and group decision matrix Pg
1  begin
2      wg ← new double[numberOfCriteria]
3      double sum ← 0.0
4      for int i ← 0 to i < wg.length i++ do
5          for int k ← 1 to k <= numberOfAlternatives k++ do
6              sum ← sum + wk[i]
7          end
8          wg[i] ← sum / numberOfAlternatives
9          sum ← 0.0
10     end
11     reinitialize sum ← 0.0
12     Pg ← new double[numberOfAlternatives][numberOfCriteria]
13     for int i ← 0 to i < numberOfAlternatives i++ do
14         for int j ← 0 to j < numberOfCriteria j++ do
15             for int k ← 1 to k <= numberOfAlternatives k++ do
16                 sum ← sum + Pk[i][j]
17             end
18             Pg[i][j] ← sum / numberOfAlternatives
19             sum ← 0.0
20         end
21     end
22     return wg and Pg
23 end
```

A.3 A simplified LSM for inconsistent matrices

The upcoming sections demonstrate an alternative algorithm to the proposed algorithm by Bozóki et al. [11]. Under stronger conditions, with regards to inconsistency and decreased accuracy, our proposed solutions run in seconds instead of days. As such, they may be useful for researchers willing to use the least squares method (LSM) instead of geometric means (GM) method.

Finding a consistent approximation for a given inconsistent pairwise comparisons (PC) matrix by the least squares method for an Euclidean metric was presented in [11]. The inspiration for developing an alternative algorithm came from the entry which indicated *three days* in Table 2 in [11] as the Central Processing Unit (CPU) time required to compute a case of a matrix for $n = 8$. We concluded that many users are too impatient to wait three days for the results to be computed. It is important to mention here that in practice we need to change the values in the pairwise comparison

Algorithm 14: Algorithm to rank alternatives with SAW and WP methods

```
Data: group weight vector w_g, group decision matrix P_g, number of alternatives: numberOfAlternatives,
      number of criteria numberOfCriteria,
Result: Simple Additive Weighting ranking in vector ranking_saw and Weighted Product ranking in vector
        ranking_wp
1  begin
2      double sum ← 0.0
3      ranking_saw ← new double[numberOfAlternatives]
4      for int i ← 0 to i < numberOfAlternatives i++ do
5          for int j ← 0 to j < numberOfCriteria j++ do
6              sum ← sum + (w_g[j] * P_g[i][j])
7          end
8          ranking_saw[i] ← sum
9          sum ← 0.0
10     end
11     double sum ← 1.0
12     ranking_wp ← new double[numberOfAlternatives]
13     for int i ← 0 to i < numberOfAlternatives i++ do
14         for int j ← 0 to j < numberOfCriteria j++ do
15             sum ← sum * ( Math.pow(w_g[j], P_g[i][j]))
16         end
17         ranking_wp[i] ← sum
18         sum ← 1.0
19     end
20 end
```

Alternatives	SAW		WP	
	Value	Ranking	Value	Ranking
P_1	3.6199	1	3.100..E-13	1
P_2	3.2525	2	2.721..E-12	2
P_3	2.4425	3	1.897..E-9	3

Table A.4: Ranking of three alternatives

matrix's upper triangle many times where fifty or more changes are not uncommon. With each change requiring three days of computations, we would need one hundred and fifty days more to complete the adjustment. As described in [11], some problems may have multiple solutions. However, all known examples having their own distinct solutions in real life situations is highly impossible. We are almost sure, and subject to further research, that multiple solutions may appear when high inconsistency is present.

A.3.1 Practicality for the simplification

It is a realistic assumption that the pairwise comparisons method is predominantly used for processing highly subjective assessments. Subjective assessments need this method for processing. For processing measurements or objective data methods are nearly always based on mathematical formulas, equations, partial differential equations, or a system of linear equations. We decided to decrease the accuracy to two significant figures, since subjective assessments do not reach one percent accuracy. We recommend a geometric means (GM) solution as a starting point for better convergence since GM and LSM solutions are identical for fully consistent matrices and they are not drastically different for matrices which have low inconsistency indicators as described in [44]. In a situation where the low inconsistency does not guarantee one solution or a unique solution, we can always select the one which is closest to GM by the Euclidean distance, or revert to GM solution. The importance of the inconsistency analysis and control was stressed in [44] but better presented in [12]. The search for a very precise solution for a highly inconsistent pairwise comparisons matrix is not feasible, since the high inconsistency indicates the presence of contradictory assessments.

A.3.2 Remarks about algorithm

The proposed simple algorithm has removed one big shortcoming of LSM which was the substantial CPU time. For subjective assessments, high accuracy for a solution is not important. Two significant digits give an accuracy of one percent. It is more than sufficient for the input data often on a scale of 1 to 5 used by [44], 1 to 9 used by [11, 12] or 0 to 4 used by [19] and the distance-based inconsistency indicator with the acceptable level assumed to be $\frac{1}{3}$, as explained in [44]. It is difficult to

Algorithm 15: Simplified least squares method algorithm for inconsistent matrices

```
   Data: square inconsistent MatrixA and precision − value
   Result: consistent approximation for inconsistent pairwise comparisons (PC) matrix
 1 begin
 2 |  compute Geometric Means Vector (GMV)
 3 |  compute Geometric Means Normalized Vector (GMVN)
 4 |  generate search − space
 5 |  delta ←precision − value
 6 |  while search − space has more unexplored solutions do
 7 |  |  current − solution ← next unexplored solution from search − space
 8 |  |  compute Geometric Means Vector (GMV)
 9 |  |  compute Geometric Means Normalized Vector (GMVN)
10 |  |  compute MatrixB where for all i and j < GMVN.length if (i == j) MatrixB[i][j] ← 1; else if (i
   |  |  < j) MatrixB[i][j] ← GMVN[i] / GMVN[j]; if (i > j) MatrixB[i][j] ← 1 / MatrixB[j][i];
11 |  |  compute MatrixD ←(MatrixA − MatrixB)^2
12 |  |  while current − solution is not fully explored do
13 |  |  |  entity ←get randomly from current − solution
14 |  |  |  add delta to entity, recompute MatrixB and MatrixD ←(MatrixA − MatrixB)^2
15 |  |  |  while SumSQ is decreasing do
16 |  |  |  |  add delta to entity; recompute MatrixB and MatrixD ←(MatrixA − MatrixB)^2
17 |  |  |  end
18 |  |  |  if SumSQ did not decrease then
19 |  |  |  |  revert made changes
20 |  |  |  end
21 |  |  |  try subtraction, recompute MatrixB and MatrixD ←(MatrixA − MatrixB)^2
22 |  |  |  while SumSQ is decreasing do
23 |  |  |  |  subtract delta from entity; recompute MatrixB and MatrixD ←
   |  |  |  |  (MatrixA − MatrixB)^2
24 |  |  |  end
25 |  |  |  if SumSQ did not decrease then
26 |  |  |  |  revert made changes
27 |  |  |  end
28 |  |  |  perform recursive call with delta ←delta/2, terminate if SumSQ did not decrease
29 |  |  |  if no addition and subtraction occurred then
30 |  |  |  |  collect final computed value for entity
31 |  |  |  end
32 |  |  |  if SumSQ did not decrease then
33 |  |  |  |  remove entity from the consideration
34 |  |  |  end
35 |  |  end
36 |  end
37 end
```

claim that our method may work for every pairwise comparison matrix, but it is fast, within seconds instead of hours, for not inconsistent pairwise comparison matrices. Inconsistent matrices have a tendency to appear in most real life problems. In order to find the optimal solution the algorithm requires 3090 changes for $n = 4$ and 22,938 changes for $n = 7$.

Appendix B

The automation scripts and their output

```
1 <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
2 <project name="run␣all␣scripts" default="run-all-scripts" basedir=".">
3     <property environment="env" />
4     <property file="${env.DEV_HOME}/modules_build/common_build.properties" />
5     <!-- will execute all scripts within the system -->
6     <target name="run-all-scripts" description="run␣certification␣scripts">
7         <ant antfile="${system_cert.file}" inheritAll="false" inheritRefs="false"
8             target="validate-system" output="summary.txt" />
9         <ant antfile="${system_cert.file}" inheritAll="false" inheritRefs="false" target="mail" />
10    </target>
11 </project>
```

Listing B.1: A main automation script

```
1 <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
2 <project name="core_build" basedir=".">
3     <!-- load build variables -->
4     <property environment="env" />
5     <property file="${basedir}/common_build.properties" />
6     <!-- clean target -->
7     <target name="clean" description="␣clean␣target">
8         <delete dir="${bin.dir}" />
9         <delete dir="${lib.dir}" />
10        <delete dir="${@dot.dir}" />
11        <delete dir="${test.dir}" />
12        <delete dir="${build}" />
13        <delete dir="${junit.results.dir}" />
14        <delete dir="${reports.dir}" />
15        <delete>
16            <fileset dir="${basedir}" includes="*.log" />
17        </delete>
18    </target>
19    <!-- compile target -->
20    <target name="compile" description="␣compile␣target">
21        <mkdir dir="${bin.dir}" />
22        <javac srcdir="${src.dir}" destdir="${bin.dir}" nowarn="${nowarn}" debug="${debug}"
23            optimize="${optimize}" deprecation="${deprecation}" target="${target}"
24            verbose="${verbose}" depend="${depend}" includeAntRuntime="${includeAntRuntime}"
```

```
25          includeJavaRuntime="${includeJavaRuntime}" failonerror="${failonerror}"
26          source="${source}">
27          <classpath refid="module.classpath" />
28      </javac>
29      <copy todir="${bin.dir}" includeEmptyDirs="no">
30          <fileset dir="${src.dir}">
31              <include name="**/*.properties" />
32              <include name="**/*.ddl" />
33              <include name="**/*.txt" />
34              <exclude name="**/.svn" />
35          </fileset>
36      </copy>
37  </target>
38  <!-- package target -->
39  <target name="package" description="␣package␣target">
40      <mkdir dir="${lib.dir}" />
41      <jar jarfile="${lib.dir}/${archive.name}">
42          <fileset dir="${bin.dir}">
43              <include name="**/*.class" />
44              <include name="**/*.properties" />
45              <include name="**/*.ddl" />
46              <include name="**/*.txt" />
47              <exclude name="**/test/" />
48          </fileset>
49          <manifest>
50              <attribute name="Vendor" value="DEV" />
51          </manifest>
52      </jar>
53  </target>
54  <!-- deploy module jar -->
55  <target name="deploy" description="deploy␣module␣jar">
56      <echo level="warning" message="deploy␣module␣jar" />
57      <copy file="${lib.dir}/${archive.name}" todir="${plugin.lib.dir}" />
58  </target>
59  <!-- emma run -->
60  <path id="emma.lib">
61      <fileset dir="${emma.dir}">
62          <include name="*.jar" />
63      </fileset>
64  </path>
65  <path id="emma.coverage.classes">
66      <pathelement location="${bin.dir}" />
67  </path>
68  <taskdef resource="emma_ant.properties" classpathref="emma.lib" />
69  <target name="emma" description="turns␣on␣EMMA's␣instrumentation␣for␣reporting">
70      <property name="emma.enabled" value="true" />
71      <mkdir dir="${instr.dir}" />
72      <property name="emma.filter" value="" />
73  </target>
74  <!-- testing -->
75  <target name="test" depends="emma" description="test␣primo␣core_database">
76      <echo level="warning" message="testing␣module" />
77      <emma enabled="${emma.enabled}">
78          <instr instrpathref="emma.coverage.classes" destdir="${instr.dir}"
79              metadatafile="${coverage.dir}/metadata.emma" merge="true">
80              <filter value="${emma.filter}" />
81          </instr>
82      </emma>
83      <junit printsummary="yes" forkmode="perBatch" haltonfailure="no" haltonerror="no"
84          includeantruntime="true" fork="true" dir="${basedir}" failureProperty="test.failure">
85          <classpath>
86              <pathelement location="${instr.dir}" />
87              <path refid="emma.lib" />
88              <path refid="module.classpath" />
89          </classpath>
90          <formatter type="brief" usefile="false" />
```

```
91          <batchtest fork="yes">
92              <fileset dir="${instr.dir}">
93                  <include name="**/*UT.class" />
94                  <include name="**/*FT.class" />
95              </fileset>
96          </batchtest>
97          <jvmarg value="-Demma.coverage.out.file=${coverage.dir}/coverage.emma" />
98          <jvmarg value="-Demma.coverage.out.merge=false" />
99      </junit>
100     <fail message="test failed" if="test.failure" />
101     <emma enabled="${emma.enabled}">
102         <report sourcepath="src" sort="+block,+name,+method,+class"
103             metrics="method:70,block:80,line:80,class:100">
104             <fileset dir="${coverage.dir}">
105                 <include name="*.emma" />
106             </fileset>
107             <html outfile="${coverage.dir}/coverage.html" depth="method"
108                 columns="name,class,method,block,line" />
109         </report>
110     </emma>
111 </target>
112 <!-- test for caverage report -->
113 <target name="test-for-cav-report">
114     <mkdir dir="${build.java}" />
115     <mkdir dir="${build.test}" />
116     <mkdir dir="${junit.results.dir}" />
117     <mkdir dir="${reports.dir}/jfeature" />
118     <javac srcdir="${src.test}" destdir="${build.test}">
119         <classpath refid="module.classpath" />
120         <classpath location="${build.java}" />
121     </javac>
122     <junit printsummary="true">
123         <classpath location="${build.test}" />
124         <classpath location="${build.java}" />
125         <classpath location="webapp" />
126         <classpath>
127             <fileset dir="${thirdparty.dir}/net.technobuff.jfeature/lib">
128                 <include name="**/*.jar" />
129             </fileset>
130         </classpath>
131         <formatter type="xml" />
132         <batchtest fork="yes" haltonerror="false" haltonfailure="false"
133             todir="${junit.results.dir}">
134             <fileset dir="${src.test}">
135                 <include name="**/*UT.java"/>
136             </fileset>
137         </batchtest>
138     </junit>
139 </target>
140 <target name="generate-caverage-report" description="generate caverage report">
141     <taskdef name="jfeaturecoveragereport"
142         classname="net.technobuff.jfeature.ant.task.JFeatureCoverageReportTask">
143         <classpath>
144             <pathelement path="${thirdparty.dir}/net.technobuff.jfeature" />
145             <fileset dir="${thirdparty.dir}/net.technobuff.jfeature">
146                 <include name="**/*.jar" />
147             </fileset>
148         </classpath>
149     </taskdef>
150     <jfeaturecoveragereport testresultsdir="${junit.results.dir}"
151         todir="${reports.dir}/jfeature" format="html">
152         <requirementsfileset dir="${basedir}/requirements">
153             <include name="**/*.jrq" />
154         </requirementsfileset>
155     </jfeaturecoveragereport>
156 </target>
```

