



# Towards An Ontology for Software Product Quality Attributes

By

**Ahmad Abdelhafiz Samhan**

Faculty of Information Technology  
Middle East University for Graduate Studies

Supervisors

**Dr. Mohammad A. Al Fayoumi**

Faculty of Information Technology  
Middle East University for Graduate Studies

**Dr. Ahmad K. A. Kayed**

Software Engineering  
Applied Science University

A thesis submitted in partial fulfillment  
of the requirements for the degree of Master of Science  
in Computer Information Systems

Amman, Jordan  
May, 2008

جامعة الشرق الأوسط للدراسات العليا

إقرار تفويض

أنا أحمد عبد الحفيظ سمحان ، أفوض جامعة الشرق الأوسط للدراسات العليا بتزويد نسخ من رسالتي للمكتبات أو المؤسسات أو الهيئات أو الأشخاص عند طلبها حسب التعليمات النافذة في الجامعة.



التوقيع:

٢٠٨ / ٥ / ٢٠٠٨

التاريخ:

**Middle East University for Graduate Studies  
Authorization Statement**

I, **Ahmad Abdelhafiz Samhan**, authorize the Middle East University for Graduate Studies to supply copies of my thesis to libraries, establishments or individuals upon their request, according to the university regulations.

Signature:



Date:

21/5/2008

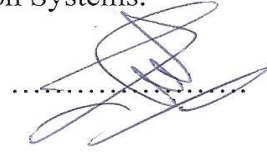
## Committee Decision

This Thesis (Towards an Ontology for Software Product Quality Attributes) was successfully defended and approved on May 21<sup>st</sup> 2008.

### Examination Committee Signatures

**Dr. Mohammad A. Al Fayoumi**

Associate Professor, Department of Computer Information Systems.  
Middle East University for Graduate Studies.



**Dr. Musbah Aqel**

Associate Professor, Department of Computer Systems.  
Middle East University for Graduate Studies.



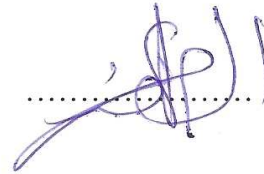
**Dr. Ahmad Kayed**

Assistant Professor, Department of Software Engineering.  
Applied Science University.



**Dr. Jehad Al-Sa'di**

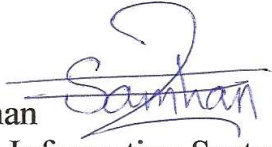
Associate Professor, Department of Computer Science.  
Arab Open University.



## Declaration

I do hereby declare the present work has been carried out by me under the supervision of Dr. Mohammad Al Fayoumi and Dr. Ahmad Kayed and this work has not been submitted elsewhere for any other degree, or any other similar title.

Date: 21/5/2008

Ahmad Abdelhafiz Samhan   
Department of Computer Information Systems  
Faculty of Information Technology  
Middle East University of Graduate Studies

## DEDICATION

*I dedicate this work to my Father, Mother, Brothers, and Sisters, for their love and Support, they were the light in my academic path and without them nothing of this would have been possible.*

*I also dedicate this work to my Father, Mother, Brothers, and Sister in Law for their kind and patience.*

*And a special dedication to the rose of my life, my fiancée Suhair for her love, patience, and support.*

## ACKNOWLEDGMENT

I would like to express my sincere appreciation to Dr. Mohammad Al-Fayoumi and Dr. Ahmad Kayed for their guidance, support and motivation through out my Master's Thesis.

I would further like to acknowledge all of the Information Technology faculty members at the Middle East University for Graduate Studies and I am particularly grateful to Professor Dr. Mohammad Alhaj Hassan for helping and encouraging my efforts during the thesis work.

Also, I would like to especially thank my brother Mohammad Samhan for supporting me during the whole time of my studying and research.

Thank you All.

## CONTENTS

<b>AUTHORIZATION FORM.....</b>	<b>I</b>
<b>COMMITTEE DECISION.....</b>	<b>II</b>
<b>DECLARATION.....</b>	<b>III</b>
<b>DEDICATION.....</b>	<b>IV</b>
<b>ACKNOWLEDGMENT.....</b>	<b>V</b>
<b>LIST OF FIGURES.....</b>	<b>IX</b>
<b>LIST OF TABLES.....</b>	<b>XI</b>
<b>GLOSSARY OF ACRONYMS.....</b>	<b>XII</b>
<b>ABSTRACT.....</b>	<b>XIII</b>
<b>ABSTRACT IN ARABIC.....</b>	<b>XIV</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 OVERVIEW.....	1
1.1.1 THE DISCIPLINE OF SOFTWARE ENGINEERING.....	1
1.1.2 SCOPE OF SOFTWARE QUALITY.....	3
1.1.3 THE ROLE OF ONTOLOGY.....	5
1.1.3.1 WHAT IS AN ONTOLOGY.....	5
1.1.3.2 WHY DEVELOP AN ONTOLOGY.....	5
1.1.3.3 WHAT IS IN AN ONTOLOGY.....	7
1.1.3.4 LEVELS OF ONTOLOGY.....	7
1.1.3.5 ONTOLOGY DEVELOPMENT PROCESS AND LIFE CYCLE.....	8
1.2 PROBLEM DEFINITION.....	10
1.3 OBJECTIVES.....	11
1.4 MOTIVATION OF THE STUDY.....	12
1.5 REQUIREMENTS.....	12
1.6 SIGNIFICANCE OF THE STUDY.....	12
1.7 CONTRIBUTION OF THE THESIS.....	13

1.8	METHODOLOGY.....	14
1.9	SOFTWARE USED IN THE WORK.....	15
1.9.1	KAON.....	15
1.9.2	TextToOnto.....	16
1.9.3	MS ACCESS AND MS VISIUAL BASIC TOOLS.....	18
1.10	HESIS ORGANIZATION.....	18
<b>CHAPTER 2: RELATED WORK .....</b>		<b>20</b>
2.1	GENERAL RELATED WORK.....	20
2.2	SPECIFIC RELATED WORK.....	23
<b>CHAPTER 3: BUILDING ONTOLOGY DOMAIN CONCEPTS.....</b>		<b>43</b>
3.1	PARING TEXT CORPORA FOR SOFTWARE PRODUCT QUALITY ATTRIBUTES DOMAIN.....	43
3.2	EXTRACTING ONTOLOGY DOMAIN CONCEPTS.....	44
3.3	ONTOLOGY EVALUATION PROCESS.....	49
3.4	THE COVERAGE PROCESS AND THE EVALUATION RESULTS.....	54
3.5	ENHANCING ONTOLOGY DOMAIN CONCEPTS.....	57
3.5.1	THE ENHANCING IDEA AND PROCESS.....	57
3.5.2	EVALUATING THE NEW SUGGESTED ONTOLOGY DOMAIN CONCEPTS.....	59
<b>CHAPTER 4: EXTRACTING ONTOLOGY DOMAIN RELATIONSHIPS .....</b>		<b>62</b>
4.1	EXTRACTING RELATIONSHIPS BETWEEN CONCEPTS .....	62
4.2	A LATTICE REPRESENTATION OF THE RELATIONSHIPS.....	67
4.3	LISTING EACH CONCEPT RELATIONSHIPS WITH OTHERS IN THE ONTOLOGY DOMAIN.....	71



<b>CHAPTER 5: CONCLUSIONS AND FUTURE WORK.....</b>	<b>76</b>
5.1 CONCLUSIONS.....	76
5.1.1 PRESENTING FINAL RESULTS.....	77
5.1.2 OUR CONTRIBUTION.....	77
5.2 FUTURE WORK.....	78
<b>REFERENCES.....</b>	<b>80</b>
<b>APPENDICES.....</b>	<b>89</b>
APPENDIX A.....	89
APPENDIX B.....	119
APPENDIX C.....	140
APPENDIX D.....	141
APPENDIX E.....	143
APPENDIX F.....	161

## LIST OF FIGURES

<b>Figure 1.1:</b> Methontology ontology development process life cycle.....	8
<b>Figure 1.2:</b> Steps of the methodology of the study.....	15
<b>Figure 1.3:</b> An overview of the KAON Tool Suite and its main components; KAON, KAON Extensions and TextToOnto.....	16
<b>Figure 1.4:</b> The Front-end of the TextToOnto tool as an extension of KAON tool.....	17
<b>Figure 1.5:</b> Part of the algorithm used in the Ms Visual Tool.....	18
<b>Figure 2.1:</b> The McCall quality model, organized around three types quality characteristics.....	24
<b>Figure 2.2:</b> McCall's Quality Model illustrated through a hierarchy of 11 quality factors (on the left hand side of the Figure) related to 23 quality criteria (on the right hand side of the Figure).....	25
<b>Figure 2.3:</b> Boehm's Software Quality Characteristics Tree.....	27
<b>Figure 2.4:</b> Principles of Dromey's Quality Model.....	30
<b>Figure 2.5:</b> The ISO 9000:2000 standards. The crosses and arrows indicate changes made from the older ISO 9000 standard to the new ISO 9000:2000 standard.....	31
<b>Figure 2.6:</b> The ISO 9126 quality model.....	32
<b>Figure 2.7:</b> ISO 9126: Software Product Evaluation: Quality Characteristics and Guidelines for their use.....	34
<b>Figure 2.8:</b> Maturity Levels of SW-CMM.....	40
<b>Figure 2.9:</b> The staged CMMI-SW/SW representation.....	41
<b>Figure 2.10:</b> The continuous CMMI-SW/SW representation.....	42
<b>Figure 3.1:</b> Creating a Corpus using TextToOnto Tool.....	44
<b>Figure 4.1:</b> Part of the resulted relationships using TextToOnto tool.....	62
<b>Figure 4.2:</b> Using TextToOnto results as an input for the second MS Access tool.....	63
<b>Figure 4.3:</b> Part of the resulted relationships groups from using the second tool.....	64

<b>Figure 4.4:</b> Group 1 relationship Lattice representation: One Level Relationship.....	67
<b>Figure 4.5:</b> Group 2 relationship Lattice representation: Two Levels Relationship.....	68
<b>Figure 4.6:</b> Group 7 relationship Lattice representation: One Level Relationship.....	68
<b>Figure 4.7:</b> Group 10 relationship Lattice representation: a One Level Relationship.....	69
<b>Figure 4.8:</b> Group 22 relationship Lattice representation: Two Level Relationship.....	69
<b>Figure 4.9:</b> Group 28 relationship Lattice representation: Two Level Relationship.....	70
<b>Figure 4.10:</b> Group 33 relationship Lattice representation: Three Level Relationship.....	70

## LIST OF TABLES

<b>Table 2.1:</b> Comparison between criteria/goals of the McCall and Boehm quality models.....	28
<b>Table 2.2:</b> Comparison between criteria/goals of the McCall, Boehm and ISO 9126 quality models.....	33
<b>Table 2.3:</b> Maturity levels with corresponding focus and key process areas for CMM.....	40
<b>Table 3.1:</b> the resulted 292 concepts (out of 2750).....	46
<b>Table 3.2:</b> The suggested 100 ontology domain concepts.....	48
<b>Table 3.3:</b> SWPQAs and their definitions from various sources references.....	50
<b>Table 3.4:</b> Coverage Process Results.....	54
<b>Table 3.5:</b> The top listed 25 uncovered chosen concepts.....	58
<b>Table 3.6:</b> The new 125 Ontology domain concepts list.....	58
<b>Table 3.7:</b> The Coverage process results using the new suggested Ontology domain concepts.....	59
<b>Table 4.1:</b> The resulted relationships between groups of our Ontology concepts after filtering.....	64
<b>Table 4.2:</b> Each Ontology domain concept relationships with other concepts in the domain.....	71
<b>Table A.1:</b> The complete common SWQPAs extracted from different sources and their definitions.....	89
<b>Table B.1:</b> The complete results from the ontology evaluation step.....	119
<b>Table C.1:</b> The final suggested ontology domain concepts list.....	140
<b>Table D.1:</b> Relationships between groups of concepts in the ontology domain.....	141
<b>Table E.1:</b> Each SWPQA definition concepts from our ontology domain concepts.....	143
<b>Table F.1:</b> Each concept relationships with others in the ontology domain concepts.....	161

## GLOSSARY OF ACRONYMS

<b>API</b>	Application Programming Interface.
<b>BWW Model</b>	Bung_Wand_Weber Model.
<b>CMM I</b>	Capability Maturity Model Integration.
<b>DoD</b>	Department of Defense.
<b>FURPS</b>	Functionality, Usability, Reliability, Performance, Supportability.
<b>GQM</b>	Goal, Question, Metric.
<b>IBM</b>	International Business Machines Corporation.
<b>IEEE</b>	Institute of Electrical and Electronics Engineers.
<b>IEEE/EIA</b>	Institute of Electrical and Electronics Engineers/Electronic Industries Association.
<b>ISO/IEC</b>	International Standards Organization / International Electrotechnical Commission.
<b>ISO-JTC1</b>	International Standards Organization/ Joint Technical Committee 1
<b>KAON</b>	KARlsruhe ONtology.
<b>MTBF</b>	Mean Time Between Failures.
<b>NATO</b>	North Atlantic Treaty Organization.
<b>PSM</b>	Practical Software Measurement.
<b>QA</b>	Quality Assurance.
<b>RDF</b>	Resource Description Framework.
<b>SEI</b>	Software Engineering Institute.
<b>SPICE</b>	Software Process Improvement and Capability dEtermination.
<b>SQA</b>	Software Quality Assurance.
<b>SQuaRE</b>	Software Product Quality Requirements and Evaluation.
<b>SWE</b>	Software Engineering.
<b>SWQ</b>	Software Quality.
<b>SWPQAs</b>	Software Product Quality Attributes.
<b>UML</b>	Unified Modeling Language.
<b>VIM</b>	International Vocabulary of Basic and General Terms.
<b>W3C</b>	World Wide Web Consortium.
<b>XML</b>	Extensible Markup Language.

## ABSTRACT

### **Towards an Ontology for Software Product Quality Attributes**

**By**

**Ahmad AbdelHafiz Samhan**

**Supervisors**

**Dr. Mohammad A. Al Fayoumi**

**Dr. Ahmad K. A. Kayed**

This work focuses on studying the most common Software Product Quality Attributes (SWPQAs) concepts and terminologies that current SWPQAs proposals present, in order to extract a conceptualization for the SWPQAs domain. We collected and studied many documents and reports that discussed SWPQAs in their contents, we extracted, studied, evaluated, and enhanced an ontology domain concepts from the most common concepts used in the semantic of the collected documents. Later we extracted and presented general relationships between the suggested ontology concepts. Those presented concepts along with the extracted relationships are introduced as an ontology that is considered as a first in the specific domain of SWPQAs. We condensed the thousands of concepts used to define the most common discussed and studied SWPQAs into a smaller set of concepts consists of 125 concepts, with a coverage percentage for the studied domain of 80%, this presented ontology can be used by software engineers, researchers, practitioners, and stakeholders as a common agreement of SWPQAs pool of knowledge in order to solve the inconsistency problem in the semantic between them while defining or using any of the definitions of the discussed SWPQAs. In addition to this, our ontology provides a base to evaluate any related presented definition semantic for one of the studied attributes.

#### **Key Words:**

Conceptualization, ontology, Software Quality Attributes, Semantic inconsistency, Relationship Lattice, ontology Text Corpus. Coverage Technique, ontology Evaluation.

## ملخص

### نحو بناء أنتولوجي لخصائص جودة البرمجيات

إعداد

أحمد عبد الحفيظ سمحان

إشراف

الدكتور محمد الفيومي

الدكتور أحمد الكايد

موضوع رسالتنا يركز على دراسة مفاهيم ومصطلحات الصفات التي تعبر عن جودة البرمجيات التي تتناولها الأبحاث في هذا المجال لاستخلاص تصوّر دلالي عام منها. لقد قمنا بتجميع ودراسة العديد من التقارير والوثائق التي ناقشت في مضمونها خصائص جودة البرمجيات، قمنا باستخلاص ودراسة وتقييم وتحسين للمفاهيم وتقديمها كمفاهيم مقترحة للأنتولوجي المقترح من خلال دراسة المفاهيم الشائعة والمستخدمه في التقارير والوثائق المجمعّة. بعد ذلك قمنا باستخلاص ودراسة وتقديم علاقات عامة بين مفاهيم الأنتولوجي المقترح. هذه المفاهيم المقدمة بالإضافة للعلاقات العامة المقترحة بينها فُدمت كأول أنتولوجي في مجال خصائص جودة البرمجيات. لقد قمنا بتلخيص آلاف المفاهيم المستخدمة في تعريف خصائص جودة البرمجيات المتداولة في الدراسة والنقاش إلى مجموعة أصغر من المفاهيم تتكون من 125 مفهوم ، وبنسبة تغطية للمفاهيم المستخدمة في تعريفات خصائص جودة البرمجيات بما يقارب 80% ، هذه الأنتولوجي المقدمة يمكن أن تستخدم من قبل لمهندسي البرمجيات ،الباحثين ، الممارسين ، وأي شخص له علاقة بها بحسب استخدامها كمنبع معرفة متفق عليه بالإجماع، الأمر الذي يحل مشكلة عدم الإجماع على معاني المفاهيم الدلالية المستخدمة في تعريف أي من الصفات المدروسة التي تعبر عن جودة البرمجيات. بالإضافة لذلك فإن الأنتولوجي المقدم ممكن أن يستخدم كأساس لتقييم أي تعريف تم تقديمه لخصائص جودة البرمجيات المدروسة.

# CHAPTER 1

## INTRODUCTION

This chapter reviews the thesis. A brief background about the field to which the thesis subject belongs is given; Software Engineering (SWE), Software Quality (SWQ), and the field of ontology and its role. Then we give an idea about our research problem and how it has been addressed. We end the chapter by giving information about tools used in the work, our own contribution, and the outline of the thesis chapters.

### 1.1 OVERVIEW

Recently, Quality Assurance (QA) concept has been widely developed to be included in many of our life existing fields; financial, industrial, trading, etc. Software Quality Attributes (SWQAs) have been created as a matter of applying the QA concept on the results of software development process, to fit the products with the organizational and global market standards and goals and to provide them with a competitive advantage value. Software quality is composed of many attributes such as portability, usability, reliability, modularity, and other software quality related attributes.

During the last years, many researchers (individuals and groups) discussed and presented software attributes in their works which show that till now there is a lack of consensus on the semantic of many of concepts and terminologies used in the field of SWQAs. According to this and in more specific our research is focusing on studying the most common SWPQAs concepts and terminologies that current SWQAs proposals present to extract a conceptualization for the SWQAs, after that we will study this conceptualization in order to build an ontology that produce a coherent and consistent semantics for SWQAs concepts and terminologies that can be used by SW engineers, researchers, practitioners, and stakeholders as a common agreement of SWQAs pool of knowledge. Before defining my research problem, a brief introduction to the related fields of this research is given.

#### 1.1.1 THE DISCIPLINE OF SOFTWARE ENGINEERING

Since the dawn of computing in the 1940s, the use of computer software has been rising enormously. Nowadays, computer software play many important roles, and considered as a way for delivering a product as it they are the basis of controlling operating systems, networks, and other applications, and also considered as products themselves [97]. They serve the human kind in almost all of the fields of his life; government, banking and finance, education, transportation, entertainment, medicine, agriculture, and law [106].



Computer software is a general term used to describe a collection of computer programs, procedures, and documentations that perform some tasks on a computer system [120]. The increasingly development of science and technology makes the need for software an important issue especially for software products that is typically a single application or suite of applications built by a software individuals/companies to be used by many customers, businesses or consumers [21].

Software products are categorized under two major types, generic products; which are stand alone products developed to be used by any customer in the market, and customized products; which are developed especially to a customer or to a group of customers [111].

The evolving of software development makes developers take a more systematic and planned way to develop their software products, Software Engineering revealed in order to help developers to do so. The IEEE Computer Society defines software engineering as: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software [54].

The term software engineering first appeared in late 1950s and early 1960s. Programmers have always known about civil, electrical, and computer engineering and debated what engineering might mean for software. The NATO Science Committee sponsored two conferences on software engineering in 1968 and 1969 [99], which gave the field its initial boost. Many believe that these conferences marked the official start of the profession of software engineering.

At the early decades of software engineering revealing, it was motivated to face the Software Crisis problem appeared at that time, researchers and practitioners tried every possible way to solve this crisis (Cost and budget overrun, property damage, and software life and death), In 1987, Fred Brooks published the No Silver Bullet [18] article, arguing that no individual technology or practice would ever make a 10-fold improvement in productivity within 10 years.

Software engineering had been widely affected by the appearance of the Internet, programmers and developers were required to deal with many new revealed issues and merge it within their developed software (images, maps, animations, web browsers usage, etc). Simpler and faster methodologies that developed running and inexpensive software products have been introduced to small organizations in order to satisfy their demands, some of these methodologies are: Rapid prototyping, Agile development, Extreme programming, and others [99].

The need for computer software has grown dramatically, thousands of billions of dollars are spent on the development of computer software. Software products provide us with a more productive, safer, and flexible working environment to help us to be more successful, accurate, trustable, efficient, and productive [109]. Despite these successes, Computer Software and Software Engineering face many key challenges such as

heterogeneity challenge, delivery challenge, trust challenge, cost challenge, timelines challenge, and quality challenge [111]. Researchers and practitioners are continuously searching to solve these challenges, they solved some of them, and searching to solve others. However this is a good characteristic of the evolving Software Engineering discipline.

### 1.1.2 SCOPE OF SOFTWARE QUALITY

Computer hardware and software are widely distributed and used in modern society. The evolving manner of business, hardware and technology, the appearance of the world wide web, and other revealing factors make users need individuals and interconnected computers, as well as sharing and exchanging information using a global information structure, processing algorithms and techniques, storage capacity and allocation dealing, data search and retrieval methodologies, all these needs and more are being met with the support of software.

This important role of software increases the dependability of human kind on them especially because they are used in so many fields of his life. Because of that developers are working hard to ensure not to be failed by their developed software by producing reliable and trusted software.

To be trusted and reliable, software must have some features and characteristics that satisfy what the customers want, this leads us to the quality world. The term quality is one of the most discussed terms these days. All of the researchers agree that quality is considered as a key business factor, as a matter of fact they considered not including it will compromise the business. Here are some of quality definitions as presented by some of quality specialists:

- Quality: The totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs [11].
- Dr. Barry Boehm [14] thinks of quality as: "Achieving high levels of user satisfaction, portability, maintainability, robustness, and fitness for use".
- Watts Humphery [47], of the software engineering institute, presented the quality as: "Achieving excellent levels of fitness for use, conformance to requirements, reliability and maintainability".
- Bill Perry [47], head of quality assurance institute has defined quality as: "High levels of user satisfaction and adherence to requirements".

Quality is a general term that can be applied on mostly anything in any field. As a consequence of that, quality is applied more specifically on software products under the term of software quality or software quality assurance.

Conformance to specifications and meeting customer's needs are two major corners when discussing the definition of software quality [10], which is defined as a planned and systematic set of activities built into software to ensure its quality. It consists of software quality assurance, software quality control, and software quality engineering.

As an attribute, software quality is defined as (1) The degree to which a system, component, or process meets specified requirements; or (2) The degree to which a system, component, or process meets customer or user needs or expectations [54].

The aimed features of quality differ from customer to customer. Factors differ upon the required requirements of the system. Quality of computer software must be planned from the beginning of software developing. So, it can not be existed at the moment it needed without planning, it must be kept in mind in every phase of software development. Before all of that, quality goals and attributes must be clearly defined, effectively monitored, and rigorously enforced.

As a consequence of the need for a planned quality the terminology of software quality assurance has appeared, ESA PSS-05-0 defines software quality assurance (SWQA) as a 'Planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements' [37]. SWQA does this by checking that [32]:

- Plans are defined according to standards.
- Procedures are performed according to plans.

Customers demand quality in the applications they use, and without making the customers happy from the software they own or use, the business will not survive. There are several reasons why business should be concerned with quality [42]:

- Quality is a competitive issue now.
- Quality is a must for survival.
- Quality gives you the global reach.
- Quality is cost effective.
- Quality helps retain customers and increase profits.
- Quality is a hallmark of world-class business.

As we mentioned earlier, software quality assurance came as a planned and systematic pattern to ensure the existence of quality features to be in the developed product.

### 1.1.3 THE ROLE OF ONTOLOGY

*“What’s the use of their having names” the Gnat said, if they won’t answer to them?”  
“No us to THEM,” said Alice, “but it’s useful to the people who name them, I suppose, if not, why do things have names at all?”*

*By Lewis Carroll, through the looking Glass.*

#### 1.1.3.1 WHAT IS AN ONTOLOGY

ontology as seen from philosophical perspective is the science of studying beings (studying of what is, of the kinds and structures of objects, properties, events, processes and relationships in every area of reality), this term which was coined in 1613 included in many philosophical areas from the metaphysics of Aristotle to the object-theory of Alexius Meinong [108].

Philosophical ontology handles the precise utilization of words as descriptors of entities; it gives an account for those words that belong to entities and those that do not [35]. In both Computer Science and Information Science, an ontology is a representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and used to define the domain [14].

#### 1.1.3.2 WHY DEVELOP AN ONTOLOGY

Recently, the term ontology has widely included in the field of computer and information science. When building frameworks for information representation of data and knowledge base systems, designers use a wide variety of terms and concepts. Studies showed that there is an inconsistency problem in the semantic of the terms that are used, e.g. identical databases labels are used but with different meanings, and also the same meaning expressed using different names. Methods must be found to resolve the terminological and conceptual incompatibilities [107]. An ontology in this context is a dictionary of terms formulated in a canonical syntax and with commonly accepted definitions designed to yield a lexical or taxonomical framework for knowledge-representation which can be shared by different information systems communities [107].

Ontologies are used in variety of current fields; Artificial intelligence [45], Software engineering [92], the Semantic web [89], Biomedical informatics [4] , Library science [45], Information architecture [92], Ecommerce content standard [46], and other fields , as a form of knowledge representation about the domain or some part of it.

In this era the presence of consistent global information has become an important issue. In every domain researchers and practitioners need to share information to conduct their works in a professional manner. To do that in a correct way inconsistencies between terms and concepts must be reduced. Ontology defines a common vocabulary for them; it contains machine-interpretable definitions of basic concepts in the domain and relationships among them.

From studying the role of ontologies in different knowledge domains, many studies showed that creating and developing and also enhancing ontologies has become important to many fields and areas of domain knowledge, because of its approved advantages effects when using them in the field of the studied knowledge domain.

Many reasons support our recommendation of creating, developing, and using ontologies, some of them are:

- Ontologies support applications (especially distributed ones) to exchange information and to process transactions independently [65].
- Ontologies make the reusing of a domain knowledge possible [31].
- Ontologies provide semantic-aware information systems, which can support enterprise, government, and personal activities at the same time [31].
- Ontologies can share different applications [90].
- Ontologies can use other ontologies [90].
- Ontologies can analyze, support, and enhance domain knowledge [90].
- Ontologies are used as a semantic support representation for many areas [65].
- Ontologies are used to capture the domain information independently of any application requirements [66].

Ontology shows enormous potential in making software more efficient, adaptive, and intelligent. It is recognized as one of the areas which will bring the next breakthrough in software development. The idea of ontology has been welcomed by visionaries and early adopters.

Since 1991, the semantic Web initiative, lead by W3C, has changed the ontology landscape completely, through the initiative, researchers and developers join forces to provide standard semantics markup languages based on XML, ontology management systems, and other useful tools. Also, the Web provides interesting applications of ontology that are critical to daily life such as search and navigation. In addition, people rediscover the value of ontology in other important applications such as information and process integration [73, 74].

### 1.1.3.3 WHAT IS IN AN ONTOLOGY

Different knowledge representation formalisms and corresponding languages exist for the formalization and implementation of ontologies. Each of them provides different components that can be used for these tasks. However, they share the following minimal set of components [20]:

- Classes represent concepts, which are taken in a broad sense. Classes in the ontology are usually organized in taxonomies through which inheritance mechanisms can be applied.
- Relationships represent a type of association between concepts of the domain. They are formally defined as any subset of a product of  $n$  sets, that is:  $R = C1 \times C2 \times \dots \times Cn$ . ontologies usually contain binary relationships. The first argument is known as the domain of the relationship, and the second argument is the range.
- Instances are used to represent elements or individuals in an ontology.

Ontology is an essential data structure for conceptualizing knowledge [117]. It is commonly used as a fundamental structure for capturing knowledge by analyzing relevant concepts and relationships in the area under search [86]. It depends mostly on the analysis of textual data over a collection of text documents by using natural language processing to do that and more such as obtaining semantic graph of a document; visualization of documents; information extraction to find relevant concepts; and visualization of context of named entities in a document collection [117].

### 1.1.3.4 LEVELS OF ONTOLOGY

Different authors like P'erez, Jones, Storre, Robert, Malka, and others have organized ontology in their studies and reports into different levels [63, 90, 94, 95, and 100]:

- Lexical, vocabulary, or data layer. The focus here is on concepts, facts, etc. that ontology included, and the vocabulary used to represent these concepts.
- Hierarchy or taxonomy. An ontology typically includes a hierarchical is-a relationships, or subsumption relationships between concepts.
- Other semantic relationships. The ontology may contain other relationships besides is-a relationship. This typically includes measures such as precision and recall.
- Context level. Ontology may be a part of a larger collection of ontologies. Another form of context is the application where the ontology is to be used

- Syntactic level. The ontology is usually described in a particular formal language and must match the syntactic requirements of that language (use of the correct keywords, etc.). Various other syntactic considerations, such as the presence of natural-language documentation, avoiding loops between definitions, etc., may also be considered.
- Structure, architecture, design. Unlike the first three levels on this list, which focus on the actual sets of concepts, instances, relationships, etc. involved in the ontology, this level focuses on higher-level design decisions that were used during the development of the ontology. This is primarily of interest in manually constructed ontologies. For some applications, it is also important that the formal definitions and statements of the ontology are accompanied by appropriate natural-language documentation, which must be meaningful, coherent, up-to-date and consistent with the formal definitions, sufficiently detailed, etc.

Let us don't forget that ontologies have been applied and played an important role in different areas of Software Engineering fields as they do in other disciplines. They provide a general framework reference of an agreed concepts and terminologies among researchers, practitioners, and stakeholders, they enhance collaboration, communication, and knowledge sharing, they represent all assumptions related to the entities and relationships between them that belong to the area under search, and finally they contribute in reducing gabs between researchers, etc created by conceptual confusion [104,106,123]. Hence, building an ontology to capture the conceptualization knowledge about Software Quality Attributes domain will achieve a significant successful solution for the semantic conflicts problem the field suffers from.

### 1.1.3.5 ONTOLOGY DEVELOPMENT PROCESS AND LIFE CYCLE

The ontology development process refers to the activities that are performed when building ontologies. It identifies three categories of activities as shown in Figure 1.1.

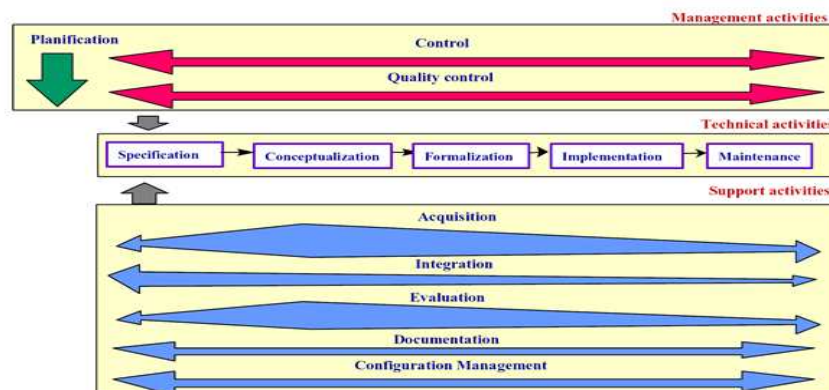


Figure 1.1: Methontology ontology development process life cycle [22].

a) **Ontology management activities:**

The management process activities are responsible for the project management issues [22, 24, 39].

1. **Scheduling** is the first activity of the ontology life cycle. The objective is to plan the main tasks to be done, how they will be arranged and the required resources, i.e. people, software and hardware.
2. **Control** is performed along the whole ontology life cycle in order to survey that there are not undesired deviations from the initial schedule.
3. **Quality** is responsible for checking that the quality of each methodology output (ontology, software and documentation) is assured.

b) **Development Process:**

The development process includes all the activities that produce the successive prototype refinement stages towards the desired ontology. The process starts with specification that produces an informal output that then evolves increasing its level of formality, as it passes through the different activities, towards the final computable model, which can be directly understood by the machine [22, 24, 39]. It consists of:

1. **Specification:** The specification establishes the ontology purpose and scope. Why the ontology is being built, what are the intended uses and end-users. The specification can be informal, in natural language, or formal, e.g. using a set of competence questions.
2. **Conceptualisation:** The objective of this activity is to organize and structure the knowledge acquired during knowledge acquisition using external representations that are independent of the knowledge representation and implementation paradigms in which the ontology will be formalised and implemented next. An informally perceived view of a domain is converted into a semi-formal model using intermediate representations based on tabular and graph notations. These intermediate representations (concept, attribute, relation, axiom and rule) are valuable because they can be understood by domain experts and ontology developers. Therefore, they bridge the gap between people's domain perception and ontology implementation languages.
3. **Formalisation:** The goal of this activity is to formalise the conceptual model. There are ontology development tools that automatically implement the conceptual model into several ontology languages using translators. Therefore, formalisation is not a mandatory activity.
4. **Implementation:** This activity builds computable models using ontology implementation languages. There are many ontology languages and they do not have the same expressiveness nor do they reason the same way.



5. **Maintenance:** This activity updates and corrects the ontology if needed due to the necessities of the current development process or other processes that reuse this ontology in order to build other ontologies or applications.

### c) Support Process:

The support activities are performed in parallel with the development-oriented activities [22, 24, 39].

1. **Knowledge Acquisition:** First of all, the source knowledge must be captured using knowledge elicitation techniques. The sources of knowledge are listed giving a description and specifying the elicitation techniques used in each case. The techniques used to extract knowledge from sources can be partially automatic by means of natural language analysis and machine learning techniques.
2. **Evaluation:** The evaluation activity judges the developed ontologies, software and documentation against a frame of reference. Ontologies should be evaluated before they are used or reused. There are two kinds of evaluation, the technical one, which is carried out by developers, and users' evaluation.
3. **Integration, merging and alignment:** The integration activity is needed if other ontologies are reused. There are two options when an ontology is integrated in the current ontological framework. First, there is ontology alignment that consists in establishing different kinds of mapping between the ontologies, hence preserving the original ontologies. Second, ontology merging that produces a new ontology from the combination of the input ontologies.
4. **Documentation:** Documentation details each completed stage and product.
5. **Configuration Management:** Configuration management records ontologies, software and documentation versions in order to control changes.

## 1.2 PROBLEM DEFINITION

Software quality attributes are one of the key revealed issues that made significant influences on Software Engineering. They play a very important role in evaluating software programs. They are considered by practitioners and researchers to be the key factor for producing high quality competitive software products to the markets which is enforced by the appearance of quality assurance issues.

As a matter of fact many initiatives such as IEEE standard releases, ISO/IEC releases, SPICE (Software Process Improvement and Capability dEtermination), and many quality models such as McCall quality model, Boehm quality model, Dromey quality model, and others, consider software quality attributes to be an important element of reaching a higher maturity levels while developing and managing the quality of software programs [68, 94].

During the last decades many developments in many fields have affected how the business is done, one of the most important affecting issues are the emerging of the Internet and the Globalization, which had a huge effect on the individuals and the organizations business processes. It created a need for sharing information and resources widely as a matter of collaboration to compete efficiently in market. In order to achieve this collaboration standards are created to provide agreed terminologies and practices that make participants avoid inconsistencies in their business [68].

However, researchers in this domain explained that there is no single standard that covers the area of software attributes in its totality, but rather there are many different standards that focus on specific areas. Without a comprehensive framework considered as a reference when managing these diverse standards, inconsistencies arise in the attributes concepts and terminologies [63].

Recently, a lot of efforts from researchers and standards institutes are done to manipulate the symptoms that software quality discipline suffers from as it is believed a young discipline. Software quality attributes concepts, principles, and terminologies are considered by those researchers and institutes to be in a stage that they are still being defined, consolidated, and agreed [68].

One way in regard to reach to a common solution for our introduced problem; software product quality attributes concepts and terminologies inconsistency among the current presented studies and reports, is by representing the conceptualization of the software product quality attributes domain by an ontology in order to reach to an understandable unified semantic framework for software product quality attributes.

So the question that points to itself: Can we condense the thousands of concepts used in the semantic of the most common software product attributes to a smaller set of concepts, and introduce the result as an ontology? Our work is focusing on that. In our thesis we studied and analyzed the presented reports, documents, and proposals concerned with SWPQAs and so we extracted the various concepts, definitions, and terminologies from them, after that a general relationships between the resulted concepts were extracted and introduced as an ontology that aims to produce a coherent and consistent conceptualization framework for SWPQAs to eliminate gabs and terminology conflicts between software engineers, researchers, practitioners, and stakeholders when using it as a common agreement software quality attributes pool of knowledge.

