INSTITUTE OF MATHEMATICS AND INFORMATICS
VYTAUTAS MAGNUS UNIVERSITY

Jelena Gasperovič

EVALUATION OF FUNCTIONALITY OF SPECIFICATION LANGUAGES

Doctoral dissertation

Physical sciences, informatics (09 P)

Vilnius, 2007

Dissertation has been prepared during the period from 2002 until 2006 in the Institute of Mathematics and Informatics.

Scientific Advisor:
Prof. Habil. Dr. (HP) Albertas Čaplinskas (Institute of Mathematics and Informatics, physical sciences, informatics – 09 P).

MATEMATIKOS IR INFORMATIKOS INSTITUTAS
VYTAUTO DIDŽIOJO UNIVERSITETAS

Jelena Gasperovič

SPECIFIKAVIMO KALBŲ FUNKCIONALUMO VERTINIMAS

Daktaro disertacija

Fiziniai mokslai, informatika (09 P)

Vilnius, 2007

Disertacija rengta 2002 - 2006 metais Matematikos ir informatikos institute.

Mokslinis vadovas:
prof. habil. dr. (HP) Albertas Čaplinskas (Matematikos ir informatikos institutas, fiziniai mokslai, informatika – 09 P).

Gratitude and Acknowledgments

I would like to particularly thank my advisor Prof. Albertas Čaplinskas for directing this dissertation. He was undoubtedly the architect of this work. I am truly grateful to him for his understanding, help and valuable advices. I would like to thank him for always inducing me to trust myself, push further my ideas, and also for all the opportunities he offered me. I say "*Thank You*" to him with all my heart.

I would like to express my gratitude to Dr. Audrone Lupeikiene, Assoc. Prof. Dr. Dale Dzemydiene for valuable criticism, comments and discussions.

I would like to express my thankfulness to all personnel and doctoral students of the Software Engineering Department of the Institute of Mathematics and Informatics for help, inducement and understanding. I'm also grateful to all people, who have contributed to this work by commenting it and giving advices.

I think that recognition should be given to the head of the Lithuanian IT company "*Algoritmu sistemos*" Rimgaudas Zaldokas, who had some impact to my choice of the research field and for his support and understanding during my writing of this work and during the four years of my doctoral studies.

A particular credit should be given to Lithuanian State Science and Studies Foundation for financial support of this work during nine last months of doctoral studies (from 2006.01.01 until 2006.10.01).

I would like to finish by thanking my family and all my friends, who morally supported me not only during writing of this work, but also during the time that I have been studying.

Abstract

The dissertation presents an approach to evaluate quality of specification languages. It is divided into five parts.

The first part is introductory. It presents the main information about the research, such as research problem, research question and objectives, motivation of the research, in the research used methodology, research findings and main results, its scientific novelty, practical importance, and approbation. Besides, the section has information about the papers, in which the main results of the research were published.

The second part presents problem statement. First, the main concepts of specification languages theory and quality evaluation theory are described. Further the scientific hypotheses are stated and described. Finally, in the dissertation proposed main scheme of specification languages quality evaluation is presented and described shortly.

The third part is analytical. It analyses and compares known approaches to evaluate quality of specification languages. First, methods of comparative analysis are described shortly, and the methodology to compare the analysed approaches is developed. Then the approaches to evaluate quality of specification languages are described in detail and compared using the proposed methodology.

The fourth part describes the research design. First of all, internal quality is distinguished from quality in use. Taxonomy of quality characteristics, which has five hierarchical levels, is developed. It is proposed to evaluate the lowest-level characteristics (elementary characteristics) of the taxonomy using the library of evaluation test examples, and to aggregate the values of elementary characteristics up to a single value, describing internal quality of the whole specification language, using aggregation techniques. Further it is considered that quality in use should be evaluated on the basis of internal quality. To evaluate quality in use it is proposed to specify the quality goals of the particular project and construct a quality model. The dissertation elaborates in the literature proposed quality model and its construction procedure, and proposes the procedure to evaluate quality in use.

The fifth part describes the research experiment. It presents the experimental evaluation of UML and Z specification languages functionality. According to the results of UML and Z evaluation conclusions about functionality of these languages are made.

Contents

List of Figures

List of Tables

Concepts

Concept	Definition
Aggregation operator	A mathematical object that has the function of reducing a set of numbers into a unique representative (or meaningful) number [Det00].
Aggregation technique	A well-defined procedure required calculating internal quality from the results of measurements.
Elementary characteristic (Quality sub-characteristic)	Internal quality sub-characteristic, which can be evaluated using the library of evaluation test examples.
Elementary problem frame	A description of a recognisable class of problems, where the class of problems has a known solution. In a sense, elementary problem frames are problem patterns.
Evaluation test example (Representative example)	For the particular category of Software Systems developed representative example used to test the sufficiency of a specification language for specification of requirements of any Software System from the current population of Software Systems.
Feature diagram	A graphical representation of a feature model.
Feature model	A model that represents the standard features of a family of systems in the domain and relationships between them.
Feature table	A tabular notation, which describes feature models and allows collecting together all information about the feature that in the original Feature-Oriented Domain Analysis approach is distributed among feature diagrams, feature definitions and rationale of features.
Formal specification language	A specification language with formally defined syntax and semantics.
Functionality	The set of features necessary to describe requirements of a future system.
Goal Interdependency Graph	User's treatment of quality goals and interdependencies between goals.
Infrastructure	A particular software tool(s) required supporting data collection, sampling, and testing of internal quality characteristics.
Internal quality	Internal quality of a specification language is the descriptive characteristic of a language as a product independently from any context of use. It means that internal quality is described by some value obtained from measurements.
Library of evaluation test examples (Library of representative examples)	The set of evaluation test examples for the particular category of Software Systems.
Lightweight formal specification language	A specification language with partially formal syntax and greatly simplified semantics.
Linguistic system	A system defining a formal structure beyond a specification language.
Multi-frame	A composition of conceptual subsystems, each from which can be described by an elementary frame.
Problem diagram	A graphical representation of a problem frames.
Quality	Totality of characteristics of an entity that bear on its