```
157 </project>
```

Listing B.2: A common automation script which is used by all components

```
1  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
2  <project name="lsm" default="all" basedir=".">
3      <!-- define build properties -->
4      <property environment="env" />
5      <property file="${env.DEV_HOME}/modules_build/common_build.properties" />
6      <property name="src.dir" value="${basedir}/src" />
7      <property name="bin.dir" value="${basedir}/bin" />
8      <property name="@dot.dir" value="${basedir}/@dot" />
9      <property name="lib.dir" value="${basedir}/lib" />
10     <property name="plugin.lib.dir" value="${basedir}/deployment" />
11     <property name="archive.name" value="lsm.jar" />
12     <property name="src.java" value="${basedir}/src" />
13     <property name="src.test" value="${basedir}/src" />
14     <property name="build" value="build" />
15     <property name="build.java" value="${build}/java" />
16     <property name="build.test" value="${build}/test" />
17     <property name="junit.results.dir" value="${basedir}/junit-results" />
18     <property name="reports.dir" value="${basedir}/caverage-report" />
19     <!-- emma -->
20     <property name="coverage.dir" value="${basedir}/temp/emma/reports/emma" />
21     <property name="instr.dir" value="${basedir}/temp/emma/target/emmainstr" />
22     <property name="emma.dir" value="${thirdparty.dir}/emma" />
23     <!-- clean lsm -->
24     <target name="clean" description="clean␣lsm">
25         <echo level="warning" message="cleaning␣lsm" />
26         <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true" target="clean" />
27     </target>
28     <!-- classpath dependencies for lsm -->
29     <path id="production.classpath">
30         <pathelement location="${thirdparty.dir}/log4j/log4j.jar" />
31         <pathelement location="${thirdparty.dir}/ant/ant-1.6.5.jar" />
32         <pathelement location="${basedir}/../ContextInterface/lib/context_interface.jar" />
33         <pathelement location=
34             "${thirdparty.dir}/equinox/org.apache.commons.logging_1.0.4.v200706111724.jar" />
35         <pathelement location=
36             "${thirdparty.dir}/equinox/org.eclipse.equinox.ds_1.0.0.v20070226.jar" />
37         <pathelement location=
38             "${thirdparty.dir}/equinox/org.eclipse.equinox.event_1.0.100.v20070516.jar" />
39         <pathelement location=
40             "${thirdparty.dir}/equinox/org.eclipse.equinox.log_1.0.100.v20070226.jar" />
41         <pathelement location=
42             "${thirdparty.dir}/equinox/org.eclipse.osgi.services_3.1.200.v20070605.jar" />
43         <pathelement location="${thirdparty.dir}/equinox/org.eclipse.osgi_3.3.0.v20070530.jar" />
44     </path>
45     <path id="testing.classpath">
46         <pathelement location="${basedir}/bin" />
47         <pathelement location="${thirdparty.dir}/net.technobuff.jfeature/lib/junit.jar" />
48     </path>
49     <path id="module.classpath">
50         <path refid="production.classpath" />
51         <path refid="testing.classpath" />
52     </path>
53     <!-- compile lsm -->
54     <target name="compile" description="compile␣lsm">
55         <echo level="warning" message="compiling␣lsm" />
56         <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true" target="compile" />
57     </target>
58     <!-- package lsm -->
59     <target name="package" description="package␣lsm">
```

```
60          <echo level="warning" message="packaging␣lsm" />
61          <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true" target="package" />
62      </target>
63      <!-- deploying jars -->
64      <target name="deploy" description="deploy␣lsm's␣jars">
65          <echo level="warning" message="deploying␣lsm's␣jars" />
66          <delete file="${plugin.lib.dir}/${archive.name}" />
67          <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true" target="deploy" />
68      </target>
69      <!-- test lsm -->
70      <target name="test" description="test␣lsm">
71          <echo level="warning" message="testing␣lsm" />
72          <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true" target="test" />
73      </target>
74      <!-- test caverage report for lsm -->
75      <target name="test-for-cav-report" description="test␣for␣caverage␣report␣for␣lsm">
76          <echo level="warning" message="testing␣for␣caverage␣report" />
77          <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true"
78              target="test-for-cav-report" />
79      </target>
80      <!-- generate caverage report for lsm-->
81      <target name="generate-caverage-report" description="generate␣caverage␣report␣for␣lsm">
82          <echo level="warning" message="generating␣caverage␣report␣for␣lsm" />
83          <ant antfile="${common.build.file}" inheritAll="true" inheritRefs="true"
84              target="generate-caverage-report" />
85      </target>
86      <!-- clean, compile, package, test, test for caverage report and generate caverage report for lsm -->
87      <target name="all"
88      depends="clean,compile,package,test,test-for-cav-report,generate-caverage-report"
89      description="clean,␣compile,␣package,␣test,␣test␣for␣cave␣report,␣generate␣caverage␣report">
90      <echo level="warning"
91      message="clean,␣compile,␣package,␣test,test␣for␣cav␣report,␣generating␣caverage␣report" />
92      </target>
93  </project>
```

Listing B.3: An upper level automation script for the component

```
1  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
2  <project name="Validate␣system" default="validate-system" basedir=".">
3      <property environment="env" />
4      <property file="${env.DEV_HOME}/modules_build/common_build.properties" />
5      <!-- run cert system -->
6      <target name="validate-system" description="cert␣system">
7          <echo level="warning" message="------------" />
8          <echo level="warning" message="CERT␣SYSTEM" />
9          <echo level="warning" message="------------" />
10         <ant dir="${basedir}/../ContextReasoner" antfile="context_reasoner_build.xml"
11             inheritAll="false" target="all" />
12         <ant dir="${basedir}/../ContextInterface" antfile="context_interface_build.xml"
13             inheritAll="false" target="all" />
14         <ant dir="${basedir}/../LSM_for_inconsistent_matrices" antfile="lsm_build.xml"
15             inheritAll="false" target="all" />
16         <ant dir="${basedir}/../ContextReasoner" antfile="certification_script.xml"
17             inheritAll="false" target="all" />
18         <echo message="VALIDATION:␣PASS" />
19     </target>
20     <target name="mail">
21         <loadfile property="message.log" srcFile="summary.txt" />
22         <condition property="condition">
23             <contains string="${message.log}" substring="VALIDATION:␣PASS" casesensitive="true" />
24         </condition>
25         <antcall target="passed" />
26         <antcall target="failed" />
```

```
27      </target>
28      <target name="passed" if="condition">
29          <loadfile property="change.log" srcFile="summary.txt" />
30          <mail mailhost="univmail.cis.mcmaster.ca" mailport="25" subject="VALIDATION:␣PASS">
31              <from address="babiyv@univmail.cis.mcmaster.ca" />
32              <replyto address="babiyv@univmail.cis.mcmaster.ca" />
33              <to address="babiyv@univmail.cis.mcmaster.ca" />
34              <message>The build was successful.
35                      ${change.log}
36              </message>
37          </mail>
38      </target>
39      <target name="failed" unless="condition">
40          <loadfile property="message.log" srcFile="summary.txt" />
41          <mail mailhost="univmail.cis.mcmaster.ca" mailport="25" subject="VALIDATION:␣FAILED">
42              <from address="babiyv@univmail.cis.mcmaster.ca" />
43              <replyto address="babiyv@univmail.cis.mcmaster.ca" />
44              <to address="babiyv@univmail.cis.mcmaster.ca" />
45              <message>${message.log}</message>
46          </mail>
47      </target>
48  </project>
```

Listing B.4: An upper level automation script for the framework management

```
1  <?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
2  <project name="certification␣script" default="all" basedir=".">
3      <!-- define build properties -->
4      <property environment="env" />
5      <property file="${env.DEV_HOME}/modules_build/common_build.properties" />
6      <property name="archive.name" value="context_reasoner.jar" />
7      <!-- classpath dependencies for certification script -->
8      <path id="production.classpath">
9          <pathelement location="${thirdparty.dir}/log4j/log4j.jar" />
10         <pathelement location="${thirdparty.dir}/ant/ant-1.6.5.jar" />
11         <pathelement location="${thirdparty.dir}/Jena-2.6.2/jena-2.6.2-sources.jar" />
12         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/arq-2.8.1.jar" />
13         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/icu4j-3.4.4.jar" />
14         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/iri-0.7.jar" />
15         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/jena-2.6.2.jar" />
16         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/jena-2.6.2-tests.jar" />
17         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/junit-4.5.jar" />
18         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/log4j-1.2.13.jar" />
19         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/lucene-core-2.3.1.jar" />
20         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/slf4j-api-1.5.6.jar" />
21         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/slf4j-log4j12-1.5.6.jar" />
22         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/stax-api-1.0.1.jar" />
23         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/wstx-asl-3.2.9.jar" />
24         <pathelement location="${thirdparty.dir}/Jena-2.6.2/lib/xercesImpl-2.7.1.jar" />
25     </path>
26     <path id="testing.classpath">
27         <pathelement location="${basedir}/bin" />
28         <pathelement location="${env.DEV_HOME}/ContextReasoner/lib/${archive.name}" />
29     </path>
30     <path id="module.classpath">
31         <path refid="production.classpath" />
32         <path refid="testing.classpath" />
33     </path>
34     <!-- target will compute certification level of the product -->
35     <target name="product-certification-level"
36         description="compute␣certification␣level␣of␣the␣product">
37         <taskdef name="productCertificationLevel"
38             classname="ContextReasoner.Internal.Certification.ReportCertLevel" >
```

```
39              <classpath refid="module.classpath"/>
40          </taskdef>
41          <productCertificationLevel />
42      </target>
43      <!-- target will compute completeness levels of the components -->
44      <target name="components-completeness-levels"
45          description="compute␣completeness␣level␣of␣the␣components">
46          <taskdef name="components-completeness-levels"
47              classname="ContextReasoner.Internal.Certification.ReportCompletenessLevel" >
48              <classpath refid="module.classpath"/>
49          </taskdef>
50          <components-completeness-levels />
51      </target>
52      <!-- target will compute conformance levels of the components -->
53      <target name="components-conformance-levels"
54          description="compute␣conformance␣levels␣of␣the␣components">
55          <taskdef name="components-conformance-levels"
56              classname="ContextReasoner.Internal.Certification.ReportConformanceLevel" >
57              <classpath refid="module.classpath"/>
58          </taskdef>
59          <components-conformance-levels />
60      </target>
61      <!-- target will compute uniformity levels of the components -->
62      <target name="components-uniformity-levels"
63          description="compute␣uniformity␣levels␣of␣the␣components">
64          <taskdef name="components-uniformity-levels"
65              classname="ContextReasoner.Internal.Certification.ReportUniformityLevel" >
66              <classpath refid="module.classpath"/>
67          </taskdef>
68          <components-uniformity-levels />
69      </target>
70      <target name="all" description="will␣execute␣all␣the␣targets␣">
71          <echo level="warning" message="will␣execute␣all␣the␣targets␣" />
72          <antcall target="components-completeness-levels"/>
73          <antcall target="components-conformance-levels"/>
74          <antcall target="components-uniformity-levels"/>
75          <antcall target="product-certification-level"/>
76      </target>
77 </project>
```

Listing B.5: A product certification script

The listing below shows the complete output as it is being reported by the automation system. This is the verbose report which is being automatically emailed to the appropriate individual.

```
1 validate-system:
2       [echo] ------------
3       [echo] CERT SYSTEM
4       [echo] ------------
5 clean:
6       [echo] cleaning context reasoner
7 clean:
8     [delete] Deleting directory E:\coding\workspace\ContextReasoner\bin
9     [delete] Deleting directory E:\coding\workspace\ContextReasoner\lib
10    [delete] Deleting directory E:\coding\workspace\ContextReasoner\temp
11    [delete] Deleting directory E:\coding\workspace\ContextReasoner\build
12    [delete] Deleting directory E:\coding\workspace\ContextReasoner\junit-results
13    [delete] Deleting directory E:\coding\workspace\ContextReasoner\caverage-report
14 compile:
```

```
15        [echo] compiling context reasoner
16 compile:
17      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\bin
18      [javac] Compiling 29 source files to E:\coding\workspace\ContextReasoner\bin
19 package:
20        [echo] packaging context reasoner
21 package:
22      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\lib
23        [jar] Building jar: E:\coding\workspace\ContextReasoner\lib\context_reasoner.jar
24 test:
25        [echo] testing context reasoner
26 emma:
27      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\temp\emma\target\emmainstr
28 test:
29        [echo] testing module
30      [instr] processing instrumentation path ...
31      [instr] instrumentation path processed in 688 ms
32      [instr] [58 class(es) instrumented, 0 resource(s) copied]
33      [instr] metadata merged into [E:\coding\workspace\ContextReasoner
34          \temp\emma\reports\emma\metadata.emma] {in 328 ms}
35      [junit] Running ContextReasoner.Internal.Test.ProductUT
36      [junit] Testsuite: ContextReasoner.Internal.Test.ProductUT
37      [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.015 sec
38      [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.015 sec
39      [junit]
40      [junit] ------------- Standard Output ---------------
41      [junit] EMMA: collecting runtime coverage data ...
42      [junit] ------------- ---------------- ---------------
43     [report] processing input files ...
44     [report] 2 file(s) read and merged in 16 ms
45     [report] writing [html] report to [E:\coding\workspace\ContextReasoner
46      \temp\emma\reports\emma\coverage.html] ...
47 test-for-cav-report:
48        [echo] testing for caverage report
49 test-for-cav-report:
50      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\build\java
51      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\build\test
52      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\junit-results
53      [mkdir] Created dir: E:\coding\workspace\ContextReasoner\caverage-report\jfeature
54      [javac] Compiling 29 source files to E:\coding\workspace\ContextReasoner\build\test
55      [junit] Running ContextReasoner.Internal.Test.ProductUT
56      [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.531 sec
57 generate-caverage-report:
58        [echo] generating caverage report for context reasoner
59 generate-caverage-report:
60 [jfeaturecoveragereport] Generating requirement coverage report under "E:\coding
␣␣␣␣␣␣␣␣\workspace\ContextReasoner/caverage-report/jfeature"... Done
62 all:
63        [echo] cleaning, compiling, packaging, testing, testing for caverage report
64              and generating caverage report for context reasoner
65 clean:
66        [echo] cleaning context interface
67 clean:
68     [delete] Deleting directory E:\coding\workspace\ContextInterface\bin
69     [delete] Deleting directory E:\coding\workspace\ContextInterface\lib
70     [delete] Deleting directory E:\coding\workspace\ContextInterface\temp
71     [delete] Deleting directory E:\coding\workspace\ContextInterface\build
72     [delete] Deleting directory E:\coding\workspace\ContextInterface\junit-results
73     [delete] Deleting directory E:\coding\workspace\ContextInterface\caverage-report
74 compile:
75        [echo] compiling context interface
76 compile:
77      [mkdir] Created dir: E:\coding\workspace\ContextInterface\bin
78      [javac] Compiling 3 source files to E:\coding\workspace\ContextInterface\bin
79 package:
80        [echo] packaging context interface
```