### **1.3 OBJECTIVES**

In this thesis we aim to:

- Extract concepts used in the semantic of the most common discussed SWPQAs.
- Extract general relationships between the extracted concepts.
- Introduce the extracted concepts and relationships as an ontology for the domain of SWPQAs.

- Introduce ways in order to use the provided ontology to solve the semantic inconsistency problem found in the field of SWPQAs.

## 1.4 MOTIVATION OF THE STUDY

During the last years, a lot of efforts from researchers and standards institutes are done to manipulate the symptoms that SWPQAs discipline suffers from as it is believed a young discipline. Software quality attributes concepts, principles, and terminologies are considered by those researchers and institutes to be in a stage that they are still being defined, consolidated, and agreed. Many researchers (individuals and groups) discussed and presented software attributes in their works which show that till now there is a lack of consensus on the semantic of many concepts and terminologies used in this field. This motivated us to do our research in this field, and provide this work which focuses on studying the most common SWPQAs concepts and terminologies that current SWPQAs proposals present, in order to extract a conceptualization for the SWPQAs domain and introduce it as an ontology for the studied field to be used in order to reach to a consistent semantic.

The meaning of consistent as we used in our work includes both generally agreement “consensus” and coherent “without conflicts” meanings. Consistent as defined in the Merriam Webster dictionary means “marked by harmony, regularity, or steady continuity: free from variation or contradiction”.

## 1.5 REQUIREMENTS

In order to reach to efficient results for our work many requirements were needed:

- Hardware device: Personal computer.
- Computer software: Operating system (Windows XP), Java Development Kit 5, Microsoft office suite, KAON tool suit. They will be discussed later.
- Human Experts: Professors, Practitioners in the field of SWE and ontology.

## 1.6 SIGNIFICANCE OF THE STUDY

This thesis aims to address the needs of two main kinds of interested audiences:

- The first kind is the software quality attributes researchers and standard developers (e.g., international standardization institutes and committees), who is responsible for producing concepts, terms, and standards in the field.
- The second kind is the software quality attributes practitioners, who may be confused by the terminology differences and conflicts in the existing standards and proposals when they would use them in their works.

## 1.7 CONTRIBUTION OF THE THESIS

SWPQAs discipline is considered in the emerging phase, and it suffers from the typical symptoms of any relatively evolving disciplines. SWPQAs are currently in the phase in which terminologies, principles, and methods are still being defined, consolidated, and agreed. In particular, there is a lack of consensus on the concepts and terminologies used in the semantic of this field. Studies showed that inconsistencies in the semantic used different research attributes proposals often occur [24, 39].

In our research we focused on studying SWPQAs concepts and terminologies that current SWQ proposals, documents, and reports present. We prepared text corpora from them to be used in tools to extract the most discussed and used concepts from it. After that experts (doctors and professors in the field of SWE and SWQ) were asked to study and filter the resulted concepts and provided them to us.

An evaluation phase depended on a coverage technique was done to the resulted concepts, followed by an enhancing step to the evaluated ontology domain concepts which led us to increase the number of the suggested concepts in the ontology domain, after that a coverage evaluation is done again to the new suggested ontology domain concepts.

In order to extract general relationships among the suggested ontology domain concepts, we returned to the prepared text corpus again and ran out two tools on it. We studied them, filtered them, listed them and represented part of them using a lattice representation.

After we have finished our research steps, and depending on the results we had, we claim that we have presented the conceptualization of the 66 common discussed SWPQAs by an ontology, which is considered as a first in this specific domain.

According to the results of the suggested ontology, we also claim that we condensed the semantic of thousands of concepts used to define any of the discussed SWPQAs into a smaller set of concepts, and that will help experts, software engineers, researchers, practitioners, and stakeholders in the field of SWPQAs to share and use a common and agreed semantic of concepts when defining any of the studied attributes, and that will lead us to resolve the inconsistencies of the semantic appeared among documents and reports that define any of the studied attributes.

In addition to this, our ontology provides a base to evaluate any related presented definition semantic for one of the studied attributes. The way of doing that is if a high percentage of the concepts used in the semantic of the presented definition are covered by our ontology domain, the presented definition semantic can be accepted, but if not we claim that it is a weak semantic to be used to define such an attribute.

## 1.8 METHODOLOGY

This research will be carried out through a theoretical and an empirical study. Our approach to study the problem as shown in Figure 1.2 is divided into 6 parts:

- The first part of this research is a literature review on almost about all existing proposals and ontologies in software quality, with the focus on a specific domain concerning with the software product quality attributes domain. This review presented, discussed, and analyzed different sources for software quality in general and for software product quality attributes in particular, such as researches, reports, documents, and proposals produced by various individuals, institutes, and committees in the field.
- The second part of our work focused on paving the way to capture and extract the ontology domain concepts from the knowledge domain prepared in the first step using some tools. Later a support from experts in the field to study and filter the results was asked.
- The third part of our work handled the evaluation of the resulted ontology domain concepts, by following a technique categorized as a coverage approach in the domain.
- In the fourth part, enhanced results were reached depending on the results of the evaluation step.
- In the fifth part, we captured and extracted general relationships between the suggested ontology domain concepts, by providing the prepared knowledge domain to two tools, after that we studied and filtered the resulted relationships into groups, a general lattice representation to a part of the resulted relationships was constructed.
- In the sixth and the final part, we showed how the results contribute in the domain, and suggested many ways to use them in order to reach to a common, shared, and agreed semantic when defining any of the studied software product quality attributes.

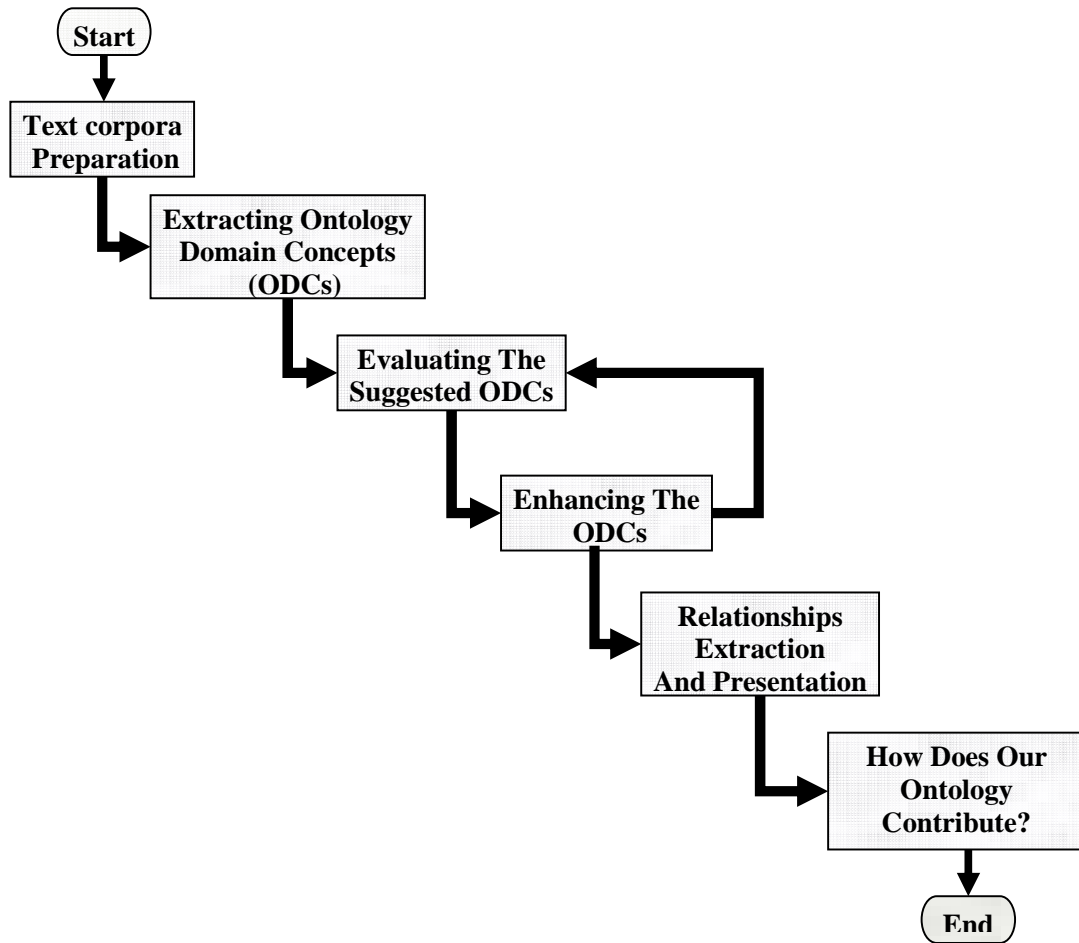


Figure 1.2: Steps of the methodology of the study

## 1.9 SOFTWARE USED IN THE WORK

In our work we used many tools in order to reach to some necessary results, below is a brief description for those tools used in this work:

### 1.9.1 KAON

KAON consists of a number of different modules providing a broad bandwidth of functionalities centered around creation, storage, retrieval, maintenance and application of ontologies. It was and currently is being further developed in a joint effort mainly by members of the Institute of Applied Informatics and Formal Description methods (AIFB) at University of Karlsruhe and the Forschungszentrum Informatik (FZI) – Research Center for Information Technologies, Karlsruhe [79].

The Karlsruhe Ontology [79] and Semantic Web tool suite a.k.a. KAON Tool Suite is an open source ontology management infrastructure. However, there exist also external components which support functionalities such as e.g. ontology learning from texts. An overview of the KAON Tool Suite and its main components; KAON, KAON Extensions and TextToOnto, is presented by Figure 1.3.

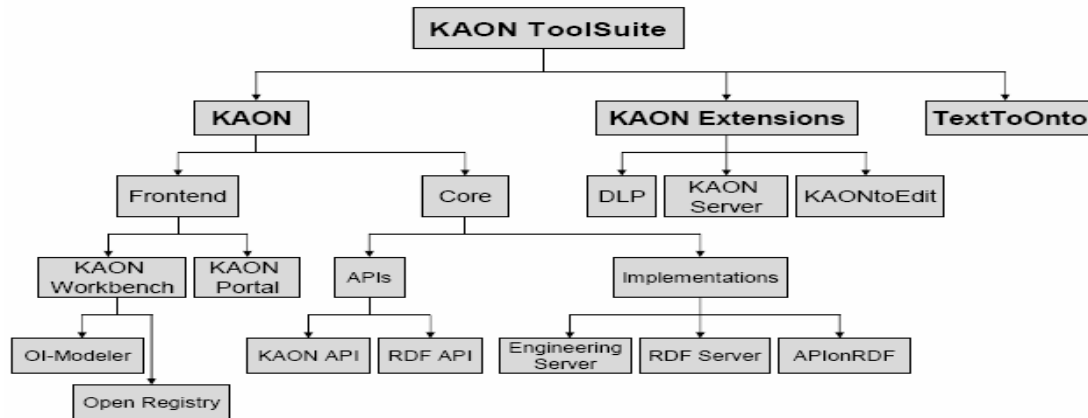


Figure 1.3: An overview of the KAON Tool Suite and its main components; KAON, KAON Extensions and TextToOnto [79].

KAON (consisting of KAON Frontend and KAON Core) includes a variety of different modules for ontology creation and management. The Frontend is represented by two applications developed in order to be used particularly by human users:

- KAON Workbench: provides a graphical environment for ontology based applications. It includes the OI-Modeler – a graphical ontology editor - and the Open Registry (a.k.a. ontology Registry), which provides mechanisms for registering and searching ontologies in a distributed context.
- KAON Portal: is a simple tool for multi-lingual, ontology-based Web portals.

The Core of KAON supports programmatic access to ontologies by including both APIs and implementations for managing local and remote ontology repositories [36].

KAON Extensions are a collection of optional components not included in the standard distribution of KAON [36].

- DLP (Description Logic Programs) supports efficient ontology reasoning by mapping Description Logic into Logic Programs.
- KAON Server can be considered as Application Server for the Semantic Web, which provides a generic infrastructure to facilitate plug'n'play engineering of ontology-based applications.

- KAONtoEdit is a plug-in for OntoEdit [93], which allows working directly on implementations of the KAON API in order to load, modify and store KAON ontology models.
- TextToOnto is a KAON-based tool suite supporting the ontology engineering process by providing a collection of independent tools for ontology learning and maintenance.

In Our work we focused on using the TextToOnto Extension because of its capability to help users to learn about ontologies from a provided text.

## 1.9.2 TextToOnto

TextToOnto [83] is a tool suite built upon KAON in order to support the ontology engineering process by text mining techniques. Providing a collection of independent tools for both automatic and semi-automatic ontology extraction. it assists the user in creating and extending OI-Models. Moreover, efficient support for ontology maintenance is given by modules for ontology pruning and comparison. In particular, the current distribution of TextToOnto comprises the following tools:

- TaxoBuilder: for building concept hierarchies
- TermExtraction: for adding concepts to an ontology
- InstanceExtraction: for adding instances to an ontology
- RelationExtraction: for semi-automatic learning of conceptual relationships
- RelationLearning: for automatic *and* semi-automatic relationship learning
- OntologyComparison: for comparing two ontologies
- OntologyPruner: for adapting an ontology to a domain-specific corpus

Figure 1.4 shows the front-end of the TextToOnto tool as an extension of KAON tool.

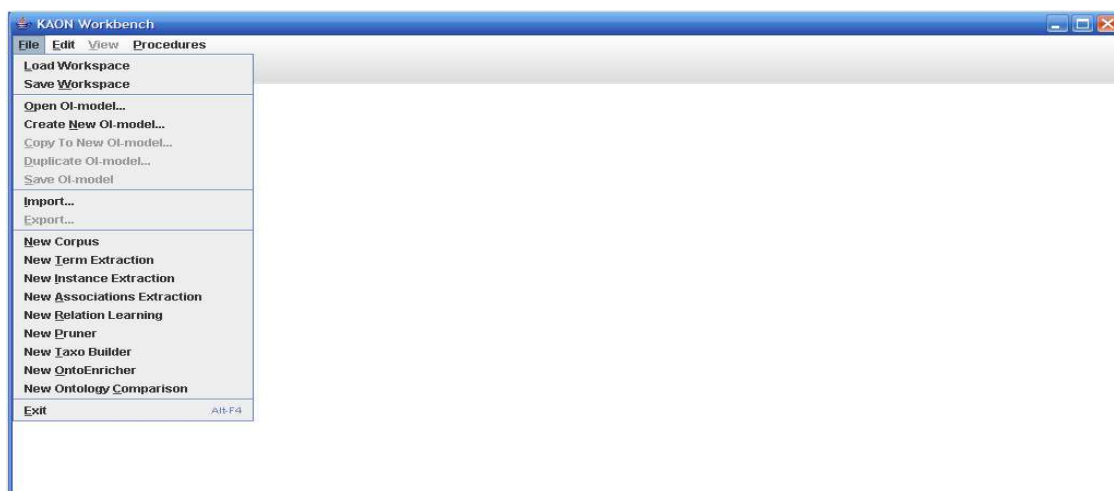
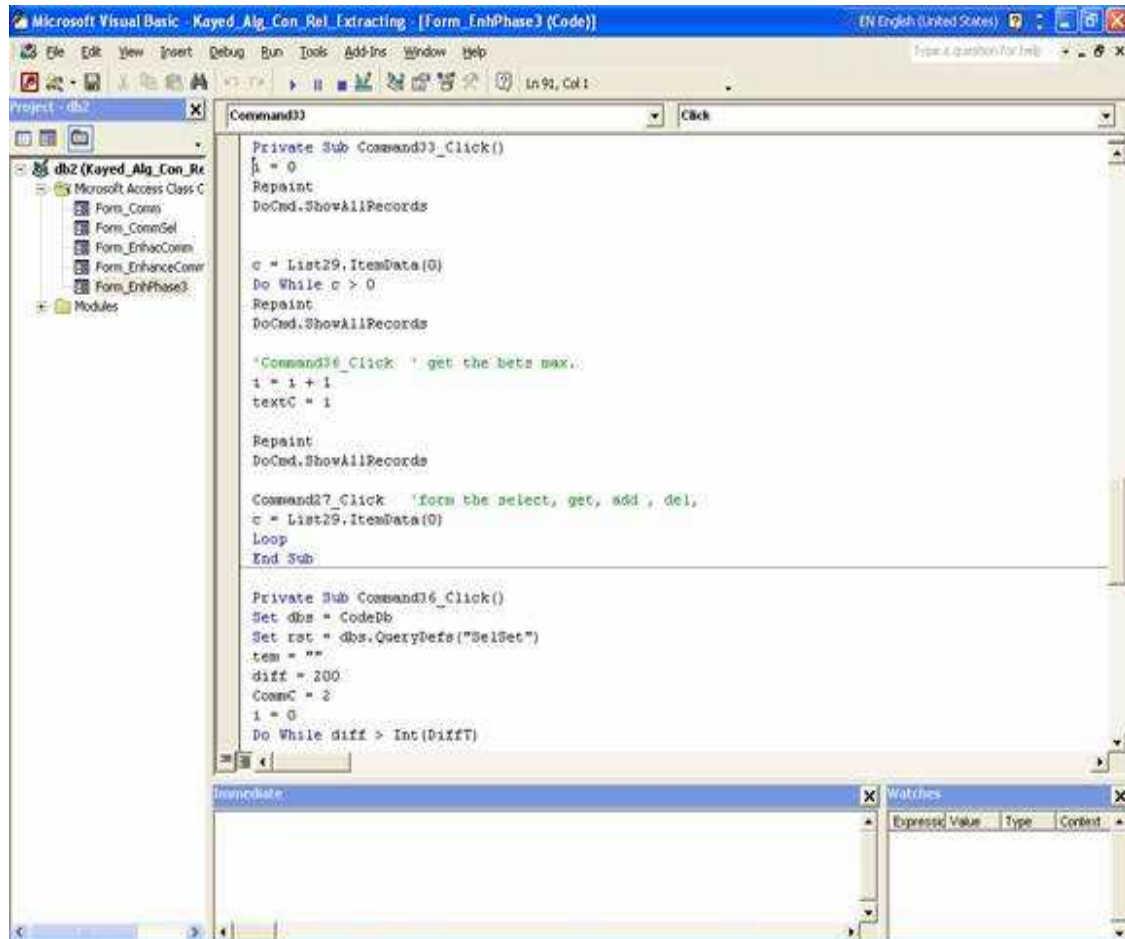


Figure 1.4: The Front-end of the TextToOnto tool as an extension of KAON tool.



### 1.9.3 MS ACCESS AND MS VISUAL BASIC TOOLS

MS Access and MS Visual Basic have been used to implement algorithms. Screen shots of the program are provided in Figure 1.5, for further reading about it you may refer to [71, 72].



```
Microsoft Visual Basic - Kayed Alg. Con. Rel. Extracting: [Form_EnhPhase3] (Code)
EN English (United States)
File Edit View Insert Debug Run Tools Add-Ins Window Help
Project - db2
db2 (Kayed Alg. Con. Rel. Extracting)
  Microsoft Access Class C
  Forms_Comm
  Forms_CommSel
  Forms_EnhanceComm
  Forms_EnhanceCommV
  Forms_EnhPhase3
  Modules
Command33
Private Sub Command33_Click()
    i = 0
    Repaint
    DoCmd.ShowAllRecords

    c = List29.ItemData(0)
    Do While c > 0
        Repaint
        DoCmd.ShowAllRecords

    'Command36_Click: get the bets max.
    i = i + 1
    textC = i

    Repaint
    DoCmd.ShowAllRecords

    Command27_Click 'form the select, get, add, del.
    c = List29.ItemData(0)
    Loop
End Sub

Private Sub Command36_Click()
    Set dbs = CodeDb
    Set rst = dbs.QueryDefs("SelSet")
    tem = ""
    diff = 200
    CoomC = 2
    i = 0
    Do While diff > Int(DiffT)
```

Figure 1.5: Part of the algorithm used in the Ms Visual Tool.

### 1.10 THESIS ORGANIZATION

- This thesis is organized into 5 chapters:

**Chapter 1:** This chapter reviews the thesis. A brief background about the field to which the thesis subject belongs is given; Software Engineering (SWE), Software Quality (SWQ), and the field of ontology and its role. Then we give an idea about our research problem and how it has been addressed. We end the chapter by giving information about tools used in the work, our own contribution, and the outline of the thesis chapters.

**Chapter 2:** In this chapter we give a brief idea about the most relevant work in the literature that are related to our study.

**Chapter 3:** This chapter is talking about the preparation of the text corpora for the SWPQAs knowledge domain. This preparation is done by collecting and studying a large number of documents and reports related to the field of software quality. The chapter also is discussing how the prepared text corpora were used to extract and create our primary ontology domain concepts using TextToOnto tool with support of an MS Access tool and then with support of human experts. In this chapter, we also focused on evaluating the suggested ontology domain concepts using a coverage methodology. After preparing the needed corpus, and by using a tool created by Kayed [72], we counted the covered concepts and calculated a coverage percentage for them. Later, we discussed what the results would be if we collect and study the uncovered concepts from the domain under discussion. The results showed an enhancement of our ontology domain concepts, and gave a much better coverage percentage. The results are shown in the discussion.

**Chapter 4:** In this chapter we extracted general relationships between the new concepts of the suggested ontology domain after studying and filtering the results of two tools. We presented the resulted relationships as groups. After that, a general lattice representation for part of the resulted relationships was done. Later, we listed each concept in the ontology domain with other concepts in the same domain that appeared with them when studying the extracted relationships.

**Chapter 5:** In this chapter we present and discuss the conclusions of our research; our final results and how we used them to contribute in the studied domain are presented among the conclusions. Future work are suggested at the end of this chapter.

## CHAPTER 2

### RELATED WORK

In this chapter we give a brief idea about the most relevant work in the literature that are related to our study.

#### 2.1 GENERAL RELATED WORK

Ontologies have been widely built and used during the last years. Many researches related to software engineering and measurements have been issued; in particular, building ontologies for software measurement engineering using potential elements (such as goals, viewpoints, data, operations, agents, scenarios and resources) have been carried out. Proposals, studies, standards, and contributions related to the work are illustrated below:

##### [Balzer, 1982]

Has started advocating the benefits of underlying ontologies of precise and formal specifications, notably for checking a specification adequacy through prototyping [5].

##### [Rumbaugh, 1991]

Has proposed multi-paradigm frameworks to combine multiple languages in a semantically meaningful way so that different facets can be captured by languages that fit them best [101].

##### [VIM, 1993]

The International Vocabulary of Basic and General Terms in Metrology [61] covers 120 terms of subjects related to measurement. Although its main target is not software, it has been successfully used by several authors, such as Alain Abran, for defining software measurement concepts [1], and is one of the bases for ISO-JTC1 software measurement harmonization efforts. The VIM is a very detailed, complete and mature reference. However, its terms remain at a very detailed level; for instance, there are no definitions for general terms such as “metric” or “measure”. The new version of the VIM, currently in preparation, is expected to deal with the software measurement specific needs.

##### [Kim, 1999]

Henry Kim [76] has proposed a formal model of enterprise quality, called “ontology of enterprise quality modeling”. This is a global ontology, whose main objective is to help evaluate the conformance of organizations to ISO/IEC 9000 standards.

As part of his global ontology, Kim also proposes measurement ontology. Although Kim's measurement ontology is not specific to software products and processes, it contains many concepts that can be applied within the context of software measurement. Under this perspective, Kim's proposal mainly focuses on targets-and-goals, including concepts such as "quality requirement", "entity", "enterprise model of quality", and "measured attribute". It does not define, however, concepts such as "measure", "metric" or "scale", for instance.

**[Kitchenham et al., 2001]**

Barbara Kitchenham et al. [78] propose a method for specifying models of software data sets in order to capture the definitions and relationships among software measures. They propose a conceptual model with three components. First, the generic component defines concepts such as attributes, units, and scale types, independently from other considerations. The development model provides the link between measures and entities of interest. Finally, the project domain represents the data values collected from real projects, linking data values to actual instances of the entities that are defined in the development model domain. This proposal is mainly concerned with both measures and targets-and-goals, but without considering the measurement process in detail. Besides, their terminology is not completely aligned with the rest of the standards and proposals. For instance, the concept of "measure" is represented by the term "DM element measure type", which significantly differs from the terms "metric" or "measure", probably the most commonly used terms in the rest of the sources for representing this concept.

**[Briand et al. 2002]**

Lionel Briand et al. propose the GQM/MEDEA approach for defining measures of product attributes in software engineering. This approach is driven by the experimental goals of measurement, expressed via the GQM [9] paradigm and a set of empirical hypotheses. This proposal provides a UML class diagram with the concepts involved in the GQM/MEDEA process. Those GQM/MEDEA concepts related to software measurement are mainly concerned with measurement targets-and-goals (e.g., entity, attribute). It does not consider, for instance, the concepts "measurement" or "scale", and does not distinguish between base and derived measures either. One of the specific characteristics of this proposal is that its concepts are not defined, but just presented for their use in the GQM/MEDEA process. This forced us to guess their real meaning when including them in the comparison analysis.

**[Devedzic, 2002]**

Has explored that ontologies are needed in all phases of software engineering lifecycle, each of which must have knowledge, whether about data structure, methods and domain. This makes ontologies everywhere and they make possible to smoothly integrate Artificial Intelligence with other software disciplines [26].

**[Zlot, 2002]**

Has defined a structure to represent the task knowledge with support to software engineers in understanding business problems starting from the understanding of the task, which comprises these problems. This structure combines task ontologies and problem solving methods to support capturing knowledge about specific domain throughout the development process [124].

**[Obrst, 2003]**

Has discussed the use of ontology for semantic interoperability in homogenous environments [91].

**[Maria Martin et al., 2003]**

Have presented a semiformal ontology for software metrics and indicators based as much as possible on the concepts from the studied standards which can be useful to support different assurance processes, methods and tools in addition to be the foundation for the cataloging web system used in their work [24].

**[Ahmad Kayed et al, 2005]**

Have used the conceptual graphs to implement ontology that built for solving problems in the E-commerce domain, and used the BWW model to evaluate the work done by using conceptual graphs, that led to build a meta-model using some of the BWW constructs [73].

**[Felix Garcia et al, 2005]**

Have presented an analysis of the software measurements terminology proposals and provided a comparison framework that can be used to identify and address the discrepancies, gaps, and terminology conflicts that current software measurement proposals present. A basic software measurement ontology is introduced, that aims at contributing to the harmonization of the different software measurement proposals and standards, by providing a coherent set of common concepts used in software measurement. The ontology is also aligned with the metrology vocabulary used in other more mature measurement engineering disciplines [39].

## 2.2 SPECIFIC RELATED WORK

In addition of previously presented related work, we separated and illustrated more specific related work to software quality in a dependent subsection, because of its important role in our work.

- **McCall's Quality Model (1977)**

One of the more renown predecessors of today's quality models is the quality model presented by Jim McCall et al. [77,84,87] (also known as the General Electrics Model of 1977). This model, as well as other contemporary models, originates from the US military (it was developed for the US Air Force, promoted within DoD) and is primarily aimed towards the system developers and the system development process. In this quality model McCall attempts to bridge the gap between users and developers by focusing on a number of software quality factors that reflect both the users' views and the developers' priorities.

The McCall quality model has, as shown in Figure 2.1, three major perspectives for defining and identifying the quality of a software product: product revision (ability to undergo changes), product transition (adaptability to new environments) and product operations (its operation characteristics). Product revision includes maintainability (the effort required to locate and fix a fault in the program within its operating environment), flexibility (the ease of making changes required by changes in the operating environment) and testability (the ease of testing the program, to ensure that it is error-free and meets its specification). Product transition is all about portability (the effort required to transfer a program from one environment to another), reusability (the ease of reusing software in a different context) and interoperability (the effort required to couple the system to another system). Quality of product operations depends on correctness (the extent to which a program fulfils its specification), reliability (the systems ability not to fail), efficiency (further categorized into execution efficiency and storage efficiency and generally meaning the use of resources, e.g. processor time, storage), integrity (the protection of the program from unauthorized access) and usability (the ease of the software).

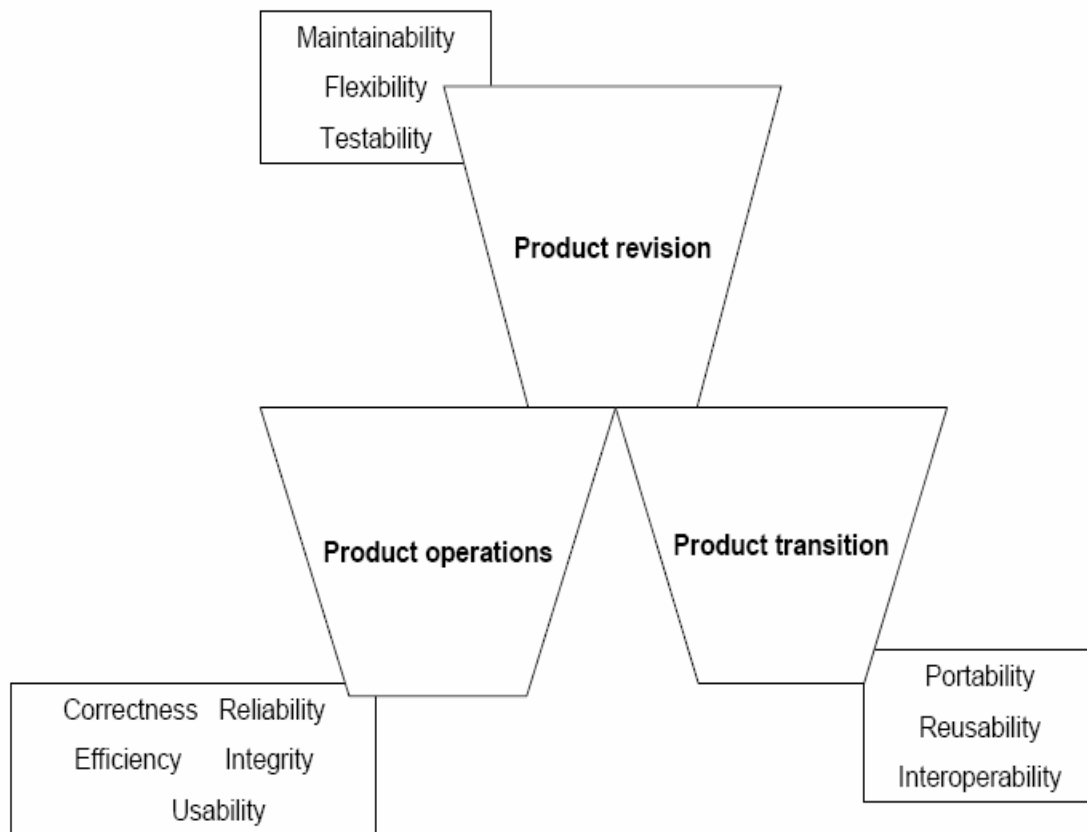


Figure 2.1: The McCall quality model, organized around three types quality characteristics.

The model furthermore details the three types of quality characteristics (major perspectives) in a hierarchy of factors, criteria and metrics:

- 11 Factors (To specify): They describe the external view of the software, as viewed by the users.
- 23 quality criteria (To build): They describe the internal view of the software, as seen by the developer.
- Metrics (To control): They are defined and used to provide a scale and method for measurement.

Figure 2.2 shows the McCall's Quality Model illustrated through a hierarchy of 11 quality factors (on the left hand side of the Figure) related to 23 quality criteria (on the right hand side of the Figure).

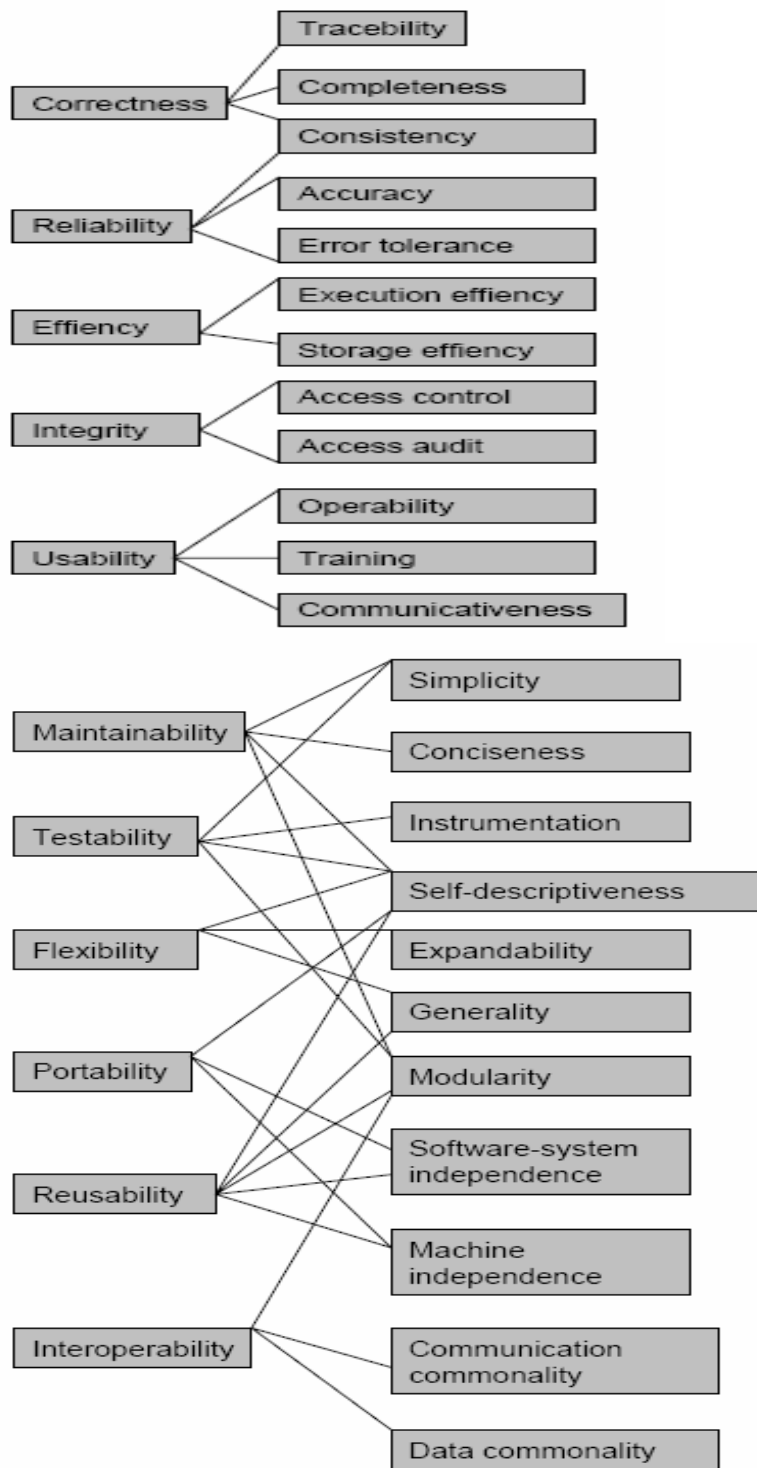


Figure 2.2: McCall's Quality Model illustrated through a hierarchy of 11 quality factors (on the left hand side of the Figure) related to 23 quality criteria (on the right hand side of the Figure).



The quality factors describe different types of system behavioral characteristics, and the quality criteria are attributes to one or more of the quality factors. The quality metric, in turn, aims to capture some of the aspects of a quality criterion. The idea behind McCall's Quality Model is that the quality factors synthesized should provide a complete software quality picture [77]. The actual quality metric is achieved by answering yes and no questions that then are put in relation to each other. That is, if answering equally amount of "yes" and "no" on the questions measuring a quality criteria you will achieve 50% on that quality criteria. The metrics can then be synthesized per quality criteria, per quality factor, or if relevant per product or service.

- **Boehm's Quality Model (1978)**

The second of the basic and founding predecessors of today's quality models is the quality model presented by Barry W. Boehm [13,14]. Boehm addresses the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. In essence his models attempts to qualitatively define software quality by a given set of attributes and metrics. Boehm's model is similar to the McCall quality model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate level characteristics, primitive characteristics, each of which contributes to the overall quality level.

- The high-level characteristics represent basic high-level requirements of actual use to which evaluation of software quality could be put – the general utility of software. The high-level characteristics address three main questions that a buyer of software has:

- As-is utility: How well (easily, reliably, efficiently) can I use it as-is?
- Maintainability: How easy is it to understand, modify and retest?
- Portability: Can I still use it if I change my environment?