Concept	Definition
	ability to satisfy stated or implied needs [ISO94].
Quality assessment function	A function used to propagate rating values of quality characteristics through the Goal Interdependency Graph from bottom towards the top of the graph.
Quality characteristics	A set of attributes of a language, by which its quality is described and evaluated. Quality characteristics may be refined into multiple levels of sub-characteristics.
Quality evaluation plan	A plan of a specification language evaluation, which contains the set of suites of quality evaluation tests, time and financial constraints, evaluators and other necessary information.
Quality evaluation test	The test that consists of a test identifier, a test example, test execution conditions, and expected results, which describe what this test allows to check.
Quality evaluation test coverage	The degree to which a given quality evaluation test or suite of tests addresses all specified requirements for the Software System of the particular category.
Quality goal	High-level requirements formulated by users (audience) of a specification language.
Quality in use	Quality in use is evaluative characteristic of a language obtained by making a judgment based on criteria that determine the worthiness of a language for a particular project.
Quality model	A formula and a set of instructions for how the obtained quality measures are to be interpreted to draw conclusions about the quality of a specification language.
Representation system	A system defining forms of representation of specification language constructs.
Sample	A part or subset of the population taken to be representative of this population as a whole for some investigative purposes of research [Co77]. In the context of this dissertation the term population refers to all systems of the particular category of Software Systems.
Sampling	The technique of selecting a suitable sample.
Semi-formal specification language	A language with formally defined syntax, but not completely defined semantic. Usually such languages depend on data-flow diagrams, entity-relationship diagrams, data dictionaries, finite state machines, or decision tables.
Specification language	A language of a high abstract level with syntax and semantics appropriate enough to define both the functionality of a system under consideration and its non-functional properties.
Suite of quality evaluation tests	A collection of tests developed to test some top-level group of the characteristics of internal quality (functionality, reliability, efficiency, and usability) for the particular specification language using the same testing infrastructure.
Taxonomy of quality characteristics	A hierarchical classification of quality characteristics, represented by tree. It consists of 5 hierarchical levels, and includes 34 elementary characteristics.

Abbreviations

Concept	Abbreviation
A Lightweight Object Modelling Notation	Alloy
Bunge-Wand-Weber	BWW
Entity-Relationship-Analysis	ERA
Feature-Oriented Domain Analysis	FODA
Good decomposition model	GDM
Goal Interdependency Graph	GIG
Object Constraint Language	OCL
Structured Analysis and Design Technique	SADT
State-tracking model	STM
Textual Representation of an Object Logic Language	TROLL
Unified Modelling Language	UML
VDM Specification Language	VDL-SL
Z notation	Z

1. Introduction

1.1. Research Problem

The dissertation analyses the problem of quality evaluation of formal, semi-formal and lightweight formal specification languages, including UML, Z, VDL-SL, TROLL, and Alloy.

1.2. Research Question and Objectives

The aim of the dissertation is to propose an approach to evaluate quality of specification languages, including both internal quality and quality in use, and detail procedures allowing evaluation of specification languages functionality.

The objectives of the dissertation are the following:

- to collect, analyse and compare known approaches to evaluate quality of specification languages;
- to develop exhaustive taxonomy of internal quality characteristics and propose how it can be used to elaborate in scientific literature proposed quality models;
- to create a methodology to evaluate functionality characteristics of internal quality;
- to develop techniques to aggregate the values of elementary characteristics of internal quality;
- to create quality in use evaluation procedure;
- to carry out experimental evaluation of the functionality of UML and Z specification languages.

1.3. Motivation

Actuality of the dissertation is determined by practical needs. There exists a great variety of languages, methods, techniques, and tools. These are specification languages, domain of discourse analysis and systems development methods, systems implementation methods, and other for different purposes used tools (packages for creation of requirements DB, development packages, specialised text editors, and etc.). This list is continuing to grow. Although systems creation tools frequently have the same functionality, they are different otherwise. Thus, systems engineers are faced with the problem of selection of tools, including specification languages, which are the most appropriate for the particular project. This problem is not trivial. First of all, a language should have expressive power that is enough to specify all aspects of any theoretically possible system. On the other hand, a language should be simple, reliable and other requirements should be satisfied depending on the context of the particular project. In other words, to select the most appropriate language for the particular project multi-criteria task should be carried out.