```
81 package:
82     [mkdir] Created dir: E:\coding\workspace\ContextInterface\lib
83       [jar] Building jar: E:\coding\workspace\ContextInterface\lib\context_interface.jar
84 test:
85      [echo] testing context interface
86 emma:
87     [mkdir] Created dir: E:\coding\workspace\ContextInterface\temp\emma\target\emmainstr
88 test:
89      [echo] testing module
90     [instr] processing instrumentation path ...
91     [instr] instrumentation path processed in 204 ms
92     [instr] [2 class(es) instrumented, 0 resource(s) copied]
93     [instr] metadata merged into [E:\coding\workspace\ContextInterface\temp
94         \emma\reports\emma\metadata.emma] {in 63 ms}
95     [junit] Running ContextInterface.Internal.Test.ContextInterfaceUT
96     [junit] Testsuite: ContextInterface.Internal.Test.ContextInterfaceUT
97     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
98     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
99     [junit]
100    [junit] ------------- Standard Output ---------------
101    [junit] EMMA: collecting runtime coverage data ...
102    [junit] ------------- ---------------- -------------
103   [report] processing input files ...
104   [report] 2 file(s) read and merged in 0 ms
105   [report] writing [html] report to [E:\coding\workspace\ContextInterface
106     \temp\emma\reports\emma\coverage.html] ...
107 test-for-cav-report:
108      [echo] testing for caverage report
109 test-for-cav-report:
110     [mkdir] Created dir: E:\coding\workspace\ContextInterface\build\java
111     [mkdir] Created dir: E:\coding\workspace\ContextInterface\build\test
112     [mkdir] Created dir: E:\coding\workspace\ContextInterface\junit-results
113     [mkdir] Created dir: E:\coding\workspace\ContextInterface\caverage-report\jfeature
114     [javac] Compiling 3 source files to E:\coding\workspace\ContextInterface\build\test
115     [junit] Running ContextInterface.Internal.Test.ContextInterfaceUT
116     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.078 sec
117 generate-caverage-report:
118      [echo] generating caverage report for context interface
119 generate-caverage-report:
120 [jfeaturecoveragereport] Generating requirement coverage report under
121 "E:\coding\workspace\ContextInterface/caverage-report/jfeature"... Done
122 all:
123      [echo] cleaning, compiling, packaging, testing, testing for caverage
124      report and generating caverage report for context interface
125 clean:
126      [echo] cleaning lsm
127 clean:
128   [delete] Deleting directory E:\coding\workspace\LSM_for_inconsistent_matrices\bin
129   [delete] Deleting directory E:\coding\workspace\LSM_for_inconsistent_matrices\lib
130   [delete] Deleting directory E:\coding\workspace\LSM_for_inconsistent_matrices\temp
131   [delete] Deleting directory E:\coding\workspace\LSM_for_inconsistent_matrices\build
132   [delete] Deleting directory E:\coding\workspace\LSM_for_inconsistent_matrices\junit-results
133   [delete] Deleting directory E:\coding\workspace\LSM_for_inconsistent_matrices\caverage-report
134 compile:
135      [echo] compiling lsm
136 compile:
137     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\bin
138     [javac] Compiling 15 source files to E:\coding\workspace\LSM_for_inconsistent_matrices\bin
139 package:
140      [echo] packaging lsm
141 package:
142     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\lib
143       [jar] Building jar: E:\coding\workspace\LSM_for_inconsistent_matrices\lib\lsm.jar
144 test:
145      [echo] testing lsm
146 emma:
```

```
147     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\temp\emma\target\emmainstr
148 test:
149      [echo] testing module
150     [instr] processing instrumentation path ...
151     [instr] instrumentation path processed in 484 ms
152     [instr] [14 class(es) instrumented, 0 resource(s) copied]
153     [instr] metadata merged into [E:\coding\workspace\LSM_for_inconsistent_matrices
154     \temp\emma\reports\emma\metadata.emma] {in 141 ms}
155     [junit] Running PairwiseComparison.Internal.Test.InconsistancyCalculatorUT
156     [junit] Testsuite: PairwiseComparison.Internal.Test.InconsistancyCalculatorUT
157     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
158     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
159     [junit]
160     [junit] ------------- Standard Output ---------------
161     [junit] EMMA: collecting runtime coverage data ...
162     [junit] ------------- ---------------- ---------------
163     [junit] Running PairwiseComparison.Internal.Test.MainUT
164     [junit] Testsuite: PairwiseComparison.Internal.Test.MainUT
165     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0 sec
166     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0 sec
167     [junit]
168     [junit] Running PairwiseComparison.Internal.Test.PairwiseComparisonUT
169     [junit] Testsuite: PairwiseComparison.Internal.Test.PairwiseComparisonUT
170     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0 sec
171     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0 sec
172     [junit]
173     [junit] Running PairwiseComparison.Internal.Test.StatusUT
174     [junit] Testsuite: PairwiseComparison.Internal.Test.StatusUT
175     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
176     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.016 sec
177     [junit]
178     [junit] Running PairwiseComparison.Internal.Test.TriadUT
179     [junit] Testsuite: PairwiseComparison.Internal.Test.TriadUT
180     [junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.015 sec
181     [junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.015 sec
182     [junit]
183     [junit] Running PairwiseComparison.Internal.Test.VectorUT
184     [junit] Testsuite: PairwiseComparison.Internal.Test.VectorUT
185     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0 sec
186     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0 sec
187     [junit]
188    [report] processing input files ...
189    [report] 2 file(s) read and merged in 0 ms
190    [report] writing [html] report to [E:\coding\workspace\LSM_for_inconsistent_matrices
191     \temp\emma\reports\emma\coverage.html] ...
192 test-for-cav-report:
193      [echo] testing for caverage report
194 test-for-cav-report:
195     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\build\java
196     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\build\test
197     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\junit-results
198     [mkdir] Created dir: E:\coding\workspace\LSM_for_inconsistent_matrices\caverage-report\jfeature
199     [javac] Compiling 15 source files to E:\coding\workspace\LSM_for_inconsistent_matrices\build\test
200     [junit] Running PairwiseComparison.Internal.Test.InconsistancyCalculatorUT
201     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.469 sec
202     [junit] Running PairwiseComparison.Internal.Test.MainUT
203     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.438 sec
204     [junit] Running PairwiseComparison.Internal.Test.PairwiseComparisonUT
205     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.219 sec
206     [junit] Running PairwiseComparison.Internal.Test.StatusUT
207     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.078 sec
208     [junit] Running PairwiseComparison.Internal.Test.TriadUT
209     [junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed: 0.078 sec
210     [junit] Running PairwiseComparison.Internal.Test.VectorUT
211     [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 0.094 sec
212 generate-caverage-report:
```

```
213         [echo] generating caverage report for lsm
214 generate-caverage-report:
215 [jfeaturecoveragereport] Generating requirement coverage report under
216 "E:\coding\workspace\LSM_for_inconsistent_matrices/caverage-report/jfeature"... Done
217 all:
218         [echo] clean, compile, package, test,test for cav report, generating caverage report
219 all:
220         [echo] will execute all the targets
221 components-completeness-levels:
222 [components-completeness-levels] Component completeness level: 4
223 components-conformance-levels:
224 [components-conformance-levels] Component conformance level: 4
225 components-uniformity-levels:
226 [components-uniformity-levels] Component uniformity level: 4
227 product-certification-level:
228 [productCertificationLevel] cert level: 4
229         [echo] VALIDATION: PASS
```

Listing B.6: Verbose output produced by automation scripts

Appendix C

Instructions to setup the framework

Below are the instructions on how to set up and use the context aware framework within OSGi environment. The version of the OSGi environment which we were using was tested on Eclipse 3.x. The software which are mentioned in the listing below is required. All other third party software will be downloaded with a code base from the repository which is located in the SourceForge domain.

- Eclipse 3.3.2 can be download from the http://archive.eclipse.org/eclipse/ downloads/ drops/ R-3.3.2-200802211800/ index.php website. This software will maintain everything.

- Protégé software can be download from the http://protege.stanford.edu/ website. This software can be used to create and edit OWL ontologies.

- OntoStudio software can be download from the http://www.ontoprise.de/ index.php?id=296 websit. This software is used to visualize OWL ontologies. The Protégé tool is better at creating and additing OWL ontologies, while the OntoStudio tool is better at visualization.

- SVN client software can be downloaded for the http://tortoisesvn.net/ downloads website. This software will be used to checkout the framework form the

SourceForge repository.

- Java 5 can be downloaded from the http://java.sun.com/ javase/downloads/index_jdk5.jsp website. Some components of framework were developed with Java 5. Therefore, it is a mandatory requirement.

- Create some new folder on your desktop or any other location. Right click on the newly created folder and from the drop down menu select "SVN Checkout ..." option.

- In the 'URL of repository:' box enter this location of the SourceForge repository. The location of the repository is https://contextaware.svn.sourceforge.net/ svnroot/contextaware. This is an open source software, therefore anybody can download it without the need of having any special access privileges.

- Under the "Revision" option select "HEAD revision" and click "Ok". These steps will allow the user to get the most recent version of the framework.

- **Note:** It is possible that the checkout may fail. If if does occur, then repeat the steps above or instead of checking out the framework from the begging simply right click on the folder which was created in a previous step and select the "SVN Update" option.

C.1 Configuration of the framework

- **Configure system variable:** In the "Control Panel" double click "System" select "Advanced" and click on "Environment Variables". Under the "System variables" click "New". Create the new variable by specifying the name DEV_HOME and location of the folder which was created in the steps above. For example, the "Variable value" may look like "C:/context aware framework".

- **Importing framework into Eclipse:** Start Eclipse software select "File" then "Import" then expand the "General" tree menu and select "Existing Projects into Workspace". Click "Next" and click "Browse" then navigate to the folder where the framework was checked out. Expand the tree by navigating to the "workspace" folder then select it, then click "OK" and "Finish". The following list of projects should be imported into the Eclipse's workspace: "ContextInterface", "ContextReasoner", "LSM_for_inconsistent_matrices", "client", "service", "remote" and "modules_build".

- **Configure system variable in Eclipse:** The framework should have the class path errors after the step above was completed. In order to remove the class path errors follow these steps. Select any project within the Eclipse environment and right click on it and then select properties. In the open window on the left hand side click "Java Build Path" then on the right hand side click "Libraries". Click "Add Variable" followed by clicking "Configure Variables". In the new window select "New", in the "Name:" entry box type DEV_HOME and in the "Path:" entry box type the location of the folder where the framework was checked out and click "Ok". Now the user should see the DEV_HOME variable under the "New Variable Classpath Entry". There is no need to do anything else, therefore click "Cancel" and then click "OK". Clean the projects within the Eclipse environment by selecting option "Project" then "Clean" and make sure to select "Clean all projects" and click "Ok". After this process the framework should not have any class path errors and is ready to be used by the user.

- **Invoking the framework:** Select "Run" and then click "Open Run Dialog...". Double click the "OSGi Framework" option. This step will create a new run environment for the OSGi framework. Within the bundles window modify the start options of the remote service and client bundles by setting the "Start

Level" of the remote bundle to 4, the service bundle to 5 and remote bundle to 6. Under the target platform almost all default bundles could be selected. It is beneficial to click the "Add Required Bundles" button which will add the missing bundles automatically. After clicking the "Run" button the following output should appear in the OSGi console.