- The intermediate level characteristic represents Boehm's 7 quality factors that together represent the qualities expected from a software system:

- Portability (General utility characteristics): Code possesses the characteristic portability to the extent that it can be operated easily and well on computer configurations other than its current one.
- Reliability (As-is utility characteristics): Code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily.
- Efficiency (As-is utility characteristics): Code possesses the characteristic efficiency to the extent that it fulfills its purpose without waste of resources.

- Usability (As-is utility characteristics, Human Engineering): Code possesses the characteristic usability to the extent that it is reliable, efficient and human-engineered.
- Testability (Maintainability characteristics): Code possesses the characteristic testability to the extent that it facilitates the establishment of verification criteria and supports evaluation of its performance.
- Understandability (Maintainability characteristics): Code possesses the characteristic understandability to the extent that its purpose is clear to the inspector.
- Flexibility (Maintainability characteristics, Modifiability): Code possesses the characteristic modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined. (Note the higher level of abstractness of this characteristic as compared with augmentability).

- The lowest level structure of the characteristics hierarchy in Boehm's model is the primitive characteristics metrics hierarchy. The primitive characteristics provide the foundation for defining qualities metrics-which was one of the goals when Boehm constructed his quality model. Consequently, the model presents one ore more metrics supposedly measuring a given primitive characteristic.



Figure 2.3: Boehm's Software Quality Characteristics Tree [14].

Boehm's and McCall's models might appear very similar, the difference is that McCall's model primarily focuses on the precise measurement of the high-level characteristics "As-is utility" (see Figure 2.3), whereas Boehm's quality mode model is based on a wider range of characteristics with an extended and detailed focus on primarily maintainability. Table 2.1 compares the two quality models, quality factor by quality factor.

Table 2.1: Comparison between criteria/goals of the McCall and Boehm quality models [53].

Criteria / Goals	McCall 1977	Boehm 1978
Correctness	*	*
Reliability	*	*
Integrity	*	*
Usability	*	*
Efficiency	*	*
Maintainability	*	*
Testability	*	*
Interoperability	*	
Flexibility	*	*
Reusability	*	*
Portability	*	*
Clarity		*
Modifiability		*
Documentation		*
Resilience		*
Understandability		*
Validity		*
Functionality		
Generality		*
Economy		*

As indicated in Table 2.1 Boehm focuses a lot on the models effort on software maintenance cost-effectiveness – which, he states, is the primary payoff of an increased capability with software quality considerations.

- **FURPS Quality Model**

A later, and perhaps somewhat less renown, model that is structured in basically the same manner as the previous two quality models (but still worth at least to be mentioned in this context) is the FURPS model originally presented by Robert Grady [43] ,(and extended by Rational Software [64,80,112]. FURPS stands for:

- Functionality – which may include feature sets, capabilities and security.
- Usability - which may include human factors, aesthetics, consistency in the user interface, online and context-sensitive help, wizards and agents, user documentation, and training materials.
- Reliability - which may include frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failures (MTBF).
- Performance - imposes conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage.
- Supportability - which may include testability, extensibility, adaptability, maintainability, compatibility, configurability, serviceability, installability, localizability (internationalization).

The FURPS-categories are of two different types: Functional (F) and Non-functional (URPS). These categories can be used as both product requirements as well as in the assessment of product quality.

- **Dromey's Quality Model**

An even more recent model similar to the McCall's, Boehm's and the FURPS quality models, is the quality model presented by R. Geoff Dromey [27, 28]. Dromey proposes a product based quality model recognizes that quality evaluation differs for each product and so a more dynamic idea for modeling the process is needed to be wide enough to apply for different systems. Dromey is focusing on the relationships between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes.

As illustrated in Figure 2.4, there are three principal elements to Dromey's generic quality model:

1. Product properties that influence quality
2. High level quality attributes
3. Means of linking the product properties with the quality attributes.

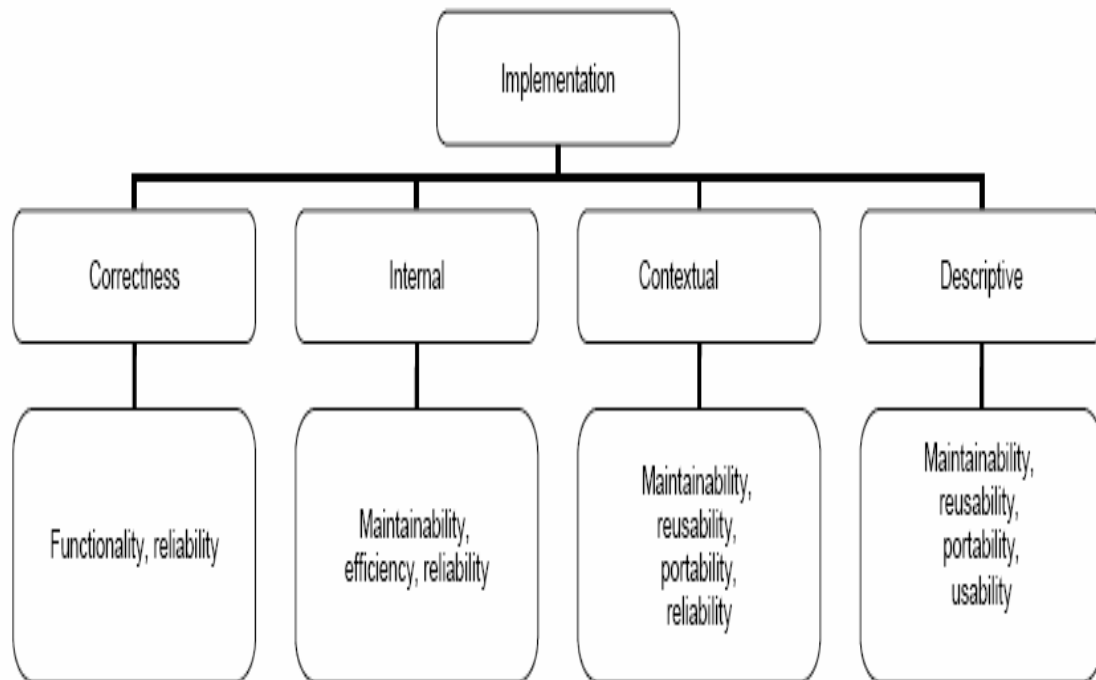


Figure 2.4: Principles of Dromey's Quality Model.

Dromey's Quality Model is further structured around a 5 steps process:

1. Choose a set of high-level quality attributes necessary for the evaluation.
2. List components/modules in your system.
3. Identify quality-carrying properties for the components/modules (qualities of the component that have the most impact on the product properties from the list above).
4. Determine how each property effects the quality attributes.
5. Evaluate the model and identify weaknesses.

- **ISO 9000**

ISO stands for International Standards Organization. The ISO organization is responsible for a whole battery of standards of which the ISO 9000 [55-63] family probably is the most well known, spread and used. Figure 1.9 shows The ISO 9000:2000 standards. The crosses and arrows indicate changes made from the older ISO 9000 standard to the new ISO 9000:2000 standard.

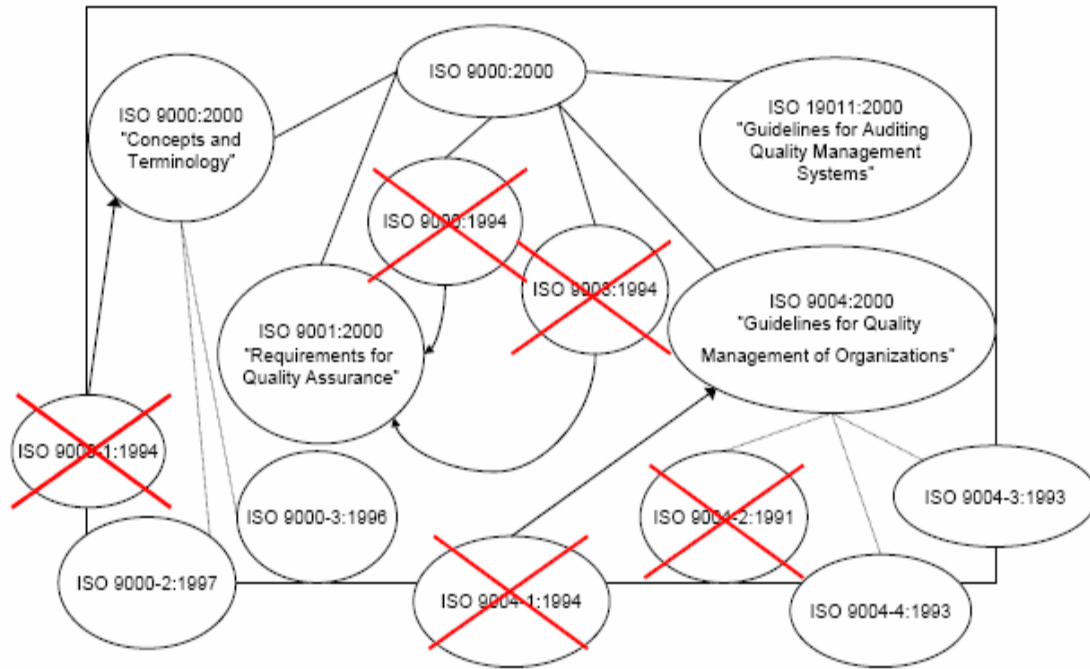


Figure 2.5: The ISO 9000:2000 standards. The crosses and arrows indicate changes made from the older ISO 9000 standard to the new ISO 9000:2000 standard.

ISO 9001 is an international quality management system standard applicable to organizations within all type of businesses. ISO 9001 internally addresses an organization's processes and methods and externally at managing (controlling, assuring etc.) the quality of delivered products and services. ISO 9001 is a process oriented approach towards quality management. That is, it proposes designing, documenting, implementing, supporting, monitoring, controlling and improving (more or less) each of the following processes [55-63]:

- Quality Management Process.
- Resource Management Process.
- Regulatory Research Process.
- Market Research Process.
- Product Design Process.
- Purchasing Process.
- Production Process.
- Service Provision Process.
- Product Protection Process.
- Customer Needs Assessment Process.
- Product Protection Process.
- Customer Needs Assessment Process.

- **ISO 9126**

Besides the famous ISO 9000, ISO has also release the ISO 9126: Software Product Evaluation: Quality Characteristics and Guidelines for their Use-standard [60] (among other standards), Figure 1.10 below shows the ISO 9126 model.

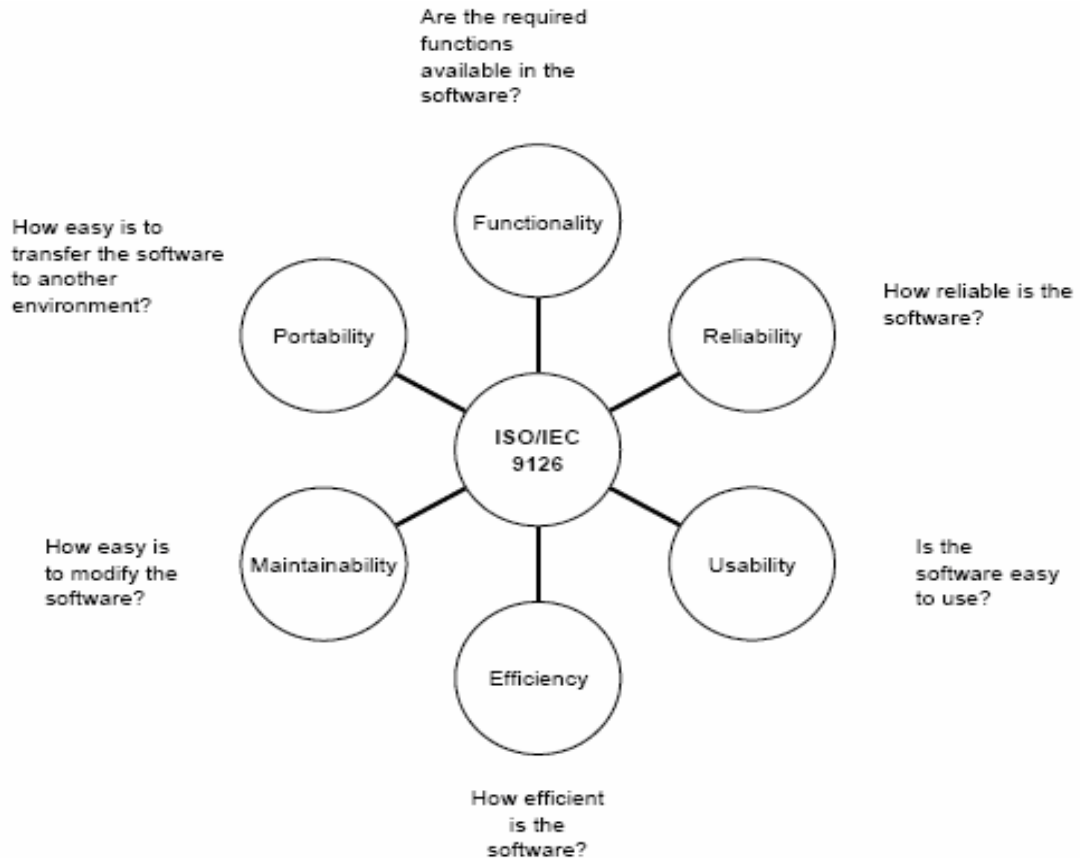


Figure 2.6: The ISO 9126 quality model.

This standard was based on the McCall and Boehm models. Besides being structured in basically the same manner as these models (see table 2.2), ISO 9126 also includes functionality as a parameter, as well as identifying both internal and external quality characteristics of software products. Figure 2.7 shows a comparison between criteria/goals of the McCall, Boehm and ISO 9126 quality models [69].

Table 2.2: Comparison between criteria/goals of the McCall, Boehm and ISO 9126 quality models [69].

<i>Criteria/goals</i>	<i>McCall, 1977</i>	<i>Boehm, 1978</i>	<i>ISO 9126, 1993</i>
Correctness	*	*	maintainability
Reliability	*	*	*
Integrity	*	*	
Usability	*	*	*
Efficiency	*	*	*
Maintainability	*	*	*
Testability	*	*	maintainability
Interoperability	*		
Flexibility	*	*	
Reusability	*	*	
Portability	*	*	*
Clarity		*	
Modifiability		*	maintainability
Documentation		*	
Resilience		*	
Understandability		*	
Validity		*	maintainability
Functionality			*
Generality		*	
Economy		*	

ISO 9126, as shown in table 2.2 below, proposes a standard which specifies six areas of importance, i.e. quality factors, for software evaluation.



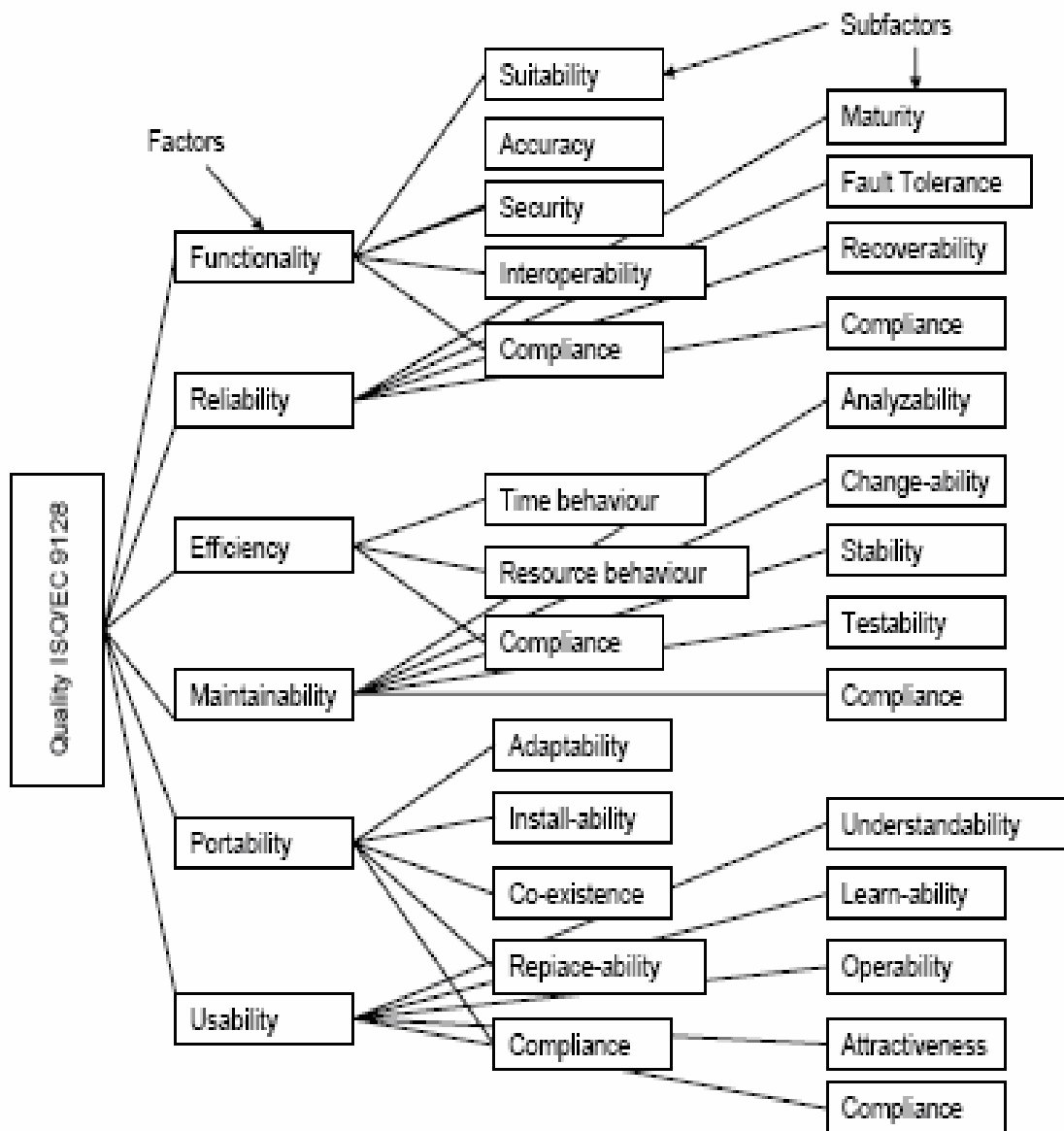


Figure 2.7: ISO 9126: Software Product Evaluation: Quality Characteristics and Guidelines for their use.

- Each quality factor and its corresponding sub-factors are defined as follows:

- Functionality: A set of attributes that relate to the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs;
  - Suitability: Attribute of software that relates to the presence and appropriateness of a set of functions for specified tasks.

- Accuracy: Attributes of software that bare on the provision of right or agreed results or effects.
  - Security: Attributes of software that relate to its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.
  - Interoperability: Attributes of software that relate to its ability to interact with specified systems.
  - Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.
- Reliability: A set of attributes that relate to the capability of software to maintain its level of performance under stated conditions for a stated period of time;
    - Maturity: Attributes of software that relate to the frequency of failure by faults in the software.
    - Fault tolerance: Attributes of software that relate to its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
    - Recoverability: Attributes of software that relate to the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.
    - Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.
- Efficiency: A set of attributes that relate to the relationship between the level of performance of the software and the amount of resources used, under stated conditions;
    - Time behavior: Attributes of software that relate to response and processing times and on throughput rates in performing its function.
    - Resource behavior: Attributes of software that relate to the amount of resources used and the duration of such use in performing its function.
    - Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.
- Maintainability: A set of attributes that relate to the effort needed to make specified modifications;
    - Analyzability: Attributes of software that relate to the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.

- Changeability: Attributes of software that relate to the effort needed for modification, fault removal or for environmental change.
  - Stability: Attributes of software that relate to the risk of unexpected effect of modifications.
  - Testability: Attributes of software that relate to the effort needed for validating the modified software.
  - Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.
- Portability: A set of attributes that relate to the ability of software to be transferred from one environment to another;
    - Adaptability: Attributes of software that relate to on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.
    - Installability: Attributes of software that relate to the effort needed to install the software in a specified environment.
    - Conformance: Attributes of software that make the software adhere to standards or conventions relating to portability.
    - Replaceability: Attributes of software that relate to the opportunity and effort of using it in the place of specified other software in the environment of that software.
  - Usability: A set of attributes that relate to the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users;
    - Understandability: Attributes of software that relate to the users' effort for recognizing the logical concept and its applicability.
    - Learnability: Attributes of software that relate to the users' effort for learning its application (for example, operation control, input, output).
    - Operability: Attributes of software that relate to the users' effort for operation and operation control.
    - Attractiveness.
    - Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.

- **ISO/IEC 14598 (1999-2001) and 9126 (2001-2004)**

ISO/IEC 14598 (Information technology- Software product evaluation) [62], is a series of international standards that provide methods for measurement, assessment and evaluation of software product quality. The different parts of this series set out a generic picture of the process of evaluation, addressing it from the point of view of developers, acquirers and (third party) evaluators. The standards of ISO/IEC 14598 series are mainly concerned with the set of concepts in the measures group, and partially covering some of the measurement process aspects. ISO/IEC 14598 series makes use of the ISO/IEC 9126 series (Software engineering “Product quality” Parts 1 to 4) [60], which propose a software product quality model, and metrics for internal quality, external quality, and quality in use. The SQuaRE project [3] has been specifically created to make them converge, trying to eliminate the gaps, conflicts, and ambiguities that they currently present. In fact, ISO/IEC TR 9126-2, 9126-3 and 9126-4 were allowed to be published as Technical Reports between 2002 and 2004 without changing their original terminology, with the agreement that they would be aligned with the new SC7 measurement terms as soon as possible.

- **ISO/IEC 15939 (2002) and PSM (2002)**

ISO/IEC 15939 standards identify the activities and tasks needed to successfully identify, define, select, apply, and improve software measurement within an overall project or organizational measurement structure. It also provides definitions for measurement terms commonly used within the software industry. The two key components included in this standard are software measurement process and measurement information model. The software measurement process is driven by the information needs of the organization. For each information need, this process produces an information product that tries to satisfy it. The measurement information model establishes the link between measures and information needs. Measured entities include processes, products, projects, and resources. The model describes how the relevant attributes are quantified, and converted to indicators that provide a basis for decision-making. It basically rests upon the concepts of ISO/IEC 14598 and ISO/IEC 9126, although changing some of the terms in order to be aligned as much as possible with the ISO VIM. Hence, it does not use the term “metric”, relating directly the terms “measurement” and “measure”. ISO/IEC 15939, together with VIM, has become the standard used by ISO-JTC1 (International Standards Organization/ Joint Technical Committee 1), as the basis for its software measurement terminology harmonization efforts [68]. Another key reference, the PSM (Practical Software Measurement) [88], is compatible with ISO/IEC 15939, and therefore uses the same terminology.

- **ISO/IEC 15504 (SPICE)**

The ISO/IEC 15504: Information Technology - software process assessment is a large international standard framework for process assessment that intends to address all processes involved in:

- Software acquisition
- Development
- Operation
- Supply
- Maintenance
- Support

ISO/IEC 15504 consists of 9 component parts covering concepts, process reference model and improvement guide, assessment model and guides, qualifications of assessors, and guide for determining supplier process capability:

1. ISO/IEC 15504-1 Part 1: Concepts and Introductory Guide.
2. ISO/IEC 15504-2 Part 2: A Reference Model for Processes and Process Capability.
3. ISO/IEC 15504-3 Part 3: Performing an Assessment.
4. ISO/IEC 15504-4 Part 4: Guide to Performing Assessments.
5. ISO/IEC 15504-5 Part 5: An Assessment Model and Indicator Guidance.
6. ISO/IEC 15504-6 Part 6: Guide to Competency of Assessors.
7. ISO/IEC 15504-7 Part 7: Guide for Use in Process Improvement.
8. ISO/IEC 15504-8 Part 8: Guide for Use in Determining Supplier Process Capability.
9. ISO/IEC 15504-9 Part 9: Vocabulary.

Given the structure and contents of the ISO/IEC 15504 documentation it is more closely related to ISO 9000, ISO/IEC 12207 and CMM, rather than the initially discussed quality models (McCall, Boehm and ISO 9126).

- **IEEE**

- IEEE has also release several standards, more or less related to the topic of our research. To name a few:

- IEEE Std. 1220-1998: IEEE Standard for Application and Management of the systems engineering process.
- IEEE Std 730-1998: IEEE Standard for SWQA Plans.
- IEEE Std 828-1998: IEEE Standard for Software Configuration Management Plans – Description.
- IEEE Std 829-1998: IEEE Standard For Software Test Documentation.

- IEEE Std 830-1998: IEEE recommended practice for software requirements specifications.
- IEEE Std 1012-1998: IEEE standard for software verification and validation plans.
- IEEE Std 1016-1998: IEEE recommended practice for software design descriptions.
- IEEE Std 1028-1997: IEEE Standard for Software Reviews.
- IEEE Std 1058-1998: IEEE standard for software project management plans.
- IEEE Std 1061-1998: IEEE standard for a software quality metrics methodology.
- IEEE Std 1063-2001: IEEE standard for software user documentation.
- IEEE Std 1074-1997: IEEE standard for developing software life cycle processes.
- IEEE/EIA 12207.0-1996: Standard Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology Software Life Cycle Processes.

Of the above mentioned standards it is probably the implementation of ISO/IEC 12207: 1995 that most resembles previously discussed models in that it describes the processes for the following life-cycle:

- Primary Processes: Acquisition, Supply, Development, Operation, and Maintenance.
- Supporting Processes: Documentation, Configuration Management, Quality Assurance, Verification, Validation, Joint Review, Audit, and Problem Resolution.
- Organization Processes: Management, Infrastructure, Improvement, and Training

In fact, IEEE/EIA 12207.0-1996 is so similar to the ISO 9000 standard that it could actually be seen as a potential replacement for ISO within software engineering organizations.

The IEEE Std 1061-1998 is another standard that is relevant from the perspective of this research as the standard provides a methodology for establishing quality requirements and identifying, implementing, analyzing and validating the process and product of software quality metrics [54].

- **Capability Maturity Model(s) (CMM)**

The Carnegie Mellon Software Engineering Institute (SEI), non-profit group sponsored by the DoD work at getting US software more reliable [50, 51, 52]. SEI has also produced a number of more extensive Capability Maturity Models that in a very IEEE and ISO 9000 similar manner addresses the topic of software quality such as:

- CMM / SW-CMM [85, 52].
- P-CMM [23].
- CMMI - PDD-CMM - SE-CMM - SA-CMM [109].

The CMM/SW-CMM addresses the issue of software quality from a process perspective, Figure 2.8 below shows the Maturity levels of SW-CMM, and also table 2.3 shows the Maturity levels with corresponding focus and key process areas for CMM.

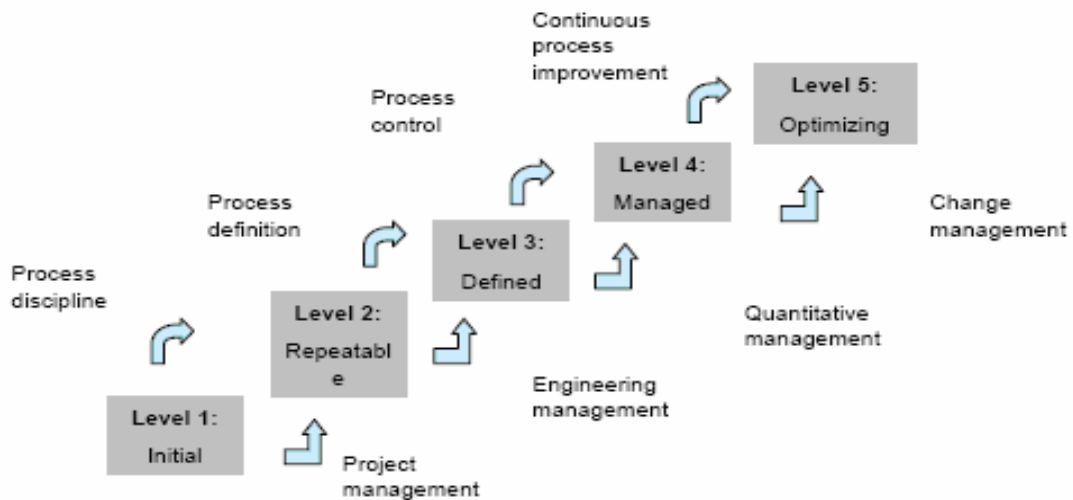


Figure 2.8: Maturity Levels of SW-CMM.

Table 2.3: Maturity levels with corresponding focus and key process areas for CMM

Level	Focus	Key Process Area
Level 5 – Optimizing level	Continuous improvement	Process Change Management Technology Change Management Defect Prevention
Level 4 – Managed level	Product and process quality	Software Quality Management Quantitative Process Management
Level 3 – Defined level	Engineering process	Organization Process Focus Organization Process Definition Peer Reviews Training Program Intergroup Coordination Software Product Engineering Integrated Software Management
Level 2 – Repeatable level	Project Management	Requirements Management Software Project Planning Software Project Tracking and Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management
Level 1 – Initial level	Heroes	No KPAs at this time

The SW-CMM is superseded by the CMMI model which also incorporates some other CMM models into a wider scope. CMMI Integrates systems and software disciplines into one process improvement framework and is structured around the following process areas [23, 85]:

- Process management.
- Project management.
- Engineering.
- Support.

And similarly to the SW-CMM the CMMI is structured around the following maturity levels [109]:

- Maturity level 5: Optimizing - Focus on process improvement.
- Maturity level 4: Quantitatively managed - Process measured and controlled.
- Maturity level 3: Defined - Process characterized for the organization and is proactive.
- Maturity level 2: Managed - Process characterized for projects and is often reactive.
- Maturity level 1: Initial - Process unpredictable, poorly controlled and reactive.
- Maturity level 0: Incomplete.

Figures 2.9 and 2.10 show the two representations of the CMMI model.

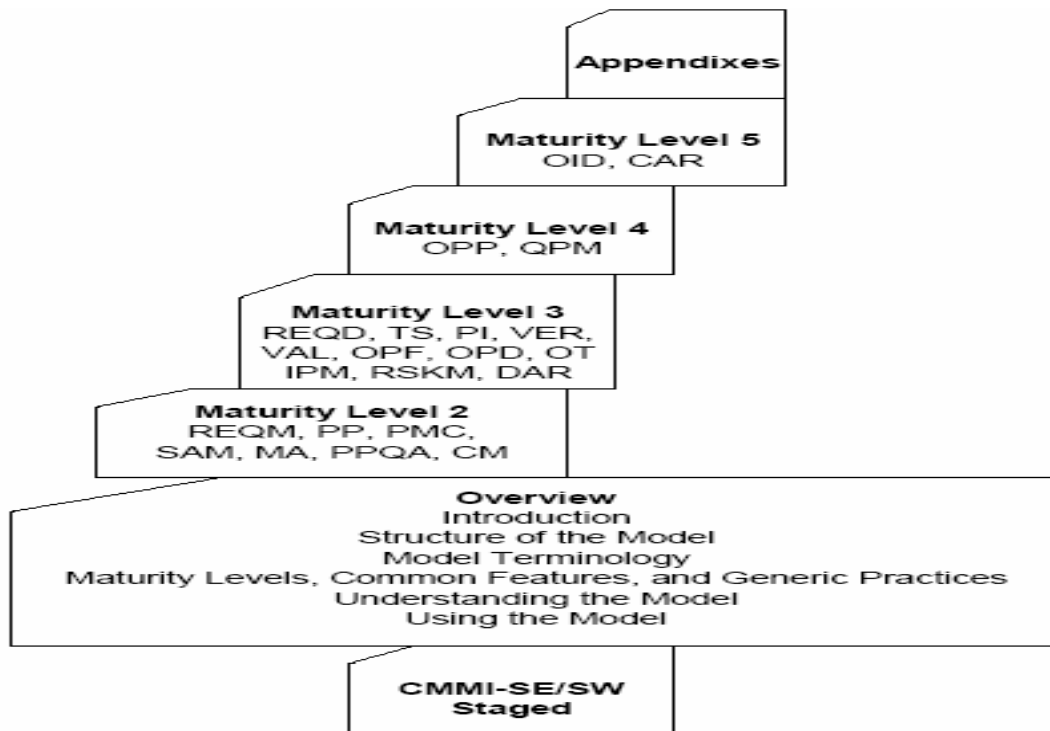


Figure 2.9: The staged CMMI-SW/SW representation.



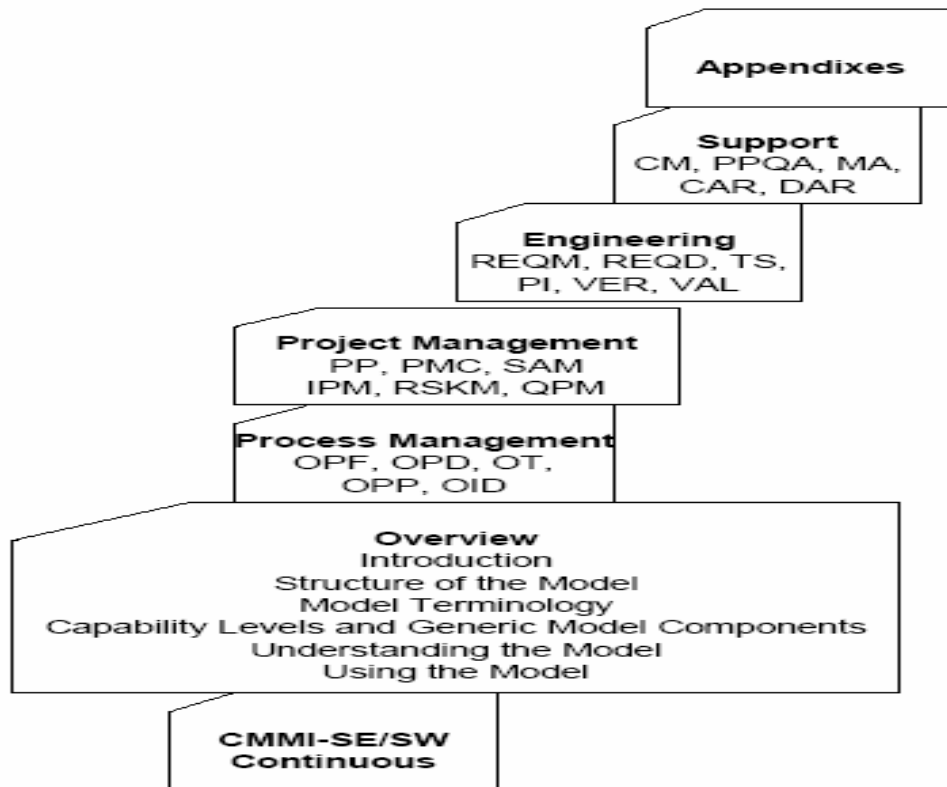


Figure 2.10: The continuous CMMI-SW/SW representation.

## CHAPTER 3

### BUILDING ONTOLOGY DOMAIN CONCEPTS

This chapter is talking about the preparation of the text corpora for the SWPQAs knowledge domain. This preparation is done by collecting and studying a large number of documents and reports related to the field of software quality. The chapter also is discussing how the prepared text corpora were used to extract and create our primary ontology domain concepts using TextToOnto tool with support of an MS Access tool and then with support of human experts. In this chapter, we also focused on evaluating the suggested ontology domain concepts using a coverage methodology. After preparing the needed corpus, and by using a tool created by Kayed [72], we counted the covered concepts and calculated a coverage percentage for them. Later, we discussed what the results would be if we collect and study the uncovered concepts from the domain under discussion. The results showed an enhancement of our ontology domain concepts, and gave a much better coverage percentage. The results are shown in the discussion.

#### 3.1 PREPARING TEXT CORPORA FOR SOFTWARE PRODUCT QUALITY ATTRIBUTES DOMAIN

As mentioned earlier, TextToOnto is a tool provided for ontology engineering process depending on text mining techniques and natural language processing algorithms [83]. To use this tool we needed to prepare text corpora, in linguistics, text corpora consists of large set of electronically processed and stored texts. They are needed when doing statistical analysis, checking occurrences, or validating linguistic rules on a specific domain. TextToOnto tool deals with corpora of text or html type.

For our research we prepared text corpora to be used within the TextToOnto tool and later within an Access tool, software quality relevant domain documents, reports, and publications were collected. In our case, we collected as much as possible of what we could reach to of publications, documents, and reports that we think they were related to the field, almost about 85 different related documents to software engineering, quality, and software quality were collected. We believed that in such a large collected domain heterogeneous and homogenous text collection, concepts, and terms can be found. Upon the discussion of SWPQAs and their definitions, a more deep study was conducted to these collected documents and they were filtered into almost 34 much related documents, reports, and publications. After that from these resulted files we created a document contains a summary from their semantic; the parts that specifically discussed SWPQAs, it included about 95 pages with almost 33,600 words. Later, we converted them into text files. By that our text corpora for the SWPQAs domain were ready, where the corpus consisted of the 34 documents were entered into the TextToOnto tool and the corpus consisted of the summary was entered into the Access tool later on. The discussion later shows the details of how they had been used.