In systems development practice attempts to carry out this task has been made for a long time. However, until now no methodology based on formally defined evaluation criteria is proposed. In the literature quality of specification languages is evaluated using comparison by subjective criteria, comparison to some standard (or norm), ontological analysis or qualitative analysis methods. However, no one of these methods is objective enough or proposes scale that could be used to measure quality characteristics of a specification language. All these methods are not developed well, and, thus, cannot be used to evaluate quality of a specification language.

Besides, this dissertation is very important from scientific point of view. Research in the field of evaluation of specification languages quality still is at the beginning. There is no comprehensive definition of quality, internal quality is not separated from quality in use, up to date exhaustive set of quality characteristics is not proposed and even no commonly accepted agreement exists about the names of quality characteristics. These problems arise because of absence of a clear picture of exactly what should be required in specifications and lack of standards for specification languages.

1.4. Research Methodology

Research methodology that is used in the dissertation includes the following methods: methods of qualitative comparative analysis, methods of taxonomical analysis, methods of non-statistical sampling, methods of domain engineering, methods of testing theory, methods of aggregation theory, statistical methods, methods of rating theory, and methods of fuzzy theory.

Methods of qualitative comparative analysis are used to compare known approaches to evaluate quality of specification languages. First, Method of conceptual analysis-based comparison is used to identify the main components of the approaches, and then Method of Agreement and Method of Difference are used to compare the identified components by their commonalities and differences.

Methods of taxonomical analysis are used to identify, define internal quality characteristics of a specification language, and organise them in the form of the hierarchical structure, called *taxonomy of quality characteristics*. This taxonomy has five levels and provides that the quality of specification language can be characterised by its functionality, reliability, usability, and efficiency. The taxonomy consists of 5 hierarchical levels, and includes 34 elementary characteristics.

Methodology to evaluate functionality characteristics of internal quality is developed by combination of *methods* of *non-statistical sampling*, *methods* of *domain engineering*, and *methods of testing theory*. It is proposed to use *method* of *non-statistical sampling* (purposeful judgment sampling) to choose a sample for each category of Software Systems from the

population of really existing systems. After this, using *methods o*f *domain analysis* for each sample a feature model should be created. These models should be used to develop a number of representative examples (evaluation test examples). Further it is proposed to use *methods of testing theory* to develop quality evaluation tests to test functionality of specification languages.

Methods of aggregation theory are used to develop techniques to aggregate the values of elementary characteristics of internal quality. Different aggregation techniques are developed according to the kind of dependencies among quality sub-characteristics in the taxonomy of quality characteristics. Besides, it is necessary to aggregate the results of measurements of a particular quality sub-characteristic obtained using several different suites of quality evaluation tests. For this aim using *statistical methods* the heuristic, which can be seen as a kind of combination of arithmetic and winsorised means, is developed.

Methods of rating theory and *methods of fuzzy theory* are used to elaborate in the literature proposed context-oriented quality model. Using these methods the quality model is supplemented by fuzzy rating functions and fuzzy interpretation functions. Fuzzy rating functions are used to map the values of quality characteristics to appropriate rating values, while fuzzy interpretation functions are used to interpret the obtained measure of the top-level quality goal.

1.5. Research Findings

The main findings of the dissertation are the following:

- Internal quality should be described by the taxonomy of quality characteristics.
- Quality in use should be evaluated on the basis of internal quality, thus the taxonomy of quality characteristics should be seen as a part of a quality model.
- The values of elementary characteristics of internal quality should be described as expected frequencies with which corresponding feature of a language L will be successfully used to specify any possible Software System development project regardless of its purpose, complexity, size and other specifics.
- Elementary characteristics of internal quality should be evaluated using the library of evaluation test examples (representative examples), developed using sampling and domain engineering theory methods.
- The values of elementary characteristics of internal quality should be obtained using suites of quality evaluation tests, developed using testing theory methods.
- The values of elementary characteristics of internal quality should be aggregated up to a single value, describing the internal quality of the whole language L.

1.6. Results

The main results of the dissertation are:

- the methodology to compare approaches to evaluate quality of specification languages;
- the taxonomy of quality characteristics;
- the methodology to evaluate functionality characteristics of internal quality, which includes the library of evaluation test examples, internal quality evaluation procedure based on development of suites of quality evaluation tests, and the procedure to interpret internal quality evaluation results;
- techniques to aggregate the values of elementary characteristics of internal quality;
- the heuristic to aggregate the results of measurements of a particular quality sub-characteristic obtained using several different suites of quality evaluation tests;
- elaborated quality model and its construction procedure;
- quality in use evaluation procedure;
- evaluation of the functionality for UML and Z specification languages.

1.7. Scientific Novelty

This dissertation is the first work, in which exhaustive quality characteristics taxonomy to evaluate internal quality is developed and the framework to evaluate quality in use is proposed. Every characteristics of the hierarchical taxonomy of quality characteristics is described in terms of the linguistic system, defining a formal structure beyond the language, and in terms of the representation system, defining forms of representation of its constructs. We hope that the proposed taxonomy will contribute both to the research on evaluation of the quality of existing specification languages and to the development of new ones.

Besides, this dissertation proposes a new approach to evaluate quality in use. It is based on the construction of quality model. This approach can be considered as more objective than the other known approaches to evaluate quality of specification languages, because, first, it allows evaluation of quality in use according to the quality goals of the particular project, second, it proposes theoretically-grounded framework for evaluation of specification languages quality in use.