```
1  osgi> Mar 22, 2010 10:55:19 PM org.mortbay.http.HttpServer doStart
2  INFO: Version Jetty/5.1.x
3  Mar 22, 2010 10:55:19 PM org.mortbay.util.Container start
4  INFO: Started org.mortbay.jetty.servlet.ServletHandler@14aa2db
5  Mar 22, 2010 10:55:19 PM org.mortbay.util.Container start
6  INFO: Started HttpContext[/,/]
7  Mar 22, 2010 10:55:19 PM org.mortbay.util.ThreadedServer start
8  WARNING: Failed to start: SocketListener0@0.0.0.0:80
9  Activated Context Reasoner layer
10 R-OSGi listens on port 9278
11 Activated Context Interface layer
12 Activated LSM in Context Providers layer
13 L-S-M Echo test context reasoners
14 SENDING EVENT org.osgi.service.event.Event [topic=test/topic]
15 Activated Remote Server
16 Invoking remote service:
17     ......
18 Activated Remote Client
```

Listing C.1: Partial output after framework activation

Appendix D

An upper ontology in OWL

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY SCS "http://2009/9/SCS.owl#" >
    <!ENTITY CL_ "http://2009/9/SCS.owl#CL_&#8722;" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://2009/9/SCS.owl#"
     xml:base="http://2009/9/SCS.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:SCS="http://2009/9/SCS.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:CL_="&SCS;CL_&#8722;">
    <owl:Ontology rdf:about=""/>
    <!--
    ///////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////
     -->
    <!-- http://2009/9/SCS.owl#articulatedWith -->
    <owl:ObjectProperty rdf:about="#articulatedWith">
        <rdfs:comment
            >Every measurement is associated with a unit of measure.
        Unit of measure is used to communicate the objective of
        measurement.</rdfs:comment>
        <rdfs:domain rdf:resource="#Attributes"/>
        <rdfs:range rdf:resource="#Metrics"/>
    </owl:ObjectProperty>
    <!-- http://2009/9/SCS.owl#belongsTo -->
    <owl:ObjectProperty rdf:about="#belongsTo">
        <rdfs:comment
            >Every scale or collection of scalers is associated with a
        type. A single type may be associated with a collection
        of classes.</rdfs:comment>
    </owl:ObjectProperty>
    <!-- http://2009/9/SCS.owl#composedOf -->
    <owl:ObjectProperty rdf:about="#composedOf">
```

```
46          <rdf:type rdf:resource="&owl;FunctionalProperty"/>
47          <rdf:type rdf:resource="&owl;TransitiveProperty"/>
48          <rdfs:comment
49              >Entity within the ontology can be composed out
50          of other entities.</rdfs:comment>
51          <rdfs:domain rdf:resource="#Attributes"/>
52          <rdfs:range rdf:resource="#Attributes"/>
53      </owl:ObjectProperty>
54      <!-- http://2009/9/SCS.owl#conversion -->
55      <owl:ObjectProperty rdf:about="#conversion">
56          <rdfs:comment
57              >It is possible for two or more measurement functions
58          to be associated with each other.</rdfs:comment>
59      </owl:ObjectProperty>
60      <!-- http://2009/9/SCS.owl#hasDelivery -->
61    <owl:ObjectProperty rdf:about="#hasDelivery">
62          <rdfs:comment
63              >Entities can maintain properties which contain
64          values</rdfs:comment>
65          <rdfs:domain rdf:resource="#Component_sections"/>
66          <rdfs:range rdf:resource="#Context_description"/>
67          <rdfs:range rdf:resource="#Correctness_proofs"/>
68          <rdfs:range rdf:resource="#Detail_design"/>
69          <rdfs:range rdf:resource="#High_level_design"/>
70          <rdfs:range rdf:resource="#Implementation"/>
71          <rdfs:range rdf:resource="#Testing"/>
72          <rdfs:range rdf:resource="#User_requirements"/>
73      </owl:ObjectProperty>
74      <!-- http://2009/9/SCS.owl#maintainsScale -->
75      <owl:ObjectProperty rdf:about="#maintainsScale">
76          <rdf:type rdf:resource="&owl;TransitiveProperty"/>
77          <rdfs:comment
78              >Every measure must have a scale associated with
79          it.</rdfs:comment>
80          <rdfs:domain rdf:resource="#Measure"/>
81          <rdfs:range rdf:resource="#Scale"/>
82      </owl:ObjectProperty>
83      <!-- http://2009/9/SCS.owl#measuredWith -->
84      <owl:ObjectProperty rdf:about="#measuredWith">
85          <rdfs:comment
86              >Every derived measurement value is achieved by applied
87          one or more measurement functions.</rdfs:comment>
88          <rdfs:domain rdf:resource="#Attributes"/>
89          <rdfs:range rdf:resource="#Metrics"/>
90      </owl:ObjectProperty>
91      <!--
92      ///////////////////////////////////////////////////////////////
93      //
94      // Data properties
95      //
96      ///////////////////////////////////////////////////////////////
97       -->
98      <!-- http://2009/9/SCS.owl#companyStandardization -->
99      <owl:DatatypeProperty rdf:about="#companyStandardization"/>
100     <!-- http://2009/9/SCS.owl#deliveryStatus -->
101     <owl:DatatypeProperty rdf:about="#deliveryStatus"/>
102     <!-- http://2009/9/SCS.owl#formalProofs -->
103     <owl:DatatypeProperty rdf:about="#formalProofs"/>
104     <!-- http://2009/9/SCS.owl#formalProofsStatus -->
105     <owl:DatatypeProperty rdf:about="#formalProofsStatus"/>
106     <!-- http://2009/9/SCS.owl#generalStandardization -->
107     <owl:DatatypeProperty rdf:about="#generalStandardization"/>
108     <!-- http://2009/9/SCS.owl#industryStandardization -->
109     <owl:DatatypeProperty rdf:about="#industryStandardization"/>
110     <!-- http://2009/9/SCS.owl#informalProofsStatus -->
111     <owl:DatatypeProperty rdf:about="#informalProofsStatus"/>
```

```
112     <!-- http://2009/9/SCS.owl#manualSpotEvaluationStatus -->
113     <owl:DatatypeProperty rdf:about="#manualSpotEvaluationStatus"/>
114     <!-- http://2009/9/SCS.owl#regularAutomatedTestingFTStatus -->
115     <owl:DatatypeProperty rdf:about="#regularAutomatedTestingFTStatus"/>
116     <!-- http://2009/9/SCS.owl#regularAutomatedTestingGUIFTStatus -->
117     <owl:DatatypeProperty rdf:about="#regularAutomatedTestingGUIFTStatus"/>
118     <!-- http://2009/9/SCS.owl#regularAutomatedTestingGUIUTStatus -->
119     <owl:DatatypeProperty rdf:about="#regularAutomatedTestingGUIUTStatus"/>
120     <!-- http://2009/9/SCS.owl#regularAutomatedTestingUTStatus -->
121     <owl:DatatypeProperty rdf:about="#regularAutomatedTestingUTStatus"/>
122     <!-- http://2009/9/SCS.owl#rigorousAuotmaticTesting -->
123     <owl:DatatypeProperty rdf:about="#rigorousAuotmaticTesting"/>
124     <!--
125     ///////////////////////////////////////////////////////////////////
126     //
127     // Classes
128     //
129     ///////////////////////////////////////////////////////////////////
130      -->
131     <!-- http://2009/9/SCS.owl#Attributes -->
132     <owl:Class rdf:about="#Attributes">
133         <owl:disjointWith rdf:resource="#Certificate_sections"/>
134         <owl:disjointWith rdf:resource="#Certification"/>
135         <owl:disjointWith rdf:resource="#Component_sections"/>
136         <owl:disjointWith rdf:resource="#Data"/>
137         <owl:disjointWith rdf:resource="#Metrics"/>
138     </owl:Class>
139     <!-- http://2009/9/SCS.owl#Base_measure -->
140     <owl:Class rdf:about="#Base_measure">
141         <rdfs:subClassOf rdf:resource="#Data"/>
142         <owl:disjointWith rdf:resource="#Concept_Models"/>
143         <owl:disjointWith rdf:resource="#Decision_criteria"/>
144         <owl:disjointWith rdf:resource="#Elementary_Models"/>
145         <owl:disjointWith rdf:resource="#Evaluation"/>
146         <owl:disjointWith rdf:resource="#Evaluation_result"/>
147         <owl:disjointWith rdf:resource="#Measurable_concepts"/>
148         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
149         <owl:disjointWith rdf:resource="#Process_description"/>
150         <owl:disjointWith rdf:resource="#Quality_models"/>
151         <rdfs:comment
152             >A basic measurement that could be applied individually
153         without any external input.</rdfs:comment>
154     </owl:Class>
155     <!-- http://2009/9/SCS.owl#CL_&#8722;_1 -->
156     <owl:Class rdf:about="#CL_&#8722;_1">
157         <rdfs:subClassOf rdf:resource="#Certification_Level"/>
158         <owl:disjointWith rdf:resource="#CL_&#8722;_2"/>
159         <owl:disjointWith rdf:resource="#CL_&#8722;_3"/>
160         <owl:disjointWith rdf:resource="#CL_&#8722;_4"/>
161         <owl:disjointWith rdf:resource="#CL_&#8722;_5"/>
162         <rdfs:comment
163             >CL &#8722; 1 &gt;= 1 &amp; CL &#8722; 2 &gt;= 1 &amp; CL
164     &#8722; 3 = 0
165     All the necessary elements of the product have been delivered and the
166     evaluator can start executing certification process.</rdfs:comment>
167     </owl:Class>
168     <!-- http://2009/9/SCS.owl#CL_&#8722;_2 -->
169     <owl:Class rdf:about="#CL_&#8722;_2">
170         <rdfs:subClassOf rdf:resource="#Certification_Level"/>
171         <owl:disjointWith rdf:resource="#CL_&#8722;_3"/>
172         <owl:disjointWith rdf:resource="#CL_&#8722;_4"/>
173         <owl:disjointWith rdf:resource="#CL_&#8722;_5"/>
174         <rdfs:comment
175        >CL &#8722; 1 &gt;= 1 &amp; CL &#8722; 2 &gt;= 1 &amp; CL
176     &#8722; 3 = 1
177     All the necessary elements of the product and their properties have been
```

```
178     manually validated. Also, all relationships between elements
179     have been manually evaluated.</rdfs:comment>
180     </owl:Class>
181     <!-- http://2009/9/SCS.owl#CL_&#8722;_3 -->
182     <owl:Class rdf:about="#CL_&#8722;_3">
183         <rdfs:subClassOf rdf:resource="#Certification_Level"/>
184         <owl:disjointWith rdf:resource="#CL_&#8722;_4"/>
185         <owl:disjointWith rdf:resource="#CL_&#8722;_5"/>
186         <rdfs:comment
187         >CL &#8722; 1 &gt;= 2 &amp; CL &#8722; 2 &gt;= 2 &amp; CL &#8722; 3 = 2
188 All the necessary elements of the product and their properties
189 have been validated with the help of automated tools. Also, all
190 relationships between elements have been evaluated with the help
191 of automated tools.</rdfs:comment>
192     </owl:Class>
193     <!-- http://2009/9/SCS.owl#CL_&#8722;_4 -->
194     <owl:Class rdf:about="#CL_&#8722;_4">
195         <rdfs:subClassOf rdf:resource="#Certification_Level"/>
196         <owl:disjointWith rdf:resource="#CL_&#8722;_5"/>
197         <rdfs:comment
198             >CL &#8722; 1 = 3 &amp; CL &#8722; 2 = 3 &amp; CL &#8722; 3 = 3
199 All the necessary elements of the product and their properties
200 have been validated with the help of mathematical methods in
201 situations where formal mathematical verification can be applied.
202 The same applies for relationships between elements.</rdfs:comment>
203     </owl:Class>
204     <!-- http://2009/9/SCS.owl#CL_&#8722;_5 -->
205     <owl:Class rdf:about="#CL_&#8722;_5">
206         <rdfs:subClassOf rdf:resource="#Certification_Level"/>
207         <rdfs:comment
208             >CL &#8722; 1 = 3 &amp; CL &#8722; 2 = 3 &amp; CL &#8722; 3 = 3
209 The same requirements as for FCL &#8722; 4, but the architecture is
210 formally verified where possible with real input data and not only
211 testing data.</rdfs:comment>
212     </owl:Class>
213     <!-- http://2009/9/SCS.owl#Calculable_Concepts -->
214   <owl:Class rdf:about="#Calculable_Concepts">
215         <rdfs:subClassOf rdf:resource="#Metrics"/>
216         <owl:disjointWith rdf:resource="#Calculation_Method"/>
217         <owl:disjointWith rdf:resource="#Direct_Metric"/>
218         <owl:disjointWith rdf:resource="#Elementary_Indicator"/>
219         <owl:disjointWith rdf:resource="#Function"/>
220         <owl:disjointWith rdf:resource="#Indirect_Metric"/>
221         <owl:disjointWith rdf:resource="#Measure"/>
222         <owl:disjointWith rdf:resource="#Scale"/>
223         <owl:disjointWith rdf:resource="#Software_Tool"/>
224         <rdfs:comment
225             >The relationship between attributes and calculable
226     Concepts</rdfs:comment>
227     </owl:Class>
228     <!-- http://2009/9/SCS.owl#Calculation_Method -->
229     <owl:Class rdf:about="#Calculation_Method">
230         <rdfs:subClassOf rdf:resource="#Metrics"/>
231         <owl:disjointWith rdf:resource="#Direct_Metric"/>
232         <owl:disjointWith rdf:resource="#Elementary_Indicator"/>
233         <owl:disjointWith rdf:resource="#Function"/>
234         <owl:disjointWith rdf:resource="#Indirect_Metric"/>
235         <owl:disjointWith rdf:resource="#Measure"/>
236         <owl:disjointWith rdf:resource="#Scale"/>
237         <owl:disjointWith rdf:resource="#Software_Tool"/>
238         <rdfs:comment
239             >A sequence of logical steps where a formula or indicator could
240     be applied in order to obtain a concept of measure</rdfs:comment>
241     </owl:Class>
242     <!-- http://2009/9/SCS.owl#Certificate_sections -->
243     <owl:Class rdf:about="#Certificate_sections">
```

```
244         <owl:disjointWith rdf:resource="#Certification"/>
245         <owl:disjointWith rdf:resource="#Component_sections"/>
246         <owl:disjointWith rdf:resource="#Data"/>
247         <owl:disjointWith rdf:resource="#Metrics"/>
248     </owl:Class>
249     <!-- http://2009/9/SCS.owl#Certification -->
250     <owl:Class rdf:about="#Certification">
251         <owl:disjointWith rdf:resource="#Component_sections"/>
252         <owl:disjointWith rdf:resource="#Data"/>
253         <owl:disjointWith rdf:resource="#Metrics"/>
254     </owl:Class>
255     <!-- http://2009/9/SCS.owl#Certification_Level -->
256     <owl:Class rdf:about="#Certification_Level">
257         <rdfs:subClassOf rdf:resource="#Certification"/>
258         <owl:disjointWith rdf:resource="#Completeness"/>
259         <owl:disjointWith rdf:resource="#Conformance"/>
260         <owl:disjointWith rdf:resource="#Uniformity"/>
261     </owl:Class>
262     <!-- http://2009/9/SCS.owl#Completeness -->
263     <owl:Class rdf:about="#Completeness">
264         <rdfs:subClassOf rdf:resource="#Certification"/>
265         <owl:disjointWith rdf:resource="#Conformance"/>
266         <owl:disjointWith rdf:resource="#Uniformity"/>
267         <rdfs:comment>Conformance</rdfs:comment>
268     </owl:Class>
269     <!-- http://2009/9/SCS.owl#Completeness_Level_1 -->
270     <owl:Class rdf:about="#Completeness_Level_1">
271         <rdfs:subClassOf rdf:resource="#Completeness"/>
272         <owl:disjointWith rdf:resource="#Completeness_Level_2"/>
273         <owl:disjointWith rdf:resource="#Completeness_Level_3"/>
274         <owl:disjointWith rdf:resource="#Completeness_Level_4"/>
275         <rdfs:comment
276             >errors in the component are found</rdfs:comment>
277     </owl:Class>
278     <!-- http://2009/9/SCS.owl#Completeness_Level_2 -->
279     <owl:Class rdf:about="#Completeness_Level_2">
280         <rdfs:subClassOf rdf:resource="#Completeness"/>
281         <owl:disjointWith rdf:resource="#Completeness_Level_3"/>