### 3.2 EXTRACTING ONTOLOGY DOMAIN CONCEPTS

Ontology domain concepts extraction is considered the most important part in building an ontology. In order to extract ontology domain concepts we must study the semantic of the prepared text corpora. To do so, at first we used the TextToOnto tool [83]. We added the prepared text corpus (34 related documents) to the tool by using the New Corpus function. Figure 3.1 shows the creation and addition of the prepared corpus to the tool.

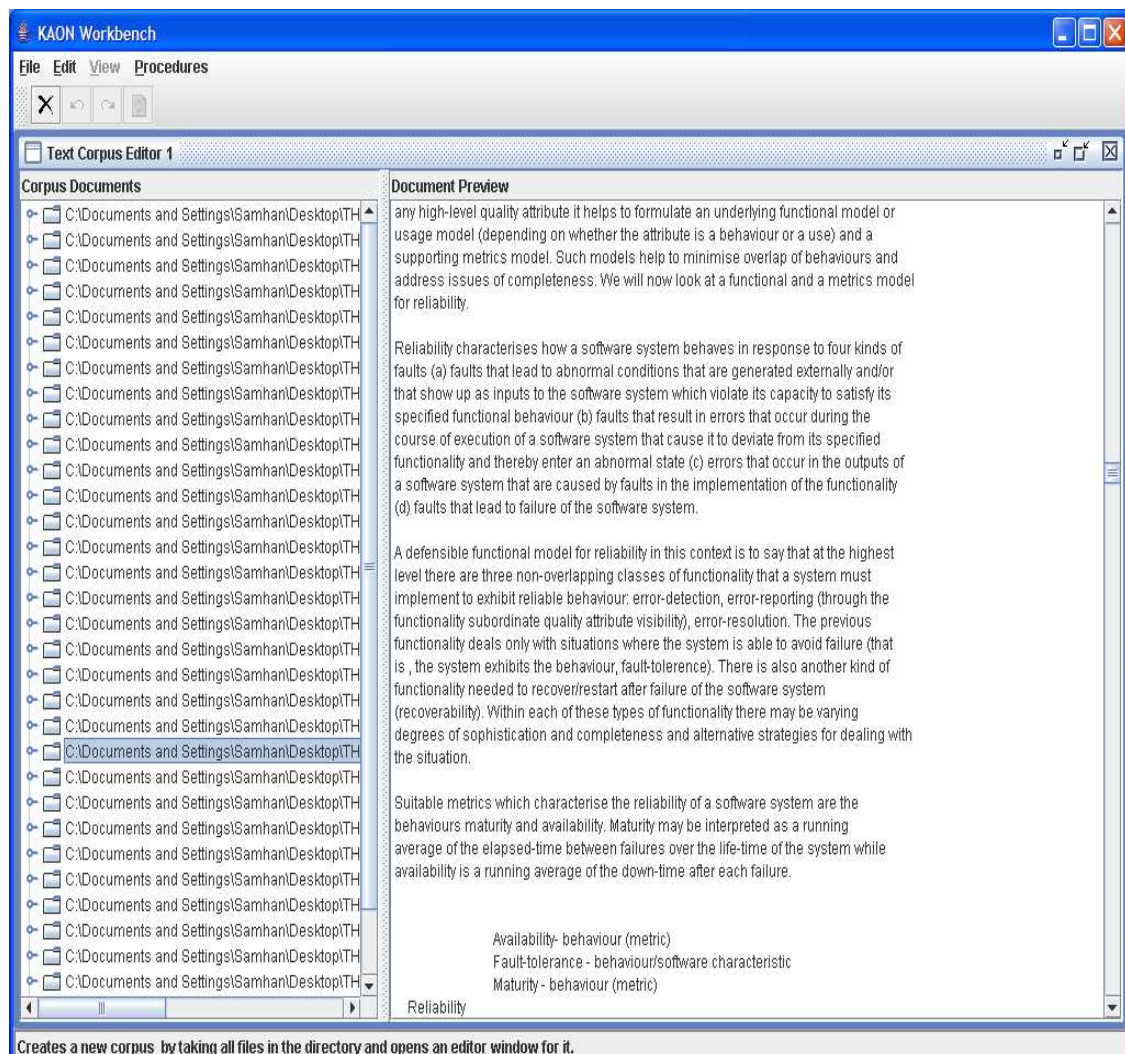


Figure 3.1: Creating a Corpus using TextToOnto Tool.

Later we used the New Term Extraction function in order to extract concepts from provided the text corpus. This tool depends on natural language processing algorithms in addition to semantic lexicon filtering techniques. When we decided to declare parameters

to be used in the tool, at first we used frequency threshold to be 5 and above but the result included more than 8500 concepts and that was very large to be considered as a initial result for the ontology domain concepts; it was difficult to be handled, so we declared 10, 15, and 20 as frequency thresholds to be taken, from the results that came we chose to stick with retrieving concepts that their frequency in the given text corpus were 10 frequencies or above, we also chose to retrieve concepts that consist on one unique word as a term; to have a suitable number of concepts (not too large and also not too small) to be collected and studied in our work. Figure 3.2 shows this step and some of the resulted concepts.

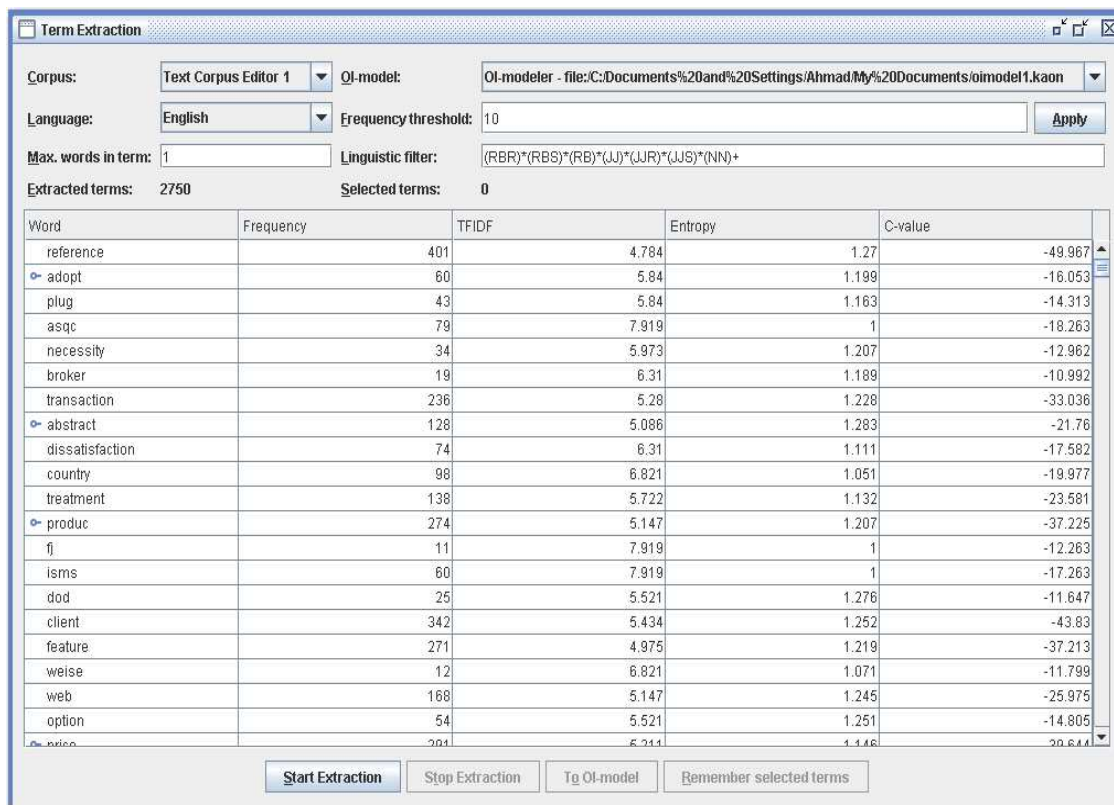


Figure 3.2: The Term Extraction process using the TextToOnto tool.

This step; using TextToOnto tool to extract concepts, provided us with about 2750 single concepts having a 10 or above as frequency of appearance in the given text corpus. After that and in order to refine these resulted concepts we used a tool created by Kayed [72], it is a combination between MS Access tool and MS Visual Basic language. We provided it with the resulted concepts (2750) and with the other previously prepared text corpus (text corpus from the abstract file), It depends on a semantic counting algorithm that counts the unique frequencies of the concepts in a given set of texts, so by using this algorithm it studied which concepts from the provided 2750 concepts were found in the semantic of the provided corpus and how many times? This tool provided us with almost 292 single concepts. Table 3.1 below lists the resulted concepts from the tool.

Table 3.1: the resulted 292 concepts (out of 2750).

Concept	Frequency	Concept	Frequency	Concept	Frequency
attribute	99	work	8	independ	3
software	83	capacity	7	nature	3
component	78	control	7	occurrence	3
system	68	express	7	people	3
ability	58	failure	7	portability	3
function	57	manner	7	precision	3
use	56	response	7	presence	3
form	52	scope	7	produce	3
characteristic	42	storage	7	relationship	3
degree	39	adapt	6	relative	3
product	38	case	6	reliability	3
can	37	complex	6	repair	3
perform	35	developer	6	risk	3
capability	32	example	6	show	3
program	30	general	6	speed	3
environment	26	make	6	stability	3
concern	25	objective	6	testability	3
design	25	period	6	transfer	3
end	25	processing	6	verification	3
user	25	support	6	adaptability	2
measure	24	table	6	adaptation	2
code	23	utilization	6	assessment	2
operation	23	communication	5	audit	2
fact	22	computing	5	certification	2
quality	22	definition	5	commonality	2
time	22	demand	5	compliance	2
data	20	documentation	5	concept	2
effort	20	error	5	configurability	2
extent	20	freedom	5	conformance	2
factor	20	impact	5	cost	2
source	19	interval	5	couple	2
requirement	18	respect	5	customer	2
change	17	see	5	database	2
ease	17	structure	5	defect	2
rate	17	sub	5	dependability	2
implementation	16	type	5	dependency	2
number	16	unit	5	descript	2
amount	15	version	5	description	2
level	15	architecture	4	domain	2
part	15	configuration	4	establishment	2
process	15	continuity	4	evaluation	2
computer	14	effectiveness	4	figure	2
interface	14	incorporation	4	flexibility	2
resource	14	mechanism	4	goal	2
state	14	memory	4	independent	2
application	13	order	4	integrity	2
develop	13	organization	4	interaction	2
input	13	platform	4	knowledge	2
performance	12	probability	4	maintainability	2
set	12	property	4	market	2
correct	11	result	4	model	2

Concept	Frequency	Concept	Frequency	Concept	Frequency
functionality	11	throughput	4	modifiability	2
operating	11	understandability	4	network	2
test	11	usability	4	practice	2
understand	11	accuracy	3	protection	2
access	10	availability	3	provision	2
document	10	being	3	range	2
efficiency	10	business	3	reason	2
information	10	completeness	3	reliance	2
mean	10	complexity	3	sense	2
object	10	correctness	3	standardization	2
context	9	custom	3	step	2
fault	9	deploy	3	terminology	2
hardware	9	development	3	testing	2
output	9	disk	3	tolerance	2
performing	9	engine	3	understanding	2
place	9	engineer	3	uniform	2
purpose	9	exchange	3	uniformity	2
specification	9	execution	3	research	1
standard	9	hand	3	responsiveness	1
effect	8	help	3	reusability	1
means	8	idea	3	safety	1
minimum	8	increase	3	second	1
modification	8	independence	1	security	1
service	8	industry	1	self	1
usage	8	interoperate	1	setting	1
utility	2	issue	1	solution	1
valid	2	language	1	specificity	1
value	2	latency	1	suitability	1
way	2	machine	1	survivability	1
absence	1	maintenance	1	top	1
abstract	1	marketing	1	traceability	1
abstraction	1	meaning	1	training	1
accessibility	1	measurement	1	transition	1
appendix	1	method	1	transport	1
applicability	1	metrics	1	try	1
assurance	1	modular	1	validating	1
breadth	1	modularity	1	verifiability	1
build	1	note	1	volume	1
clarity	1	operator	1	web	1
compatibility	1	improvement	1	existence	1
confidence	1	overlap	1	explanation	1
consistency	1	point	1	extendability	1
coupling	1	predict	1	future	1
deployment	1	predictability	1	guide	1
evaluator	1	producibility	1	removal	1
readiness	1	reference	1		
replacement	1	report	1		

After we had the 292 concepts resulted from the used Access tool, we reached to the final part of extracting the ontology domain concepts. We took those concepts, and applied an elimination process for the stopping words (extremely common words like use, can, the, of, etc) from them. The concepts set resulted from the elimination process was sent to human experts (professors, doctors, and practitioners) in the field of SWE and SWQ, where we asked them to help us in condensing the sent set of concepts into a smaller one. After a while the results were sent back to us, we collected them, studied them upon the agreement of all experts on the sent concepts; they all considered them related and important to the studied field, and merged them into one set of concepts. The result of this part was 100 concepts, which we suggested as an ontology domain concept. Table 3.2 below shows the suggested ontology domain concepts.

Table 3.2: The suggested 100 ontology domain concepts.

Concept	Concept	Concept	Concept	Concept
ability	develop	meaning	responsiveness	control
access	developer	measure	scope	data
accessibility	development	memory	service	definition
accuracy	documentation	modification	set	degree
adapt	ease	notation	setting	design
adaptability	effect	number	software	information
adaptation	effectiveness	object	source	level
amount	efficiency	objective	specification	maintenance
applicability	effort	operating	storage	manner
application	environment	operation	structure	mean
attribute	error	operator	system	purpose
capability	extent	output	test	quality
change	factor	performance	throughput	rate
characteristic	failure	period	time	respect
code	freedom	portability	understand	response
component	function	precision	understandability	usage
computer	functionality	probability	understanding	user
computing	hardware	product	uniform	utility
concern	implementation	program	uniformity	utilization
context	incorporation	property	usability	work

An evaluation process for the suggested ontology domain concepts must be done in order to see how much the reached concepts belong to the knowledge domain of software quality , and to know if the resulted concepts are enough to build a good ontology from or not. That what we are discussing in the next chapter.

### 3.3 ONTOLOGY EVALUATION PROCESS

Various methodologies to evaluate ontologies have been presented in the last decade, most of them belong to one of the following categories:

- Evaluations based on using the ontology in a context of an application or project, to evaluate how effective it is. The use of the system may reveal weakness or strength points in the ontology [16]. For our research it is hard to build an application in order to be used considering the time we have, also we could not find an application in the field to use the suggested ontology in its context.
- Evaluations based on the effort done by human experts, who try to assess how well the ontology meets a set of predefined criteria, standards, and requirements [96]. To reduce the role of human intervention in our work especially after we depended on human experts when extracting the suggested ontology domain concepts, we did not use this approach to evaluate the suggested ontology domain concepts.
- Evaluations based on comparing the ontology with other ontologies in the same domain [17]. As we declared earlier, our ontology is presented as a first in the specific domain of SWQPAs, so we could not use this approach for evaluation.
- Evaluations based on studying ontology relationships considering some criteria [17]. For our ontology we extracted and presented general and basic relationships between the extracted concepts from the domain, it is not adequate to be evaluated using this approach.
- Evaluations based on studying and comparing the formal representation of the ontology with other ontologies formal representations, criteria, or measures [117]. As mentioned earlier, our ontology is presented as a first in the specific domain of SWQPAs, so we could not use this approach for evaluation.
- Evaluations based on fitting or covering techniques between an ontology and a domain of knowledge that the ontology is created for [16, 25].

The last methodology; the coverage methodology, can be decomposed into two different coverage approaches. The first is done by comparing the new ontology domain concepts with a considered existing gold standard domain concepts, to see how much does the studied domain fit in the resulted ontology. The second approach is done by comparing the ontology domain concepts with concepts of a prepared knowledge domain to see how much does the suggested ontology concepts cover from the studied domain concepts.



For our research we used a coverage technique; the requirements for this approach are available (text corpora, tools, etc). We combined the two last discussed approaches of the coverage methodology. We prepared a corpus that combined between the semantic of golden standards and the semantic of SWQ knowledge domain. From many related documents, reports, and publications we extracted the semantic of the most common discussed SWPQAs and their various discussed definitions in those files. We reached to almost 66 SWPQAs and a wide range of definitions for them. Table 3.3 shows part of the extracted 66 common discussed SWPQAs. The complete extracted 66 SWPQAs and there definitions in addition to the sources they are taken from are presented in Appendix A.

Table 3.3: SWPQAs and their definitions from various sources references.

Att ID	Quality Attribute	Definition(s)
1	Accuracy	<p>Attributes of software that bare on the provision of right or agreed results or effects.</p> <p>Those attributes of the software which provide the required precision in calculations and outputs.</p> <p>This quality factor addresses the concern that programs provide the precision required for each output. Accuracy is important because most computer manipulations are not exact, but are limited approximations.</p> <p>A software product possesses accuracy to the extent that its outputs are sufficiently precise to satisfy their intended use</p> <p>The capability of the software product to provide the right or agreed results or effects with the needed degree of precision.</p> <p>The characteristics of the software which provide the required precision in calculations and outputs</p> <p>(1) A qualitative assessment of correctness, or freedom from error. (2) A quantitative measure of the magnitude of error. <i>Contrast with:</i> precision</p> <p>Correctness</p>

Att ID	Quality Attribute	Definition(s)
		<p>The degree to which a system, as built, is free from error, especially with respect to quantitative outputs. Accuracy differs from correctness; it is a determination of how well a system does the job it is designed for rather than whether it was implemented correctly</p> <p>The capability of the software product to provide the right or agreed results or effects with the needed degree of precision</p> <p>The provision of right or agreed results or effects</p>
7	Complexity	<p>This quality factor addresses the concern that programs not be complex</p> <p>Is the extent to which it is involved or intricate, composed of many interwoven parts?</p> <p>The degree to which a component or system has a design and/or internal structure that is difficult to understand, maintain and verify.</p> <p>A code measure, which is a combination of code, data, data flow, structure and control flow metrics</p> <p>(1) The degree to which a system or component has a design or implementation that is difficult to understand and verify.  (2) Pertaining to any of a set of structure-based metrics that measure the attribute in (1</p>
12	Functionality	<p>This characteristic express the ability of a component to provide the required services, when used under specified conditions</p> <p>The responsibilities assigned to the classes of a design, which are made available by the classes through their public interfaces.</p> <p>A set of attributes that relate to the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs</p>

Att ID	Quality Attribute	Definition(s)
		<p>The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.</p> <p>The extent to which a component satisfies its specifications and fulfills the stated or implied needs of the user</p> <p>The capability of the software product to provide functions that meets stated and implied needs when the software is used under specified conditions.</p> <p>Is the essential purpose of any product or service</p> <p>Is expressed as a totality of essential functions that the software product provides</p> <p>Characteristics relating to achievement of the basic purpose for which the software is being engineered</p>
40	Dependability	<p>This attribute indicates if the component is not self-contained, i.e. if the component depend of other component to provide its specified services</p> <p>Is that property of a computer system such that reliance can justifiably be placed on the service it delivers</p> <p>That property of a system such that reliance can justifiably be placed in the service it provides</p> <p>Availability. The degree to which a system or component is operational and accessible when required for use. Dependability is that property of a computer system such that reliance can justifiably be placed on the service it deliver</p>

Att ID	Quality Attribute	Definition(s)
45	Integrity	<p>The protection of the program from unauthorized access</p> <p>Extent to which unauthorized access to the software or data can be controlled</p> <p>Quality factor addresses the concern that programs must continue to perform their function even under adverse conditions: inputs that are unexpected, improper, or harmful</p> <p>Ability of software to prevent purposeful or accidental damage to the data or software</p> <p>The extent to which access to software or data by unauthorized persons should be controlled</p> <p>The degree to which a system or component or application prevents unauthorized access to, or modification of, computer programs or data.</p> <p>Non-occurrence of improper alterations of information</p> <p>Is the requirement that data and process be protected from unauthorized modification</p> <p>Protection of the program from unauthorized access.</p> <p>The extent to which access to a software component, a component-based software using the software component or the companion data by unauthorized persons can be controlled</p> <p>THE degree to which a system prevents unauthorized or improper access or modification to its code and data or other system resources and/or the degree to which it ensures that data or object state is maintained in a coherent and correct manner. The idea of integrity includes restricting unauthorized user access as well as ensuring that data is accessed properly by its intended users and other software.</p>
65	Readability	The ease with which a developer can read and understand the source code and technical documentation of a system, especially at the detailed source code statement level
66	Productivity	The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

As shown in the table above, the most common and discussed 66 software product quality attributes were extracted from the studied knowledge domain found in various documents and reports we collected earlier. The table shows that every attribute has many definitions came from many sources. If we take a deep look to them, we will see inconsistencies on the semantic of the used concepts. That really makes the researchers in the field confused about those attribute semantic. After completing the ontology that we aim to build through our work, we will show how to use it in order to solve the problem appeared from using various semantics in the definitions of any of the studied attributes.

### 3.4 THE COVERAGE PROCESS AND THE EVALUATION RESULTS

After we prepared the text corpus for the evaluation process (as seen in the table before), we used two tools to help us in conducting the coverage technique. First, for each quality attribute definition(s) we extracted its single and unique concepts using the TextToOnto tool as used before. After that we eliminated the stopping words from the resulted concepts.

Later, in order to know how much does our suggested ontology concepts cover from each attribute definition concepts which were extracted earlier, we used a program created by Kayed [72]; we provided the program with two groups of concepts, the first group consisted of the single concepts of each attribute definition(s), and the second group consisted of our suggested ontology domain concepts; it is the same tool used in the refinement process for the extracted ontology concepts. The results after that appeared with which concepts from our ontology domain covered concepts from each attribute definition(s).

Depending on the results provided by the program, and for each quality attribute definition(s), we counted how many concepts did our ontology domain concepts cover, and calculated the average coverage for each one. Finally we calculated the average of all the resulted coverage averages for all of the attributes definition(s). The result of the coverage process showed that an average of 73% of the definitions concepts was covered by our ontology domain concepts. That was a very good coverage percentage for the studied domain text corpus. Table 3.4 shows part of the results from these steps. The complete results are provided in Appendix B.

Table 3.4: Coverage Process Results.

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
1	Accuracy	assessment	accuracy	17 from 24
		computer	capability	0.708333333
		concern	computer	
		determination	concern	
		extent	degree	
		factor	error	
		freedom	extent	
		job	factor	

		magnitude	freedom	
		measure	measure	
		output	output	
		quality	precision	
		respect	product	
		capability	quality	
		provision	respect	
		right	software	
		system	system	
		accuracy		
		correctness		
		degree		
		error		
		product		
		precision		
		software		
7	Complexity	attribute	attribute	16 from 19
		code	code	0.842105263
		combination	component	
		component	concern	
		concern	control	
		control	data	
		data	degree	
		degree	design	
		design	extent	
		extent	factor	
		factor	implementation	
		flow	measure	
		implementation	quality	
		measure	set	
		metrics	structure	
		quality	system	
		set		
		structure		
		system		
12	Functionality	ability	ability	12 from 16
		achievement	capability	0.75
		capability	characteristic	
		characteristic	component	
		component	design	
		design	extent	
		existence	product	
		express	purpose	
		extent	service	
		product	set	
		purpose	software	
		service	user	
		set		
		software		
		totality		
		user		
40	Dependability	attribute	attribute	7 from 10
		availability	component	0.7
		component	computer	

		computer	degree	
		degree	property	
		property	service	
		reliance	system	
		self		
		service		
		system		
45	Integrity	ability	ability	21 from 30
		access	access	0.7
		application	application	
		code	code	
		companion	component	
		component	computer	
		computer	concern	
		concern	data	
		damage	degree	
		data	extent	
		degree	factor	
		extent	function	
		factor	information	
		function	manner	
		idea	modification	
		information	object	
		integrity	object	
		manner	program	
		modification	quality	
		object	software	
		occurrence	system	
		process	user	
		program		
		protection		
		quality		
		requirement		
		software		
		state		
		system		
		User		
65	Readability	code	code	7 from 8
		developer	developer	0.875
		documentation	documentation	
		ease	ease	
		level	level	
		source	source	
		statement	system	
		system		
66	Productivity	capability	capability	5 from 7
		context	context	0.714285714
		effectiveness	effectiveness	
		expend	product	
		product	software	
		relation		
		software		
<b>The Average of Coverage Averages is : 0.734520723</b>				

As shown in the table above, the evaluation process revealed that our ontology domain concepts covered almost 73% from the given knowledge domain. This result supports our claim; that we can condense the thousands of concepts used to define the 66 most common and discussed software product quality attributes into a smaller set of concepts (100 concepts), and those 100 concepts covered about 73% of the semantic used to define them. In other words, a range of 73% of the semantic of each studied SWPQA is covered by our ontology domain concepts.

What about the uncovered concepts? Can we get benefits from the evaluation process that had been done to our ontology domain concepts in order to enhance the work? Next section will be devoted to answer this question.

### **3.5 ENHANCING ONTOLOGY DOMAIN CONCEPTS**

After studying the results of the evaluation process for the suggested ontology domain concepts, we devoted this section for answering a specific question that says: what the result would be if we collect and study the uncovered concepts from the domain under discussion? The results show that we were able to enhance our ontology domain concepts with a much better coverage percentage. Such results are shown in the discussion below.

#### **3.5.1 THE ENHANCING IDEA AND PROCESS**

As mentioned earlier, ontology domain concepts are considered to be the most important part of the ontology building process, reaching to coherent ontology domain concepts is like accomplishing about 70% of the ontology building process road. In order to make the reached coverage percentage of our ontology domain concepts better, an enhancing idea was suggested.

From the results of the previous section; evaluating the ontology domain concepts through concepts knowledge domain coverage technique, we could reach to the uncovered concepts for the studied domain. A question popped up in our minds which says: can we get benefits from those uncovered concepts in order to enhance our suggested ontology domain concepts? So we took a look again on the concepts of each definition (mentioned earlier in table 3.4), we collected and studied the uncovered concepts for each definition and made a list from them.

A study for the resulted list of the uncovered concepts depending on their appearance frequency in the domain of SWPQAs has been conducted. After that we rearranged the list according to the studied criterion, we found that the frequencies range was between 1 and 5, when we studied them we found that the number of concepts had frequency of 5,4,and 3 were 25 concepts, and concepts that had frequency of 2 or/and 1 were in hundreds, so we chose the top listed 25 concepts and studied them upon if we can add them to our ontology domain concepts and get a noticed better covering average percentage when evaluating the new suggested ontology concepts. Table 3.5 shows the new chosen concepts.



Table 3.5: The top listed 25 uncovered chosen concepts.

Concept	Concept	Concept
means	meeting	idea
capacity	minimum	impact
interface	nature	interval
requirement	people	variety
state	presence	Verification
architecture	relationship	
availability	reliability	
demand	resource	
exchange	risk	
express	testability	

We took the new resulted concepts and merged them with our ontology domain concepts. This gave us a new suggested ontology domain concepts consisted of 125 concepts. Table 3.6 shows the new 125 suggested ontology domain concepts.

Table 3.6: The new 125 Ontology domain concepts list.

Concept	Concept	Concept	Concept
ability	documentation	memory	risk
access	ease	minimum	scope
accessibility	effect	modification	service
accuracy	effectiveness	nature	set
adapt	efficiency	notation	setting
adaptability	effort	number	software
adaptation	environment	object	source
amount	error	objective	specification
applicability	exchange	operating	state
application	express	operation	storage
architecture	extent	operator	structure
attribute	factor	output	system
availability	failure	people	test
capability	freedom	performance	testability
capacity	function	period	throughput
change	functionality	portability	time
characteristic	hardware	precision	understand
code	idea	presence	understandability
component	impact	probability	understanding
computer	implementation	product	uniform
computing	incorporation	program	uniformity
concern	information	property	usability
context	interface	purpose	usage
control	interval	quality	user
data	level	rate	utility
definition	maintenance	relationship	utilization
degree	manner	reliability	variety
demand	mean	requirement	verification
design	meaning	resource	
develop	means	respect	
developer	measure	response	
development	meeting	responsiveness	

### 3.5.2 EVALUATING THE NEW SUGGESTED ONTOLOGY DOMAIN CONCEPTS

In order to see how the new suggested ontology concepts effected on our work, we evaluated it using the same technique (the coverage technique) we used to evaluate our first suggested ontology domain concepts. For each software product quality attribute definition, we counted how many new covered concepts we got when we used the new ontology domain concepts to cover it, and we calculated the new coverage percentage for each one. After that, the average of the covering averages was calculated. Table 3.7 shows the new results.

Table 3.7: The Coverage process results using the new suggested Ontology domain concepts.

Att. ID	The Old Count	The New Count	The new Percentage
1	17 from 24	no change	0.708333333
2	13 from 15	14 from 15	0.933333333
3	6 from 10	no change	0.6
4	4 from 4	no change	1
5	12 from 26	17 from 26	0.653846154
6	9 from 12	no change	0.75
7	16 from 19	no change	0.842105263
8	4 from 7	no change	0.571428571
9	17 from 22	no change	0.772727273
10	4 from 5	no change	0.8
11	38 from 66	42 from 66	0.636363636
12	12 from 16	13 from 16	0.8125
13	5 from 5	no change	1
14	13 from 17	16 from 17	0.941176471
15	10 from 14	no change	0.714285714
16	39 from 61	45 from 61	0.737704918
17	8 from 12	no change	0.666666667
18	13 from 17	no change	0.764705882
19	32 from 45	36 from 45	0.8
20	29 from 36	30 from 36	0.833333333
21	11 from 12	no change	0.916666667
22	33 from 41	35 from 41	0.853658537
23	7 from 13	8 from 13	0.615384615
24	5 from 8	6 from 8	0.75
25	10 from 19	13 from 19	0.684210526
26	14 from 17	16 from 17	0.941176471
27	20 from 44	24 from 44	0.545454545
28	5 from 9	7 from 9	0.777777778
29	9 from 11	10 from 11	0.909090909
30	22 from 40	27 from 40	0.675
31	12 from 30	14 from 30	0.466666667
32	26 from 35	no change	0.742857143
33	31 from 48	33 from 48	0.6875
34	8 from 10	9 from 10	0.9
35	7 from 9	8 from 9	0.888888889
36	15 from 19	no change	0.789473684
37	19 from 24	20 from 24	0.833333333

Att. ID	The Old Count	The New Count	The new Percentage
38	17 from 22	19 from 22	0.863636364
39	12 from 15	13 from 15	0.866666667
40	7 from 10	8 from 10	0.8
41	3 from 4	4 from 4	1
42	3 from 3	no change	1
43	17 from 23	19 from 23	0.826086957
44	20 from 21	no change	0.952380952
45	21 from 30	24 from 30	0.8
46	9 from 13	10 from 13	0.769230769
47	3 from 5	4 from 5	0.8
48	7 from 8	no change	0.875
49	10 from 11	11 from 11	1
50	7 from 11	9 from 11	0.818181818
51	13 from 15	14 from 15	0.933333333
52	7 from 8	no change	0.875
53	12 from 17	13 from 17	0.764705882
54	10 from 15	11 from 15	0.733333333
55	11 from 12	no change	0.916666667
56	13 from 14	no change	0.928571429
57	6 from 11	8 from 11	0.727272727
58	10 from 13	11 from 13	0.846153846
59	4 from 6	no change	0.666666667
60	15 from 22	19 from 22	0.863636364
61	5 from 6	no change	0.833333333
62	7 from 13	no change	0.538461538
63	13 from 16	15 from 16	0.9375
64	8 from 13	9 from 13	0.692307692
65	7 from 8	no change	0.875
66	5 from 7	no change	0.714285714
<b>The New Average For Coverage Averages is : 0.7989858 ≈ 0.80</b>			

From studying the results above, we can see that this step enhanced our ontology domain concepts coverage percentage from seventy three percent to about eighty percent. It is clear that the new suggested ontology domain gave us a much better result in the evaluation process.

The new results of the evaluation process showed that an average of 80% from the semantic used to define one of the 66 studied software product quality attributes are covered by our new suggested ontology domain concepts. This 80% is shared and agreed knowledge concepts among a very large number of experts and practitioners in the field, who used to define software product quality attributes. These new results enhanced our claim that we can condense the thousands of concepts used in the semantic of the most 66 common and discussed software product quality attributes into a smaller set of concepts consists of 125 concepts, which cover about 80% from it. In other words, a range of 80% of the semantic used to define 66 software product quality attributes can be condensed by our ontology domain concepts.

What about the remaining 20% uncovered concepts? When we studied the remaining 20% uncovered concepts, we found that those concepts had not been used in a very large shared manner like the 80% covered concepts in the domain. They were used by some individuals in the field to define one of the discussed software product quality attributes. About 50% of the uncovered concepts had an appearance frequency value of 2 in the studied domain, and the second half of them had 1 as an appearance frequency value in the semantic of the whole definitions. In addition to that, and in their semantic, they were not related to the domain as much as the 80% covered concepts. So we can claim that a large percentage of the uncovered concepts are not important to the shared knowledge that we want to reach as much as the 80% covered concepts, and we can eliminate them from the semantic used to define one of the studied attributes. So not mentioning them as a part of our ontology domain concepts has a small negative effect on our work.

The idea of condensing concepts used in the studied semantic resulting with information loss for sure, but if we take a look on the condensing results, we will see that we condensed the thousands of concepts used in the semantic of the studied domain into a smaller set of concepts, as we saw earlier we managed to condensed the 2750 extracted concepts that have a coverage average of almost 100% for the concepts in the domain into 125 concepts that have a coverage average of almost 80% for the concepts used in the studied domain, we managed to condense about 95% of the used concepts with just 20% of information loss.

## CHAPTER 4

### EXTRACTING ONTOLOGY DOMAIN RELATIONSHIPS

In this chapter we extracted general relationships between the new concepts of the suggested ontology domain after studying and filtering the results of two tools. We presented the resulted relationships as groups. After that, a general lattice representation for part of the resulted relationships was done. Later, we listed each concept in the ontology domain with other concepts in the same domain that appeared with them when studying the extracted relationships.

#### 4.1 EXTRACTING RELATIONSHIPS BETWEEN CONCEPTS

Extracting structured information and relationships from text between concepts has been widely studied lately and became a rich subject of research. Many works and documents and even theses have been published about this subject exclusively. When we decided to proceed in this step; extracting and creating relationships between concepts of our ontology domain, we found that if we want to create a detailed and complete ontology relation taxonomy, then this work will be large enough to be a thesis by itself. Such details go beyond our work scope, and we suggested it to be done in the future work. So, we went for extracting and presenting a basic and general representation of the relationships that we could extract between our ontology domain concepts.

In this step, and in order to extract relationships between our suggested ontology domain concepts, we used and studied the results of two tools. Firstly, we used the TextToOnto tool in order to extract relationships (associations) between concepts. We provided the tool with the text corpus we prepared previously to extract concepts from, and also we provided it with the concepts we want to study the relationships between. When we ran this step the used tool provided us with about 1467 relationships. Figure 4.1 shows part of the results of this step.

Premise	Conclusion	Conclusion Freque...	Support	Confidence	Abs. Freq.	Pattern Names	Property
measur	usage	7	0.001	0.053	1		1
storag	efficien	12	0.001	0.167	1		1
compon	environ	25	0.001	0.012	1		1
effect	capabi	21	0.001	0.067	1		1
system	hardwar	8	0.001	0.007	1		1
time	measur	16	0.002	0.054	2		2
maintain	code	17	0.001	0.167	1		1
amount	inform	8	0.001	0.053	1		1
output	control	6	0.001	0.067	1		1
properti	environ	25	0.001	0.1	1		1
measur	number	7	0.002	0.106	2		2
modifi	data	20	0.002	0.111	2		2
function	product	19	0.001	0.013	1		1
program	characterist	29	0.002	0.037	2		2
purpos	code	17	0.002	0.167	2		2
hardwar	system	43	0.001	0.083	1		1
effect	context	7	0.001	0.067	1		1
design	number	7	0.001	0.039	1		1
system	applic	14	0.002	0.014	2		2
inform	usag	7	0.001	0.126	1		1
level	compon	32	0.001	0.053	1		1
respect	utili	9	0.001	0.167	1		1
accuraci	output	10	0.002	0.333	2		2

Figure 4.1: Part of the resulted relationships using TextToOnto tool.

To get benefits from the resulted relationships from the TextToOnto tool, we used them as an input for another tool; a tool created by Kayed et al [71]. Such a tool accepts the relationships resulted from the TextToOnto tool as an input, and implements a counting and relevancy algorithm on them. This tool provided us with about 65 relationships categorized in groups of concepts. Figures 4.2 and 4.3 show part of using this tool and part of its results. Later, we took the 65 resulted groups from this tool, studied them, filtered them upon containing our ontology domain concepts or not (because the text corpus we provided to the tool contained much more concepts than our ontology domain did). Table 4.1 shows the results of this step.