1.8. Practical Importance

First of all, in the dissertation proposed taxonomy of quality characteristics could be used as a guide during creation of new specification languages, because it provides valuable information about the conceptual structure of specification languages and about the features, on which their

internal quality and quality in use depends. Besides, the proposed taxonomy, internal quality and quality in use evaluation procedures could be used to compare several different specification languages both independently from their context of use or according to their worthiness for the particular project.

Second, the work that is done in the dissertation can be used to start storing representative examples in the library. Of course, such kind of library cannot be created at once. For this aim several projects should be initiated. During each such project representative examples for the systems of the particular category should be developed. It follows from the theory of problem frames that 5-6 projects should be enough, thus, such kind of work is rather expensive, but is possible to implement in practice.

Third, in the dissertation proposed quality in use evaluation procedure can be used in practice, when it is necessary to choose the language that is the most suitable to describe requirements of the particular project. The evaluation should be done by an expert, who should apply the proposed quality in use evaluation procedure for several chosen specification languages and compare the obtained results of evaluation according to achievement of for the particular project formulated quality goals. Besides, it can be done only in such case, when internal quality for in the project created system of the particular category has been already evaluated.

1.9. Approbation

The main results of the dissertation were presented and discussed during the following in Lithuania and abroad hold national and international conferences:

1. "*Informacinės technologijos '2003*", January 28-29, 2003, Kaunas, Lithuania, Kaunas Technology University.
2. "*Kompiuterininku dienos-2003*" (the conference was held by Lithuanian Computer Society), August 28-30, 2003, Vilnius, Lithuania, Parliament of the Republic of Lithuania.
3. XLIV Conference of Lithuanian Mathematicians (the conference was held by Lithuanian Mathematicians Society), June 19-20, 2003, Vilnius, Lithuania, Vilnius Pedagogical University.
4. The Sixth International Baltic Conference "*DB&IS'2004*", June 6-8, 2004, Riga, Latvia, Institute of Mathematics and Computer Science of University of Latvia.
5. The 13th International Conference on IS Development "*ISD'2004*", September 9-11, 2004, Vilnius, Lithuania, Vilnius Gediminas Technical University.

6. XLV Conference of Lithuanian Mathematicians (the conference was held by Lithuanian Mathematicians Society), June 17-18, 2004, Kaunas, Lithuania, Lithuanian University of Agriculture.
7. "*Kompiuterininku dienos-2005*" (the conference was held by Lithuanian Computer Society), September 15-17, 2005, Klaipeda, Lithuania, University of Klaipeda.
8. XLVI Conference of Lithuanian Mathematicians (the conference was held by Lithuanian Mathematicians Society), June 15-16, 2005, Vilnius, Lithuania, Vilnius University.
9. XLVII Conference of Lithuanian Mathematicians (the conference was held by Lithuanian Mathematicians Society), June 20-21, 2006, Kaunas, Lithuania, Kaunas Technology University.

1.10. Publications

The main results of the dissertation were published in **12** papers:

- **2 papers** published in the scientific journals included into the ISI Master Journal list;
- **6 papers** published in other referred scientific journals;
- **2 papers** published in other reviewed periodical scientific publications;
- **1 paper** published in the international conference proceedings included into the ISI Proceedings list;
- **1 paper** published in the proceedings of the local scientific conference.

The list of papers is presented in the section "The List of Publications".

1.11. Synopsis

Dissertation consists of six sections.

The first section is introductory. It presents research problem, research question and objectives, motivation of the research, in the research used methodology, research findings and results, its scientific novelty, practical importance, approbation, and information about the papers, in which the main results of the research were published.

The second section presents problem statement. First, it provides definition and classification of specification languages. Then the definitions of quality, quality in use, and internal quality are provided. After that the scientific hypotheses that will be tested in the dissertation are stated and described in detail. Finally, the main specification languages quality evaluation scheme is presented.

The third section is analytical. The object of the analysis is known approaches to evaluate quality of specification languages. First, methods of comparative analysis are described, and the methodology to compare known approaches to evaluate quality of specification languages is

proposed. The approaches to evaluate quality of specification languages are described in detail and compared using the proposed methodology of comparative analysis. Finally, it is concluded that all analysed methods are incomplete, have weak theoretical background, and, thus, cannot be used to evaluate and compare quality of different specification languages.

The fourth section describes the research design. An approach to evaluate quality of specification languages is presented and described in detail. First, the taxonomy of quality characteristics is developed, and functionality characteristic and its sub-characteristics are defined and described in detail. Then the the framework to evaluate elementary characteristics of internal quality is developed. It contains the methodology to evaluate functionality characteristics of internal quality, which includes the library of evaluation test examples, internal quality evaluation procedure based on development of suites of quality evaluation tests, and the procedure to interpret internal quality evaluation results; techniques to aggregate the values of elementary characteristics of internal quality; the heuristic to aggregate the results of several measurements of a particular quality sub-characteristic. The next issues are related to evaluation of quality in use on the basis of internal quality. First of all, it is proposed how to elaborate the quality model and its construction procedure. Further, quality in use evaluation procedure is developed. It aims to evaluate quality of a specification language in context of the particular project. The quality model, the particular part of the taxonomy, which is relevant to the needs of the particular project, and quality in use evaluation procedure constitute in the dissertation proposed framework to evaluate specification language quality in use. It is concluded that this dissertation proposes a sketch of the systematic specification languages quality evaluation and, thus, it should make an impact on the whole specification languages theory. Besides, the systematic evaluation of internal quality and quality in use provides valuable experience, which could be used during construction of new specification languages.