282         <owl:disjointWith rdf:resource="#Completeness_Level_4"/>
283         <rdfs:comment
284             >no errors were found with manual
285     evaluation and testing of the component</rdfs:comment>
286     </owl:Class>
287     <!-- http://2009/9/SCS.owl#Completeness_Level_3 -->
288     <owl:Class rdf:about="#Completeness_Level_3">
289         <rdfs:subClassOf rdf:resource="#Completeness"/>
290         <owl:disjointWith rdf:resource="#Completeness_Level_4"/>
291         <rdfs:comment
292             >rigorous automatic testing did not detect
293     any errors</rdfs:comment>
294     </owl:Class>
295     <!-- http://2009/9/SCS.owl#Completeness_Level_4 -->
296     <owl:Class rdf:about="#Completeness_Level_4">
297         <rdfs:subClassOf rdf:resource="#Completeness"/>
298         <rdfs:comment
299             >formal verification of the component pass</rdfs:comment>
300     </owl:Class>
301     <!-- http://2009/9/SCS.owl#Component -->
302     <owl:Class rdf:about="#Component">
303         <rdfs:subClassOf rdf:resource="#Certificate_sections"/>
304         <owl:disjointWith rdf:resource="#Documentation"/>
305         <owl:disjointWith rdf:resource="#Final_certification_level"/>
306         <owl:disjointWith rdf:resource="#Input_data"/>
307         <owl:disjointWith rdf:resource="#Measuring_methods"/>
308         <owl:disjointWith rdf:resource="#Properties"/>
309         <rdfs:comment
```

```
310             >Description of the component</rdfs:comment>
311     </owl:Class>
312     <!-- http://2009/9/SCS.owl#Component_sections -->
313     <owl:Class rdf:about="#Component_sections">
314         <rdfs:subClassOf>
315             <owl:Restriction>
316                 <owl:onProperty rdf:resource="#hasDelivery"/>
317                 <owl:someValuesFrom rdf:resource="#Correctness_proofs"/>
318             </owl:Restriction>
319         </rdfs:subClassOf>
320         <rdfs:subClassOf>
321             <owl:Restriction>
322                 <owl:onProperty rdf:resource="#hasDelivery"/>
323                 <owl:someValuesFrom rdf:resource="#Context_description"/>
324             </owl:Restriction>
325         </rdfs:subClassOf>
326         <rdfs:subClassOf>
327             <owl:Restriction>
328                 <owl:onProperty rdf:resource="#hasDelivery"/>
329                 <owl:someValuesFrom rdf:resource="#High_level_design"/>
330             </owl:Restriction>
331         </rdfs:subClassOf>
332         <rdfs:subClassOf>
333             <owl:Restriction>
334                 <owl:onProperty rdf:resource="#hasDelivery"/>
335                 <owl:someValuesFrom rdf:resource="#User_requirements"/>
336             </owl:Restriction>
337         </rdfs:subClassOf>
338         <rdfs:subClassOf>
339             <owl:Restriction>
340                 <owl:onProperty rdf:resource="#hasDelivery"/>
341                 <owl:someValuesFrom rdf:resource="#Detail_design"/>
342             </owl:Restriction>
343         </rdfs:subClassOf>
344         <rdfs:subClassOf>
345             <owl:Restriction>
346                 <owl:onProperty rdf:resource="#hasDelivery"/>
347                 <owl:someValuesFrom rdf:resource="#Testing"/>
348             </owl:Restriction>
349         </rdfs:subClassOf>
350         <rdfs:subClassOf>
351             <owl:Restriction>
352                 <owl:onProperty rdf:resource="#hasDelivery"/>
353                 <owl:someValuesFrom rdf:resource="#Implementation"/>
354             </owl:Restriction>
355         </rdfs:subClassOf>
356         <owl:disjointWith rdf:resource="#Data"/>
357         <owl:disjointWith rdf:resource="#Metrics"/>
358     </owl:Class>
359     <!-- http://2009/9/SCS.owl#Concept_Models -->
360     <owl:Class rdf:about="#Concept_Models">
361         <rdfs:subClassOf rdf:resource="#Data"/>
362         <owl:disjointWith rdf:resource="#Decision_criteria"/>
363         <owl:disjointWith rdf:resource="#Elementary_Models"/>
364         <owl:disjointWith rdf:resource="#Evaluation"/>
365         <owl:disjointWith rdf:resource="#Evaluation_result"/>
366         <owl:disjointWith rdf:resource="#Measurable_concepts"/>
367         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
368         <owl:disjointWith rdf:resource="#Process_description"/>
369         <owl:disjointWith rdf:resource="#Quality_models"/>
370         <rdfs:comment
371             >The collection of sub-concepts and associations between
372     them.</rdfs:comment>
373     </owl:Class>
374     <!-- http://2009/9/SCS.owl#Conformance -->
375     <owl:Class rdf:about="#Conformance">
```

```
376         <rdfs:subClassOf rdf:resource="#Certification"/>
377         <owl:disjointWith rdf:resource="#Uniformity"/>
378         <rdfs:comment>Completeness</rdfs:comment>
379     </owl:Class>
380     <!-- http://2009/9/SCS.owl#Conformance_Level_1 -->
381     <owl:Class rdf:about="#Conformance_Level_1">
382         <rdfs:subClassOf rdf:resource="#Conformance"/>
383         <owl:disjointWith rdf:resource="#Conformance_Level_2"/>
384         <owl:disjointWith rdf:resource="#Conformance_Level_3"/>
385         <owl:disjointWith rdf:resource="#Conformance_Level_4"/>
386         <rdfs:comment
387     >errors were found during regular automated testing</rdfs:comment>
388     </owl:Class>
389     <!-- http://2009/9/SCS.owl#Conformance_Level_2 -->
390     <owl:Class rdf:about="#Conformance_Level_2">
391         <rdfs:subClassOf rdf:resource="#Conformance"/>
392         <owl:disjointWith rdf:resource="#Conformance_Level_3"/>
393         <owl:disjointWith rdf:resource="#Conformance_Level_4"/>
394         <rdfs:comment
395             >no errors were found with manual spot evaluation and regular
396     automated testing</rdfs:comment>
397     </owl:Class>
398     <!-- http://2009/9/SCS.owl#Conformance_Level_3 -->
399     <owl:Class rdf:about="#Conformance_Level_3">
400         <rdfs:subClassOf rdf:resource="#Conformance"/>
401         <owl:disjointWith rdf:resource="#Conformance_Level_4"/>
402         <rdfs:comment
403             >rigorous automatic testing did not detect any errors
404     (includes stress testing)</rdfs:comment>
405     </owl:Class>
406     <!-- http://2009/9/SCS.owl#Conformance_Level_4 -->
407     <owl:Class rdf:about="#Conformance_Level_4">
408         <rdfs:subClassOf rdf:resource="#Conformance"/>
409         <rdfs:comment
410             >all formal verification of the component pass and
411     have beed delivered</rdfs:comment>
412     </owl:Class>
413     <!-- http://2009/9/SCS.owl#Context_description -->
414     <owl:Class rdf:about="#Context_description">
415         <rdfs:subClassOf rdf:resource="#Component_sections"/>
416         <owl:disjointWith rdf:resource="#Correctness_proofs"/>
417         <owl:disjointWith rdf:resource="#Detail_design"/>
418         <owl:disjointWith rdf:resource="#High_level_design"/>
419         <owl:disjointWith rdf:resource="#Implementation"/>
420         <owl:disjointWith rdf:resource="#Testing"/>
421         <owl:disjointWith rdf:resource="#User_requirements"/>
422         <rdfs:comment
423             >Describes main objectives and environment of the
424     component</rdfs:comment>
425     </owl:Class>
426     <!-- http://2009/9/SCS.owl#Correctness_proofs -->
427     <owl:Class rdf:about="#Correctness_proofs">
428         <rdfs:subClassOf rdf:resource="#Component_sections"/>
429         <owl:disjointWith rdf:resource="#Detail_design"/>
430         <owl:disjointWith rdf:resource="#High_level_design"/>
431         <owl:disjointWith rdf:resource="#Implementation"/>
432         <owl:disjointWith rdf:resource="#Testing"/>
433         <owl:disjointWith rdf:resource="#User_requirements"/>
434         <rdfs:comment
435             >Formal proofs for some sections of the component
436     </rdfs:comment>
437     </owl:Class>
438     <!-- http://2009/9/SCS.owl#Data -->
439     <owl:Class rdf:about="#Data">
440         <owl:disjointWith rdf:resource="#Metrics"/>
441     </owl:Class>
```

```
442     <!-- http://2009/9/SCS.owl#Decision_criteria -->
443     <owl:Class rdf:about="#Decision_criteria">
444         <rdfs:subClassOf rdf:resource="#Data"/>
445         <owl:disjointWith rdf:resource="#Elementary_Models"/>
446         <owl:disjointWith rdf:resource="#Evaluation"/>
447         <owl:disjointWith rdf:resource="#Evaluation_result"/>
448         <owl:disjointWith rdf:resource="#Measurable_concepts"/>
449         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
450         <owl:disjointWith rdf:resource="#Process_description"/>
451         <owl:disjointWith rdf:resource="#Quality_models"/>
452         <rdfs:comment
453             >Description on how to achieve certain level on confidence
454     in a particular result.</rdfs:comment>
455     </owl:Class>
456     <!-- http://2009/9/SCS.owl#Dedicated_External -->
457     <owl:Class rdf:about="#Dedicated_External">
458         <rdfs:subClassOf rdf:resource="#Attributes"/>
459         <owl:disjointWith rdf:resource="#Dedicated_Internal"/>
460         <owl:disjointWith rdf:resource="#General_External"/>
461         <owl:disjointWith rdf:resource="#General_Internal"/>
462     </owl:Class>
463     <!-- http://2009/9/SCS.owl#Dedicated_External_High -->
464     <owl:Class rdf:about="#Dedicated_External_High">
465         <rdfs:subClassOf rdf:resource="#Dedicated_External"/>
466         <owl:disjointWith rdf:resource="#Dedicated_External_Low"/>
467         <owl:disjointWith rdf:resource="#Dedicated_External_Moderate"/>
468     </owl:Class>
469     <!-- http://2009/9/SCS.owl#Dedicated_External_Low -->
470     <owl:Class rdf:about="#Dedicated_External_Low">
471         <rdfs:subClassOf rdf:resource="#Dedicated_External"/>
472         <owl:disjointWith rdf:resource="#Dedicated_External_Moderate"/>
473     </owl:Class>
474     <!-- http://2009/9/SCS.owl#Dedicated_External_Moderate -->
475     <owl:Class rdf:about="#Dedicated_External_Moderate">
476         <rdfs:subClassOf rdf:resource="#Dedicated_External"/>
477     </owl:Class>
478     <!-- http://2009/9/SCS.owl#Dedicated_Internal -->
479     <owl:Class rdf:about="#Dedicated_Internal">
480         <rdfs:subClassOf rdf:resource="#Attributes"/>
481         <owl:disjointWith rdf:resource="#General_External"/>
482         <owl:disjointWith rdf:resource="#General_Internal"/>
483     </owl:Class>
484     <!-- http://2009/9/SCS.owl#Dedicated_Internal_High -->
485     <owl:Class rdf:about="#Dedicated_Internal_High">
486         <rdfs:subClassOf rdf:resource="#Dedicated_Internal"/>
487         <owl:disjointWith rdf:resource="#Dedicated_Internal_Low"/>
488         <owl:disjointWith rdf:resource="#Dedicated_Internal_Moderate"/>
489     </owl:Class>
490     <!-- http://2009/9/SCS.owl#Dedicated_Internal_Low -->
491     <owl:Class rdf:about="#Dedicated_Internal_Low">
492         <rdfs:subClassOf rdf:resource="#Dedicated_Internal"/>
493         <owl:disjointWith rdf:resource="#Dedicated_Internal_Moderate"/>
494     </owl:Class>
495     <!-- http://2009/9/SCS.owl#Dedicated_Internal_Moderate -->
496     <owl:Class rdf:about="#Dedicated_Internal_Moderate">
497         <rdfs:subClassOf rdf:resource="#Dedicated_Internal"/>
498     </owl:Class>
499     <!-- http://2009/9/SCS.owl#Detail_design -->
500     <owl:Class rdf:about="#Detail_design">
501         <rdfs:subClassOf rdf:resource="#Component_sections"/>
502         <owl:disjointWith rdf:resource="#High_level_design"/>
503         <owl:disjointWith rdf:resource="#Implementation"/>
504         <owl:disjointWith rdf:resource="#Testing"/>
505         <owl:disjointWith rdf:resource="#User_requirements"/>
506         <rdfs:comment
507             >Could be a collection of designs that demonstrate
```

```
508     every aspect of the component</rdfs:comment>
509     </owl:Class>
510     <!-- http://2009/9/SCS.owl#Direct_Metric -->
511     <owl:Class rdf:about="#Direct_Metric">
512         <rdfs:subClassOf rdf:resource="#Metrics"/>
513         <owl:disjointWith rdf:resource="#Elementary_Indicator"/>
514         <owl:disjointWith rdf:resource="#Function"/>
515         <owl:disjointWith rdf:resource="#Indirect_Metric"/>
516         <owl:disjointWith rdf:resource="#Measure"/>
517         <owl:disjointWith rdf:resource="#Scale"/>
518         <owl:disjointWith rdf:resource="#Software_Tool"/>
519         <rdfs:comment
520             >An independent metric that can be applied individually
521     and does not depend on other metrics</rdfs:comment>
522     </owl:Class>
523     <!-- http://2009/9/SCS.owl#Documentation -->
524     <owl:Class rdf:about="#Documentation">
525         <rdfs:subClassOf rdf:resource="#Certificate_sections"/>
526         <owl:disjointWith rdf:resource="#Final_certification_level"/>
527         <owl:disjointWith rdf:resource="#Input_data"/>
528         <owl:disjointWith rdf:resource="#Measuring_methods"/>
529         <owl:disjointWith rdf:resource="#Properties"/>
530         <rdfs:comment
531             >Description of goals and what is being planed to be achieved
532     with the certification</rdfs:comment>
533     </owl:Class>
534     <!-- http://2009/9/SCS.owl#Elementary_Indicator -->
535     <owl:Class rdf:about="#Elementary_Indicator">
536         <rdfs:subClassOf rdf:resource="#Metrics"/>
537         <owl:disjointWith rdf:resource="#Function"/>
538         <owl:disjointWith rdf:resource="#Indirect_Metric"/>
539         <owl:disjointWith rdf:resource="#Measure"/>
540         <owl:disjointWith rdf:resource="#Scale"/>
541         <owl:disjointWith rdf:resource="#Software_Tool"/>
542         <rdfs:comment
543             >An indicator that is independent and does not depend on
544     other indicators to deduce calculable concept</rdfs:comment>
545     </owl:Class>
546     <!-- http://2009/9/SCS.owl#Elementary_Models -->
547     <owl:Class rdf:about="#Elementary_Models">
548         <rdfs:subClassOf rdf:resource="#Data"/>
549         <owl:disjointWith rdf:resource="#Evaluation"/>
550         <owl:disjointWith rdf:resource="#Evaluation_result"/>
551         <owl:disjointWith rdf:resource="#Measurable_concepts"/>
552         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
553         <owl:disjointWith rdf:resource="#Process_description"/>
554         <owl:disjointWith rdf:resource="#Quality_models"/>
555         <rdfs:comment
556             >A collection of models with are based on on some known
557     algorithms which can evaluate known criteria.</rdfs:comment>
558     </owl:Class>
559     <!-- http://2009/9/SCS.owl#Evaluation -->
560     <owl:Class rdf:about="#Evaluation">
561         <rdfs:subClassOf rdf:resource="#Data"/>
562         <owl:disjointWith rdf:resource="#Evaluation_result"/>
563         <owl:disjointWith rdf:resource="#Measurable_concepts"/>
564         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
565         <owl:disjointWith rdf:resource="#Process_description"/>