ID	Con1	con2	con3	n1	n2
1441	user	set	7.73E-04	11	1
1442	user	softwar	0.002320186	59	3
1443	user	system	7.73E-04	51	1
1444	user	understand	7.73E-04	14	1
1445	util	characterist	7.73E-04	33	1
1446	util	execut	7.73E-04	6	1
1447	util	measur	7.73E-04	17	1
1448	util	meet	7.73E-04	10	1
1449	util	natur	7.73E-04	2	1
1450	util	resourc	0.002320186	20	3
1451	util	space	7.73E-04	3	1
1452	util	standard	7.73E-04	7	1
1453	varieta	compon	0.00154679	30	2
1454	varieta	oper	7.73E-04	21	1
1455	verif	compon	7.73E-04	30	1
1456	verif	evalu	7.73E-04	6	1
1457	verif	interfac	7.73E-04	7	1
1458	verif	set	7.73E-04	11	1
1459	wast	execut	7.73E-04	6	1
1460	wast	purpos	0.00154679	9	2
1461	wast	rate	7.73E-04	5	1
1462	work	abl	7.73E-04	26	1
1463	work	effort	7.73E-04	26	1
1464	work	period	7.73E-04	6	1
1465	work	process	7.73E-04	6	1
1466	work	product	7.73E-04	22	1
1467	work	system	0.00154679	51	2
*	(AutoNumber)			0	0

Figure 4.2: Using TextToOnto results as an input for the second MS Access tool.

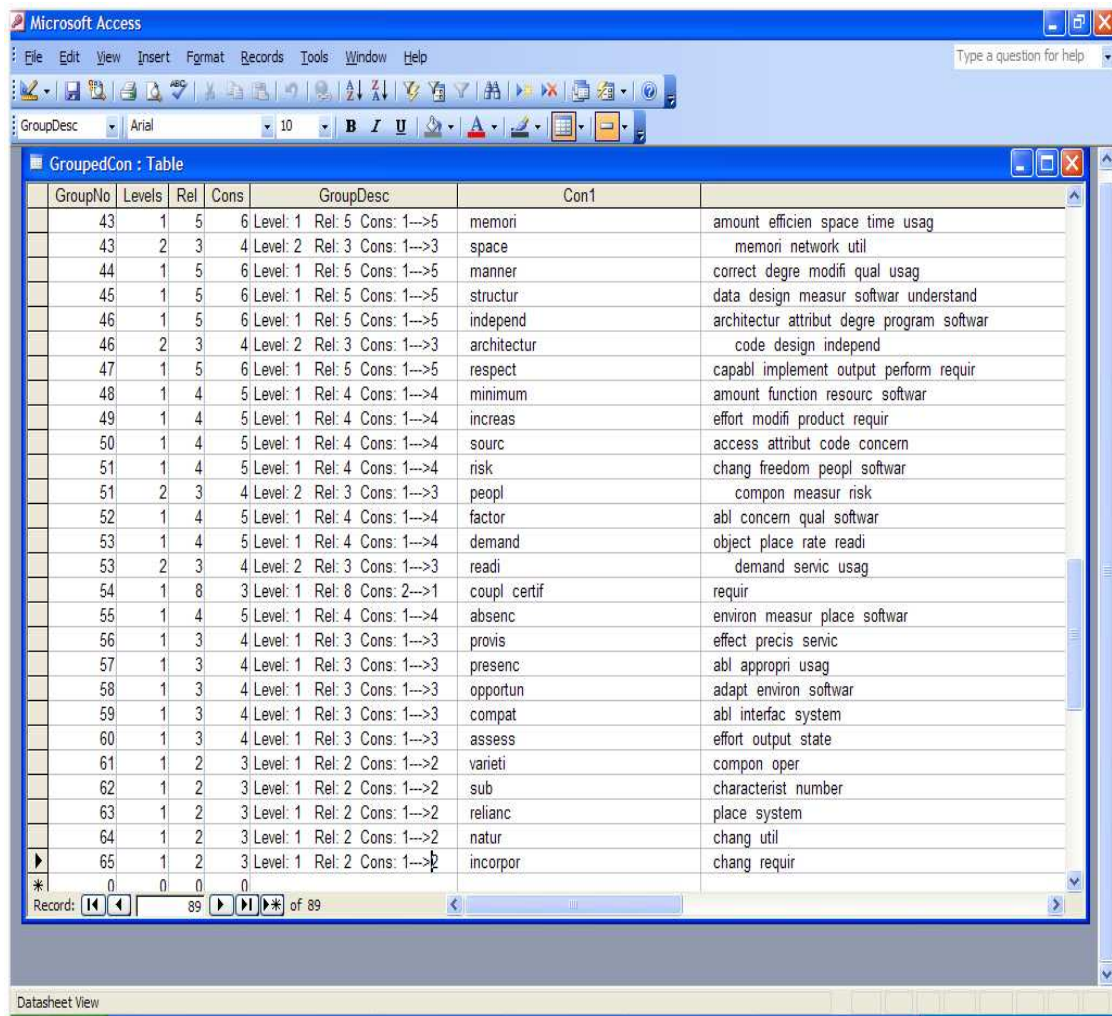


Figure 4.3: Part of the resulted relationships groups from using the second tool.

Table 4.1: The resulted relationships between groups of our Ontology concepts after filtering.

Group No	Level	Group 1	Group 2
1	1	Software, system, requirement characteristic, function,	Attribute, design, test ,user
2	1	Performance, degree, component	Data, effort, function, software ,system
2	2	Data, effort	Component, degree, performance, requirement, function ,software, system, user
3	1	Environment, program, ability	Component, requirement, software
4	1	Extent, time ,product	software ,system
5	1	Operate (ion), ease, change	Environment, software, system
6	1	Resource, specification, implementation	extent

Group No	Level	Group 1	Group 2
7	1	Capability, code	Environment, performance, requirement
8	1	Modification, measure	Ability, requirement
9	1	amount ,state	Function, resource, software
10	1	Application, applicability, understand	Modification, requirement, system ,user
11	1	Level, modification	Product, software, system
12	1	Service, access	Requirement, system, user
13	1	Effect, set	Attribute, resource, system, user
14	1	Develop(er), failure	Ease, product, software
15	1	Output, computer	Amount, system
16	1	Efficiency, quality	characteristic
17	1	meeting	Modification, performance
18	1	Documentation, concern	Software, system
19	1	Hardware, purpose	Environment, software
20	1	Number	amount ,specification
21	1	information	Ability, data, degree, documentation, exchange, object, software, system
22	1	control	Access, attribute, characteristic, data, degree, operation, user, idea
22	2	idea	Ease
23	1	precision	Requirement, service
24	1	Adapt, utility (ization)	characteristic
25	1	probability	Availability, express, extent ,failure, function, performance, program, time
26	1	interface	software
27	1	Mean, context	change
28	1	probability	Ability, characteristic, code, degree, function ,time, Verification
28	2	Verification	Component, interface, set
29	1	Freedom, uniform	Environment
30	1	Storage, reliability	code
31	1	response	Design, measure, meet ,system ,throughput, time
31	2	throughput	Rate, requirement, response, time
33	1	error	Maintenance, measure, precision, program, requirement, system
33	2	maintenance	Adaptability, attribute, ease, error, impact ,state
33	3	impact	component ,maintenance ,system
34	1	Scope, accuracy	extent
35	1	Usage, usability	resource
36	1	work ,period	system
37	1	relationship	Attribute, degree, function, modification, product
38	1	notation	Definition, degree, implementation, quality, uniform



Group No	Level	Group 1	Group 2
38	2	Definition	Implementation, level, notation
39	1	testability	Characteristic, code ,effort ,extent, number
40	1	memory	Amount, efficiency, time, usage
41	1	manner	degree, modification, quality, usage
42	1	structure	data ,design, measure, software, understand
43	2	architecture	code ,design,
44	1	respect	Capability, implementation, output, performance, requirement
45	1	minimum	Amount, function, resource, software
46	1	source	Access, attribute, code ,concern
47	1	risk	Change, freedom ,people, software
47	2	people	Component, measure, risk
48	1	factor	Ability, concern, quality, software
49	1	demand	Object, rate,
50	1	presence	Ability, usage
51	1	variety	Component, operation
52	1	nature	Change, utility
53	1	incorporation	Change, requirement

If we have a look at the table above, which shows the relationships between groups of our ontology concepts, we would see a group number column; which refers to the ID of the group of concepts that had a relationship in between. The second column shows the level number which refers to the number of levels of the relationships when concepts from the same group have relationships with other concepts. Before filtering, every group was consisted of two levels: level one indicated that there is a relationship between a group of concepts; call it g1, and another group of concepts, say g2, while in level 2, the reverse of the relationship is given, that is the relationship that g2 has with g1. So we filtered and eliminated them from the table above and said that g1 had a relationship with g2 and vice versa instead. Also the second and the third level from a group may show that a concept or (concepts), which considered as a part of a group of concepts, had a relationship with other concepts from the domain. We did not eliminate this type of relationships and we used it later in our representation. Also, we would see the group 1 column; which refers to a side of the group of concepts that had a relationship with another group of concepts shown in group 2 column.

## 4.2 A LATTICE REPRESENTATION OF THE RELATIONSHIPS

After we studied and filtered the resulted relationships from the tools we used, as shown in table 4.1 above, we considered representing them in a general form of lattice representation, but if we did it to all relationships groups it will be a large and complex representation in addition to time consuming. So we took part of those groups and represented them as shown in the Figures as follows:

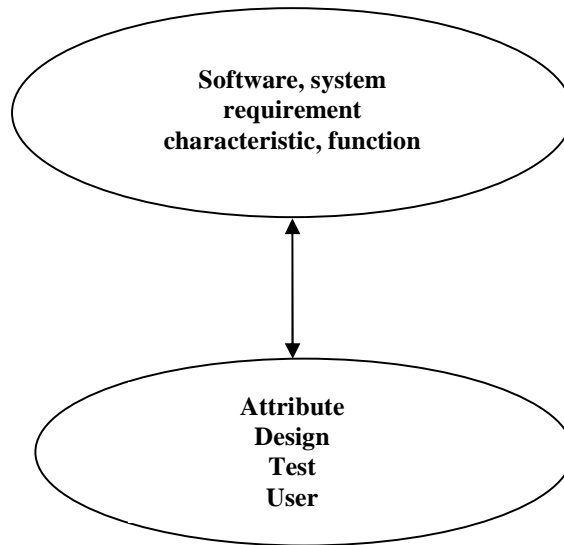


Figure 4.4: Group 1 relationship Lattice representation: One Level Relationship.

Figure 4.4 above shows one level relationship, which indicates that a group of concepts consists of (software, system, requirement, and function) has a relationship with another group of concepts consists of (attribute, design, test, and user) and vice versa. When we looked at the semantic used to define the studied attributes, we found that concepts from group one came in the semantic along with concepts from the second group.

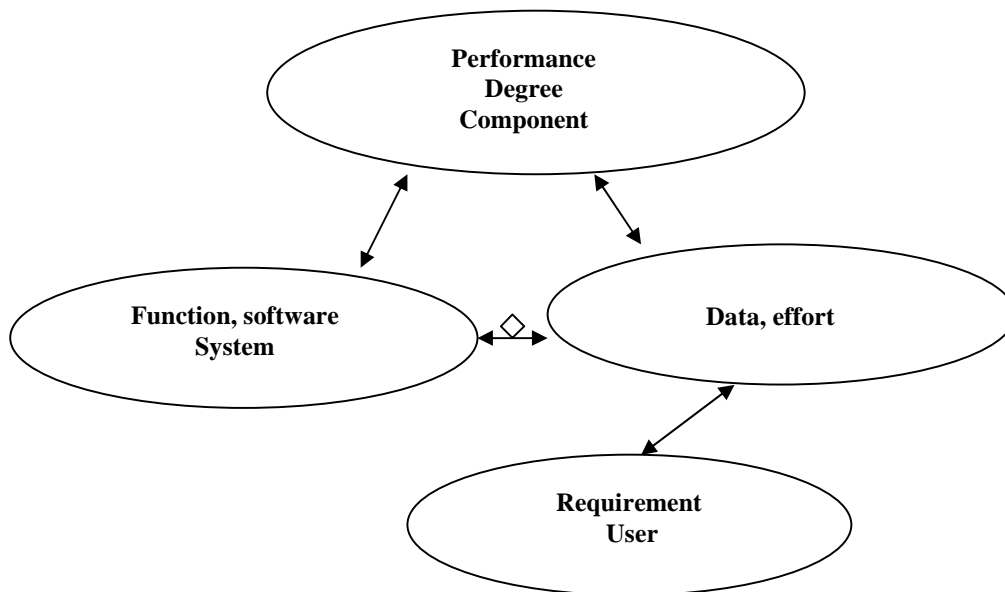


Figure 4.5: Group 2 relationship Lattice representation: Two Levels Relationship.

Figure 4.5 above shows a two levels relationship, which indicates that a group of concepts consists of (performance, degree, and component) has a relationship with a group of concepts consists of (function, software, system, data, effort) and vice versa. We separated the second group concepts from each other because we needed to connect a part from it (data, effort) with a second level group of concepts consists of (requirements, user). These groups are used together in the semantic when defining some of the studied attributes. The diamond shape on the arrow between some concepts means that those concepts together considered as a group, but separated for a needed reason.

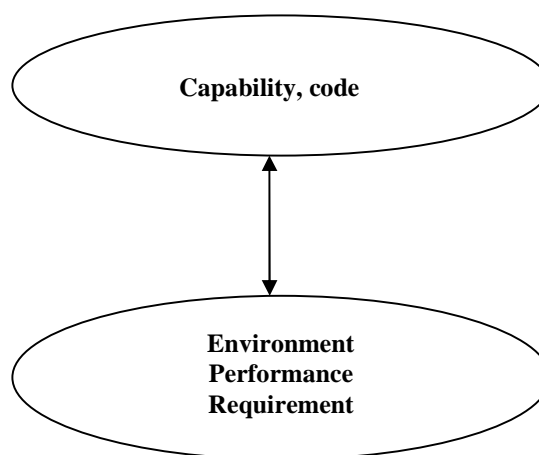


Figure 4.6: Group 7 relationship Lattice representation: One Level Relationship.

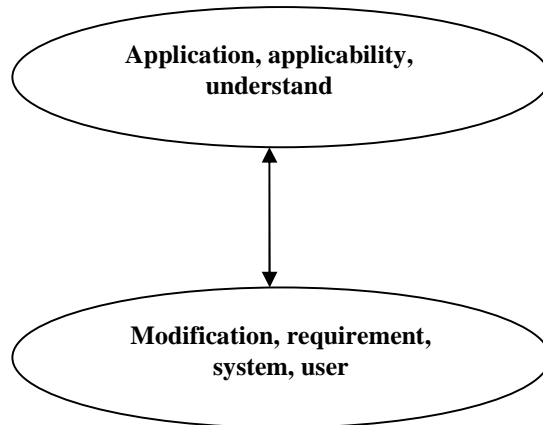


Figure 4.7: Group 10 relationship Lattice representation: a One Level Relationship.

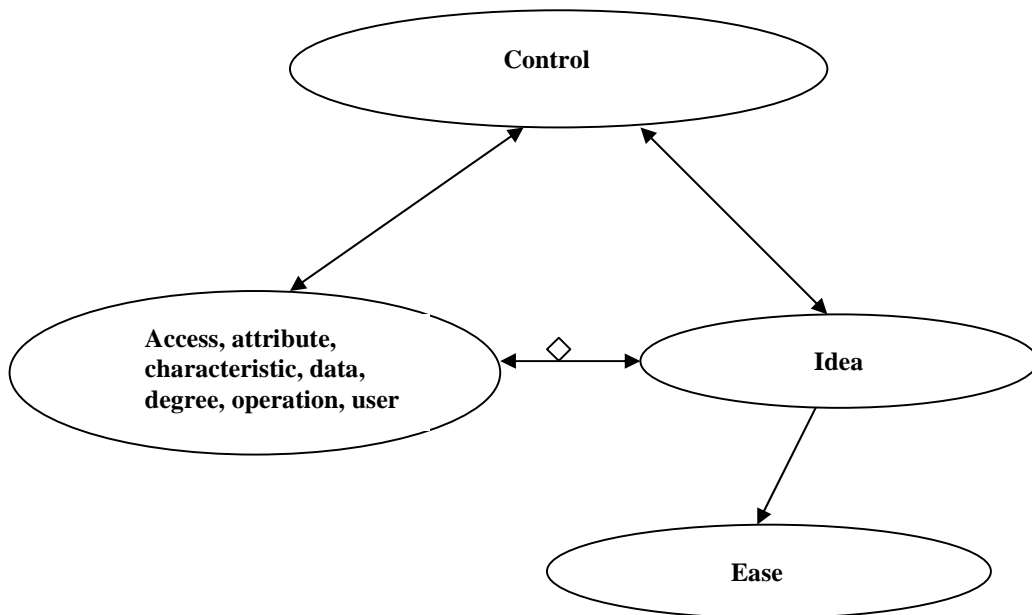


Figure 4.8: Group 22 relationship Lattice representation: Two Level Relationship.

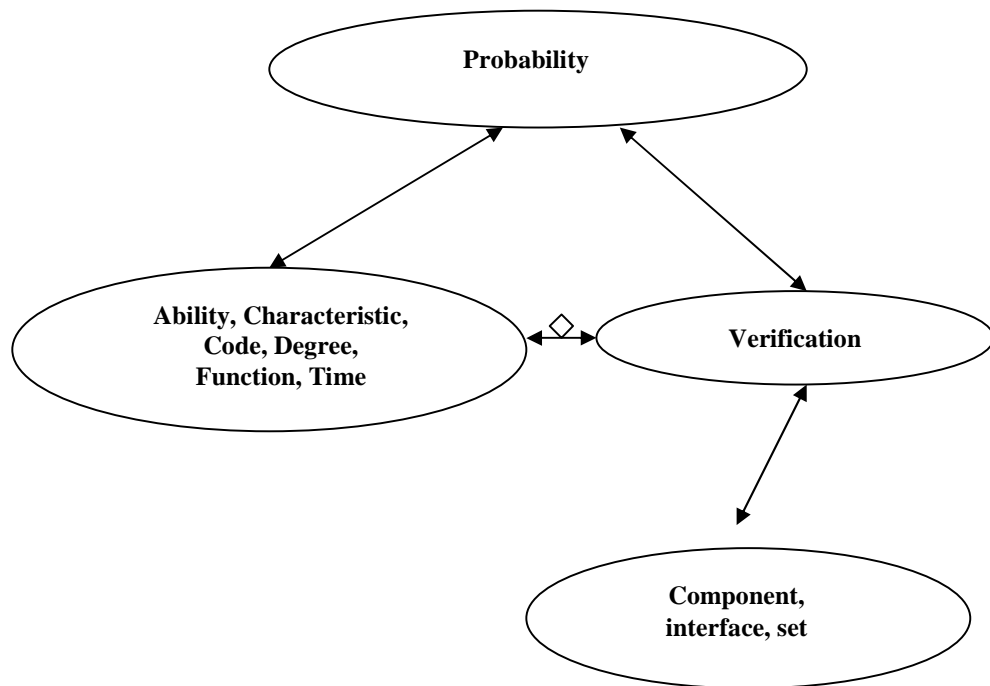


Figure 4.9: Group 28 relationship Lattice representation: Two Level Relationship.

Similarly, Figures 4.6 to 4.9 show similar ideas, but for different groups.

Figure 4.10 shows a three levels relationship, the first level consists of the concept (error) as the first group, and it has a relationship with another group of concepts consists of (measure, precision, program, requirement, system and maintenance). We separated (maintenance) and (system) from it because we needed to connect them with another group in the second level, the second level consists of the group (adaptability, attribute, ease, state, and impact) which has a relationship with (maintenance) in the first level of the relationship. We separated (impact) from the group because it has a relationship with the third level group of concepts (component) and with the concept (system) in the first level. Again, the diamond shape on the arrows between concepts means that those concepts together considered as a group, but separated for a needed reason.

So, for all the groups that appears in the table, we can make a lattice representation as shown, either for one or two or three levels relationship.

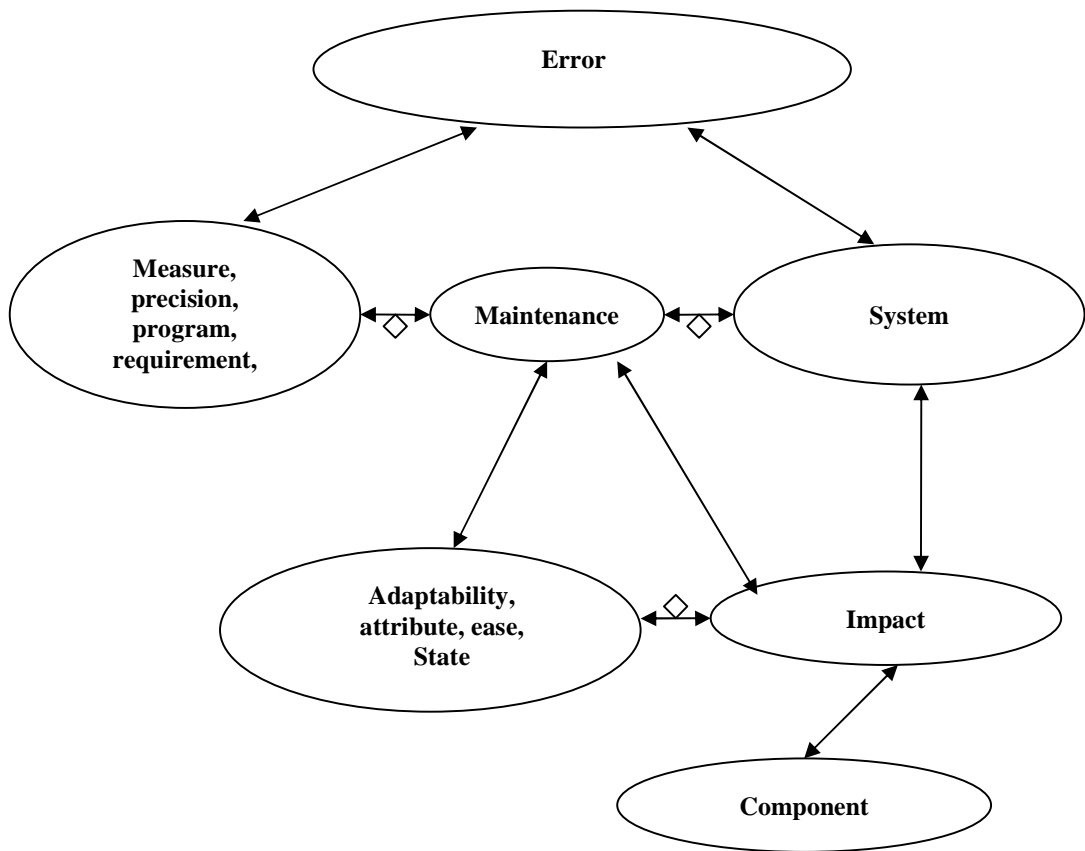


Figure 4.10: Group 33 relationship Lattice representation: Three Level Relationship.

### 4.3 LISTING EACH CONCEPT RELATIONSHIPS WITH OTHERS IN THE ONTOLOGY DOMAIN

Finally, after representing the relationships groups as shown earlier, we looked at them again and we knew that if we studied them we could list each concept relationships with other concepts in the ontology domain. So we took them and reviewed them again, but this time for each single concept in our ontology domain. We studied what other concepts in the ontology domain (in all groups) appeared with each concept in the previously presented relationships groups. The result, shown in table 4.2, gave us each concept in the ontology domain and other concepts from the domain it has a relationship with.

Table 4.2: Each Ontology domain concept relationships with other concepts in the domain.

ID	Concept	Other Concepts that have relationships with the first concept
1	ability	Component, requirement, software, Modification, measure, information, portability, factor, presence, portability
2	access	Requirement, system, user, control, source
3	accessibility	Requirement, system, user, control, source
4	accuracy	Extent
5	adapt	Characteristic
6	adaptability	Maintenance
7	adaptation	Environment, Software, Maintenance, Modification
8	amount	Function, resource, software, Output, computer, Number, memory, minimum, computing
9	applicability	Modification, requirement, system ,user
10	application	Modification, requirement, system ,user
11	architecture	code ,design
12	attribute	Software, system, requirement characteristic, function, Effect, set, control, maintenance, relationship, source, quality, meaning, means, portability, property, responsiveness
13	availability	Probability
14	capability	Environment, performance, requirement, respect
15	capacity	Function, System, Requirement, Software
16	change	Environment, software, system, Mean, context, risk, nature, means
17	characteristic	Attribute, design, test ,user, Efficiency, quality, control, Adapt, utility (ization), portability, testability, property
18	code	Environment, performance, requirement, portability, Storage, reliability, testability, architecture, source
19	component	Data, effort, function, software ,system, Environment, program, ability, Verification, impact, people, variety
20	computer	Amount, system
21	computing	Amount, system
22	concern	Software, system, source, factor
23	context	Change
24	control	Access, attribute, characteristic, data, degree, operation, user, accessibility
25	data	Performance, degree, component, Component, degree, performance, requirement ,software, system, user, Information, control, structure
26	definition	Notation, Implementation, level, notation, meaning, means
27	degree	Data, effort, function, software ,system, information, control, portability, relationship, notation, manner, level
28	demand	Object, rate
29	design	Software, system, requirement characteristic, function, response, structure, architecture
30	develop	Ease, product, software
31	developer	Ease, product, software
32	development	Ease, product, software
33	documentation	Software, system, information, meaning, means,

ID	Concept	Other Concepts that have relationships with the first concept
		<b>understanding</b>
34	ease	Environment, software, system, Develop(er), failure, idea, maintenance
35	effect	Attribute, resource, system, user
36	effectiveness	Efficiency, Quality, Program, Function, Factor
37	efficiency	Characteristic, memory, effectiveness
38	effort	Performance, degree, component, requirement ,software, system, user, Testability
39	environment	Component, requirement, software, Operate(ion), ease, change, Capability, code, Hardware, purpose Freedom, uniform, adaptation
40	error	Maintenance, measure, precision, program, requirement, system
41	exchange	Information
42	express	Probability
43	extent	software ,system, Resource, specification, implementation, probability, Scope, accuracy, testability
44	factor	Ability, concern, quality, software, effectiveness
45	failure	Ease, product, software, probability
46	freedom	Environment, risk
47	function	Attribute, design, test ,user, Performance, degree, component, amount ,state, probability, relationship, minimum, capacity, effectiveness, responsiveness
48	functionality	Attribute, design, test ,user, Performance, degree, component, amount ,state, probability, relationship, minimum
49	hardware	Environment, software
50	idea	Ease
51	impact	Maintenance, component, system, interval
52	implementation	Extent, notation, definition, respect
53	incorporation	Change, requirement
54	information	Ability, data, degree, documentation, exchange, object, software, system
55	interface	Software, Verification
56	interval	Time, Period, impact, minimum
57	level	Performance, Degree
58	maintenance	Error, Adaptability, attribute, ease, impact ,state, adaptation
59	manner	degree, modification, quality, usage
60	mean	Change
61	meaning	Definition, Attribute, Documentation
62	means	Change, Definition, Attribute, Documentation
63	measure	Ability, requirement, response, error, structure, people
64	Meeting (meet)	Modification, performance, response
65	memory	Amount, efficiency, time, usage, storage
66	minimum	Amount, function, resource, software, interval
67	modification	Ability, requirement, Application, applicability, understand, Product, software, system, meeting, relationship, manner, adaptation , understanding
68	nature	Change, utility
69	notation	Definition, degree, implementation, quality, uniform



ID	Concept	Other Concepts that have relationships with the first concept
70	number	amount ,specification, testability
71	object	Information, demand
72	objective	Information, demand
73	operating	Environment, software, system, control, variety
74	operation	Environment, software, system, control, variety
75	operator	Environment, software, system, control, variety
76	output	Amount, system, respect
77	people	Risk, Component, measure
78	performance	Data, effort, function, software ,system, Capability, code, meeting, probability, respect, level
79	period	System, time, interval
80	portability	Ability, Program, Software, Attribute, utilization
81	precision	Requirement, service, error
82	presence	Ability, usage
83	probability	Availability, express, extent ,failure, function, performance, program, time, Ability, characteristic, code, degree
84	product	software ,system, Level, modification, Develop(er), failure, relationship
85	program	Component, requirement, software, probability, error, effectiveness, portability, responsiveness, utilization
86	property	Characteristic, Attribute, Software
87	purpose	Environment, software
88	quality	Characteristic, notation, manner, factor, Attribute, effectiveness
89	rate	Throughput, demand
90	relationship	Attribute, degree, function, modification, product
91	reliability	Code
92	requirement	Attribute, design, test ,user, Data, effort, Environment, program, ability, Capability, code, Modification, measure Application, applicability, understand, Service, access, precision, throughput, error, respect, incorporation, accessibility, capacity, understanding
93	resource	Extent, amount ,state, Effect, set, Usage, usability, minimum
94	respect	Capability, implementation, output, performance, requirement
95	response	Design, measure, meet ,system ,throughput, time, responsiveness
96	responsiveness	Response, function, software, Attribute, program, time
97	risk	Change, freedom ,people, software
98	scope	Extent
99	service	Requirement, system, user, precision,
100	set	Attribute, resource, system, user, Verification
101	Setting	Attribute, resource, system, user, Verification
102	software	Attribute, design, test ,user, Performance, degree, component, Data, effort, Environment, program, ability, Extent, time ,product, Operate(ion), ease, change, amount ,state, Level, modification, Develop(er), failure, portability Documentation, concern, Hardware, purpose, information, interface, structure, minimum, risk, factor, adaptation,

ID	Concept	Other Concepts that have relationships with the first concept
		capacity, property, responsiveness, utilization
103	source	Access, attribute, code ,concern, accessibility
104	specification	Extent, Number
105	state	Function, resource, software, maintenance, uniformity
106	storage	Code, memory
107	structure	data ,design, measure, software, understand, understanding
108	system	Attribute, design, test ,user, Performance, degree, component, Data, effort, Extent, time ,product, Operate(ion), ease, change, Application, applicability, understand, Level, modification, Service, access, Effect, set, Output, computer Documentation, concern, information, response, error, impact, work ,period, accessibility, capacity, computing, understanding, uniformity
109	test	Software, system, requirement characteristic, function
110	testability	Characteristic, code ,effort ,extent, number
111	throughput	Response, Rate, requirement, time
112	time	Period, software ,system, probability, response, throughput, memory, interval, responsiveness
113	understand	Modification, requirement, system ,user, structure, Documentation
114	understandability	Modification, requirement, system ,user, structure, Documentation
115	understanding	Modification, requirement, system ,user, structure, Documentation
116	uniform	Environment, notation, uniformity
117	uniformity	State, uniform, System
118	usability	Resource
119	usage	Resource, memory, manner, presence,
120	user	Software, system, requirement characteristic, function, Data, effort, Application, applicability, understand, Service, access, Effect, set, control, accessibility, understanding, utilization
121	utility	Characteristic, nature
122	utilization	Software, User, Program, Portability
123	variety	Component, operation
124	verification	Component, interface, set
125	work	System

Those presented relationships in the table above as you will see later, support us in a way or another in our contribution and that what the next chapter is discussing.

## CHAPTER 5

### CONCLUSIONS AND FUTURE WORK

In this chapter we present and discuss the conclusions of our research; our final results and how we used them to contribute in the studied domain are presented among the conclusions. Future work are suggested at the end of this chapter.

#### 5.1 CONCLUSIONS

SWPQAs discipline is considered in the emerging phase, and it suffers from the typical symptoms of any relatively evolving disciplines. SWPQAs are currently in the phase in which terminologies, principles, and methods are still being defined, consolidated, and agreed. In particular, there is a lack of consensus on the concepts and terminologies used in the semantic of this field. Studies showed that inconsistencies in the semantic used different research attributes proposals often occur.

In our research we focused on studying SWPQAs concepts and terminologies that current software quality proposals, documents, and reports present. We prepared text corpora from them to be used in a tool to extract the most discussed and used concepts from it. After that experts were asked to study and filter the resulted concepts and provided them to us.

An evaluation phase depended on a coverage technique was done to the resulted concepts, followed by an enhancing step to the evaluated ontology domain concepts which led us to increase the number of the suggested concepts in the ontology domain, after that a coverage evaluation is done again to the new suggested ontology domain concepts.

In order to extract general relationships among the suggested ontology domain concepts, we returned to the prepared text corpus again and ran out two tools on it. We studied them, filtered them, listed them and represented part of them using a lattice representation.

Through completing the steps of our work, which have been previously summarized, we reached to many important results. These results are studied filtered and used to support our claim. The sections below present our work final results and how we used them to support our contribution in the field.

### 5.1.1 PRESENTING FINAL RESULTS

Through completing the steps of our work we reached to many important results, which could be summarized as follows:

- **Ontology domain concepts:** resulted from preparation, studying, and filtering a text corpus related to the domain of software product quality attributes. First, we reached to a result that we can condense the semantic of thousands of concepts used to define the discussed 66 attributes into a smaller set of concepts consisted of 100 concepts with a coverage percentage for the studied knowledge domain of 73%. But after an evaluation process and what came from studying its results, we enhanced our suggested ontology domain concepts to be consisted of 125 concepts with an average of 80% of coverage percentage for the studied knowledge domain, (final results of the new 125 suggested ontology domain concepts are shown in Appendix C).
- **Relationships between groups of concepts for the suggested ontology domain:** resulted from studying and filtering the results of two tools; the associations resulted from using the TextToOnto tool after we provided it with a related knowledge domain text corpus and concepts, After that we took those associations and provided another tool created by Kayed et al [71] with them. This process provided us with relationships between groups of concepts from our suggested ontology domain. Again we studied them, filtered them, and finally presented them, (the resulted relationships are shown in Appendix D).
- **Each Software product quality attribute concepts that belong to our ontology domain:** from the evaluation and enhancing phase for the ontology domain concepts. We reached to every attribute definition concepts that belong to our ontology domain concepts, (the resulted concepts are shown in Appendix E).
- **Finally, Relationships between each concept in the ontology domain and other concepts also in the same domain:** resulted from studying the groups of relationships that appeared in appendix D in addition to the domain itself (these results are shown in appendix F).

### 5.1.2 OUR CONTRIBUTION

By reaching and providing those final results discussed in the previous section, let us don't forget that our main focus in this work is to provide experts mainly, researchers, and practitioners in the field of SWQ with an ontology to be considered as a base and a common agreement knowledge. This supports them in defining the most common discussed SWPQAs (66 attributes) that we extracted from the fields documents and reports, and reaching to a common, shared, and consistent semantic for them. This solves the inconsistencies of the semantic appears in the definitions of those attributes among many documents and reports as shown earlier.

We have presented the conceptualization of the common discussed SWPQAs by an ontology, which considered as a first in this specific domain.

We also have condensed the semantic of thousands of concepts used to define any of the 66 discussed SWPQAs into a smaller set of concepts consists of 125 concepts with a high percentage of coverage average for the studied domain reached to 80% of coverage.

Also by the results of this research, we provide the experts and practitioners in the field of SWQ who want to define any of the discussed 66 attributes in the domain with an ontology which contains a set of common used and agreed concepts (for each attribute definition, and for general studied domain), and also with relationships between them (as groups, or relationships between concepts belong to the same attribute that can be inferred from the presented relationships, or relationships between concepts in the studied domain). So when an expert decide to define an attribute from the discussed domain, we suggest two ways to use our ontology to have a consistent semantic with other definitions in the field. First after an expert wrote down his own definition he can compare the concepts he used in the semantic of his definition with our ontology domain concepts and try to map from his used concepts to our concepts from the ontology domain if needed, and try to use the provided relationships between them to connect the semantic of the concepts together in a strong, meaningful, and consistent manner. The second way that we suggest to reach to an agreed semantic is that before the expert write down his own definition we recommend to take a look on the ontology domain and use its concepts and relationships along with his experience as a base knowledge to consist the definition he wants.

If experts in the field follow one of these suggested ways when defining one of the discussed SWPQAs, eventually they will reach to a common, agreed, and consistent semantic between them, and this will be a successful way to solve the presented problem.

In addition to this, our ontology provides a base to evaluate any related presented definition semantic for one of the 66 studied attributes. The way of doing that is if a high percentage of the concepts used in the semantic of the presented definition are covered by our ontology domain, the presented definition semantic can be accepted, but if not we claim that it is a weak semantic to be used defining such an attribute.

## 5.2 FUTURE WORK

By working on our thesis step by step, many ideas and issues were appeared but not accomplished yet because of time, resources, and other constraints. We would like to suggest them as a future work. To mention:

- Providing a description for each concept used in the provided ontology domain in order to help experts and practitioners who want to use them while defining one of the discussed software product quality attributes.
- A formal representation for the ontology.