The fifth section describes the research experiment. It is demonstrated how to evaluate functionality of UML and Z specification languages by application of the framework to evaluate elementary characteristics of internal quality. The results of experimental evaluation of UML and Z languages functionality are presented. It is concluded that evaluation of the particular aspects of the particular specification language allows preparing of valuable recommendations how to use the language in more appropriate way as well as how to improve it.

The sixth section contains the main conclusions.

At the end of the dissertation there are references, the list of publications and appendixes.

2. Problem statement

2.1. Specification Languages

Definition 1: Specification language *is the language of a high abstract level with syntax and semantics appropriate enough to define both the functionality of a system under consideration and its non-functional properties. In general case, the system under consideration can be some real-world business system, information system, software system, hardware system or any other system.*

First specification languages were created at the end of the fifth decade of the last century. In 1979 about 150 different specification languages were used [JS80]. We consider only languages that are intended to specify systems, which are relevant to the field of Information Systems engineering, namely, real-world systems, Information Systems and Software Systems. The specifications may be external (requirements specification) or internal (design specification). A specification language that supports the specification process throughout many phases of a life-cycle model is called a wide-spectrum language, and a language that supports only one or two phases of a life-cycle model is known as a narrow-spectrum language. We use the term "specification language" in the broad sense that covers both wide-spectrum and narrow-spectrum languages and suppose that wide-spectrum languages should support business modelling as well as requirement specification and design. Usually specification languages are classified into formal and semi-formal ones.

Definition 2: Formal specification language *is the language that has formally defined vocabulary, syntax, and semantics.*

The main advantage of formal specification languages is that they are amenable to automatic semantic and syntax analysis. These features increase correctness of formal specifications, because it is possible to check completeness, ambiguity and some other features of created specification. On the other hand, formal specification languages have some deficiencies, the main of which is that formal specifications are cumbersome and not easy to write and understand. These are the main reasons why formal specification languages are not widely spread in software development industry. Well-known formal specification languages are Z, VDL-SL, TROLL, OCL, and etc.

Definition 3: Semi-formal specification language *is the language with formally defined syntax, but not completely defined semantic. Usually such languages depend on data-flow diagrams, entity-relationship diagrams, data dictionaries, finite state machines, or decision tables.*

Thus, semi-formal specification languages are less strict than formal. They have long been generally recognised as an improvement over purely natural language specification. These languages represent an attempt to solve some of the ambiguity and completeness problems of natural language specifications by forcing the system behaviour description to be made using a more controlled syntax. Well-known semi-formal specification language is UML, SADT, and etc.

Recently an attempt to combine strengths of formal and semi-formal specification languages was made. A new group of languages has appeared - lightweight formal specification languages [Jac99], [Jac02], [JW96].

Definition 4: Lightweight formal specification language *is the specification language with partially formal syntax and greatly simplified semantics.*

Lightweight formal specification language like formal specification languages has vocabulary and formally defined semantics. The main difference is that syntax of such languages is less formal and semantics is simplified. The idea of simplification is related to partial specification. In other words, using these languages only some aspects of specified system are described formally, while other aspects remain semi-formal. These aspects of the system are not described completely, and for their complete specification it is necessary to write separate specifications. The main elements of lightweight approach are partiality in language, partiality in modelling, partial analysis, and partiality in composition [JW96]. A lightweight approach, in comparison to the traditional approach, lacks power of expression and breadth of coverage. On the other hand, it makes software development process more efficient and reduces the cost of creation and usage of software. Well-known lightweight formal specification language is Alloy [Jac02].

2.2. The Notion of Quality

The ISO 8402 standard [ISO94] defines quality as a "*totality of characteristics of an entity that bear on its ability to satisfy stated or implied needs*". In contractual environment needs are specified whereas in other environments implied needs should be identified and defined. Following this approach, the quality of a specification language is defined in the following way.

Definition 5: Specification language quality *is the totality of features and characteristics of this language that bear on its ability to satisfy stated or implied needs [CLV02].*

The ISO/IEC 9126 standard [ISO91] distinguishes three kinds of quality: internal quality, external quality and quality in use. Internal quality describes the level, to which the product was developed following "good engineering" practices. In other words, it evaluates the product from the point of view of developers. External quality describes the level, to which the product

correctly provides the expected services. It treats the product as "black box" and evaluates it from the point of view of potential users. Quality in use describes the level, to which a product used by particular users meets their needs to achieve some specified goals. External quality of software in many aspects depends on its internal quality. Sometimes it is not simple to distinguish external quality and internal quality at all. Although in the field of software engineering such distinction is deeply reasonable, because "good engineering" practices play in this field very important role, in many other fields, including specification languages, the situation is slightly different. To distinguish internal and external quality of a specification language is very difficult and sometimes even impossible. In addition, there are no serious reasons to make such distinction for specification languages. Thus, in this dissertation we will distinguish only two kinds of quality: internal quality and quality in use and use the term "internal quality" to address both internal and external quality.