566         <owl:disjointWith rdf:resource="#Quality_models"/>
567         <rdfs:comment
568             >A collection of evaluation which produces measuring results for a
569     single attribute by applying one or more measuring approaches.</rdfs:comment>
570     </owl:Class>
571     <!-- http://2009/9/SCS.owl#Evaluation_result -->
572     <owl:Class rdf:about="#Evaluation_result">
573         <rdfs:subClassOf rdf:resource="#Data"/>
```

```
574         <owl:disjointWith rdf:resource="#Measurable_concepts"/>
575         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
576         <owl:disjointWith rdf:resource="#Process_description"/>
577         <owl:disjointWith rdf:resource="#Quality_models"/>
578         <rdfs:comment
579             >A number or an abstract value which indicates some
580     level of achievement.</rdfs:comment>
581     </owl:Class>
582     <!-- http://2009/9/SCS.owl#Final_certification_level -->
583     <owl:Class rdf:about="#Final_certification_level">
584         <rdfs:subClassOf rdf:resource="#Certificate_sections"/>
585         <owl:disjointWith rdf:resource="#Input_data"/>
586         <owl:disjointWith rdf:resource="#Measuring_methods"/>
587         <owl:disjointWith rdf:resource="#Properties"/>
588         <rdfs:comment
589             >An overall level of certification</rdfs:comment>
590     </owl:Class>
591     <!-- http://2009/9/SCS.owl#Function -->
592     <owl:Class rdf:about="#Function">
593         <rdfs:subClassOf rdf:resource="#Metrics"/>
594         <owl:disjointWith rdf:resource="#Indirect_Metric"/>
595         <owl:disjointWith rdf:resource="#Measure"/>
596         <owl:disjointWith rdf:resource="#Scale"/>
597         <owl:disjointWith rdf:resource="#Software_Tool"/>
598         <rdfs:comment
599             >Can be a formula or an algorithm that associates
600     two or more metrics</rdfs:comment>
601     </owl:Class>
602    <!-- http://2009/9/SCS.owl#General_External -->
603     <owl:Class rdf:about="#General_External">
604         <rdfs:subClassOf rdf:resource="#Attributes"/>
605         <owl:disjointWith rdf:resource="#General_Internal"/>
606     </owl:Class>
607     <!-- http://2009/9/SCS.owl#General_External_High -->
608     <owl:Class rdf:about="#General_External_High">
609         <rdfs:subClassOf rdf:resource="#General_External"/>
610         <owl:disjointWith rdf:resource="#General_External_Low"/>
611         <owl:disjointWith rdf:resource="#General_External_Moderate"/>
612     </owl:Class>
613     <!-- http://2009/9/SCS.owl#General_External_Low -->
614     <owl:Class rdf:about="#General_External_Low">
615         <rdfs:subClassOf rdf:resource="#General_External"/>
616         <owl:disjointWith rdf:resource="#General_External_Moderate"/>
617     </owl:Class>
618     <!-- http://2009/9/SCS.owl#General_External_Moderate -->
619     <owl:Class rdf:about="#General_External_Moderate">
620         <rdfs:subClassOf rdf:resource="#General_External"/>
621     </owl:Class>
622     <!-- http://2009/9/SCS.owl#General_Internal -->
623     <owl:Class rdf:about="#General_Internal">
624         <rdfs:subClassOf rdf:resource="#Attributes"/>
625     </owl:Class>
626     <!-- http://2009/9/SCS.owl#General_Internal_High -->
627     <owl:Class rdf:about="#General_Internal_High">
628         <rdfs:subClassOf rdf:resource="#General_Internal"/>
629         <owl:disjointWith rdf:resource="#General_Internal_Low"/>
630         <owl:disjointWith rdf:resource="#General_Internal_Moderate"/>
631     </owl:Class>
632     <!-- http://2009/9/SCS.owl#General_Internal_Low -->
633     <owl:Class rdf:about="#General_Internal_Low">
634         <rdfs:subClassOf rdf:resource="#General_Internal"/>
635         <owl:disjointWith rdf:resource="#General_Internal_Moderate"/>
636     </owl:Class>
637     <!-- http://2009/9/SCS.owl#General_Internal_Moderate -->
638     <owl:Class rdf:about="#General_Internal_Moderate">
639         <rdfs:subClassOf rdf:resource="#General_Internal"/>
```

```
640     </owl:Class>
641     <!-- http://2009/9/SCS.owl#High_level_design -->
642     <owl:Class rdf:about="#High_level_design">
643         <rdfs:subClassOf rdf:resource="#Component_sections"/>
644         <owl:disjointWith rdf:resource="#Implementation"/>
645         <owl:disjointWith rdf:resource="#Testing"/>
646         <owl:disjointWith rdf:resource="#User_requirements"/>
647         <rdfs:comment
648             >Represents mapping between customer&amp;#8217;s require-
649     ments and system&amp;#8217;s design</rdfs:comment>
650     </owl:Class>
651     <!-- http://2009/9/SCS.owl#Implementation -->
652     <owl:Class rdf:about="#Implementation">
653         <rdfs:subClassOf rdf:resource="#Component_sections"/>
654         <owl:disjointWith rdf:resource="#Testing"/>
655         <owl:disjointWith rdf:resource="#User_requirements"/>
656         <rdfs:comment
657             >Represents relationship between component&amp;#8217;s code
658     and its documentation</rdfs:comment>
659     </owl:Class>
660     <!-- http://2009/9/SCS.owl#Indirect_Metric -->
661     <owl:Class rdf:about="#Indirect_Metric">
662         <rdfs:subClassOf rdf:resource="#Metrics"/>
663         <owl:disjointWith rdf:resource="#Measure"/>
664         <owl:disjointWith rdf:resource="#Scale"/>
665         <owl:disjointWith rdf:resource="#Software_Tool"/>
666         <rdfs:comment
667             >A metric that is constructed from other metric or metrics
668     that are being utilized for other attributes</rdfs:comment>
669     </owl:Class>
670     <!-- http://2009/9/SCS.owl#Input_data -->
671     <owl:Class rdf:about="#Input_data">
672         <rdfs:subClassOf rdf:resource="#Certificate_sections"/>
673         <owl:disjointWith rdf:resource="#Measuring_methods"/>
674         <owl:disjointWith rdf:resource="#Properties"/>
675         <rdfs:comment
676             >Precise description of the input data for the certification
677     process</rdfs:comment>
678     </owl:Class>
679     <!-- http://2009/9/SCS.owl#Measurable_concepts -->
680     <owl:Class rdf:about="#Measurable_concepts">
681         <rdfs:subClassOf rdf:resource="#Data"/>
682         <owl:disjointWith rdf:resource="#Measuring_approaches"/>
683         <owl:disjointWith rdf:resource="#Process_description"/>
684         <owl:disjointWith rdf:resource="#Quality_models"/>
685         <rdfs:comment
686             >Abstract description of components that could be
687     measured.</rdfs:comment>
688     </owl:Class>
689     <!-- http://2009/9/SCS.owl#Measure -->
690     <owl:Class rdf:about="#Measure">
691         <rdfs:subClassOf rdf:resource="#Metrics"/>
692         <owl:disjointWith rdf:resource="#Scale"/>
693         <owl:disjointWith rdf:resource="#Software_Tool"/>
694         <rdfs:comment
695             >A value that is associated with an attribute after
696     evaluation process</rdfs:comment>
697     </owl:Class>
698     <!-- http://2009/9/SCS.owl#Measuring_approaches -->
699     <owl:Class rdf:about="#Measuring_approaches">
700         <rdfs:subClassOf rdf:resource="#Data"/>
701         <owl:disjointWith rdf:resource="#Process_description"/>
702         <owl:disjointWith rdf:resource="#Quality_models"/>
703         <rdfs:comment
704             >The measurement approach could be measurement function,
705     analysis model which is interrelated with quality model or particular
```

```
706     measuring approach.</rdfs:comment>
707     </owl:Class>
708     <!-- http://2009/9/SCS.owl#Measuring_methods -->
709     <owl:Class rdf:about="#Measuring_methods">
710         <rdfs:subClassOf rdf:resource="#Certificate_sections"/>
711         <owl:disjointWith rdf:resource="#Properties"/>
712         <rdfs:comment
713             >Precise description of the measuring methods</rdfs:comment>
714     </owl:Class>
715     <!-- http://2009/9/SCS.owl#Metrics -->
716     <owl:Class rdf:about="#Metrics"/>
717     <!-- http://2009/9/SCS.owl#Process_description -->
718     <owl:Class rdf:about="#Process_description">
719         <rdfs:subClassOf rdf:resource="#Data"/>
720         <owl:disjointWith rdf:resource="#Quality_models"/>
721         <rdfs:comment
722             >Information which describes objectives, risks
723     and goals.</rdfs:comment>
724     </owl:Class>
725     <!-- http://2009/9/SCS.owl#Properties -->
726     <owl:Class rdf:about="#Properties">
727         <rdfs:subClassOf rdf:resource="#Certificate_sections"/>
728         <rdfs:comment
729             >The list of properties that should be
730     evaluated</rdfs:comment>
731     </owl:Class>
732     <!-- http://2009/9/SCS.owl#Quality_models -->
733     <owl:Class rdf:about="#Quality_models">
734         <rdfs:subClassOf rdf:resource="#Data"/>
735         <rdfs:comment
736             >Specification for the quality requirements and description
737     of entity class relationships.</rdfs:comment>
738     </owl:Class>
739     <!-- http://2009/9/SCS.owl#Scale -->
740     <owl:Class rdf:about="#Scale">
741         <rdfs:subClassOf rdf:resource="#Metrics"/>
742         <owl:disjointWith rdf:resource="#Software_Tool"/>
743         <rdfs:comment
744             >A collection of values that have specific meaning
745     associated with them</rdfs:comment>
746     </owl:Class>
747     <!-- http://2009/9/SCS.owl#Software_Tool -->
748     <owl:Class rdf:about="#Software_Tool">
749         <rdfs:subClassOf rdf:resource="#Metrics"/>
750         <rdfs:comment
751             >A tool or set of tools that is used during the
752     evaluation</rdfs:comment>
753     </owl:Class>
754     <!-- http://2009/9/SCS.owl#Testing -->
755     <owl:Class rdf:about="#Testing">
756         <rdfs:subClassOf rdf:resource="#Component_sections"/>
757         <owl:disjointWith rdf:resource="#User_requirements"/>
758         <rdfs:comment
759             >Description of the testing architecture</rdfs:comment>
760     </owl:Class>
761     <!-- http://2009/9/SCS.owl#Uniformity -->
762     <owl:Class rdf:about="#Uniformity">
763         <rdfs:subClassOf rdf:resource="#Certification"/>
764         <rdfs:comment>Uniformity</rdfs:comment>
765     </owl:Class>
766     <!-- http://2009/9/SCS.owl#Uniformity_Level_1 -->
767     <owl:Class rdf:about="#Uniformity_Level_1">
768         <rdfs:subClassOf rdf:resource="#Uniformity"/>
769         <owl:disjointWith rdf:resource="#Uniformity_Level_2"/>
770         <owl:disjointWith rdf:resource="#Uniformity_Level_3"/>
771         <owl:disjointWith rdf:resource="#Uniformity_Level_4"/>
```

```
772         <rdfs:comment
773             >uniformity and standardization is missing</rdfs:comment>
774     </owl:Class>
775     <!-- http://2009/9/SCS.owl#Uniformity_Level_2 -->
776     <owl:Class rdf:about="#Uniformity_Level_2">
777         <rdfs:subClassOf rdf:resource="#Uniformity"/>
778         <owl:disjointWith rdf:resource="#Uniformity_Level_3"/>
779         <owl:disjointWith rdf:resource="#Uniformity_Level_4"/>
780         <rdfs:comment
781             >some uniformity and standardization is present</rdfs:comment>
782     </owl:Class>
783     <!-- http://2009/9/SCS.owl#Uniformity_Level_3 -->
784     <owl:Class rdf:about="#Uniformity_Level_3">
785         <rdfs:subClassOf rdf:resource="#Uniformity"/>
786         <owl:disjointWith rdf:resource="#Uniformity_Level_4"/>
787         <rdfs:comment
788             >component conforms to uniformity and
789     standardization based on companies expectations</rdfs:comment>
790     </owl:Class>
791     <!-- http://2009/9/SCS.owl#Uniformity_Level_4 -->
792     <owl:Class rdf:about="#Uniformity_Level_4">
793         <rdfs:subClassOf rdf:resource="#Uniformity"/>
794         <rdfs:comment
795             >component conforms to uniformity and
796     standardization based on industry expectatio</rdfs:comment>
797     </owl:Class>
798     <!-- http://2009/9/SCS.owl#User_requirements -->
799     <owl:Class rdf:about="#User_requirements">
800         <rdfs:subClassOf rdf:resource="#Component_sections"/>
801         <rdfs:comment
802             >List of expectations from the component</rdfs:comment>
803     </owl:Class>
804     <!-- http://www.w3.org/2002/07/owl#Thing -->
805     <owl:Class rdf:about="&owl;Thing"/>
806     <!--
807     ///////////////////////////////////////////////////////////////
808     //
809     // Individuals
810     //
811     ///////////////////////////////////////////////////////////////
812      -->
813     <!-- http://2009/9/SCS.owl#company_standardization_delivery -->
814     <owl:Thing rdf:about="#company_standardization_delivery">
815         <rdf:type rdf:resource="#Implementation"/>
816         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
817         <companyStandardization rdf:datatype="&xsd;string">conforms
818     </companyStandardization>
819     </owl:Thing>
820     <!-- http://2009/9/SCS.owl#context_description_delivery -->
821     <owl:Thing rdf:about="#context_description_delivery">
822         <rdf:type rdf:resource="#Context_description"/>
823         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
824     </owl:Thing>
825     <!-- http://2009/9/SCS.owl#correctness_proofs_delivery -->
826     <Correctness_proofs rdf:about="#correctness_proofs_delivery">
827         <rdf:type rdf:resource="&owl;Thing"/>
828         <formalProofsStatus rdf:datatype="&xsd;string">complete
829         </formalProofsStatus>
830         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
831         <informalProofsStatus rdf:datatype="&xsd;string">complete
832         </informalProofsStatus>
833         <formalProofs rdf:datatype="&xsd;string">pass</formalProofs>
834     </Correctness_proofs>
835     <!-- http://2009/9/SCS.owl#detail_design_delivery -->
836     <owl:Thing rdf:about="#detail_design_delivery">
837         <rdf:type rdf:resource="#Detail_design"/>
```

```
838         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
839     </owl:Thing>
840     <!-- http://2009/9/SCS.owl#general_standardization_delivery -->
841     <owl:Thing rdf:about="#general_standardization_delivery">
842         <rdf:type rdf:resource="#Implementation"/>
843         <generalStandardization rdf:datatype="&xsd;string">above average
844     </generalStandardization>
845         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
846     </owl:Thing>
847     <!-- http://2009/9/SCS.owl#high_level_design_delivery -->
848     <owl:Thing rdf:about="#high_level_design_delivery">
849         <rdf:type rdf:resource="#High_level_design"/>
850         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
851     </owl:Thing>
852     <!-- http://2009/9/SCS.owl#implementation_delivery -->
853     <owl:Thing rdf:about="#implementation_delivery">
854         <rdf:type rdf:resource="#Implementation"/>
855         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
856     </owl:Thing>
857     <!-- http://2009/9/SCS.owl#industry_standardization_delivery -->