- Some suggestions for the tools we used to be more user friendly, as the possibility of copying records as all in all not a record in a time, and putting some notes in the interface that help the users to use the tool easily.
- Extracting and presenting the detailed types of relationships between our ontology domain concepts.
- Completing the lattice representation for the ontology domain relationships.
- Use another approach to evaluate and enhance the ontology domain, and compare the results with the results we already had. A critique on evaluating the ontology may be done by conducting set of experiments and trying to deploy the ontology in some applications to show how effective, useful, and expressive is the proposed ontology to the audience in a context of software engineering domain and especially to the audience in the context of software quality attributes domain.
- Using our ontology domain and convert it into an Arabic ontology for the same studied domain but in Arabic language.

## REFERENCES

- [1] Abram A.; Sellami A., “Initial modeling of the measurement concepts in the ISO vocabulary of terms in metrology”, In Proc. of IWSM2002, Magdeburg (Germany), Oct. 2002.
- [2] Alvero A.; Santana de Almeida E.; Romero de Meira S.,”Quality Attributes for a Component Quality Model”, in the proceeding of 10th International Workshop on Component-Oriented Programming, Glasgow, Scotland, at ECOOP 2005, Glasgow, Scotland, July 25--29, 2005.
- [3] Azuma M., “SQuaRE: The next generation of the ISO/IEC 9126 and 14598 International Standards Series on software product quality”. In Proc. of European Software Control and Metrics, London (England), April 2001.
- [4] Baker G.; Brass A.; Bechhofer S.; Goble C.; Paton N.; Tambis R., “Transparent access to multiple biological information sources”, Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology , ISMB-98, Menlow Park, California, AAAI Press, 25-34, June 28-July 1 1998.
- [5] Balzer R.; Dayer D.; Feahling M., Sawnders S.,”Specification Based Computing Environments”, Proceedings of the 8th International Conference on Very Large Data Bases, ISBN: 0-934613-14-1, PP: 237 – 279, 1982.
- [6] Barbacci M., “Software Quality Attributes: Modifiability and usability”, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA 15213, Sponsored by the U.S. Department of Defense, 2004.
- [7] Barbacci M.; Ellison R.; et al., “Quality Attribute Workshop Participants Handbook”, special report, CMU/SEI-2000, SR-001, January 2000.
- [8] Barbacci M.; et al., “Quality Attributes”, Technical report CMU/SEI-95, TR- 021, ESC-TR-95-021 , Dec. 1995.
- [9] Basili V.; Rombach D., “The Tame project: Towards improvement oriented software environments”, IEEE Transactions on Software Engineering, Vol. 14, Issue 6, PP: 758-773, June 1998.
- [10] Berander P.; et al., “Software quality attributes and trade-offs”, Blekinge Institute of Technology, produced in a Ph.D. course on “Quality attributes and trade-offs”. The 11 Ph.D. students that followed the course all worked in the same research project: BESQ (Blekinge – Engineering Software Qualities), <http://www.bth.se/besq>, June 2005.
- [11] Bevan N., “Quality in Use: Meeting User Needs for Quality”, Journal of System and Software, Vol. 49, Issue 1, PP: 89-96, 15 Dec. 1999.
- [12] Black A., “Software Quality Assurance In A Remote Client/Contractor Context”, A thesis submitted in fulfillment of the requirements for the degree of master of science of Rhodes University, Dec. 2005.

- [13] Boehm B.; Brown J.; Lipow M., "Quantitative evaluation of software quality, International Conference on Software Engineering", Proceedings of the 2nd international conference on Software engineering, IEEE Computer Society Press, San Francisco, California, United States, Pages: 592 – 605, 1976.
- [14] Boehm B.; Brown J., Kaspar H.; Lipow M.; McLeod G.; Merritt M., "Characteristics of Software Quality", Elsevier, North Holland, 1st Edition, ISBN: 0444851054, 1978.
- [15] Børretzen J., "The impact of component-based development on software quality attributes", borrette@idi.ntnu.no, Essay, DT8100, 2005.
- [16] Brank J.; Grobelnik M.; Mladenić D., "A Survey of Ontology Evaluation Techniques", in proc. Of the 8th international multi conference information society, SIKDD, 2005.
- [17] Brewster C.; Alani H.; Dasmahapatra S.; Wilks Y., "Data-driven ontology evaluation". In Proc. of the 4th International Conference on Language Resources and Evaluation, Lisbon, 2004.
- [18] Brooks F., "No Silver Bullet - essence and accidents of software engineering", Computer magazine, Vol. 20, Issue 4, PP: 10-19, Apr. 1987.
- [19] Burnstein I., "Practical Software Testing", Springer-Verlag, New York, Inc., 1st Edition, ISBN 0-387-95131-8, 2003.
- [20] Calero C.; Ruiz F.; Piattini M., "Ontologies for Software Engineering and Software Technology", Springer Berlin Heidelberg, New York, ISBN-10 3-540-34517-5, 1998.
- [21] Conde D., "Software Product Management: Managing Software Development from Idea to Product to Marketing to Sales (Execenablers)", Aspature Books, 1st Edition, ISBN 1-58762-202-5, Sep. 2002.
- [22] Corcho O.; Fernandez M.; et al., "Building Legal Ontologies with METHONTOLOGY and WebODE", In Benjamins, R.; Casanovas, P.; Breuker, J. & Gangemi, A. (ed.): "Law and the Semantic Web".
- [23] Curtis B.; Hefley B.; Miller S., "People Capability Maturity Model® (P-CMM®)", Carnegie Mellon University, Software Engineering Institute, Version 2.0, MM-001, 2001.
- [24] De los Angeles Martin M.; Olsina L., "Toward an Ontology for SW metrics and indicators as the Foundation For Catalog Web system". Web Congress, First Latin American, Santiago, Chile, Vol. 30 , Issue 3, 10-12,PP: 103 -113, ,2003.
- [25] Dellschaft K.; Staab S., "On How to Perform a Gold Standard Based Evaluation of Ontology Learning". International Semantic Web Conference, PP: 228-241, 2006.
- [26] Devedzic V., "Understanding Ontological Engineering". Commune ACM, Vol. 45, Issue 4, PP: 136-144, 2002.
- [27] Dromey R., "A model for software product quality", IEEE Transactions on Software Engineering, Vol. 21, Issue 2, pp: 146-163, 1995.
- [28] Dromey R., "Concerning the Chimera [software quality]", IEEE Software, Vol. 13, Issue. 1, PP: 33-43, 1996.



- [29] Dromey R., “Software Product Quality: Theory, Model and Practice. Software Quality Institute”, Griffith University, Nathan, Brisbane, QLD 4111, Australia, accessed on 2008 April, Available online at:  
<http://scholar.google.com/url?sa=U&q=http://www.sqi.gu.edu.au/docs/sqi/misc/SPQ-Theory.pdf>, 1998.
- [30] Ehrig M.; Haase P.; Hefke M.; Stojanovic N., “Similarity for ontologies a comprehensive framework”. In Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM 2004, DEC 2004. <http://citeseer.ist.psu.edu/ehrig04similarity.html>.
- [31] Ernst M., “Ontology building: A survey of editing tools”, Online, accessed 2008 April, Available from URL: <http://www.xml.com/pub/a/2002/11/06/ontologies.html>.
- [32] ESA Board for Software Standardisation and Control (BSSC), “Guide to software quality assurance”, european space agency / agence spatiale européenne, 8-10, rue Mario-Nikis, 75738 Paris Cedex, France, ESA PSS-05-11, Issue 1, Revision 1, Mar. 1995.
- [33] Fitzpatrick R.; Higgins C., “Usable software and its attributes: A synthesis of software quality”, European Community law and human-computer interaction, In People and Computers XIII, Proceedings of HCI98 Conference, Springer, London, UK 1998.
- [34] Florac W., “Software Quality Measurement: A Framework for Counting Problems and Defects” , CMU/SEI-92-TR-22, Software Engineering Institute -Carnegie Mellon University, Pittsburgh, Pennsylvania, Sep. 1992.
- [35] Floridi L., “Blackwell Guide to the Philosophy of Computing and Information”, Preprint version of chapter “Ontology”, Oxford: Blackwell, PP: 155– 166, 2003.
- [36] FZI Karlsruhe; AIFB Karlsruhe, “KAON: the Karlsruhe ontology and semantic web framework developer’s guide for KAON 1.2.7”, University of Karlsruhe, Germany, 2004.
- [37] Galin D., “Software Quality Assurance -from theory to implementation”, Pearson Addison Wesley, ISBN 0-201-70945-7, Sep. 2004.
- [38] Gang Z.; Gao Y.; Meersman R., "An ontology-based approach to business modeling", In Proceedings of the International Conference of Knowledge Engineering and Decision Support, Vol. 31, Issue 7, 2005.
- [39] Garcia F.; Bertoa M.; Calero C., “Towards a Consistent Terminology for Software measurements”, Information of Software Technology, Vol. 48, Issue 8, PP: 631-644, 27 June 2005.
- [40] Glossary of Computerized System and Software Development Terminology, a reference material for Investigators and other FDA personnel. Available on [http://www.fda.gov/ora/inspect\\_ref/igs/gloss.html](http://www.fda.gov/ora/inspect_ref/igs/gloss.html), Visited 21-4-2008.
- [41] Glossary of Software Engineering terms, SEGlossary, Digital publications LLC Version 1.0d, 2005, available on <http://www.shellmethod.com/refs/seglossary.pdf>.
- [42] Godbole N., “Software Quality Assurance - principles and practice”, Alpha science international ltd., ISBN 1-84265-176-5, 2004.
- [43] Grady R., “Practical software metrics for project management and process improvement”, Prentice Hall, ISBN: 978-0137203840, 1992.

- [44] Gruber T., "Ontology", to appear in the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag, 2008.
- [45] Guarino N.; "Formal ontology and information systems", Proceedings of the international conference on Formal Ontology in Information Systems, Trento, Italy, Amsterdam: IOS Press, Netherlands, PP: 3-15,1998.
- [46] Guarino N.; Persidis A., "Evaluation framework for content standards", Technical Report, OntoWeb Deliverable 3.5, Padova; 2003.
- [47] Horch J., "Practical Guide to Software Quality Management", 2nd Edition, ISBN: 1580535275, 2003.
- [48] Hoyle D., "ISO 9000 Quality Systems Handbook", Elsevier, Fifth Edition, ISBN 0 7506 6785 0, Dec. 2005.
- [49] [Http://www.sqa.net/iso9126.html](http://www.sqa.net/iso9126.html), "An overview of the ISO 9126-1 software quality model definition, with an explanation of the major characteristics", Article, last visit on 27-4-2008.
- [50] Humphrey W., "Introduction to the Personal Software Process", Addison-Wesley Pub Co, 1st Edition, ISBN: 978-0201548099, 1996.
- [51] Humphrey W., "Introduction to the team software process", Addison-Wesley Pub Co, 1st Edition, ISBN: 978-0201477191, 2000.
- [52] Humphrey W., "Managing the software process", Addison-Wesley Pub. Co, 1st Edition, ISBN: 978-0201180954, 1989.
- [53] Hyatt L.; Rosenberg L., "A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality", Product Assurance Symposium and Software Product Assurance Workshop, ESTEC, Noordwijk, the Netherlands, European Space Agency, p.209, 1996.
- [54] IEEE, "IEEE Standard Glossary of Software Engineering Terminology/IEEE Std 610.12-1990", (Revision and redesignation of IEEE Std 729-1983), Sponsor Standards Coordinating Committee of the IEEE Computer Society, Approved September 28, 1990 IEEE Standards Board, ISBN: 978-1559370677, 1990.
- [55] ISO, International Organization for Standardization, "ISO 9000:2000, Quality management systems - Fundamentals and vocabulary", 1st Edition, ISBN: 92-67-10332-6, 2001.
- [56] ISO, International Organization for Standardization, "ISO 9000-2:1997, Quality management and quality assurance standards — Part 2: Generic guidelines for the application of ISO 9001, ISO 9002 and ISO 9003", 1997.
- [57] ISO, International Organization for Standardization, "ISO 9000-3:1998 - Quality management and quality assurance standards – Part 3: Guidelines for the application of ISO 9001\_1994 to the development, supply, installation and maintenance of computer software (ISO 9000-3:1997)", 1998.
- [58] ISO, International Organization for Standardization, "ISO 9001:2000, Quality management systems – Requirements", 3rd Edition, 2000.

- [59] ISO, International Organization for Standardization, "ISO 9004:2000, Quality management systems - Guidelines for performance improvements", 2000.
- [60] ISO, International Organization for Standardization, "ISO 9126-1:2001, Software engineering - Product quality, Part 1: Quality model", 2001.
- [61] ISO. International Organization for Standardization, "International Vocabulary of Basic and General Terms in Metrology". International Standard organization, Geneva, Switzerland, 2nd Edition, 1993.
- [62] ISO/IEC, "ISO/IEC 15504-1:2003, Information technology – Process assessment – Part 1: Concepts and vocabulary" 2004.
- [63] ISO/IEC, "Software and Systems Engineering - Guidelines for the Application of ISO/IEC 9001:2000 to Computer Software". International Standards Organization, Geneva, witzerland, 2004.
- [64] Jacobson I.; Booch G.; Rumbaugh J., "The Unified Software Development Process", Addison Wesley Longman, Inc., ISBN: 0201571692, Feb. 1999.
- [65] Jarrar M., "Towards methodological principles for ontology engineering". PhD Dissertation, Vrije Universities Brussels, 2005.
- [66] Jarrar M.; Demy J.; Meersman R., "On reusing conceptual data modeling for ontology engineering". In: Aberer K, March S, Spaccapietra S (eds.), Journal on Data, Vol. 2800, Issue 1, PP: 185-207, 2003.
- [67] Jetter A., "Assessing software quality attributes", A thesis submitted in fulfillment of the requirements for the degree of master of science, Software Evolution & Architecture Lab, Department of Informatics, University of Zurich, Zurich, Switzerland, January 2007.
- [68] Jones C., "Making measurement work", Crosstalk, The Journal of Defense Software Engineering, Vol. 16, Issue 1, PP: 15, 19, Jan. 2001.
- [69] Jones B.; Storey V.; Sugumaran V.; Ahluwalia P., "A semiotic metrics suite for assessing the quality of ontologies". Data and Knowledge Engineering, Vol. 55, Issue 1, PP: 84 – 102, Oct. 2005.
- [70] Juran J.; et al., "Juran's Quality Handbook", McGraw-Hill, ISBN: 007034003X, Fifth Edition, 1999.
- [71] Kayed A.; Hirzallah N.; Al Shalabi L. A.; and Najjar M., "Building Ontological Relationships: A new approach", Journal of the American Society for Information Science and Technology, John Wiley & Sons Inc, 111 River ST , Hoboken , USA , NJ, 07030, 2008.
- [72] Kayed A., "Building e-laws ontology: New approach", Springer, Lecture Notes in Computer Science ,Springer-Verlag Vol. 3762 , Germany, ISBN/ISSN: 0302-9743, pp 826-830,2006.
- [73] Kayed A.; Colomb R., "Using BWW Model to Evaluate Building Ontologies in CGs Formalism". Information Systems. Vol. 30, Issue 5, PP: 379 – 398, ISSN: 0306-4379, July 2005.
- [74] Kayed A.; Colomb R., "Extracting Ontological Concepts for Tendering Conceptual Structures", Data & Knowledge Engineering, Vol. 40, Issue 1, Pages: 71 - 89, 2002.

- [75] Khosravi KH.; Gael Y, "On issues with software quality models", 2005. Jun. 2006. Available on [www.iro.umontreal.ca/~sahraouh/-qaoose2005/paper7.pdf](http://www.iro.umontreal.ca/~sahraouh/-qaoose2005/paper7.pdf).
- [76] Kim H., "Representing and Reasoning about Quality using Enterprise Models". PhD thesis, Dept. of Mechanical and Industrial Engineering, University of Toronto, Canada, 1999.
- [77] Kitchenham B.; Pfleeger S., "Software quality: the elusive target [special issues section]", IEEE Software, Vol.13, Issue 1, pp: 12-21, ISSN: 0740-7459, 2002.
- [78] Kitchenham B.; Hughes R.; Linkman S., "Modeling software measurement data". IEEE Transactions on Software Engineering, Vol. 27, Issue: 9, PP: 788-804, ISSN: 0098-5589, Sept. 2001.
- [79] Knowledge Management Group (WBS); Research Group Knowledge Management (WIM), "Extensions to the Karlsruhe Ontology and Semantic Web Framework, KAON Extensions, Developer's Guide for KAON Extensions 0.6", University of Karlsruhe, Germany, Aug. 2003.
- [80] Kruchten P., "The Rational Unified Process an Introduction", Addison Wesley Longman, Inc., 3rd Edition, ISBN: 978-0321197702, Dec., 2003.
- [81] Land R., "Measurements of Software Maintainability", In Proceedings of Second Conference on Software Engineering Research and Practice in Sweden (SERPS), Blekinge Institute of Technology Research Report 10, 2002. Available on <http://www.mrtc.mdh.se/publications/0436.pdf>.
- [82] Maedche A.; Motik B.; Stojanovic L.; Studer R.; Volz R., "An infrastructure for searching, reusing and evolving distributed ontologies". In Proceedings of the twelfth international conference on World Wide Web, Budapest, Hungary, ACM Press., PP 439-448, 2003.
- [83] Maedche A.; Volz R., "The ontology extraction and maintenance framework text-to-onto". In Proceedings of the ICDM'01 Workshop on Integrating Data Mining and Knowledge Management, 2001.
- [84] Marciniak J., "Encyclopedia of software engineering", 2 vol. set, 2nd ed., Chichester Wiley, ISBN: 978-0-471-37737-5, 2002.
- [85] Mark C.; Weber W.; Charles V.; Garcia F.; Suzanne M.; et al, "Capability Maturity Model for Software", Technical Report, Carnegie Mellon University, Software Engineering Institute, TR-024, Version 1.1, 1993.
- [86] Marko G.; Mladeni D., "Automated Knowledge Discovery in Advanced Knowledge Management", Journal of Knowledge Management, Vol. 9, Issue 5, PP: 132-149, 2005.
- [87] McCall J.; Richards P.; Walters G., "Factors in Software Quality", Nat'l Tech. Information Service, General Electronic Co Sunnyvale Calif, Vol. 1, 2 and 3, 1977.
- [88] McGarry J.; Card D.; et al., "Practical Software Measurement Objective Information for Decision Makers", Addison-Wesley, ISBN 4-320-09741-6, 2002.
- [89] Mika P., "Social networks and the semantic web: the next challenge", IEEE Intelligent Systems, Vol. 20, Issue 1, PP: 80-93, 2005.

- [90] Noy N.; McGuinness D., "Ontology development 101: A guide to creating your first ontology", Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, and Technical Report SMI-2001-0880, Stanford Medical Informatics, California, USA, 2001.
- [91] Obrst L., "Ontologies for Semantically Interoperable Systems", Conference on Information and Knowledge Management, Proceedings of the twelfth international conference on Information and knowledge management, Industry session 2: meditation and data sharing, PP: 366 – 369, 2003.
- [92] Ontology, computer science, Online, accessed on 2008 April, Available from URL [http://en.wikipedia.org/wiki/Ontology\\_\(computer\\_science\)#\\_note-0](http://en.wikipedia.org/wiki/Ontology_(computer_science)#_note-0).
- [93] Ontoprise GmbH, "How to work with OntoEdit — user's guide for OntoEdit version 2.6" Online, accessed 2008 April, available from URL [http://www.ontoprise.de/documents/tutorial\\_ontoedit.pdf](http://www.ontoprise.de/documents/tutorial_ontoedit.pdf).
- [94] Pérez G., "Some ideas and examples to evaluate ontologies". Technical Report KSL-94-65, Knowledge Systems Laboratory, Stanford, 1994.
- [95] Pérez G., "Towards a framework to verify knowledge sharing technology", Expert Systems with applications, Vol. 11, Issue 4, PP: 519–529, 1996.
- [96] Parekh V.; Gwo J.; Finin T.; "Mining Domain Specific Texts and Glossaries to Evaluate and Enrich Domain Ontologies", International Conference on Information and Knowledge Engineering, PP: 533-540, USA, CSREA, ISBN 1-932415-27-0, 2004.
- [97] Pressman R., "Software Engineering, a practitioner's approach", sixth Edition, Mc Graw Hill Companies, ISBN: 978-0073019338, April 2005.
- [98] Presson E.; Tsai J.; Bowen T.; Post J.; Schmidt R., "Software Interoperability and Reusability Guidebook for Software Quality Measurement", Boeing Aerospace Co., Seattle, WA, Rome Air Development Center (RADC), Griffiss AFB, NY, RADC-TR-83-174, July 1983.
- [99] Randell B.; Naur P., "Software Engineering", Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th Oct.1968.
- [100] Robert P.; Malaka R., "A task-based approach for ontology evaluation". In Proc. ECAI 2004 Workshop on ontology Learning and Population, PP: 9–16, 2004.
- [101] Rumbaugh J.; Balha M.; Premelani W., "Object-Oriented Modeling and Design with UML". Printicehall, International Edition, 2nd Edition, ISBN: 9780131968592, Dec. 2004.
- [102] Russell R., "Defining Software Quality" - An Essay - [www.io.com/~richardr](http://www.io.com/~richardr), February 2002 Updated April 2004.
- [103] S2ESC Plans and Policies, Fundamental Policies, "FP-03 Software Product Quality", online, accessed on 2008 April, available from URL [http://standards.computer.org/sesc/s2esc\\_pols/polIndex.htm](http://standards.computer.org/sesc/s2esc_pols/polIndex.htm).
- [104] Scacchi W.; Jensen C.; Noll J.; Elliott M., "Multi-Modal Modeling, Analysis and Validation of Open Source Software Requirements Processes", Intern. J. Information Technology and Web Engineering, Vol. 1, Issue 3, PP: 49-63, 2006.

- [105] Schulmeyer G.; Mcmanus J., "Software Quality Assurance handbook", Prentice Hall PTR, 3 Edition, ISBN-13: 978-0130104700, Sep. 1998.
- [106] SEI, "Software Engineering 2004", Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A Volume of the Computing Curricula Series, Aug.2004.
- [107] Smith B., "Ontology", Draft version of chapter published in Luciano Floridi (ed.), Blackwell Guide to the Philosophy of Computing and Information, Oxford: Blackwell, PP:155-166, 2003.
- [108] Smith B.; Welty C., "Ontology: Towards a New Synthesis", Ogunquit, Maine, USA.,ACM1-58113-377-4/01/0010, FOIS'01, Oct. 17-19, 2001.
- [109] Software Engineering Institute, CMMI® Web Site, Carnegie Mellon, online, accessed on 2008 April, available from URL <http://www.sei.cmu.edu/cmmi/cmmi.html>, 2004.
- [110] Software Measurements Guide Book, Software productivity Consortium Services Corporation, Van Nostrand Reinhold Company, ISBN: 978-0442020095 Version 02.01.00, 1995.
- [111] Sommerville I., "Software Engineering", Pearson Addison Wesley, Seventh Edition, ISBN: 020139815, 2007.
- [112] Spacer IBM Certified Solution Designer, "IBM Rational Unified Process V7.0", IBM. online, Accessed on 2008 April, available from URL <http://www-03.ibm.com/certify/certs/38008003.shtml>.
- [113] Spinellis D., "Software engineering glossary", IEEE Software, version control, part I.Vol. 22, Issue 5, PP: 107, Sept./Oct. 2005.
- [114] Standard glossary of terms used in Software Testing, Version 1.2,Produced by the 'Glossary Working Party' International Software Testing Qualification Board, Editor : Erik van Veenendaal, The Netherlands, 2006.
- [115] U.S. Department of Transportation Federal Aviation Administration, Handbook Volume II Digital Systems Validation, Atlantic city international airport, New Jersey 08405, Technical Center, May 1992.
- [116] Usrey M.; Dooley K., "The Dimensions of Software Quality", Quality Management Journal, Vol. 3, Issue 3, PP: 67-86, 1996.
- [117] Volker J.; Vrande D.; Sure Y., "Automatic evaluation of ontologies (AEON)".In Proc. of the 4th International Semantic Web Conference (ISWC'05), Springer Verlag Berlin-Heidelberg, Vol. 3729, PP: 716-731, Nov. 2005.
- [118] Wallace D.; Ippolito L.; Kuhn D., "High Integrity Software Standards and Guidelines", National Institute of Standards and Technology Special Publication 500-204,Gaithersburg, MD 20899 GPO Stock Number is SN003-03171-2, July 1992.
- [119] William R.; Pawlowski S.; Volkov V., "Requirements interaction management", ACM computing surveys, Vol. 35, Issue 2, PP: 132-190, June 2003.
- [120] Wordreference.com: WordNet® 2.0, Princeton University, Princeton, NJ. , online, Accessed on 2008 April, available from URL [WWW.Wordreference.com](http://WWW.Wordreference.com).

- [121] Zeyu Gao J.; Jacob Tsao H. S.;Wu Y.; “Testing and Quality Assurance for Component-Based Software”, Artech House, Inc., ISBN 1-58053-480-5, 2003.
- [122] Zhang J., “Quality Oriented Exploration Techniques for Component Based Architectures”, Technische University Eindhoven Department of Mathematics and Computer Science, Master’s Thesis, Eindhoven, August 2005.
- [123] Zieliński K.; Szmuc T., “Software Engineering: Evolution and Emerging Technologies”, IOS Press, ISBN 1-58603-559-2, 2005.
- [124] Zolet F.; Oliveira K.; Regina A., “Modeling Task Knowledge to Support Software Development”, ACM International Conference Proceeding Series, Vol. 27, Proceedings of the 14th international conference on Software engineering and knowledge engineering, Ischia, Italy, PP: : 35 - 42 , ISBN:1-58113-556-4 , 2002.

## APPENDICES

### APPENDIX A

The complete common extracted SWPQAs from various documents and reports related to the field of study, and their different definitions found in them.

Table A.1: The complete common SWQPAs extracted from different sources and their definitions.

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
1	Accuracy	Attributes of software that bare on the provision of right or agreed results or effects.	10
		Those attributes of the software which provide the required precision in calculations and outputs.	98
		This quality factor addresses the concern that programs provide the precision required for each output. Accuracy is important because most computer manipulations are not exact, but are limited approximations.	115
		A software product possesses accuracy to the extent that its outputs are sufficiently precise to satisfy their intended use	14
		The capability of the software product to provide the right or agreed results or effects with the needed degree of precision.	114
		The characteristics of the software which provide the required precision in calculations and outputs	105
		(1) A qualitative assessment of correctness, or freedom from error. (2) A quantitative measure of the magnitude of error. <i>Contrast with:</i> precision	54
		Correctness	116
		The degree to which a system, as built, is free from error, especially with respect to quantitative outputs. Accuracy differs from correctness; it is a determination of how well a system does the job it is designed for rather than whether it was implemented correctly	102
The capability of the software product to provide the right or agreed results or effects with the needed degree of precision	103		



Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		The provision of right or agreed results or effects	118
2	Adaptability	Attributes of software that relate to on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered	10
		The capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered	10 114 103
		The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed	54
		The degree to which a system can be used, without modification, in applications or environments other than those for which it was specifically designed	102
		Characterizes the ability of the system to change to new specifications or operating environments.	49
		The opportunity for its adaptation to different specified environment	118
3	Analyzability	Attributes of software that relate to the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.	10
		The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.	114
		The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or the capability to identify the parts to be modified	103
		Characterizes the ability to identify the root cause of a failure within the software	49
4	Attractiveness	The capability of the software product to be attractive to the user	114
5	Availability	The product's readiness for use on demand	118
		The degree to which a component or system is operational and accessible when required for use. Often expressed as a percentage (probability)	114 , 54
		Readiness for usage	8
		Is the requirement that data and processes be protected from denial of service to authorized users?	8

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		The system's readiness for delivery of service, or reliability, the system's continuity	7
		The policies required to provide a particular level of availability, such as checkpoint, recovery and restart	12
		The probability that the program ( software ) is performing successfully ( meeting requirements) , according to specification , at a given point of time	110
6	Changeability	A set of attributes that bear on the effort needed to make specified modifications.	67
		Attributes of software that relate to the effort needed for modification, fault removal or for environmental change	10
		"The capability of the software product to enable a specified modification to be implemented."	10
		Characterizes the amount of effort to change a system.	49
7	Complexity	This quality factor addresses the concern that programs not be complex	115
		Is the extent to which it is involved or intricate, composed of many interwoven parts?	115
		The degree to which a component or system has a design and/or internal structure that is difficult to understand, maintain and verify..	114
		A code measure, which is a combination of code, data, data flow, structure and control flow metrics	113
		(1) The degree to which a system or component has a design or implementation that is difficult to understand and verify. (2) Pertaining to any of a set of structure-based metrics that measure the attribute in (1).	54
8	Compliance	Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions Where appropriate certain industry (or government) laws and guidelines need to be complied with, i.e. SOX. This sub-characteristic addresses the compliant capability of software.	10, 49

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		Adherence to application-related standards, conventions, regulations in laws and protocols.	118
9	Consistency	<p>Those attributes of the software which provide for uniform design and implementation techniques and notation</p> <p>This quality factor addresses the concern that the source code syntax and constructs in programs be implemented uniformly "Those characteristics of software which provide for uniform design and implementation techniques and notation"</p> <p>The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a component or system.</p> <p>Commands consistent with environs</p> <p>Uniform notation, terminology, and symbology through each definition level</p>	<p>98,105</p> <p>115</p> <p>114,54</p> <p>29</p> <p>116</p>
10	Co-existence	The capability of the software product to co-exist with other independent software in a common environment sharing common resources.	10
11	Efficiency	<p>This characteristic express the ability of a component to provide appropriate performance, relative to the amount of resources used;</p> <p>Further categorized into execution efficiency and storage efficiency and generally meaning the use of resources, e.g. processor time, storage</p> <p>The code executes its intention without waste of resources</p> <p>A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions</p> <p>(As-is utility characteristics): Code possesses the characteristic efficiency to the extent that it fulfills its purpose without waste of resources</p> <p>Degree of utilization of resources (processing time, storage, communication time) in performing functions.</p> <p>Quality factor addresses the concern that programs optimally use any computer resources</p> <p>The amount of computing resources and code required by the Software (program) to perform a function</p>	<p>2</p> <p>67 , 87</p> <p>67 , 14</p> <p>67, 60</p> <p>10 , 14</p> <p>98</p> <p>115</p> <p>105,110</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		Software utilization of resources	105
		The degree to which a system or component performs its designated functions with minimum consumption of resources.	54
		Ability of a software to place as few demands as possible on hardware resources, such as processor time, memory space occupied, or network bandwidth, to achieve a given task.	110
		Rate of value and waste added per resource consumed	116
		Efficient to use	110
		Use of resources execution and storage	122
		This is an attribute that is used to evaluate the ability of a software system to perform its specified functions under stated or implied Measurements and TMM Levels conditions within appropriate time frames. One useful measure is response time—the time it takes for the system to respond to a user request	19
		Is a characteristic that captures the ability of a correct software product to provide appropriate performance in relation to the amount of resources used. Efficiency can be considered an indication of how well a system works, provided that the functionality requirements are met.	123
		The measure of resources usage such as: memory, CPU utilization, disk space, network bandwidth, screen real estate and amount of user interaction to complete key tasks	102
		This characteristic is concerned with the system resources used when providing the required functionality. The amount of disk space, memory, network etc. provides a good indication of this characteristic. As with a number of these characteristics, there are overlaps.	49
12	Functionality	This characteristic express the ability of a component to provide the required services, when used under specified conditions	2
		The responsibilities assigned to the classes of a design, which are made available by the classes through their public interfaces.	67

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>A set of attributes that relate to the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs</p> <p>The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.</p> <p>The extent to which a component satisfies its specifications and fulfills the stated or implied needs of the user</p> <p>The capability of the software product to provide functions that meets stated and implied needs when the software is used under specified conditions.</p> <p>Is the essential purpose of any product or service</p> <p>Is expressed as a totality of essential functions that the software product provides</p> <p>Characteristics relating to achievement of the basic purpose for which the software is being engineered</p>	<p>10</p> <p>114</p> <p>121</p> <p>103</p> <p>49</p> <p>49</p> <p>118 60</p>
13	Installability	<p>Attributes of software that relate to the effort needed to install the software in a specified environment.</p> <p>The capability of the software product to be installed in a specified environment.</p> <p>Characterizes the effort required to install the software</p>	<p>10 , 60</p> <p>10 ,60</p> <p>49</p>
14	Interoperability	<p>The effort required to couple the system to an other system</p> <p>Attributes of software that relate to its ability to interact with specified systems.</p> <p>The ability of two or more systems or components to exchange information and to use the information that has been exchanged.</p> <p>The capability of the software product to interact with one or more specified components or systems.</p> <p>The Effort required to couple the software of one system to the software of another system.</p>	<p>67 , 87</p> <p>10 , 60</p> <p>10, 87 And 54</p> <p>10, 60,103</p> <p>98, 105</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>How easy it is to interface the software with another system</p> <p>Effort required interconnecting or relating two different applications, running possibly in different computing environment</p> <p>The degree to which the software can be connected easily with other systems and operated.</p> <p>The extent to which a software component can be assembled with a possibly wide variety of component-based software systems employing the component or with other software components</p> <p>The effort required to couple a software component with other programs in general, not necessarily with component-based software systems employing the component or with other software components</p> <p>The extent to which a software system will function or communicate correctly, reliably and robustly with other system using externally defined interfaces (hardware or software) or communications protocols.</p>	<p>105</p> <p>110</p> <p>19</p> <p>121</p> <p>121</p> <p>102</p>
15	Learnability	<p>Attributes of software that relate to the users' effort for learning its application (for example, operation control, input, output).</p> <p>The capability of the software product to enable the user to learn its application.</p> <p>Easy to learn how to use</p> <p>Easy to learn; novices can readily start getting some work done</p> <p>Learning effort for different users, i.e. novice, expert, casual etc.</p> <p>The effort required for the user to learn its application, operation, input, out</p>	<p>10, 60</p> <p>114 , 103</p> <p>29</p> <p>6</p> <p>49</p> <p>118</p>
16	Maintainability	<p>This characteristic describes the ability of a component to be modified;</p> <p>The effort required to locate and fix a fault in the program within its operating environment</p>	<p>2</p> <p>67, 10 , 87, 118,122</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		How easy is it to understand modify and retest?	67,10,14
		The ease of maintenance and upgrade	118
		To be testable: Code possesses the characteristic testability to the extent that it facilitates the establishment of verification criteria and supports evaluation of its performance.	10
		To be understandable: Code possesses the characteristic understandability to the extent that its purpose is clear to the inspector.	10
		To be flexible and modifiable: Code possesses the characteristic modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined.	10
		A set of attributes that relate to the effort needed to make specified modifications.	10 , 60
		The ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment.	10, 81
		Average effort to locate , fix a software failure	98
		This quality factor addresses the concern that programs be easy to fix, once a failure is identified.	115
		"Ease of effort for locating and fixing a software failure within a specified time period"	115
		The ease with which a software product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment.	114
		Is defined as the effort to perform maintenance tasks, the impact domain of the maintenance actions, and the error rate caused by those actions.	113
		The effort required to locate and fix an error in the operational software, program, it environment	105
		Is concerned with how easy the software is to repair	105

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		Extendability	54
		(1) The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. <i>See also:</i> extendability ; flexibility. (2) The ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions.	54,6
		the <i>probability</i> that a maintenance activity can be carried out within a stated time interval ranges from 0 to 1	81
		Aptitude to undergo repairs and evolution	8
		The capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.	11
		Effort required modifying, updating, evolving, or repairing a program during its operation.	110
		The level of maintainability of the system should be specified in terms of the ability for maintenance.	12
		An attribute that relates to the amount of effort needed to make changes in the software	19
		The effort required to replace a software component with a corrected version, to upgrade a current software component (of an operational component-based software system), and to migrate an existing software component from a current component-based software system to a new version of the system	121
		Describes the ease with which the software product can be analyzed, changed and tested. The capability to avoid unexpected effects from modifications to the software is also within the scope of this characteristic. All types of modifications, i.e. corrections, improvements and adaptation to changes in requirements and in environment are covered by this characteristic.	123



Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		The ease with which a developer can modify a software system to change or add capabilities, improve performance or efficiency, or correct defects without adversely affecting other internal or external quality characteristics.	102
17	Maturity	Attributes of software that relate to (bear on) the frequency of failure by faults in the software  (1) The capability of an organization with respect to the effectiveness and efficiency of its processes and work practices. (2) The capability of the software product to avoid failure as a result of defects in the software.	118, 10 , 60  114 and 103
18	Operability	Attributes of software that relate to the users' effort for operation and operation control.  Those attributes of the software which determine operations and procedures concerned with the operation of the software  The capability of the software product to enable the user to operate and control it.  The characteristics of the software which determine operations and procedures concerned with operations of the software and which provide useful inputs and outputs which can be assimilated  Easy and efficient to apply functionality  The degree to which the operation of the software matches the purpose, environment, and physiological characteristics of users; this includes ergonomic factors such as color, shape, sound, font size, etc.  Ability of the software to be easily operated by a given user in a given environment  The ease of operation and control by users	10 , 60  98  114, 103  105  29  19  49  118
19	Performance	Imposes conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage  This quality factor addresses the concern of how well a program attribute or function is implemented with respect to some standard. Often, this is related to the utilization of resources The effectiveness with which resources of the host system are utilized toward meeting the objective of the software system	10  115