Definition 6: Internal quality *of a specification language is the quality of a specification language itself, and it is described by some value obtained from measurements.*

Definition 7: Quality in use *of a specification language describes the extent to which a language used by specified users meets their needs to achieve specified goals in specified context of use.*

Thus, internal quality is the descriptive characteristic of a language as a product independently from any context of use, while quality in use is evaluative characteristic of a language obtained by making a judgment based on criteria that determine the worthiness of a language for the particular project.

2.3. Hypotheses

The dissertation aims to prove the following hypotheses:

Hypothesis 1: *Internal quality should be described by the taxonomy of quality characteristics.*

Internal quality is described by a set of quality characteristics. Usually a set of characteristics forms some hierarchical taxonomy, in which top-level characteristics may be refined into multiple levels of sub-characteristics. Every characteristic should be described in terms of the linguistic system, defining a formal structure beyond the language, and in terms of the representation system, defining forms of representation of its constructs.

Hypothesis 2: *Quality in use should be evaluated on the basis of internal quality, thus the taxonomy of quality characteristics should be seen as a part of a quality model.*

Quality is use should be evaluated according to the needs of the particular project, for which the high-level quality goals should be identified and quality model should be constructed. It can

be done only after evaluation of specification language internal quality independently from any context of use. The taxonomy of quality characteristics should be included into quality model, and quality characteristics should be related to quality goals. Because for the evaluation of quality in use not all internal quality characteristics may be important for the particular project, in the particular quality model only some part of the taxonomy may be used.

Hypothesis 3: *The values of elementary characteristics of internal quality should be described as expected frequencies with which corresponding feature of a language L will be successfully used to specify any possible Software System development project.*

Because it is impossible to measure quality characteristics using some kind of quantitative measurement units (such as the units used to measure distance or mass), it is supposed that internal quality of specification language L should be described as expected frequency with which corresponding feature of a language L will be successfully used to specify the current population of Software Systems. In other words, expected frequency can be considered as the frequency, with which the specification language will satisfy the needs of any possible Software System development project regardless of its purpose, complexity, size and other specifics. The value of the expected frequency should belong to the interval [0,1].

Hypothesis 4: *Elementary characteristics of internal quality should be evaluated using the library of evaluation test examples (representative examples), developed using sampling and domain engineering theory methods.*

To develop a library of evaluation test examples (representative examples), first of all, taxonomy of Software Systems should be proposed. Then, for each category of Software Systems that belong to this taxonomy the particular library of representative examples should be developed combining non-statistical sampling, domain engineering, and testing theory methods. These examples should be used as evaluation test examples to test the sufficiency of specification language L for specification of requirements of any Software System from the current population of Software Systems. For this aim we propose to use a multi-stage sampling scheme, which provides that all theoretically possible Software Systems should be first divided into categories of systems that meet principally different requirements, then from the population of really existing systems a sample for each category of Software Systems should be chosen via purposive judgment sampling. After this using domain analysis methods for each sample feature model should be developed. These models should be used to develop a number of representative examples (evaluation test examples). Further quality evaluation tests to test functionality, reliability, efficiency and usability of specification languages should be developed. It should be noted that for every internal quality sub-characteristic (functionality, reliability, efficiency, and

usability) the representative test examples and quality evaluation tests should be developed and adapted separately.

Hypothesis 5: *The values of elementary characteristics of internal quality should be obtained using suites of quality evaluation tests, developed using testing theory methods.*

Suite of quality evaluation tests should be a collection of tests developed to test some top-level group of the characteristics of internal quality (functionality, reliability, efficiency, and usability) for the particular specification language using the same testing infrastructure. It should consist of quality evaluation tests, from which each is developed to test the different category of Software Systems. In order to minimise prior arrangement efforts, the suite should be developed for the particular collection of tools required supporting the testing (i.e. for a particular testing infrastructure).

After development of suites of quality evaluation tests the results of testing should be interpreted, and the values of quality characteristics should be calculated.

Hypothesis 6: *The values of elementary characteristics of internal quality should be aggregated up to a single value, describing the internal quality of the whole language L.*

To evaluate internal quality the values of the lower-level sub-characteristics of the taxonomy should be aggregated into characteristics of the higher-level. This process should be executed using aggregation techniques, and it should be repeated until finally the internal quality is described at the higher level by single aggregated characteristic.

2.4. The Main Quality Evaluation Scheme

According to [ISO99] "*quality evaluation is a systematic examination of the extent to which an entity (part, product, service or organisation) is capable of meeting specified requirements*".

The main quality evaluation scheme is presented in Figure 1.

It is proposed to describe internal quality of a specification language by the taxonomy of quality characteristics, which has five levels and includes 34 elementary characteristics. This proposal has been based on the careful conceptual analysis of wide spectrum of specification languages, including UML, Z, VDL, TROLL, and Alloy, as well as on the analysis of quality characteristics that are used to describe quality of similar artefacts, such as programming languages and conceptual models. The proposed taxonomy is based on the classification scheme that is similar to the one described by ISO/IEC 9126 standard [ISO91]. This taxonomy provides that the quality of a specification language can be characterised by its functionality, reliability, usability, and efficiency. However, these characteristics are decomposed further in the different way than in ISO/IEC 9126.