858     <owl:Thing rdf:about="#industry_standardization_delivery">
859         <rdf:type rdf:resource="#Implementation"/>
860         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
861         <industryStandardization rdf:datatype="&xsd;string">conforms
862     </industryStandardization>
863     </owl:Thing>
864     <!-- http://2009/9/SCS.owl#manual_spot_evaluation_delivery -->
865     <owl:Thing rdf:about="#manual_spot_evaluation_delivery">
866         <rdf:type rdf:resource="#Correctness_proofs"/>
867         <manualSpotEvaluationStatus rdf:datatype="&xsd;string">complete
868     </manualSpotEvaluationStatus>
869         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
870     </owl:Thing>
871     <!-- http://2009/9/SCS.owl#regular_automated_testing_delivery -->
872     <owl:Thing rdf:about="#regular_automated_testing_delivery">
873         <rdf:type rdf:resource="#Correctness_proofs"/>
874         <regularAutomatedTestingFTStatus rdf:datatype="&xsd;string">
875         complete</regularAutomatedTestingFTStatus>
876         <regularAutomatedTestingUTStatus rdf:datatype="&xsd;string">
877         complete</regularAutomatedTestingUTStatus>
878         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
879         <regularAutomatedTestingGUIUTStatus rdf:datatype="&xsd;string">
880         complete</regularAutomatedTestingGUIUTStatus>
881         <regularAutomatedTestingGUIFTStatus rdf:datatype="&xsd;string">
882         complete</regularAutomatedTestingGUIFTStatus>
883     </owl:Thing>
884     <!-- http://2009/9/SCS.owl#rigorous_automatic_testing_delivery -->
885     <Correctness_proofs rdf:about="#rigorous_automatic_testing_delivery">
886         <rdf:type rdf:resource="&owl;Thing"/>
887         <rigorousAuotmaticTesting rdf:datatype="&xsd;string">complete
888     </rigorousAuotmaticTesting>
889         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
890     </Correctness_proofs>
891     <!-- http://2009/9/SCS.owl#testing_delivery -->
892     <owl:Thing rdf:about="#testing_delivery">
893         <rdf:type rdf:resource="#Testing"/>
894         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
895     </owl:Thing>
896     <!-- http://2009/9/SCS.owl#user_requirements_delivery -->
897     <owl:Thing rdf:about="#user_requirements_delivery">
898         <rdf:type rdf:resource="#User_requirements"/>
899         <deliveryStatus rdf:datatype="&xsd;string">complete</deliveryStatus>
900     </owl:Thing>
901     <!--
902     ////////////////////////////////////////////////////////////////////////
903     //
```

```
904     // General axioms
905     //
906     ////////////////////////////////////////////////////////////////////////
907      -->
908     <rdf:Description>
909         <rdf:type rdf:resource="&owl;AllDifferent"/>
910         <owl:distinctMembers rdf:parseType="Collection">
911             <rdf:Description rdf:about="#testing_delivery"/>
912             <rdf:Description rdf:about="#implementation_delivery"/>
913         </owl:distinctMembers>
914     </rdf:Description>
915     <rdf:Description>
916         <rdf:type rdf:resource="&owl;AllDifferent"/>
917         <owl:distinctMembers rdf:parseType="Collection">
918             <rdf:Description rdf:about="#user_requirements_delivery"/>
919             <rdf:Description rdf:about="#high_level_design_delivery"/>
920         </owl:distinctMembers>
921     </rdf:Description>
922     <rdf:Description>
923         <rdf:type rdf:resource="&owl;AllDifferent"/>
924         <owl:distinctMembers rdf:parseType="Collection">
925             <rdf:Description rdf:about="#testing_delivery"/>
926             <rdf:Description rdf:about="#industry_standardization_delivery"/>
927         </owl:distinctMembers>
928     </rdf:Description>
929     <rdf:Description>
930         <rdf:type rdf:resource="&owl;AllDifferent"/>
931         <owl:distinctMembers rdf:parseType="Collection">
932             <rdf:Description rdf:about="#testing_delivery"/>
933             <rdf:Description rdf:about="#context_description_delivery"/>
934         </owl:distinctMembers>
935     </rdf:Description>
936     <rdf:Description>
937         <rdf:type rdf:resource="&owl;AllDifferent"/>
938         <owl:distinctMembers rdf:parseType="Collection">
939             <rdf:Description rdf:about="#implementation_delivery"/>
940             <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
941         </owl:distinctMembers>
942     </rdf:Description>
943     <rdf:Description>
944         <rdf:type rdf:resource="&owl;AllDifferent"/>
945         <owl:distinctMembers rdf:parseType="Collection">
946             <rdf:Description rdf:about="#high_level_design_delivery"/>
947             <rdf:Description rdf:about="#context_description_delivery"/>
948         </owl:distinctMembers>
949     </rdf:Description>
950     <rdf:Description>
951         <rdf:type rdf:resource="&owl;AllDifferent"/>
952         <owl:distinctMembers rdf:parseType="Collection">
953             <rdf:Description rdf:about="#implementation_delivery"/>
954             <rdf:Description rdf:about="#correctness_proofs_delivery"/>
955         </owl:distinctMembers>
956     </rdf:Description>
957     <rdf:Description>
958         <rdf:type rdf:resource="&owl;AllDifferent"/>
959         <owl:distinctMembers rdf:parseType="Collection">
960             <rdf:Description rdf:about="#testing_delivery"/>
961             <rdf:Description rdf:about="#user_requirements_delivery"/>
962         </owl:distinctMembers>
963     </rdf:Description>
964     <rdf:Description>
965         <rdf:type rdf:resource="&owl;AllDifferent"/>
966         <owl:distinctMembers rdf:parseType="Collection">
967             <rdf:Description rdf:about="#implementation_delivery"/>
968             <rdf:Description rdf:about="#context_description_delivery"/>
969         </owl:distinctMembers>
```

```
970     </rdf:Description>
971     <rdf:Description>
972         <rdf:type rdf:resource="&owl;AllDifferent"/>
973         <owl:distinctMembers rdf:parseType="Collection">
974             <rdf:Description rdf:about="#user_requirements_delivery"/>
975             <rdf:Description rdf:about="#industry_standardization_delivery"/>
976         </owl:distinctMembers>
977     </rdf:Description>
978     <rdf:Description>
979         <rdf:type rdf:resource="&owl;AllDifferent"/>
980         <owl:distinctMembers rdf:parseType="Collection">
981             <rdf:Description rdf:about="#correctness_proofs_delivery"/>
982             <rdf:Description rdf:about="#company_standardization_delivery"/>
983         </owl:distinctMembers>
984     </rdf:Description>
985     <rdf:Description>
986         <rdf:type rdf:resource="&owl;AllDifferent"/>
987         <owl:distinctMembers rdf:parseType="Collection">
988             <rdf:Description rdf:about="#high_level_design_delivery"/>
989             <rdf:Description rdf:about="#general_standardization_delivery"/>
990         </owl:distinctMembers>
991     </rdf:Description>
992     <rdf:Description>
993         <rdf:type rdf:resource="&owl;AllDifferent"/>
994         <owl:distinctMembers rdf:parseType="Collection">
995             <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
996             <rdf:Description rdf:about="#detail_design_delivery"/>
997         </owl:distinctMembers>
998     </rdf:Description>
999     <rdf:Description>
1000        <rdf:type rdf:resource="&owl;AllDifferent"/>
1001        <owl:distinctMembers rdf:parseType="Collection">
1002            <rdf:Description rdf:about="#implementation_delivery"/>
1003            <rdf:Description rdf:about="#company_standardization_delivery"/>
1004        </owl:distinctMembers>
1005    </rdf:Description>
1006    <rdf:Description>
1007        <rdf:type rdf:resource="&owl;AllDifferent"/>
1008        <owl:distinctMembers rdf:parseType="Collection">
1009            <rdf:Description rdf:about="#testing_delivery"/>
1010            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1011        </owl:distinctMembers>
1012    </rdf:Description>
1013    <rdf:Description>
1014        <rdf:type rdf:resource="&owl;AllDifferent"/>
1015        <owl:distinctMembers rdf:parseType="Collection">
1016            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1017            <rdf:Description rdf:about="#detail_design_delivery"/>
1018        </owl:distinctMembers>
1019    </rdf:Description>
1020    <rdf:Description>
1021        <rdf:type rdf:resource="&owl;AllDifferent"/>
1022        <owl:distinctMembers rdf:parseType="Collection">
1023            <rdf:Description rdf:about="#company_standardization_delivery"/>
1024            <rdf:Description rdf:about="#general_standardization_delivery"/>
1025        </owl:distinctMembers>
1026    </rdf:Description>
1027    <rdf:Description>
1028        <rdf:type rdf:resource="&owl;AllDifferent"/>
1029        <owl:distinctMembers rdf:parseType="Collection">
1030            <rdf:Description rdf:about="#user_requirements_delivery"/>
1031            <rdf:Description rdf:about="#general_standardization_delivery"/>
1032        </owl:distinctMembers>
1033    </rdf:Description>
1034    <rdf:Description>
1035        <rdf:type rdf:resource="&owl;AllDifferent"/>
```

```
1036        <owl:distinctMembers rdf:parseType="Collection">
1037            <rdf:Description rdf:about="#high_level_design_delivery"/>
1038            <rdf:Description rdf:about="#detail_design_delivery"/>
1039        </owl:distinctMembers>
1040    </rdf:Description>
1041    <rdf:Description>
1042        <rdf:type rdf:resource="&owl;AllDifferent"/>
1043        <owl:distinctMembers rdf:parseType="Collection">
1044            <rdf:Description rdf:about="#context_description_delivery"/>
1045            <rdf:Description rdf:about="#general_standardization_delivery"/>
1046        </owl:distinctMembers>
1047    </rdf:Description>
1048    <rdf:Description>
1049        <rdf:type rdf:resource="&owl;AllDifferent"/>
1050        <owl:distinctMembers rdf:parseType="Collection">
1051            <rdf:Description rdf:about="#implementation_delivery"/>
1052            <rdf:Description rdf:about="#general_standardization_delivery"/>
1053        </owl:distinctMembers>
1054    </rdf:Description>
1055    <rdf:Description>
1056        <rdf:type rdf:resource="&owl;AllDifferent"/>
1057        <owl:distinctMembers rdf:parseType="Collection">
1058            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1059            <rdf:Description rdf:about="#detail_design_delivery"/>
1060        </owl:distinctMembers>
1061    </rdf:Description>
1062    <rdf:Description>
1063        <rdf:type rdf:resource="&owl;AllDifferent"/>
1064        <owl:distinctMembers rdf:parseType="Collection">
1065            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1066            <rdf:Description rdf:about="#testing_delivery"/>
1067        </owl:distinctMembers>
1068    </rdf:Description>
1069    <rdf:Description>
1070        <rdf:type rdf:resource="&owl;AllDifferent"/>
1071        <owl:distinctMembers rdf:parseType="Collection">
1072            <rdf:Description rdf:about="#company_standardization_delivery"/>
1073            <rdf:Description rdf:about="#context_description_delivery"/>
1074        </owl:distinctMembers>
1075    </rdf:Description>
1076    <rdf:Description>
1077        <rdf:type rdf:resource="&owl;AllDifferent"/>
1078        <owl:distinctMembers rdf:parseType="Collection">
1079            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1080            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1081        </owl:distinctMembers>
1082    </rdf:Description>
1083    <rdf:Description>
1084        <rdf:type rdf:resource="&owl;AllDifferent"/>
1085        <owl:distinctMembers rdf:parseType="Collection">
1086            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1087            <rdf:Description rdf:about="#high_level_design_delivery"/>
1088        </owl:distinctMembers>
1089    </rdf:Description>
1090    <rdf:Description>
1091        <rdf:type rdf:resource="&owl;AllDifferent"/>
1092        <owl:distinctMembers rdf:parseType="Collection">
1093            <rdf:Description rdf:about="#implementation_delivery"/>
1094            <rdf:Description rdf:about="#high_level_design_delivery"/>
1095        </owl:distinctMembers>
1096    </rdf:Description>
1097    <rdf:Description>
1098        <rdf:type rdf:resource="&owl;AllDifferent"/>
1099        <owl:distinctMembers rdf:parseType="Collection">
1100            <rdf:Description rdf:about="#testing_delivery"/>
1101            <rdf:Description rdf:about="#general_standardization_delivery"/>
```

```
1102        </owl:distinctMembers>
1103    </rdf:Description>
1104    <rdf:Description>
1105        <rdf:type rdf:resource="&owl;AllDifferent"/>
1106        <owl:distinctMembers rdf:parseType="Collection">
1107            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1108            <rdf:Description rdf:about="#general_standardization_delivery"/>
1109        </owl:distinctMembers>
1110    </rdf:Description>
1111    <rdf:Description>
1112        <rdf:type rdf:resource="&owl;AllDifferent"/>
1113        <owl:distinctMembers rdf:parseType="Collection">
1114            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1115            <rdf:Description rdf:about="#general_standardization_delivery"/>
1116        </owl:distinctMembers>
1117    </rdf:Description>
1118    <rdf:Description>
1119        <rdf:type rdf:resource="&owl;AllDifferent"/>
1120        <owl:distinctMembers rdf:parseType="Collection">
1121            <rdf:Description rdf:about="#implementation_delivery"/>
1122            <rdf:Description rdf:about="#user_requirements_delivery"/>
1123        </owl:distinctMembers>
1124    </rdf:Description>
1125    <rdf:Description>
1126        <rdf:type rdf:resource="&owl;AllDifferent"/>
1127        <owl:distinctMembers rdf:parseType="Collection">
1128            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1129            <rdf:Description rdf:about="#general_standardization_delivery"/>
1130        </owl:distinctMembers>
1131    </rdf:Description>
1132    <rdf:Description>
1133        <rdf:type rdf:resource="&owl;AllDifferent"/>
1134        <owl:distinctMembers rdf:parseType="Collection">
1135            <rdf:Description rdf:about="#testing_delivery"/>
1136            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1137        </owl:distinctMembers>
1138    </rdf:Description>
1139    <rdf:Description>
1140        <rdf:type rdf:resource="&owl;AllDifferent"/>
1141        <owl:distinctMembers rdf:parseType="Collection">
1142            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1143            <rdf:Description rdf:about="#general_standardization_delivery"/>
1144        </owl:distinctMembers>
1145    </rdf:Description>
1146    <rdf:Description>
1147        <rdf:type rdf:resource="&owl;AllDifferent"/>
1148        <owl:distinctMembers rdf:parseType="Collection">