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>The degree to which a system or component accomplishes its designated functions within given constraints regarding processing time and throughput rate. such as speed, accuracy, or memory usage.</p> <p>Performance quality factors characterize how well the software functions</p> <p>Performance as a software quality attribute refers to the timeliness aspects of how software systems behave</p> <p>“Performance refers to responsiveness: either the time required to respond to specific events or the number of events processed in a given interval of time”</p> <p>Performance is that attribute of a computer system that characterizes the timeliness of the service delivered by the system.</p> <p>Responsiveness of the system—either the time required to respond to specific events or the number of events processed in a given interval of time</p> <p>Primary operating characteristics</p> <p>Speed or throughput: minimizing the time, or perceived time, between a system’s input events and output events - optimizing or maximizing the amount of useful work done in a given period of time. Note that software can be very fast, but still be a memory or CPU hog (see efficiency)</p>	<p>114, 8, 54</p> <p>105</p> <p>8</p> <p>8</p> <p>8</p> <p>8, 7</p> <p>116</p> <p>102</p>
20	Portability	<p>The ability of a component to be transferred from one environment to another</p> <p>The effort required to transfer a program from one environment to another</p> <p>Can I still use it if I change my environment?</p> <p>The code can be operated easily and well on other environments</p> <p>A set of attributes that bear on the ability of software to be transferred from one environment to another</p>	<p>2</p> <p>67,10,122, 87</p> <p>67,10 ,14</p> <p>67,14</p> <p>67,10, 60</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		The ease with which a system or component can be transferred from one hardware or software environment to another.	10, 54, 114,60
		Effort to convert the software for use in another operating environment (hardware configuration, software system environment).	98
		This quality factor addresses the concern that programs be changed easily to operate on a different set of equipment "How quickly and cheaply the software system can be converted to perform the same functions using different equipment"	115
		Portability is concerned with how easy it is to transport the system	105
		Effort required to transfer a program from one hardware configuration and/or software system environment to another	70
		The capability of software to be transferred from one environment to	11,60
		The ability for the product to be used on different machines or operating systems	12
		The extent to which a software component can be ported to a possibly wide variety of operational environments, including operating systems and hardware, and the amount of effort required for porting	121
		Is a measure of the effort that is needed to move software to another computing platform	123
		The degree to which a system, or a system's components, can be used in an operating environment different from that for which it was originally designed or developed without adversely affecting other quality characteristics. There are two types of portability - run time and compile time	102
		This characteristic refers to how well the software can adopt to changes in its environment or with its requirements. The sub characteristics of this characteristic include adaptability. Object oriented design and implementation practices can contribute to the extent to which this characteristic is present in a given system	49
		the effort required to transport the software for use in other environment	105

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
21	Recoverability	<p>Attributes ( capability) of software that relate to the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.</p> <p>Ability to bring back a failed system to full operation, including data and network connections</p> <p>Capability and effort needed to reestablish level of performance and recover affected data after possible failure</p>	<p>10,114, 60 103</p> <p>49</p> <p>118</p>
22	Reliability	<p>This characteristics express the ability of the component to maintain a specified level of performance, when used under specified conditions</p> <p>The systems ability not to fail</p> <p>The code performs its intended functions satisfactorily</p> <p>A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time</p> <p>The longevity of product performance</p> <p>Probability that the software will perform its logical operations in the specified environment without failure</p> <p>This quality factor addresses the concern that programs continue to perform properly over time.</p> <p>The probability that a software system will operate without failure for at least a given period of time when used under stated conditions</p> <p>The ability of the software product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations.</p> <p>The extent to which the software performs its intended function without failures for a given time period.</p> <p>Reliability is concerned with what confidence can be placed in the software</p>	<p>2</p> <p>67,10,122, 87</p> <p>67, 14</p> <p>10, 67, 60</p> <p>118</p> <p>98</p> <p>115</p> <p>54</p> <p>114</p> <p>105</p> <p>105</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		Extent to which a program can be expected to perform its intended function with required precision	70
		Continuity of service	8
		A measure of the ability of a system to keep operating over time	8
		A measure of the rate of failure in the system that renders the system unusable. A measure of the ability of a system to keep operating over time	8
		The capability of the software to maintain its level of performance when used under specified conditions	11
		Ability of a program to achieve precisely its intended mission	110
		The ability of a software application or component to perform its required functions under design-compliant conditions for a specified period of time	41
		The extent to which a component can be expected to fulfill its functions for a stated period of time under stated conditions	121
		Is defined as the ability of software to maintain a specified level of performance within the specified usage conditions	123
		A system's ability to perform its required functions under stated conditions whenever required. Also: having a long mean time between failures	102
		The probability that software will not cause the failure of a system for a specified time under specified conditions	103
		The capability of the system to maintain its service provision under defined conditions for defined periods of time.	49
		The extent to which a program can be expected to perform its intended function with required precision	110
		The probability that the program performs successfully in compliance with its specification for a given time period	110
		The probability that there are no failures in the time interval 0-t	110
		the ability of a system or a component to perform its required functions under stated conditions for a specified period of time	110

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
23	Replaceability	Attributes of software that relate to the opportunity and effort of using it in the place of specified other software in the environment of that software	10 , 60
		The capability of the software product to be used in place of another specified software product for the same purpose in the same environment.	10 , 114,60 103
		Characterizes the <i>plug and play</i> aspect of software components, that is how easy is it to exchange a given software component within a specified environment.	49
		The opportunity and effort of using it in the place of other software in a particular environment	118
24	Robustness	The degree to which a component or system can function correctly in the presence of invalid inputs or stressful environmental conditions.	114 , 54, 103
		Ability of a program to react appropriately to abnormal conditions	110
		Marginal cost of surviving unforeseen changes	116
25	Safety	Means simply put that the system does not ever perform anything "bad",	10
		The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use.	114
		Non-occurrence of catastrophic consequences on the environment	8
		The absence of catastrophic consequences on the environment	8
		As freedom from accidents and loss.	8
		Property of a computer system such that reliance can justifiably be placed in the absence of accidents.	8
		A measure of the absence of unsafe software conditions. The absence of catastrophic consequences to the environment	8
Freedom from physical danger	116		

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
26	Scalability	The capability of the software product to be upgraded to accommodate increased loads.	114
		The ability of a system to continue to meet its response time or throughput objectives as the demand for the software functions increases	75
		The degree to which a software system's capacity, efficiency, or performance is not limited by its design, implementation, the hardware platform on which it runs, other software systems with which it interoperates or communicates.	102
27	Security	Attributes of software that relate to its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.	10,60
		A general definition of security is provided in Appendix F of the National Research Council's report, "Computers at Risk":	8
		1. Freedom from danger; safety.	.
		2. Protection of system data against disclosure, modification, or destruction. Protection of computer systems themselves. Safeguards can be both technical and administrative.	.
		3. The property that a particular security policy is enforced, with some degree of assurance.	.
		4. Often used in a restricted sense to signify confidentiality, particularly in the case of multilevel security.	8
		Freedom from risk or doubt	116
		Secure systems are those that can be trusted to keep secrets and safeguard privacy.	6
		The degree to which the software can detect and prevent information leak, information loss, illegal use, and system resource destruction	19
		The extent to which access to a software component, a component-based software using the software component or the companion data by unauthorized persons can be controlled	47
Integrity	121,102		
The degree to which a system prevents unauthorized or improper access or modification to its code and data or other system resources and/or the degree to which it ensures that data or object state is maintained in a coherent and correct manner. The idea of integrity includes restricting unauthorized user access as well as ensuring that data is accessed properly by its intended users and other software.	102		

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<b>The capability of the software product to protect information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them.</b>	<b>103</b>
		<b>subcharacteristic relates to unauthorized access to the software functions.</b>	<b>49</b>
28	<b>Stability</b>	<b>Attributes of software that relate to the risk of unexpected effect of modifications</b>	<b>10, 60</b>
		<b>The capability of the software product to avoid unexpected effects from modifications of the software.</b>	<b>10, 60</b>
		<b>Predictability</b>	<b>116</b>
		<b>Characterizes the sensitivity to change of a given system that is the negative impact that may be caused by system changes</b>	<b>49</b>
		<b>The risk of unexpected effect of modifications</b>	<b>118</b>
29	<b>Suitability</b>	<b>Attribute of software that relates to the presence and appropriateness of a set of functions for specified tasks.</b>	<b>10 , 60</b>
		<b>The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives.</b>	<b>114</b>
		<b>This is the essential Functionality characteristic and refers to the appropriateness (to specification) of the functions of the software</b>	<b>49</b>
		<b>The presence and appropriateness of a set of functions for specified tasks</b>	<b>118</b>
30	<b>Testability</b>	<b>the ease of testing the program, to ensure that it is error-free and meets its specification</b>	<b>67,10, 87 122</b>
		<b>The code eases setting up verification criteria and supports evaluation of its performance.</b>	<b>67, 14</b>
		<b>Attributes of software that relate to the effort needed for validating the modified software.</b>	<b>10, 60</b>
		<b>The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.</b>	<b>10,40</b>
		<b>Addresses the concern that programs be easy to test "A software product possesses the characteristic Testability to the extent that it facilitates the establishment of acceptance criteria and supports evaluation of its performance."</b>	<b>115</b>



Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>The capability of the software product to enable modified software to be tested</p> <p>(1) The degree to which a requirement is stated in terms that permit establishment of test criteria and performance of tests to determine whether those criteria have been met.</p> <p>Effort required to test a program to ensure that it performs its intended function</p> <p>Effort required to test a program</p> <p>As an indication of the degree of testing effort required</p> <p>This attribute is related to the effort needed to test a software system to ensure it performs its intended functions A quantification of testability could be the number of test cases required to adequately test a system, or the cyclomatic complexity of an individual module.</p> <p>Which refers to the effort required to ensure that it performs its intended function and performance, and, for software components, includes the verification of interface, assembly, porting, and certification requirements in the scope</p> <p>The degree to which someone can unit-test, system test and functionally test a software system. This idea also extends to the ease with which a test plan can be developed from the projects requirements</p> <p>The capability of the software product to enable modified software to be validated.</p> <p>Characterizes the effort needed to verify</p>	<p>114</p> <p>40</p> <p>70</p> <p>110</p> <p>19</p> <p>19</p> <p>121</p> <p>102</p> <p>103</p> <p>49</p>
31	Traceability	<p>Those attributes of the software which provide a thread of origin from the implementation to the requirements with respect to the specific development envelope and operational environment.</p> <p>The ability to identify related items in documentation and software, such as requirements with associated tests.</p> <p>The characteristics of the software that provide a thread from the requirements to the implementation with respect to the specific development and operational environment</p>	<p>98</p> <p>114</p> <p>105</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>(1) The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match. <i>See also</i>: consistency.</p> <p>(2) The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement that it satisfies.</p>	54
		Traceability would make it possible to know the relationships of a particular entity to other entities,	10
		Allows a modification of one system artefact to be traced to other system artefacts that also will be affected.	10
32	Understandability	<p>The code is easy to read in the sense, that inspectors can rapidly recognize its purpose.</p> <p>The properties of the design that enable it to be easily learned and comprehended. This directly relates to the complexity of the design structure</p> <p>Attributes of software that relate to the users' effort for recognizing the logical concept and its applicability</p> <p>The degree to which the purpose of the system or component is clear to the evaluator.</p> <p>“The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.”</p> <p>This quality factor addresses the concern that programs be easy to understand</p> <p>Ease with which the implementation can be understood</p> <p>The amount of effort required to understand the software</p> <p>The ease with which someone can comprehend a software system at both the system-organizational and detailed-statement levels. Understandability has to do with the coherence and cohesiveness of the system at a more general level than readability. Understanding includes not only understanding what the system does, but <u>why</u> it does it. Good detailed design documents can greatly enhance a systems</p>	<p>67, 14</p> <p>67</p> <p>10, 60</p> <p>10, 14</p> <p>10,114</p> <p>115</p> <p>115</p> <p>19</p> <p>102</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<b>Determines the ease of which the systems functions can be understood, relates to user mental models in Human Computer Interaction methods.</b>	<b>49</b>
		<b>The effort for a user to learn its application , operation, input and output</b>	<b>118</b>
33	Usability	<b>This characteristic express the ability of a component to be understood, learned, used, configured, and executed, when used under specified conditions;</b>	<b>2</b>
		<b>Its ability to be used by the application developer when constructing a software product or a system with it.</b>	<b>2</b>
		<b>The ease of the software</b>	<b>67,87</b>
		<b>The code is reliable, efficient and human-friendly-engineered</b>	<b>67, 14</b>
		<b>A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.</b>	<b>67, 60,33</b>
		<b>Effort to convert a software component for use In another application.</b>	<b>98</b>
		<b>Effort for training and software operation -familiarization, input preparation, execution, output interpretation</b>	<b>98</b>
		<b>A software product possesses the characteristic Usability to the extent that it is convenient and practicable to use."</b>	<b>115</b>
		<b>The capability of the software to be understood, learned, used and attractive to the user when used under specified conditions.</b>	<b>114,11</b>
		<b>The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.</b>	<b>40,54</b>
		<b>The effort required to learn, operate, prepare input, and interpret output of the software (program)</b>	<b>105,33</b>
		<b>The extent to which an end-user is able to carry out required tasks successfully, and without difficulty using the computer application system.</b>	<b>33</b>
		<b>The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use</b>	<b>33</b>
		<b>The ease with which a user can learn to operate a software application</b>	<b>41</b>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p><b>Usability: The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.</b></p>	6
		<p><b>Usability is a measure of how well users can take advantage of some system functionality. Usability is different from utility, a measure of whether that functionality does what is needed</b></p>	6
		<p><b>The extent of ease to which a software component can be unpacked by possibly a variety of users, configured by these users for selecting the particular configurations that best satisfy the needs of these users (if such configurability is provided), and assembled by these users into the application environments of their component-based application software systems (this also includes understandability and ease of learning)</b></p>	121
		<p><b>The ease with which users can learn about and effectively use a system. The quality of end user documentation and technical support can radically effect this characteristic. This includes traditional documentation, on-line help and web based information.</b></p>	102
		<p><b>Characteristics relating to the effort needed for use , and on the individual assessment of such use, by a stated or implied set of users</b></p>	118
34	Utility	<p><b>To be portable: Code possesses the characteristic portability to the extent that it can be operated easily and well on computer configurations other than its current one.</b></p>	10, 14
		<p><b>To be reliable : Code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily</b></p>	10, 14
		<p><b>To efficiency : Code possesses the characteristic efficiency to the extent that it fulfills its purpose without waste of resources</b></p>	10, 14
		<p><b>To be usable: Code possesses the characteristic usability to the extent that it is reliable, efficient and human-engineered.</b></p>	10, 14
		<p><b>How well (easily, reliably, efficiently) can I use it as-is?</b></p>	67, 14
35	fault tolerance	<p><b>Attributes of software that relate to its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.</b></p>	10 , 60
		<p><b>That is the ability of a system to withstand component failure</b></p>	49
		<p><b>The ability of software to withstand (and recover) from component, or environmental, failure.</b></p>	49

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		Ability to maintain a specified level of performance in cases of software faults or un expected inputs	118
36	Reusability	<p>The ease of reusing software in a different context</p> <p>“The degree to which a software module or other work product can be used in more than one computing program or software system.”</p> <p>Addresses the concern that programs be easy to reuse in a different application. "Relative effort to convert a software component for use in a different application"</p> <p>is concerned with how easy it is to convert the software for use in another application,</p> <p>Extent to which a program can be used in other application—related to the packaging and scope of the functions that programs perform</p> <p>The extent to which a software component can be reused in developing component-based software systems, other software components or other software products in general</p>	<p>10,87</p> <p>10, 87</p> <p>115</p> <p>105</p> <p>70, 110</p> <p>121</p>
37	Correctness	<p>This attribute evaluates the percentage of the results obtained with precision, specified by the user requirements</p> <p>The extent to which a program conforms to its specification</p> <p>The extent to which a program fulfils its specification</p> <p>“The degree to which a system or component is free from faults in its specification, design, and implementation”].</p> <p>-Extent to which the software satisfies its specifications and fulfills the user's mission objectives.</p> <p>The concern that software design and documentation formats conform to the specifications and standards set for them. It is not concerned with any content affecting software operation or performance. "Extent to which the software conforms to its specifications and standards"</p> <p>Is concerned with how well the software conforms to the requirements</p> <p>(1) The degree to which software, documentation, or other items meet specified requirements. (2) The degree to which software, documentation, or other items meet user needs and expectations, whether specified or not.</p>	<p>2</p> <p>67 , 87</p> <p>10 , 87</p> <p>10,54</p> <p>98,70</p> <p>115</p> <p>115</p> <p>105</p> <p>54</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>Ability of software products to perform their exact tasks, as defined by their specification</p> <p>The degree to which the software performs its required functions.</p> <p>The degree to which a system is free from faults in its requirements, scope, specification, architecture, design, implementation and deployment</p>	<p>110</p> <p>19</p> <p>102</p>
38	Modifiability	<p>This attribute indicates the component behavior when accomplished some modification on it;</p> <p>The degree to which a system or component facilitates the incorporation of changes, once the nature of the desired change has been determined.</p> <p>Addresses the concern that programs be easy to change, regardless of the reason for the change. "A software product possesses modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined."</p> <p>Considers how the system can accommodate anticipated and unanticipated changes and is largely a measure of how changes can be made locally, with little ripple effect on the system at large.</p> <p>Modifiability encompasses two aspects: "Maintainability. (1) The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. (2) The ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions."</p>	<p>2</p> <p>10,14</p> <p>115</p> <p>7</p> <p>6</p>
39	Completeness	<p>It is possible that some implementations do not completely cover the services specified. This attribute measure the number of implemented operations compared to the total number of specified operations;</p> <p>Those attributes (characteristics) of the software which provide full implementation of the functions required.</p> <p>Quality factor addresses the concern that program functions be implemented completely</p> <p>Each part full developed</p> <p>The degree to which the software possesses the necessary and sufficient functions to satisfy the users needs.</p>	<p>2</p> <p>98, 115</p> <p>115</p> <p>116</p> <p>19</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		The degree to which a system implements its planned scope with a particular focus on meeting requirements and delivering features	102
40	Dependability	<p>This attribute indicates if the component is not self-contained, i.e. if the component depend of other component to provide its specified services</p> <p>Is that property of a computer system such that reliance can justifiably be placed on the service it delivers</p> <p>That property of a system such that reliance can justifiably be placed in the service it provides</p> <p>Availability. The degree to which a system or component is operational and accessible when required for use. Dependability is that property of a computer system such that reliance can justifiably be placed on the service it delivers</p>	2 8 7 6
41	Extensibility	This attribute indicates the capacity to extend a certain component functionality;	2
42	Customizability	This attribute measures the number of customizable parameters that the component offers	2
43	Modularity	<p>This attribute indicates the modularity level of the component, if it has modules, packages or all the source files are only grouped.</p> <p>Those attributes of the software which provide a structure of highly cohesive modules with optimum coupling</p> <p>Quality factor addresses the concern that programs be composed of many small, simple, independent steps that are clearly delineated by the code. "Formal way of dividing a program into a number of sub-units each having a well defined function and relationship to the rest of the program"</p> <p>The characteristics of the software which provide a structure of highly independent modules</p> <p>The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.</p>	2 98 115 105 54
44	Flexibility	The ease of making changes required by changes in the operating environment	10 , 67 ,122

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<b>The code is easy to change, when a desired change has been determined</b>	<b>67</b>
		<b>Characteristics that allow the incorporation of changes in a design. The ability of a design to be adapted to provide functional related capabilities</b>	<b>67</b>
		<b>The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.</b>	<b>10</b>
		<b>Effort to extend the software missions, functions, or data to satisfy other requirements.</b>	<b>98</b>
		<b>This quality factor addresses the concern that programs be easy to change to meet different requirements, with no change in the context. Ease of effort for changing the software missions, functions, or data to satisfy other requirements</b>	<b>115</b>
		<b>The effort required to modify operational software</b>	<b>105, 70</b>
		<b>Marginal cost to extend Features</b>	<b>116</b>
		<b>The extent to which a developer can modify a software system for uses or environments other than those for which it was specifically designed without adversely affecting other internal or external quality characteristics</b>	<b>102</b>



Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
45	Integrity	The protection of the program from unauthorized access	67,10
		Extent to which unauthorized access to the software or data can be controlled	98
		Quality factor addresses the concern that programs must continue to perform their function even under adverse conditions: inputs that are unexpected, improper, or harmful Ability of software to prevent purposeful or accidental damage to the data or software	115
		The extent to which access to software or data by unauthorized persons should be controlled	105,70,110
		The degree to which a system or component or application prevents unauthorized access to, or modification of, computer programs or data.	54
		Non-occurrence of improper alterations of information	8
		Is the requirement that data and process be protected from unauthorized modification	8
		Protection of the program from unauthorized access.	122
46	Accessibility	Means that the system allows usage of its parts in a selective manner, which helps testing as test cases can be constructed with higher flexibility	10
		System accessibility : Those attributes of the software which provide for control and audit of access of software and data	98, 105
47	Communicat_iveness	Means that it is possible to easily specify and understand inputs to and outputs from the system , which again facilitates the construction of test cases	10
		Those attributes of the software which provide useful inputs and outputs which can be assimilated	98

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		The degree to which the software is designed in accordance with the psychological characteristics of the users	19
48	Self Descriptiveness	Those attributes of the software which provide explanation of the implementation of a function.  The degree to which a system or component contains enough information to explain its objectives and properties.	98  54
49	Conciseness	Those attributes of the software which provide for implementation of a function with a minimum amount of code.  This quality factor addresses the concern that programs not contain any extraneous information. The ability to satisfy functional requirements with minimum amount of software  No excess information is present	98  115  116
50	Extendability	Refers to the presence and usage of properties in an exiting design that allow for the incorporation of new requirements in the design  The ease with which a system or component can be modified to increase its storage or functional capacity	67  54
51	Effectiveness	This refers to a design's ability to achieve the desired functionality and behavior using object-oriented design concepts an techniques  Those attributes of the software which provide for minimum utilization of resources (processing time, storage, operator time) in performing functions.  The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.	67  98  103
52	Resource Utilization	The amount of resources used and the duration of such use in performing its function  The capability of the software product to use appropriate amounts and types of resources, for example the amounts of main and secondary memory used by the program and the sizes of required temporary or overflow files, when the software performs its function under stated conditions.	118  114, 103
53	Compatibility	A measure (characteristics) of the hardware, software and communication compatibility of two systems	98,105

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>(1) The ability of two or more systems or components to perform their required functions while sharing the same hardware or software environment.</p> <p>(2) The ability of two or more systems or components to exchange information.</p>	54
		The degree to which new software can be installed without changing environments and conditions that were prepared for the replaced software.	19
		The extent to which a software system will function or communicate correctly, reliably and robustly with other similar systems that share the same data types, file formats, or user interfaces. Backward compatibility specifically applies to a software systems' ability to work with previously versions from which it was derived or with versions ported to other systems	102
54	Independence	<p><b>APPLICATION INDEPENDENCE</b> Attributes of the software which determine its dependency on the software application (database system, data structure, system libraries routines, microcode, computer architecture and algorithms)</p>	98
		<p><b>INDEPENDENCE</b> Those attributes of the software which determine its non-dependency on the software environment (computing system, operating system, utilities ,input/output routines, libraries</p>	98
		<p><b>Executable in hardware environment other than current one</b></p>	116
55	Simplicity	<p>Those attributes of the software which provide for the definition and implementation of functions in the most non-complex and understandable manner.</p>	98
		<p>Quality factor addresses the concern that, as much as possible, programs be implemented in strictly sequential steps that depend only on the step before it</p>	115
		<p>Those characteristics of software which provide for definition and implementation of functions in the most noncomplex and understandable manner</p>	
		<p>The degree to which a system or component has a design and implementation that is straightforward and easy to understand</p>	54
		<p>How complicated</p>	116
56	Expandability	<p>How easy to add new functionality to it</p>	10
		<p>Quality factor addresses the concern that program limitations be easy to extend</p> <p>The "Relative effort [required] to increase the software capability or performance by enhancing current functions or by adding new functions or data"</p>	115,105

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>Concerned with how easy it is to expand or upgrade the software capability or performance.</p> <p>The degree of effort required to improve or modify the efficiency or functions of the software.</p> <p>The effort required to increase the capability of a software component</p>	<p>105</p> <p>19</p> <p>121</p>
57	Generality	<p>Means general solutions that by nature are prepared for being utilized in other contexts than the ones for which they were constructed</p> <p>Those attributes of the software which provide breadth to the functions performed with respect to the application</p> <p>The degree to which a system or component performs a broad range of functions</p>	<p>10</p> <p>98,105</p> <p>54</p>
58	System Clarity	<p>Those attributes (characteristics) of the software which provide clear description of program structure in the most non-complex, easily understandable and modifiable manner.</p> <p>This quality factor addresses the concern that programs be easily understood by people Measure of how clear a program is, i.e., how easy it is to read, understand, and use</p> <p>Then clarity only addresses the ease of reading and understanding the program.</p>	<p>98,105</p> <p>115</p> <p>115</p>
59	Survivability	<p>The extent (Probability that) to which the software will continue to perform or support critical functions when a portion of the system is inoperable</p> <p>Is concerned with how well the software will perform under adverse conditions. The attributes which support survivability</p>	<p>98,105</p> <p>105</p>
60	Verifiability	<p>Effort to verify the specified software operation, and performance</p> <p>The effort required to test and verify (ensure) that the software performs its intended designed function</p> <p>Is concerned with how easy it is to verify the software performance. The attributes which support verifiability are these'</p>	<p>98</p> <p>105, 110</p> <p>105</p>

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
		<p>The effort required to verify, with or without access to the source code, architecture, design and the developers, that the software design and implementation satisfies the specifications of the software component (This goes beyond the testability, which refers to the effort required to ensure that it performs its intended function and performance, and, for software components, includes the verification of interface, assembly, porting, and certification requirements in the Scope.</p> <p>(The extent to which a software component can be certified), the extent to which certification implies the quality of the software component, and the effort required for such extents of certification</p>	<p>121</p> <p>121</p>
61	Repeatability = Reproducibility	The degree to which a system will repeatedly produce the same results given a consistent set of inputs and a consistent operating environment. Sometimes called Reproducibility	102
62	Conformance	<p>Attributes of software that make the software adhere to standards or conventions relating to portability</p> <p>Degree to which a products design and operating characteristics meet the stated requirements ; "all parts present" portion of "Completeness" Characteristic</p> <p>Similar to compliance for functionality, but this characteristic relates to portability. One example would be Open SQL conformance which relates to portability of database used.</p>	<p>10</p> <p>116</p> <p>49</p>
63	Capacity	<p>Ability to produce at least at the rate of demand</p> <p>How much demand can be placed on the system while continuing to meet latency and throughput requirements?</p> <p>Is a measure of the amount of work a system can perform</p> <p>The ability or suitability for holding, storing, or accommodating data or information. The maximum amount or number of something that can be contained or accommodated. Capacity may be dictated by design, hard coded limits or requirements. It may also be dictated by the operating environment .</p>	<p>116</p> <p>8</p> <p>8</p> <p>102</p>
64	Buildability	The ease with which a software product can be reliably built from its individual components. Typically this focuses on people other than the original developer. However, it also applies to the scenario where the original developer has not built the system for a extended period of time. The use of SCM tools typically focuses on this characteristic. The term reliably is important: it implies that the build system is repeatable, dependable, timely and that when given the same inputs it will always build the same thing.	102

Att ID	Quality Attribute	Definition(s)	Source(s) reference(s)
65	Readability	The ease with which a developer can read and understand the source code and technical documentation of a system, especially at the detailed source code statement level	102
66	Productivity	The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.	102

## APPENDIX B

The complete results from the ontology evaluation step; each SWPQA extracted concepts and the covered concepts from our ontology in addition to the coverage percentage.

Table B.1: The complete results from the ontology evaluation step.

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
1	Accuracy	assessment	accuracy	17 from 24
		computer	capability	0.708333333
		concern	computer	
		determination	concern	
		extent	degree	
		factor	error	
		freedom	extent	
		job	factor	
		magnitude	freedom	
		measure	measure	
		output	output	
		quality	precision	
		respect	product	
		capability	quality	
		provision	respect	
		right	software	
		system	system	
2	Adaptability	accuracy		
		correctness		
		degree		
		error		
		product		
		precision		
		software		
2	Adaptability	product	ability	13 from 15
		environment	adaptation	0.866666667
		ease	change	
		operating	component	
		component	degree	
		degree	ease	
		modification	environment	
ability	modification			

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		change	operating	
		purpose	product	
		opportunity	purpose	
		means	software	
		adaptation	system	
		system		
		software		
3	Analyzability	diagnosis	ability	6 from 10
		identification	capability	0.6
		cause	effort	
		failure	failure	
		ability	product	
		root	software	
		effort		
		product		
		capability		
		software		
4	Attractiveness	product	capability	4 from 4
		user	product	1
		software	software	
		capability	user	
5	Availability	availability	component	12 from 26
		checkpoint	data	0.461538462
		component	degree	
		continuity	level	
		data	probability	
		degree	program	
		delivery	service	
		demand	software	
		denial	specification	
		level	system	
		meeting	time	
		percentage	usage	
		point		
		probability		
		program		
		readiness		
		recovery		
		reliability		
		requirement		
		restart		
		service		
		software		
		specification		
		system		
		time		
		usage		
6	Changeability	amount	amount	9 from 12
		capability	capability	0.75
		change	change	
		effort	effort	
		fault	modification	
		make	product	
		modification	set	
		product	software	
		removal	system	
		set		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		software		
		system		
7	Complexity	attribute	attribute	16 from 19
		code	code	0.842105263
		combination	component	
		component	concern	
		concern	control	
		control	data	
		data	degree	
		degree	design	
		design	extent	
		extent	factor	
		factor	implementation	
		flow	measure	
		implementation	quality	
		measure	set	
		metrics	structure	
		quality	system	
		set		
		structure		
		system		
8	Compliance	adherence	application	4 from 7
		application	capability	0.571428571
		capability	characteristic	
		characteristic	software	
		government		
		industry		
		software		
9	Consistency	code	code	17 from 22
		component	component	0.772727273
		concern	concern	
		contradiction	definition	
		definition	degree	
		degree	design	
		design	factor	
		factor	freedom	
		freedom	implementation	
		implementation	level	
		level	notation	
		notation	quality	
		quality	software	
		software	source	
		source	system	
		standardization	uniform	
		symbology	uniform	
		syntax	uniformity	
		system		
		terminology		
		uniform		
		uniformity		
10	Co-existence	capability	capability	4 from 5
		environment	environment	0.8
		independent	product	
		product	software	
		software		
11	Efficiency	ability	ability	38 from 65
		amount	amount	0.575757576



Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		attribute	attribute	
		bandwidth	characteristic	
		characteristic	code	
		code	component	
		communication	computer	
		component	computing	
		computer	concern	
		computing	degree	
		concern	efficiency	
		consumption	extent	
		cpu	factor	
		degree	function	
		disk	function	
		efficiency	function	
		estate	functionality	
		execution	hardware	
		express	level	
		extent	meaning	
		factor	measure	
		function	memory	
		functionality	number	
		hardware	performance	
		indication	product	
		intention	program	
		interaction	purpose	
		issue	quality	
		key	rate	
		level	response	
		meaning	set	
		measure	software	
		memory	storage	
		minimum	system	
		network	time	
		number	usability	
		performance	usage	
		place	user	
		processor	utility	
		product	utilization	
		program		
		purpose		
		quality		
		rate		
		relation		
		relationship		
		relative		
		request		
		resource		
		response		
		screen		
		set		
		software		
		space		
		storage		
		system		
		task		
		time		
		usability		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		usage		
		user		
		utility		
		utilization		
		value		
		waste		
		works		
12	Functionality	ability	ability	12 from 16
		achievement	capability	0.75
		capability	characteristic	
		characteristic	component	
		component	design	
		design	extent	
		existence	product	
		express	purpose	
		extent	service	
		product	set	
		purpose	software	
		service	user	
		set		
		software		
		totality		
		user		
13	Installability	capability	capability	5 from 5
		effort	effort	1
		environment	environment	
		product	product	
		software	software	
14	Interoperability	ability	ability	13 from 17
		capability	capability	0.764705882
		component	component	
		computing	computing	
		couple	degree	
		degree	effort	
		effort	environment	
		environment	extent	
		exchange	hardware	
		extent	information	
		hardware	product	
		information	software	
		interface	system	
		product		
		software		
		system		
		variety		
15	Learnability	application	application	10 from 14
		capability	capability	0.714285714
		control	control	
		effort	effort	
		expert	operation	
		input	output	
		learning	product	
		novice	software	
		operation	user	
		output	work	
		product		
		software		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		user		
		work		
16	Maintainability	ability	ability	39 from 61
		activity	adaptation	0.639344262
		adaptation	amount	
		amount	attribute	
		aptitude	capability	
		attribute	change	
		average	characteristic	
		capability	code	
		change	component	
		characteristic	concern	
		code	developer	
		component	ease	
		concern	efficiency	
		developer	effort	
		domain	environment	
		ease	error	
		efficiency	extent	
		effort	factor	
		environment	failure	
		error	hardware	
		establishment	incorporation	
		evaluation	level	
		evolution	maintenance	
		extendability	operating	
		extent	operation	
		factor	performance	
		failure	period	
		fault	probability	
		fix	product	
		flexibility	program	
		hardware	purpose	
		impact	quality	
		incorporation	rate	
		inspector	scope	
		interval	set	
		level	software	
		maintainability	system	
		maintenance	time	
		modifiability	understandability	
		nature		
		operating		
		operation		
		performance		
		period		
		probability		
		product		
		program		
		purpose		
		quality		
		rate		
		repair		
		scope		
		set		
		software		
		state		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
17		system		
		testability		
		time		
		understandability		
		verification		
		version		
	Maturity	bear	capability	8 from 12
		capability	effectiveness	0.666666667
		effectiveness	efficiency	
		efficiency	failure	
		failure	product	
		frequency	respect	
		organization	software	
		product	work	
		respect		
		result		
		software		
work				
18	Operability	ability	ability	13 from 17
		capability	capability	0.764705882
		color	control	
		control	degree	
		degree	ease	
		ease	effort	
		effort	environment	
		environment	functionality	
		functionality	operation	
		operation	product	
		product	purpose	
		purpose	software	
		shape	user	
		size		
		software		
sound				
user				
19	Performance	accuracy	accuracy	32 from 45
		amount	amount	0.711111111
		attribute	attribute	
		availability	component	
		component	computer	
		computer	concern	
		concern	degree	
		cpu	effectiveness	
		degree	efficiency	
		effectiveness	factor	
		efficiency	function	
		factor	memory	
		function	number	
		host	objective	
		input	operating	
		interval	output	
		meeting	performance	
		memory	period	
		note	program	
number	quality			
objective	rate			
operating	respect			

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		output	response	
		performance	responsiveness	
		period	service	
		processing	software	
		program	system	
		quality	throughput	
		rate	time	
		recovery	usage	
		resource	utilization	
		respect	work	
		response		
		responsiveness		
		service		
		software		
		speed		
		standard		
		system		
		throughput		
		time		
		timeliness		
usage				
utilization				
work				
20	Portability	ability	ability	29 from 36
		adaptability	adaptability	0.805555556
		amount	amount	
		capability	capability	
		characteristic	characteristic	
		code	code	
		component	component	
		computing	computing	
		concern	concern	
		configuration	degree	
		degree	design	
		design	ease	
		ease	effort	
		effort	environment	
		environment	extent	
		equipment	factor	
		extent	hardware	
		factor	implementation	
		hardware	measure	
		implementation	object	
		measure	operating	
		move	portability	
		object	product	
		operating	program	
		platform	quality	
		portability	set	
		product	software	
		program	system	
		quality	time	
		set	time	
		software		
		system		
		time		
		transfer		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
21	Recoverability	transport		
		variety		
		ability	ability	11 from 12
		capability	capability	0.916666667
		data	data	
		effort	effort	
		failure	failure	
		level	level	
		network	operation	
		operation	performance	
		performance	software	
		software	system	
system	time			
time				
22	Reliability	ability	ability	33 from 41
		application	application	0.804878049
		capability	capability	
		code	code	
		compliance	component	
		component	concern	
		concern	design	
		confidence	environment	
		continuity	extent	
		design	factor	
		environment	fail	
		extent	failure	
		factor	function	
		fail	level	
		failure	mean	
		function	measure	
		interval	number	
		level	operating	
		longevity	performance	
		mean	period	
		measure	precision	
		mission	probability	
		number	product	
		operating	program	
		performance	quality	
		period	rate	
		precision	service	
		probability	set	
		product	software	
		program	specification	
		provision	system	
		quality	time	
		rate	usage	
reliability				
service				
set				
software				
specification				
system				
time				
usage				
23	Replaceability	aspect	capability	7 from 13
		capability	component	0.538461538