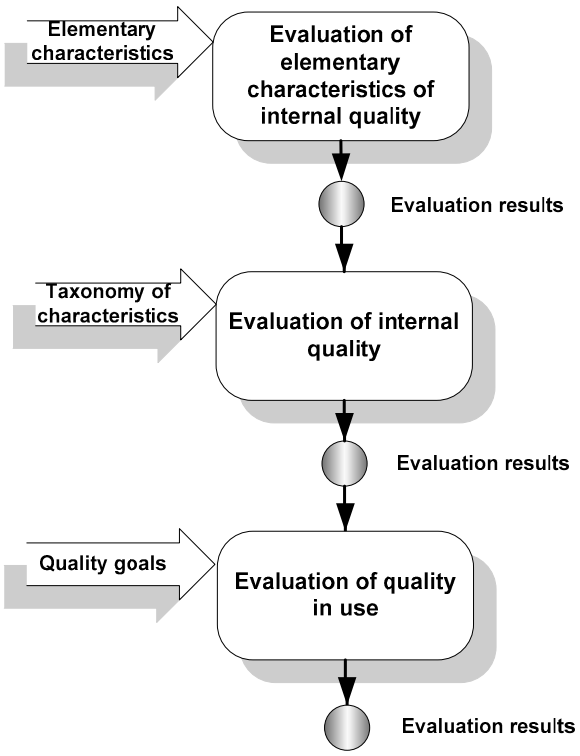


Figure 1. The main quality evaluation scheme

It is proposed to express the values of characteristics of internal quality as expected frequencies with which corresponding feature of a language L will be successfully used to specify the current population of Software Systems, and to evaluate expected frequencies using the library of evaluation test examples (representative examples).

To evaluate internal quality, different groups of related elementary characteristics are aggregated into characteristics of higher level and this process is repeated until finally the quality is described at the higher level by single aggregated characteristic. For this aim it is proposed to use methods for aggregation of the values of characteristics of internal quality. Using these methods the values of elementary characteristics can be aggregated up to a single value, describing the internal quality of the whole language L.

It is also proposed how to evaluate the quality in use of the IS specification language L on the basis of its internal quality. Quality in use is evaluated for a given Software System development project P and describes the degree of appropriateness of the language L for the project P. In order to evaluate quality in use, the quality goals of the project P must be specified.

3. Comparative Analysis of Approaches to Evaluate Quality of Specification Languages

The main results of this section are published in papers [GC03a], [GC03b], [GC03c].

3.1. Aims and Objectives

The aim of comparative analysis is to analyse and compare approaches to evaluate quality of specification languages. Comparative analysis should help to determine how by combination of the strengths of the analysed approaches and by eliminating of their weaknesses to develop an approach to evaluate quality of specification languages.

The objectives of comparative analysis are the following:

- to identify strengths and weaknesses of the analysed approaches to evaluate quality of specification languages;
- to identify evaluation criteria, which should be used during comparative analysis;
- to compare the analysed approaches by their commonalities and differences;
- to evaluate the results of comparison and make conclusions about appropriateness of the analysed approaches to evaluate quality of specification languages.

3.2. Subject of the Comparative Analysis

The subject of the comparative analysis is approaches to evaluate quality of specification languages. The following approaches are analysed:

- Bunge ontology based approach and its extension;
- Chisholm ontology based approach;
- ontological categories based approach;
- quality model based approach.

These approaches are compared using in this dissertation proposed methodology of comparative analysis, which combines the particular qualitative comparative analysis methods.

3.3. Main Principles of the Comparative Analysis

Comparative analysis is the method of theoretical research that is used practically in all sciences. Every science uses different comparative analysis methods and has specific collection of analysis tools. Generally, comparative analysis is used to determine commonalities and differences between two or more analysed phenomena. Commonalities and differences between compared phenomena are separated considering goals of comparative analysis. Usually the aim of comparative analysis is to find the best and the most appropriate instances of the particular phenomena class. Besides, during comparative analysis not only differences between instances

are determined and compared, but also an effort to explain their causes and influence on analysed phenomena class is made.

The ultimate goal of comparative analysis is to structure and classify analysed phenomena. Both qualitative and quantitative methods of comparative analysis can be used for this aim. Quantitative methods are precise, but frequently they can't be applied in practice, because it is impossible to collect all necessary data. Thus, in this dissertation attention is paid mostly to qualitative methods of comparative analysis.

Extremely important in comparative analysis are so called *essential* elements of analysed phenomenon. These are the elements that determine the nature of phenomenon by making influence on its main features directly or indirectly. The other elements of analysed phenomenon are called *secondary*. They make influence only on some individual aspects of the particular phenomenon. In other words, essential elements are associated to phenomenon commonalities, secondary – to phenomenon differences. Essential elements are more important than secondary, besides usually they are easier to understand and learn. Thus, during comparison of phenomena first it is necessary to determine and compare their essential elements and only then analyse their secondary elements. It should be noticed that essential elements usually are related to each other by dependencies (relations) or, in other words, they can be described using some pattern. Pattern matching is one of the most important qualitative methods of comparative analysis. It can be done in many different ways and using different methods. One of these methods is so called *normative* comparison, when phenomena are compared to particular standard or norm.