1149            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1150            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1151        </owl:distinctMembers>
1152    </rdf:Description>
1153    <rdf:Description>
1154        <rdf:type rdf:resource="&owl;AllDifferent"/>
1155        <owl:distinctMembers rdf:parseType="Collection">
1156            <rdf:Description rdf:about="#implementation_delivery"/>
1157            <rdf:Description rdf:about="#detail_design_delivery"/>
1158        </owl:distinctMembers>
1159    </rdf:Description>
1160    <rdf:Description>
1161        <rdf:type rdf:resource="&owl;AllDifferent"/>
1162        <owl:distinctMembers rdf:parseType="Collection">
1163            <rdf:Description rdf:about="#testing_delivery"/>
1164            <rdf:Description rdf:about="#company_standardization_delivery"/>
1165        </owl:distinctMembers>
1166    </rdf:Description>
1167    <rdf:Description>
```

```
1168        <rdf:type rdf:resource="&owl;AllDifferent"/>
1169        <owl:distinctMembers rdf:parseType="Collection">
1170            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1171            <rdf:Description rdf:about="#implementation_delivery"/>
1172        </owl:distinctMembers>
1173    </rdf:Description>
1174    <rdf:Description>
1175        <rdf:type rdf:resource="&owl;AllDifferent"/>
1176        <owl:distinctMembers rdf:parseType="Collection">
1177            <rdf:Description rdf:about="#general_standardization_delivery"/>
1178            <rdf:Description rdf:about="#detail_design_delivery"/>
1179        </owl:distinctMembers>
1180    </rdf:Description>
1181    <rdf:Description>
1182        <rdf:type rdf:resource="&owl;AllDifferent"/>
1183        <owl:distinctMembers rdf:parseType="Collection">
1184            <rdf:Description rdf:about="#high_level_design_delivery"/>
1185            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1186        </owl:distinctMembers>
1187    </rdf:Description>
1188    <rdf:Description>
1189        <rdf:type rdf:resource="&owl;AllDifferent"/>
1190        <owl:distinctMembers rdf:parseType="Collection">
1191            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1192            <rdf:Description rdf:about="#general_standardization_delivery"/>
1193        </owl:distinctMembers>
1194    </rdf:Description>
1195    <rdf:Description>
1196        <rdf:type rdf:resource="&owl;AllDifferent"/>
1197        <owl:distinctMembers rdf:parseType="Collection">
1198            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1199            <rdf:Description rdf:about="#context_description_delivery"/>
1200        </owl:distinctMembers>
1201    </rdf:Description>
1202    <rdf:Description>
1203        <rdf:type rdf:resource="&owl;AllDifferent"/>
1204        <owl:distinctMembers rdf:parseType="Collection">
1205            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1206            <rdf:Description rdf:about="#detail_design_delivery"/>
1207        </owl:distinctMembers>
1208    </rdf:Description>
1209    <rdf:Description>
1210        <rdf:type rdf:resource="&owl;AllDifferent"/>
1211        <owl:distinctMembers rdf:parseType="Collection">
1212            <rdf:Description rdf:about="#testing_delivery"/>
1213            <rdf:Description rdf:about="#detail_design_delivery"/>
1214        </owl:distinctMembers>
1215    </rdf:Description>
1216    <rdf:Description>
1217        <rdf:type rdf:resource="&owl;AllDifferent"/>
1218        <owl:distinctMembers rdf:parseType="Collection">
1219            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1220            <rdf:Description rdf:about="#company_standardization_delivery"/>
1221        </owl:distinctMembers>
1222    </rdf:Description>
1223    <rdf:Description>
1224        <rdf:type rdf:resource="&owl;AllDifferent"/>
1225        <owl:distinctMembers rdf:parseType="Collection">
1226            <rdf:Description rdf:about="#testing_delivery"/>
1227            <rdf:Description rdf:about="#high_level_design_delivery"/>
1228        </owl:distinctMembers>
1229    </rdf:Description>
1230    <rdf:Description>
1231        <rdf:type rdf:resource="&owl;AllDifferent"/>
1232        <owl:distinctMembers rdf:parseType="Collection">
1233            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
```

```
1234            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1235        </owl:distinctMembers>
1236    </rdf:Description>
1237    <rdf:Description>
1238        <rdf:type rdf:resource="&owl;AllDifferent"/>
1239        <owl:distinctMembers rdf:parseType="Collection">
1240            <rdf:Description rdf:about="#user_requirements_delivery"/>
1241            <rdf:Description rdf:about="#detail_design_delivery"/>
1242        </owl:distinctMembers>
1243    </rdf:Description>
1244    <rdf:Description>
1245        <rdf:type rdf:resource="&owl;AllDifferent"/>
1246        <owl:distinctMembers rdf:parseType="Collection">
1247            <rdf:Description rdf:about="#high_level_design_delivery"/>
1248            <rdf:Description rdf:about="#company_standardization_delivery"/>
1249        </owl:distinctMembers>
1250    </rdf:Description>
1251    <rdf:Description>
1252        <rdf:type rdf:resource="&owl;AllDifferent"/>
1253        <owl:distinctMembers rdf:parseType="Collection">
1254            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1255            <rdf:Description rdf:about="#company_standardization_delivery"/>
1256        </owl:distinctMembers>
1257    </rdf:Description>
1258    <rdf:Description>
1259        <rdf:type rdf:resource="&owl;AllDifferent"/>
1260        <owl:distinctMembers rdf:parseType="Collection">
1261            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1262            <rdf:Description rdf:about="#high_level_design_delivery"/>
1263        </owl:distinctMembers>
1264    </rdf:Description>
1265    <rdf:Description>
1266        <rdf:type rdf:resource="&owl;AllDifferent"/>
1267        <owl:distinctMembers rdf:parseType="Collection">
1268            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1269            <rdf:Description rdf:about="#detail_design_delivery"/>
1270        </owl:distinctMembers>
1271    </rdf:Description>
1272    <rdf:Description>
1273        <rdf:type rdf:resource="&owl;AllDifferent"/>
1274        <owl:distinctMembers rdf:parseType="Collection">
1275            <rdf:Description rdf:about="#user_requirements_delivery"/>
1276            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1277        </owl:distinctMembers>
1278    </rdf:Description>
1279    <rdf:Description>
1280        <rdf:type rdf:resource="&owl;AllDifferent"/>
1281        <owl:distinctMembers rdf:parseType="Collection">
1282            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1283            <rdf:Description rdf:about="#user_requirements_delivery"/>
1284        </owl:distinctMembers>
1285    </rdf:Description>
1286    <rdf:Description>
1287        <rdf:type rdf:resource="&owl;AllDifferent"/>
1288        <owl:distinctMembers rdf:parseType="Collection">
1289            <rdf:Description rdf:about="#user_requirements_delivery"/>
1290            <rdf:Description rdf:about="#context_description_delivery"/>
1291        </owl:distinctMembers>
1292    </rdf:Description>
1293    <rdf:Description>
1294        <rdf:type rdf:resource="&owl;AllDifferent"/>
1295        <owl:distinctMembers rdf:parseType="Collection">
1296            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1297            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1298        </owl:distinctMembers>
1299    </rdf:Description>
```

```
1300    <rdf:Description>
1301        <rdf:type rdf:resource="&owl;AllDifferent"/>
1302        <owl:distinctMembers rdf:parseType="Collection">
1303            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1304            <rdf:Description rdf:about="#high_level_design_delivery"/>
1305        </owl:distinctMembers>
1306    </rdf:Description>
1307    <rdf:Description>
1308        <rdf:type rdf:resource="&owl;AllDifferent"/>
1309        <owl:distinctMembers rdf:parseType="Collection">
1310            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1311            <rdf:Description rdf:about="#context_description_delivery"/>
1312        </owl:distinctMembers>
1313    </rdf:Description>
1314    <rdf:Description>
1315        <rdf:type rdf:resource="&owl;AllDifferent"/>
1316        <owl:distinctMembers rdf:parseType="Collection">
1317            <rdf:Description rdf:about="#user_requirements_delivery"/>
1318            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1319        </owl:distinctMembers>
1320    </rdf:Description>
1321    <rdf:Description>
1322        <rdf:type rdf:resource="&owl;AllDifferent"/>
1323        <owl:distinctMembers rdf:parseType="Collection">
1324            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1325            <rdf:Description rdf:about="#company_standardization_delivery"/>
1326        </owl:distinctMembers>
1327    </rdf:Description>
1328    <rdf:Description>
1329        <rdf:type rdf:resource="&owl;AllDifferent"/>
1330        <owl:distinctMembers rdf:parseType="Collection">
1331            <rdf:Description rdf:about="#company_standardization_delivery"/>
1332            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1333        </owl:distinctMembers>
1334    </rdf:Description>
1335    <rdf:Description>
1336        <rdf:type rdf:resource="&owl;AllDifferent"/>
1337        <owl:distinctMembers rdf:parseType="Collection">
1338            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1339            <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1340        </owl:distinctMembers>
1341    </rdf:Description>
1342    <rdf:Description>
1343        <rdf:type rdf:resource="&owl;AllDifferent"/>
1344        <owl:distinctMembers rdf:parseType="Collection">
1345            <rdf:Description rdf:about="#implementation_delivery"/>
1346            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1347        </owl:distinctMembers>
1348    </rdf:Description>
1349    <rdf:Description>
1350        <rdf:type rdf:resource="&owl;AllDifferent"/>
1351        <owl:distinctMembers rdf:parseType="Collection">
1352            <rdf:Description rdf:about="#company_standardization_delivery"/>
1353            <rdf:Description rdf:about="#detail_design_delivery"/>
1354        </owl:distinctMembers>
1355    </rdf:Description>
1356    <rdf:Description>
1357        <rdf:type rdf:resource="&owl;AllDifferent"/>
1358        <owl:distinctMembers rdf:parseType="Collection">
1359            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1360            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1361        </owl:distinctMembers>
1362    </rdf:Description>
1363    <rdf:Description>
1364        <rdf:type rdf:resource="&owl;AllDifferent"/>
1365        <owl:distinctMembers rdf:parseType="Collection">
```

```
1366            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1367            <rdf:Description rdf:about="#high_level_design_delivery"/>
1368        </owl:distinctMembers>
1369    </rdf:Description>
1370    <rdf:Description>
1371        <rdf:type rdf:resource="&owl;AllDifferent"/>
1372        <owl:distinctMembers rdf:parseType="Collection">
1373            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1374            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1375        </owl:distinctMembers>
1376    </rdf:Description>
1377    <rdf:Description>
1378        <rdf:type rdf:resource="&owl;AllDifferent"/>
1379        <owl:distinctMembers rdf:parseType="Collection">
1380            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1381            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1382        </owl:distinctMembers>
1383    </rdf:Description>
1384    <rdf:Description>
1385        <rdf:type rdf:resource="&owl;AllDifferent"/>
1386        <owl:distinctMembers rdf:parseType="Collection">
1387            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1388            <rdf:Description rdf:about="#industry_standardization_delivery"/>
1389        </owl:distinctMembers>
1390    </rdf:Description>
1391    <rdf:Description>
1392        <rdf:type rdf:resource="&owl;AllDifferent"/>
1393        <owl:distinctMembers rdf:parseType="Collection">
1394            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1395            <rdf:Description rdf:about="#context_description_delivery"/>
1396        </owl:distinctMembers>
1397    </rdf:Description>
1398    <rdf:Description>
1399        <rdf:type rdf:resource="&owl;AllDifferent"/>
1400        <owl:distinctMembers rdf:parseType="Collection">
1401            <rdf:Description rdf:about="#rigorous_automatic_testing_delivery"/>
1402            <rdf:Description rdf:about="#implementation_delivery"/>
1403        </owl:distinctMembers>
1404    </rdf:Description>
1405    <rdf:Description>
1406        <rdf:type rdf:resource="&owl;AllDifferent"/>
1407        <owl:distinctMembers rdf:parseType="Collection">
1408            <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1409            <rdf:Description rdf:about="#context_description_delivery"/>
1410        </owl:distinctMembers>
1411    </rdf:Description>
1412    <rdf:Description>
1413        <rdf:type rdf:resource="&owl;AllDifferent"/>
1414        <owl:distinctMembers rdf:parseType="Collection">
1415            <rdf:Description rdf:about="#regular_automated_testing_delivery"/>
1416            <rdf:Description rdf:about="#context_description_delivery"/>
1417        </owl:distinctMembers>
1418    </rdf:Description>
1419    <rdf:Description>
1420        <rdf:type rdf:resource="&owl;AllDifferent"/>
1421        <owl:distinctMembers rdf:parseType="Collection">
1422            <rdf:Description rdf:about="#user_requirements_delivery"/>
1423            <rdf:Description rdf:about="#company_standardization_delivery"/>
1424        </owl:distinctMembers>
1425    </rdf:Description>
1426    <rdf:Description>
1427        <rdf:type rdf:resource="&owl;AllDifferent"/>
1428        <owl:distinctMembers rdf:parseType="Collection">
1429            <rdf:Description rdf:about="#context_description_delivery"/>
1430            <rdf:Description rdf:about="#detail_design_delivery"/>
1431        </owl:distinctMembers>
```

```
1432      </rdf:Description>
1433      <rdf:Description>
1434          <rdf:type rdf:resource="&owl;AllDifferent"/>
1435          <owl:distinctMembers rdf:parseType="Collection">
1436              <rdf:Description rdf:about="#correctness_proofs_delivery"/>
1437              <rdf:Description rdf:about="#manual_spot_evaluation_delivery"/>
1438          </owl:distinctMembers>
1439      </rdf:Description>
1440 </rdf:RDF>
1441 <!-- Generated by the OWL API (version 2.2.1.1138)
1442 http://owlapi.sourceforge.net -->
```

Listing D.1: An upper ontology in OWL for the product based software certification