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		component	effort	
		effort	environment	
		environment	product	
		exchange	purpose	
		opportunity	software	
		place		
		play		
		plug		
		product		
		purpose		
		software		
24	Robustness	ability	ability	5 from 8
		component	component	0.625
		cost	degree	
		degree	program	
		invalid	system	
		presence		
		program		
		system		
25	Safety	absence	capability	10 from 19
		business	computer	0.526315789
		capability	context	
		computer	environment	
		context	freedom	
		danger	measure	
		environment	product	
		freedom	property	
		loss	software	
		means	system	
		measure		
		occurrence		
		people		
		product		
		property		
		reliance		
		risk		
		software		
		system		
26	Scalability	ability	ability	14 from 17
		capability	capability	0.823529412
		capacity	degree	
		degree	design	
		demand	efficiency	
		design	hardware	
		efficiency	implementation	
		hardware	performance	
		implementation	product	
		performance	response	
		platform	software	
		product	system	
		response	throughput	
		software	time	
		system		
		throughput		
		Time		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
27	Security	ability	ability	20 from 44
		access	access	0.454545455
		assurance	capability	
		capability	code	
		case	component	
		code	computer	
		companion	data	
		component	definition	
		computer	degree	
		confidentiality	extent	
		danger	freedom	
		data	information	
		definition	manner	
		degree	modification	
		destruction	object	
		disclosure	product	
		doubt	property	
		extent	software	
		freedom	system	
		idea	user	
		information		
		integrity		
		leak		
		loss		
		manner		
		modification		
		multilevel		
		object		
		policy		
		privacy		
		product		
		property		
		protection		
report				
research				
resource				
risk				
safety				
security				
sense				
software				
state				
system				
user				
28	Stability	change	change	5 from 9
		effect	effect	0.555555556
		impact	product	
		predictability	software	
		product	system	
		risk		
		sensitivity		
		software		
29	Suitability	appropriateness	attribute	9 from 11
		attribute	capability	0.818181818
		capability	characteristic	
		characteristic	functionality	



Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		functionality	product	
		presence	set	
		product	software	
		set	specification	
		software	user	
		specification		
		user		
30	Testability	acceptance	attribute	22 from 39
		assembly	capability	0.55
		attribute	characteristic	
		capability	code	
		certification	component	
		characteristic	concern	
		code	degree	
		complexity	ease	
		component	effort	
		concern	error	
		degree	extent	
		ease	function	
		effort	number	
		error	performance	
		establishment	product	
		evaluation	program	
		extent	scope	
		function	setting	
		idea	software	
		indication	specification	
		interface	system	
		module	test	
		number		
		performance		
		plan		
		product		
		program		
		quantification		
		requirement		
		scope		
		setting		
		software		
		specification		
		system		
		test		
		testability		
		unit		
		validating		
		verification		
31	Traceability	ability	ability	12 from 30
		artefact	component	0.4
		bubble	degree	
		chart	design	
		component	development	
		consistency	documentation	
		degree	environment	
		design	modification	
		development	product	
		documentation	respect	
		element	software	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		entity	system	
		envelope		
		environment		
		master		
		match		
		modification		
		origin		
		predecessor		
		process		
		product		
		reason		
		relationship		
		requirement		
		respect		
		software		
		successor		
		system		
		thread		
		traceability		26 from 35
32	Understandability	amount	amount	0.742857143
		applicability	applicability	
		application	application	
		code	code	
		coherence	component	
		cohesiveness	computer	
		complexity	concern	
		component	degree	
		computer	design	
		concept	ease	
		concern	effort	
		degree	factor	
		design	implementation	
		ease	level	
		effort	operation	
		evaluator	output	
		factor	product	
		implementation	purpose	
		interaction	quality	
		level	software	
		operation	structure	
		output	system	
		product	understand	
		purpose	understandability	
		quality	understanding	
		readability	user	
		sense		
		software		
		statement		
		structure		
		system		
		understand		
		understandability		
		understanding		
		user		
33	Usability	ability	ability	31 from 48
		advantage	application	0.645833333
		application	capability	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		assessment	characteristic	
		capability	code	
		characteristic	component	
		code	computer	
		component	context	
		computer	developer	
		configurability	documentation	
		context	ease	
		developer	effect	
		difficulty	effectiveness	
		documentation	efficiency	
		ease	effort	
		effect	end	
		effectiveness	extent	
		efficiency	functionality	
		effort	information	
		end	measure	
		execution	operation	
		express	output	
		extent	product	
		familiarization	program	
		functionality	quality	
		help	set	
		information	software	
		interpretation	system	
		line	understandability	
		measure	usability	
		operation	user	
		output	utility	
		preparation		
		product		
		program		
		quality		
		satisfaction		
		set		
		software		
		support		
		system		
		training		
		understandability		
		usability		
		user		
		utility		
		variety		
		web		
34	Utility	characteristic	characteristic	8 from 10
		code	code	0.8
		computer	computer	
		efficiency	efficiency	
		extent	extent	
		portability	portability	
		purpose	purpose	
		reliability	usability	
		usability		
		waste		
35	fault tolerance	ability	ability	7 from 9
		component	component	0.777777778

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		failure	failure	
		infringement	level	
		interface	performance	
		level	software	
		performance	system	
		software		
		system		
36	Reusability	application	application	15 from 19
		component	component	0.789473684
		computing	computing	
		concern	concern	
		context	context	
		convert	degree	
		degree	ease	
		ease	effort	
		effort	extent	
		extent	product	
		module	program	
		packaging	scope	
		product	software	
		program	system	
		reusing	work	
		scope		
		software		
		system		
		work		
37	Correctness	ability	ability	19 from 24
		architecture	attribute	0.791666667
		attribute	component	
		component	concern	
		concern	degree	
		content	design	
		degree	documentation	
		deployment	extent	
		design	implementation	
		documentation	operation	
		extent	performance	
		implementation	precision	
		mission	program	
		operation	scope	
		percentage	set	
		performance	software	
		precision	specification	
		program	system	
		scope	user	
		set		
		software		
		specification		
		system		
		user		
38	Modifiability	attribute	attribute	17 from 22
		change	change	0.772727273
		component	component	
		concern	concern	
		degree	degree	
		ease	ease	
		effect	effect	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		environment	environment	
		extent	extent	
		hardware	hardware	
		incorporation	incorporation	
		measure	measure	
		modifiability	modification	
		modification	performance	
		nature	product	
		performance	software	
		product	system	
		reason		
		ripple		
		software		
		state		
		system		
39	Completeness	attribute	attribute	12 from 15
		concern	concern	0.8
		degree	degree	
		factor	factor	
		focus	implementation	
		implementation	measure	
		measure	number	
		meeting	program	
		number	quality	
		part	scope	
		program	software	
		quality	system	
		scope		
		software		
		system		
40	Dependability	attribute	attribute	7 from 10
		availability	component	0.7
		component	computer	
		computer	degree	
		degree	property	
		property	service	
		reliance	system	
		self		
		service		
		system		
41	Extensibility	attribute	attribute	3 from 4
		capacity	component	0.75
		component	functionality	
		functionality		
42	Customizability	attribute	attribute	3 from 3
		component	component	1
		number	number	
43	Modularity	attribute	attribute	17 from 23
		change	change	0.739130435
		code	code	
		component	component	
		computer	computer	
		concern	concern	
		coupling	degree	
		degree	factor	
		factor	function	
		function	level	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		impact	number	
		level	program	
		modularity	quality	
		number	software	
		program	source	
		quality	structure	
		relationship	system	
		rest		
		software		
		source		
		structure		
		system		
		way		
44	Flexibility	ability	ability	20 from 21
		change	change	0.952380952
		code	code	
		component	component	
		concern	concern	
		context	context	
		cost	data	
		data	design	
		design	developer	
		developer	ease	
		ease	effort	
		effort	environment	
		environment	extent	
		extent	factor	
		factor	incorporation	
		incorporation	operating	
		operating	quality	
		quality	software	
		software	system	
		system	use	
		use		
45	Integrity	ability	ability	21 from 30
		access	access	0.7
		application	application	
		code	code	
		companion	component	
		component	computer	
		computer	concern	
		concern	data	
		damage	degree	
		data	extent	
		degree	factor	
		extent	function	
		factor	information	
		function	manner	
		idea	modification	
		information	object	
		integrity	object	
		manner	program	
		modification	quality	
		object	software	
		occurrence	system	
		process	user	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		program		
		protection		
		quality		
		requirement		
		software		
		state		
		system		
		User		
46	Accessibility	access	access	9 from 13
		accessibility	accessibility	0.692307692
		audit	control	
		control	data	
		data	manner	
		flexibility	software	
		manner	system	
		means	test	
		software	usage	
		system		
		test		
		testing		
		usage		
47	Communicativeness	construction	degree	3 from 5
		degree	software	0.6
		means	system	
		software		
		system		
48	Self Descriptiveness	component	component	7 from 8
		degree	degree	0.875
		explanation	function	
		function	implementation	
		implementation	information	
		information	software	
		software	system	
		system		
49	Conciseness	ability	ability	10 from 11
		amount	amount	0.909090909
		code	code	
		concern	concern	
		factor	factor	
		function	function	
		implementation	implementation	
		information	information	
		minimum	quality	
		quality	software	
		software		
50	Extendability	capacity	component	7 from 11
		component	design	0.636363636
		design	ease	
		ease	incorporation	
		exiting	storage	
		incorporation	system	
		increase	usage	
		presence		
		storage		
		system		
		usage		

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
51	Effectiveness	ability	ability	13 from 15
		accuracy	accuracy	0.866666667
		capability	capability	
		completeness	context	
		context	design	
		design	functionality	
		functionality	object	
		minimum	operator	
		object	product	
		operator	software	
		product	storage	
		software	time	
		storage	utilization	
time				
utilization				
52	Resource Utilization	amount	amount	7 from 8
		capability	capability	0.875
		duration	function	
		function	memory	
		memory	product	
		product	program	
		program	software	
software				
53	Compatibility	ability	ability	12 from 17
		communication	data	0.705882353
		compatibility	degree	
		data	environment	
		degree	extent	
		environment	hardware	
		exchange	information	
		extent	measure	
		file	software	
		hardware	system	
		information	user	
		measure	work	
		share		
software				
system				
user				
work				
54	Independence	application	application	10 from 15
		architecture	computer	0.666666667
		computer	computing	
		computing	data	
		data	environment	
		database	hardware	
		dependency	operating	
		environment	software	
		hardware	structure	
		independence	system	
		microcode		
		operating		
		software		
structure				
system				
55	Simplicity	component	component	11 from 12
		concern	concern	0.916666667



Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		definition	definition	
		degree	degree	
		design	design	
		factor	factor	
		implementation	implementation	
		manner	manner	
		quality	quality	
		software	software	
		step	system	
		system		
56	Expandability	capability	capability	13 from 14
		component	component	0.928571429
		concern	concern	
		data	data	
		degree	degree	
		efficiency	efficiency	
		effort	effort	
		factor	factor	
		functionality	functionality	
		increase	performance	
		performance	program	
		program	quality	
		quality	software	
		software		
57	Generality	application	application	6 from 11
		being	component	0.545454545
		breadth	degree	
		component	respect	
		degree	software	
		means	system	
		nature		
		range		
		respect		
		software		
		system		
58	System Clarity	clarity	concern	10 from 13
		concern	ease	0.769230769
		description	factor	
		ease	manner	
		factor	measure	
		manner	program	
		measure	quality	
		people	software	
		program	structure	
		quality	understanding	
		software		
		structure		
		understanding		
59	Survivability	extent	extent	4 from 6
		portion	probability	0.666666667
		probability	software	
		software	system	
		survivability		
		system		
60	Verifiability	access	access	15 from 22
		architecture	code	0.681818182
		assembly	component	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		certification	design	
		code	effort	
		component	extent	
		design	function	
		effort	implementation	
		extent	operation	
		function	performance	
		implementation	quality	
		interface	scope	
		operation	software	
		performance	source	
		quality	test	
		scope		
		software		
		source		
		test		
		testability		
		verifiability		
		verification		
61	Repeatability = Reproducibility	degree	degree	5 from 6
		environment	environment	0.833333333
		operating	operating	
		reproducibility	set	
		set	system	
		system		
62	Conformance	characteristic	characteristic	7 from 13
		completeness	degree	0.538461538
		compliance	design	
		conformance	functionality	
		database	operating	
		degree	portability	
		design	software	
		functionality		
		operating		
		portability		
		portion		
		software		
		sql		
63	Capacity	ability	ability	13 from 16
		amount	amount	0.8125
		capacity	data	
		data	design	
		demand	environment	
		design	information	
		environment	measure	
		information	number	
		measure	operating	
		number	rate	
		operating	system	
		rate	throughput	
		suitability	work	
		system		
		throughput		
		work		
64	Buildability	build	characteristic	8 from 13
		characteristic	developer	0.615384615
		developer	ease	

Att. ID	Attribute	Def. Concepts	Onto. Concepts that cover	Count and Average
		ease	period	
		people	product	
		period	software	
		product	system	
		scenario	time	
		software		
		system		
		term		
		thing		
		Time		
65	Readability	code	code	7 from 8
		developer	developer	0.875
		documentation	documentation	
		ease	ease	
		level	level	
		source	source	
		statement	system	
		system		
66	Productivity	capability	capability	5 from 7
		context	context	0.714285714
		effectiveness	effectiveness	
		expend	product	
		product	software	
		relation		
		software		
The Average of Coverage Averages is :			0.734520723	

## APPENDIX C

The suggested ontology domain concepts:

Table C.1: The final suggested ontology domain concepts list.

Concept	Concept	Concept	Concept
ability	documentation	memory	risk
access	ease	minimum	scope
accessibility	effect	modification	service
accuracy	effectiveness	nature	set
adapt	efficiency	notation	setting
adaptability	effort	number	software
adaptation	environment	object	source
amount	error	objective	specification
applicability	exchange	operating	state
application	express	operation	storage
architecture	extent	operator	structure
attribute	factor	output	system
availability	failure	people	test
capability	freedom	performance	testability

capacity	function	period	throughput
change	functionality	portability	time
characteristic	hardware	precision	understand
code	idea	presence	understandability
component	impact	probability	understanding
computer	implementation	product	uniform
computing	incorporation	program	uniformity
concern	information	property	usability
context	interface	purpose	usage
control	interval	quality	user
data	level	rate	utility
definition	maintenance	relationship	utilization
degree	manner	reliability	variety
demand	mean	requirement	verification
design	meaning	resource	
develop	means	respect	
developer	measure	response	
development	meeting	responsiveness	

## APPENDIX D

Relationships between groups of concepts in the suggested ontology domain:

Table D.1: Relationships between groups of concepts in the ontology domain.

Group No	Level	Con1	Con2
1	1	Software, system, requirement characteristic, function,	Attribute, design, test ,user
2	1	Performance, degree, component	Data, effort, function, software ,system
2	2	Data, effort	Component, degree, performance, requirement, function ,software, system, user
3	1	Environment, program, ability	Component, requirement, software
4	1	Extent, time ,product	software ,system
5	1	Operate (ion), ease, change	Environment, software, system
6	1	Resource, specification, implementation	extent
7	1	Capability, code	Environment, performance, requirement
8	1	Modification, measure	Ability, requirement
9	1	amount ,state	Function, resource, software
10	1	Application, applicability, understand	Modification, requirement, system ,user
11	1	Level, modification	Product, software, system
12	1	Service, access	Requirement, system, user
13	1	Effect, set	Attribute, resource, system, user
14	1	Develop (er), failure	Ease, product, software

Group No	Level	Con1	Con2
15	1	Output, computer	Amount, system
16	1	Efficiency, quality	characteristic
17	1	meeting	Modification, performance
18	1	Documentation, concern	Software, system
19	1	Hardware, purpose	Environment, software
20	1	Number	amount ,specification
21	1	information	Ability, data, degree, documentation, exchange, object, software, system
22	1	control	Access, attribute, characteristic, data, degree, operation, user, idea
22	2	idea	Ease
23	1	precision	Requirement, service
24	1	Adapt, utility (ization)	characteristic
25	1	probability	Availability, express, extent ,failure, function, performance, program, time
26	1	interface	software
27	1	Mean, context	change
28	1	probability	Ability, characteristic, code, degree, function ,time, Verification
28	2	Verification	Component, interface, set
29	1	Freedom, uniform	Environment
30	1	Storage, reliability	code
31	1	response	Design, measure, meeting ,system ,throughput, time
31	2	throughput	Rate, requirement, response, time
33	1	error	Maintenance, measure, precision, program, requirement, system
33	2	maintenance	Adaptability, attribute, ease, error, impact ,state
33	3	impact	component ,maintenance ,system
34	1	Scope, accuracy	extent
35	1	Usage, usability	resource
36	1	work ,period	system
37	1	relationship	Attribute, degree, function, modification, product
38	1	notation	Definition, degree, implementation, quality, uniform
38	2	definition	Implementation, level, notation
39	1	testability	Characteristic, code ,effort ,extent, number
40	1	memory	Amount, efficiency, time, usage
41	1	manner	degree, modification, quality, usage
42	1	structure	data ,design, measure, software, understand
43	2	architecture	code ,design,
44	1	respect	Capability, implementation, output,

Group No	Level	Con1	Con2
			<b>performance, requirement</b>
45	1	<b>minimum</b>	<b>Amount, function, resource, software</b>
46	1	<b>source</b>	<b>Access, attribute, code ,concern</b>
47	1	<b>risk</b>	<b>Change, freedom ,people, software</b>
47	2	<b>people</b>	<b>Component, measure, risk</b>
48	1	<b>factor</b>	<b>Ability, concern, quality, software</b>
49	1	<b>demand</b>	<b>Object, rate,</b>
50	1	<b>presence</b>	<b>Ability, usage</b>
51	1	<b>variety</b>	<b>Component, operation</b>
52	1	<b>nature</b>	<b>Change, utility</b>
53	1	<b>incorporation</b>	<b>Change, requirement</b>

## APPENDIX E

Each SWPQA concepts that belong to our ontology domain:

Table E.1: Each SWPQA definition concepts from our ontology domain concepts.

Def. ID	Attribute	Def. Concepts From Ontology Domain
1	Accuracy	accuracy
		capability
		computer
		concern
		degree
		error
		extent
		factor
		freedom
		measure
		output
		precision
		product
		quality
respect		
2	Adaptability	software
		system
		ability
		adaptation
		change
		component
		degree
		ease
		environment
		modification
operating		
product		
purpose		

Def. ID	Attribute	Def. Concepts From Ontology Domain
		software
		system
		means
3	Analyzability	ability
		capability
		effort
		failure
		product
		software
4	Attractiveness	capability
		product
		software
		user
5	Availability	component
		data
		degree
		level
		probability
		program
		service
		software
		specification
		system
		time
		usage
		availability
		meeting
		reliability
		requirement
		demand
6	Changeability	amount
		capability
		change
		effort
		modification
		product
		set
		software
		System
7	Complexity	attribute
		code
		component
		concern
		control
		data
		degree
		design
		extent
		factor
		implementation
		measure
		quality

Def. ID	Attribute	Def. Concepts From Ontology Domain
		set
		structure
		system
8	Compliance	application
		capability
		characteristic
		software
9	Consistency	code
		component
		concern
		definition
		degree
		design
		factor
		freedom
		implementation
		level
		notation
		quality
		software
		source
		system
		uniform
		uniform
		uniformity
10	Co-existence	capability
		environment
		product
		software
11	Efficiency	ability
		amount
		attribute
		characteristic
		code
		component
		computer
		computing
		concern
		degree
		efficiency
		extent
		factor
		function
		function
		function
		functionality
		hardware
		level
		meaning
		measure
		memory
		number



Def. ID	Attribute	Def. Concepts From Ontology Domain
		performance
		product
		program
		purpose
		quality
		rate
		response
		set
		software
		storage
		system
		time
		usability
		usage
		user
		utility
		utilization
		minimum
		relationship
		resource
		express
12	Functionality	ability
		capability
		characteristic
		component
		design
		extent
		product
		purpose
		service
		set
		software
		user
		express
13	Installability	capability
		effort
		environment
		product
		software
14	Interoperability	ability
		capability
		component
		computing
		degree
		effort
		environment
		extent
		hardware
		information
		product
		software
		system

Def. ID	Attribute	Def. Concepts From Ontology Domain
		exchange
		variety
		interface
15	Learnability	application
		capability
		control
		effort
		operation
		output
		product
		software
		user
		work
16	Maintainability	ability
		adaptation
		amount
		attribute
		capability
		change
		characteristic
		code
		component
		concern
		developer
		ease
		efficiency
		effort
		environment
		error
		extent
		factor
		failure
		hardware
		incorporation
		level
		maintenance
		operating
		operation
		performance
		period
		probability
		product
		program
		purpose
		quality
		rate
		scope
		set
		software
		system
		time
		understandability

Def. ID	Attribute	Def. Concepts From Ontology Domain
17		impact
		interval
		verification
		testability
		state
		Nature
	Maturity	capability
		effectiveness
		efficiency
		failure
		product
		respect
		software
		Work
18	Operability	ability
		capability
		control
		degree
		ease
		effort
		environment
		functionality
		operation
		product
		purpose
		software
		user
		19
amount		
attribute		
component		
computer		
concern		
degree		
effectiveness		
efficiency		
factor		
function		
memory		
number		
objective		
operating		
output		
performance		
period		
program		
quality		
rate		
respect		
response		
responsiveness		
service		

Def. ID	Attribute	Def. Concepts From Ontology Domain
		software
		system
		throughput
		time
		usage
		utilization
		work
		availability
		interval
		meeting
		Resource
20	Portability	ability
		adaptability
		amount
		capability
		characteristic
		code
		component
		computing
		concern
		degree
		design
		ease
		effort
		environment
		extent
		factor
		hardware
		implementation
		measure
		object
		operating
		portability
		product
		program
		quality
		set
		software
		system
		time
		time
		variety
21	Recoverability	ability
		capability
		data
		effort
		failure
		level
		operation
		performance
		software
		system

Def. ID	Attribute	Def. Concepts From Ontology Domain
22	Reliability	time
		ability
		application
		capability
		code
		component
		concern
		design
		environment
		extent
		factor
		fail
		failure
		function
		level
		mean
		measure
		number
		operating
		performance
		period
		precision
		probability
		product
		program
		quality
		rate
		service
		set
		software
		specification
system		
time		
usage		
interval		
reliability		
23	Replaceability	capability
		component
		effort
		environment
		product
		purpose
		software
		exchange
24	Robustness	ability
		component
		degree
		program
		system
25	Safety	presence
		capability
		computer

Def. ID	Attribute	Def. Concepts From Ontology Domain
		context
		environment
		freedom
		measure
		product
		property
		software
		system
		means
		people
		risk
26	Scalability	ability
		capability
		degree
		design
		efficiency
		hardware
		implementation
		performance
		product
		response
		software
		system
		throughput
		time
		capacity
		Demand
27	Security	ability
		access
		capability
		code
		component
		computer
		data
		definition
		degree
		extent
		freedom
		information
		manner
		modification
		object
		product
		property
		software
		system
		user
		idea
		resource
		risk
		state

Def. ID	Attribute	Def. Concepts From Ontology Domain
28	Stability	change
		effect
		product
		software
		system
		impact
		risk
29	Suitability	attribute
		capability
		characteristic
		functionality
		product
		set
		software
		specification
		user
		Presence
30	Testability	attribute
		capability
		characteristic
		code
		component
		concern
		degree
		ease
		effort
		error
		extent
		function
		number
		performance
		product
		program
		scope
		setting
		software
		specification
		system
		test
		idea
		interface
requirement		
testability		
verification		
31	Traceability	ability
		component
		degree
		design
		development
		documentation
environment		

Def. ID	Attribute	Def. Concepts From Ontology Domain
		modification
		product
		respect
		software
		system
		relationship
		requirement
32	Understandability	amount
		applicability
		application
		code
		component
		computer
		concern
		degree
		design
		ease
		effort
		factor
		implementation
		level
		operation
		output
		product
		purpose
		quality
		software
		structure
		system
		understand
		understandability
		understanding
		User
33	Usability	ability
		application
		capability
		characteristic
		code
		component
		computer
		context
		developer
		documentation
		ease
		effect
		effectiveness
		efficiency
		effort
		end
		extent
		functionality
		information



Def. ID	Attribute	Def. Concepts From Ontology Domain
		measure
		operation
		output
		product
		program
		quality
		set
		software
		system
		understandability
		usability
		user
		utility
		express
		variety
34	Utility	characteristic
		code
		computer
		efficiency
		extent
		portability
		purpose
		usability
		reliability
35	fault tolerance	ability
		component
		failure
		level
		performance
		software
		system
		interface
36	Reusability	application
		component
		computing
		concern
		context
		degree
		ease
		effort
		extent
		product
		program
		scope
		software
		system
		work
37	Correctness	ability
		attribute
		component
		concern
		degree

Def. ID	Attribute	Def. Concepts From Ontology Domain
		design
		documentation
		extent
		implementation
		operation
		performance
		precision
		program
		scope
		set
		software
		specification
		system
		user
		architecture
38	Modifiability	attribute
		change
		component
		concern
		degree
		ease
		effect
		environment
		extent
		hardware
		incorporation
		measure
		modification
		performance
		product
		software
		system
		state
39	Completeness	attribute
		concern
		degree
		factor
		implementation
		measure
		number
		program
		quality
		scope
		software
		system
		meeting
40	Dependability	attribute
		component
		computer
		degree
		property
		service

Def. ID	Attribute	Def. Concepts From Ontology Domain
		system
		Availability
41	Extensibility	attribute
		component
		functionality
		capacity
42	Customizability	attribute
		component
		number
43	Modularity	attribute
		change
		code
		component
		computer
		concern
		degree
		factor
		function
		level
		number
		program
		quality
		software
		source
		structure
		system
		impact
		relationship
44	Flexibility	ability
		change
		code
		component
		concern
		context
		data
		design
		developer
		case
		effort
		environment
		extent
		factor
		incorporation
		operating
		quality
		software
		system
		use
45	Integrity	ability
		access
		application

Def. ID	Attribute	Def. Concepts From Ontology Domain
		code
		component
		computer
		concern
		data
		degree
		extent
		factor
		function
		information
		manner
		modification
		object
		object
		program
		quality
		software
		system
		user
		idea
		requirement
		state
46	Accessibility	access
		accessibility
		control
		data
		manner
		software
		system
		test
		usage
		means
47	Communicativeness	degree
		software
		system
		means
48	Self Descriptiveness	component
		degree
		function
		implementation
		information
		software
		system
49	Conciseness	ability
		amount
		code
		concern
		factor
		function
		implementation
		information
		quality

Def. ID	Attribute	Def. Concepts From Ontology Domain
50	Extendability	software
		minimum
		component
		design
		ease
		incorporation
		storage
		system
		usage
		capacity
51	Effectiveness	presence
		ability
		accuracy
		capability
		context
		design
		functionality
		object
		operator
		product
		software
		storage
		time
utilization		
Minimum		
52	Resource Utilization	amount
		capability
		function
		memory
		product
		program
53	Compatibility	software
		ability
		data
		degree
		environment
		extent
		hardware
		information
		measure
		software
		system
		user
		work
exchange		
54	Independence	application
		computer
		computing
		data
		environment
		hardware
operating		

Def. ID	Attribute	Def. Concepts From Ontology Domain
		software
		structure
		system
		architecture
55	Simplicity	component
		concern
		definition
		degree
		design
		factor
		implementation
		manner
		quality
		software
		system
56	Expandability	capability
		component
		concern
		data
		degree
		efficiency
		effort
		factor
		functionality
		performance
		program
		quality
		Software
57	Generality	application
		component
		degree
		respect
		software
		system
		means
		nature
58	System Clarity	concern
		ease
		factor
		manner
		measure
		program
		quality
		software
		structure
		understanding
		people
59	Survivability	extent
		probability
		software
		system
60	Verifiability	access

Def. ID	Attribute	Def. Concepts From Ontology Domain
		code
		component
		design
		effort
		extent
		function
		implementation
		operation
		performance
		quality
		scope
		software
		source
		test
		architecture
		interface
		testability
		verification
61	Repeatability = Reproducibility	degree
		environment
		operating
		set
		system
62	Conformance	characteristic
		degree
		design
		functionality
		operating
		portability
		Software
63	Capacity	ability
		amount
		data
		design
		environment
		information
		measure
		number
		operating
		rate
		system
		throughput
		work
		capacity
		demand
64	Buildability	characteristic
		developer
		ease
		period
		product
		software
		system

Def. ID	Attribute	Def. Concepts From Ontology Domain
		time
		people
65	Readability	code
		developer
		documentation
		ease
		level
		source
		system
66	Productivity	capability
		context
		effectiveness
		product
		software

## APPENDIX F

Relationships between each concept in the ontology domain and other concepts also in the domain:

Table F.1: Each concept relationships with others in the ontology domain concepts.

ID	Concept	Other Concepts that have relationships with the first concept
1	ability	Component, requirement, software, Modification, measure, information, portability, factor, presence, portability
2	access	Requirement, system, user, control, source
3	accessibility	Requirement, system, user, control, source
4	accuracy	Extent
5	adapt	Characteristic
6	adaptability	Maintenance
7	adaptation	Environment, Software, Maintenance, Modification
8	amount	Function, resource, software, Output, computer, Number, memory, minimum, computing
9	applicability	Modification, requirement, system ,user
10	application	Modification, requirement, system ,user
11	architecture	code ,design
12	attribute	Software, system, requirement characteristic, function, Effect, set, control, maintenance, relationship, source, quality, meaning, means, portability, property, responsiveness
13	availability	Probability
14	capability	Environment, performance, requirement, respect
15	capacity	Function, System, Requirement, Software
16	change	Environment, software, system, Mean, context, risk, nature, means
17	characteristic	Attribute, design, test ,user, Efficiency, quality, control, Adapt, utility (ization), portability, testability, property
18	code	Environment, performance, requirement, portability,



ID	Concept	Other Concepts that have relationships with the first concept
		Storage, reliability, testability, architecture, source
19	component	Data, effort, function, software ,system, Environment, program, ability, Verification, impact, people, variety
20	computer	Amount, system
21	computing	Amount, system
22	concern	Software, system, source, factor
23	context	Change
24	control	Access, attribute, characteristic, data, degree, operation, user, accessibility
25	data	Performance, degree, component, Component, degree, performance, requirement ,software, system, user, Information, control, structure
26	definition	Notation, Implementation, level, notation, meaning, means
27	degree	Data, effort, function, software ,system, information, control, portability, relationship, notation, manner, level
28	demand	Object, rate
29	design	Software, system, requirement characteristic, function, response, structure, architecture
30	develop	Ease, product, software
31	developer	Ease, product, software
32	development	Ease, product, software
33	documentation	Software, system, information, meaning, means, understanding
34	ease	Environment, software, system, Develop(er), failure, idea, maintenance
35	effect	Attribute, resource, system, user
36	effectiveness	Efficiency, Quality, Program, Function, Factor
37	efficiency	Characteristic, memory, effectiveness
38	effort	Performance, degree, component, requirement ,software, system, user, Testability
39	environment	Component, requirement, software, Operate(ion), ease, change, Capability, code, Hardware, purpose Freedom, uniform, adaptation
40	error	Maintenance, measure, precision, program, requirement, system
41	exchange	Information
42	express	Probability
43	extent	software ,system, Resource, specification, implementation, probability, Scope, accuracy, testability
44	factor	Ability, concern, quality, software, effectiveness
45	failure	Ease, product, software, probability
46	freedom	Environment, risk
47	function	Attribute, design, test ,user, Performance, degree, component, amount ,state, probability, relationship, minimum, capacity, effectiveness, responsiveness
48	functionality	Attribute, design, test ,user, Performance, degree, component, amount ,state, probability, relationship, minimum
49	hardware	Environment, software
50	idea	Ease

ID	Concept	Other Concepts that have relationships with the first concept
51	impact	Maintenance, component, system, interval
52	implementation	Extent, notation, definition, respect
53	incorporation	Change, requirement
54	information	Ability, data, degree, documentation, exchange, object, software, system
55	interface	Software, Verification
56	interval	Time, Period, impact, minimum
57	level	Performance, Degree
58	maintenance	Error, Adaptability, attribute, ease, impact ,state, adaptation
59	manner	degree, modification, quality, usage
60	mean	Change
61	meaning	Definition, Attribute, Documentation
62	means	Change, Definition, Attribute, Documentation
63	measure	Ability, requirement, response, error, structure, people
64	Meeting (meet)	Modification, performance, response
65	memory	Amount, efficiency, time, usage, storage
66	minimum	Amount, function, resource, software, interval
67	modification	Ability, requirement, Application, applicability, understand, Product, software, system, meeting, relationship, manner, adaptation , understanding
68	nature	Change, utility
69	notation	Definition, degree, implementation, quality, uniform
70	number	amount ,specification, testability
71	object	Information, demand
72	objective	Information, demand
73	operating	Environment, software, system, control, variety
74	operation	Environment, software, system, control, variety
75	operator	Environment, software, system, control, variety
76	output	Amount, system, respect
77	people	Risk, Component, measure
78	performance	Data, effort, function, software ,system, Capability, code, meeting, probability, respect, level
79	period	System, time, interval
80	portability	Ability, Program, Software, Attribute, utilization
81	precision	Requirement, service, error
82	presence	Ability, usage
83	probability	Availability, express, extent ,failure, function, performance, program, time, Ability, characteristic, code, degree
84	product	software ,system, Level, modification, Develop(er), failure, relationship
85	program	Component, requirement, software, probability, error, effectiveness, portability, responsiveness, utilization
86	property	Characteristic, Attribute, Software
87	purpose	Environment, software
88	quality	Characteristic, notation, manner, factor, Attribute, effectiveness
89	rate	Throughput, demand
90	relationship	Attribute, degree, function, modification, product
91	reliability	Code

ID	Concept	Other Concepts that have relationships with the first concept
92	requirement	Attribute, design, test ,user, Data, effort, Environment, program, ability, Capability, code, Modification, measure Application, applicability, understand, Service, access, precision, throughput, error, respect, incorporation, accessibility, capacity, understanding
93	resource	Extent, amount ,state, Effect, set, Usage, usability, minimum
94	respect	Capability, implementation, output, performance, requirement
95	response	Design, measure, meet ,system ,throughput, time, responsiveness
96	responsiveness	Response, function, software, Attribute, program, time
97	risk	Change, freedom ,people, software
98	scope	Extent
99	service	Requirement, system, user, precision,
100	set	Attribute, resource, system, user, Verification
101	Setting	Attribute, resource, system, user, Verification
102	software	Attribute, design, test ,user, Performance, degree, component, Data, effort, Environment, program, ability, Extent, time ,product, Operate(ion), ease, change, amount ,state, Level, modification, Develop(er), failure, portability Documentation, concern, Hardware, purpose, information, interface, structure, minimum, risk, factor, adaptation, capacity, property, responsiveness, utilization
103	source	Access, attribute, code ,concern, accessibility
104	specification	Extent, Number
105	state	Function, resource, software, maintenance, uniformity
106	storage	Code, memory
107	structure	data ,design, measure, software, understand, understanding
108	system	Attribute, design, test ,user, Performance, degree, component, Data, effort, Extent, time ,product, Operate(ion), ease, change, Application, applicability, understand, Level, modification, Service, access, Effect, set, Output, computer Documentation, concern, information, response, error, impact, work ,period, accessibility, capacity, computing, understanding, uniformity
109	test	Software, system, requirement characteristic, function
110	testability	Characteristic, code ,effort ,extent, number
111	throughput	Response, Rate, requirement, time
112	time	Period, software ,system, probability, response, throughput, memory, interval, responsiveness
113	understand	Modification, requirement, system ,user, structure, Documentation
114	understandability	Modification, requirement, system ,user, structure, Documentation
115	understanding	Modification, requirement, system ,user, structure, Documentation
116	uniform	Environment, notation, uniformity
117	uniformity	State, uniform, System
118	usability	Resource
119	usage	Resource, memory, manner, presence,
120	user	Software, system, requirement characteristic, function, Data,

ID	Concept	Other Concepts that have relationships with the first concept
		effort, Application, applicability, understand, Service, access, Effect, set, control, accessibility, understanding, utilization
121	utility	Characteristic, nature
122	utilization	Software, User, Program, Portability
123	variety	Component, operation
124	verification	Component, interface, set
125	work	System