Generally comparative analysis is accomplished in three steps (see Figure 2).

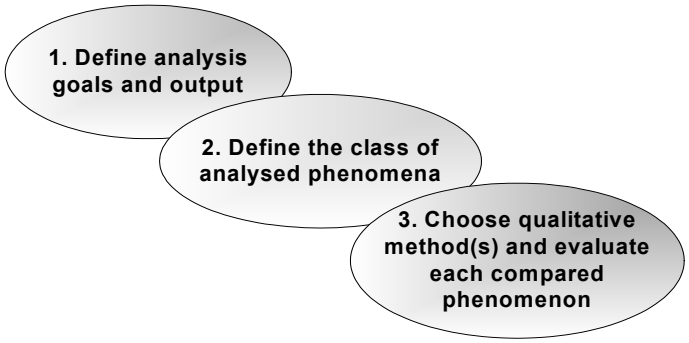


Figure 2. Steps of comparative analysis

In the first step it is necessary to determine what the goals of comparative analysis are and what results are expected after doing it. In the second step the class of analysed phenomenon is defined. Finally, in the third step qualitative method (or several methods) of comparative analysis is chosen and each compared phenomenon is evaluated (using particular evaluation

criteria and methodology of comparison), in other words, differences between phenomena, positive and negative influence of these differences on compared phenomena are identified.

3.4. Qualitative Comparative Analysis Methods

Qualitative comparative analysis methods can be defined as the ways of collecting information about different phenomena and identification of differences between these phenomena. As a rule, information gathered using qualitative comparative analysis methods is not numerically measurable. The following comparative analysis methods can be found in the literature:

- method of conventional comparison;
- method of agreement and method of difference;
- method of conceptual analysis based comparison;
- method of normative comparison;
- method of ontological categories based comparison;
- method of quality model based comparison.

Method of conventional comparison has no common comparison framework. Using this method researchers analyse strengths and weaknesses of the chosen phenomena. This is done subjectively without using any objective evaluation criteria. For example, method of conventional comparison is used in [Jac99], where specification languages Alloy [Jac02], UML and Z are evaluated and compared. The main idea of this work is to use attestation techniques or, in other words, to construct a library of characteristic situations that should be created and used to evaluate quality of a specification language. For this aim representative fragment of social reality is conceptualised in different ways (using different specification languages). Given specifications are evaluated to determine strengths and weaknesses of compared specification languages.

Such a comparison method seems to be well grounded (it is based on the methods of controlled experiments), but it cannot be considered objective, because the fragment of social reality is constructed using subjective criteria. Besides, no goals or criteria of comparison are used. Thus, it is not clear what result is expected specifying some social reality fragment using different specification languages.

Method of agreement and method of difference have been proposed by English philosopher John Stuart Mill [Mil02]. These methods are intended to shed light on issues of causation, which refers to the set of all particular "causal" or "cause-and-effect" relations between instances of the phenomena. During comparison by agreements one phenomenon is compared to the other in order to find their commonalities. In the process of comparison by differences it is attempted to

determine how one phenomenon differs from the other. Then, comparison table is filled in and used to compare different phenomena by their commonalities and differences.

The method of agreement and the method of difference can be used, for example, to identify two similar or two different Software or Information Systems. During identification of similar systems an assumption that systems are identical by essential elements is made, while during identification of different systems their differences are compared. Identification of differences results in secondary elements of analysed systems.

Both method of agreement and method of difference have some serious weaknesses. First of all, it is not clear if in practice it is really possible to determine the best (most appropriate) features of phenomena only by separating and comparing their commonalities and differences. The other deficiency is that no distinction is made between contingent and necessary features of phenomenon. This means that separate cases can be not properly generalised into common case. And, finally, if some hypothesis is stated and it affirming case is found, then, in general, that does not prove anything. Thus, both method of agreement and method of difference are not subjective enough.

More objective method of comparative analysis is the *method of conceptual analysis based comparison*. It can be used, for example, to compare Software Systems design methodologies [Son92], [SO91], [SO92]. The main idea of conceptual analysis based comparison is to separate components of each phenomenon and compare for the same purposes used ones. Components should be formalised before comparison. Formalising is considered to be conceptual modelling of a component, when model is specified using some semi-formal modelling language. Phenomena should be modelled using structural models that represent relationships between components of the compared phenomena. Separation and classification of components is based on their functional purposes criteria.

The main strength of the method of conceptual analysis based comparison is that formal basis of comparison is proposed that can be used repeatedly to satisfy in objectivity of obtained results. However, this method also has some weaknesses. First of all, in practice separation of components is rather complicated task, because a great variety of concepts exists. Each compared phenomenon may be described using different terms. That is why it is almost impossible to identify what characterises the particular component of one phenomenon, what is the equivalent of this component in the other phenomenon, and what is the difference between these two components. To solve these problems it is necessary to have a consistent system of concepts, formulated using the same conceptualisation way. In other words, it is necessary to have the ontology of the analysed phenomena. Second, functional purposes criteria are not enough for separation of components. Using this criteria it is possible to identify what

components each phenomenon has and what is their functionality, but the context of the solved problem (the particular project) is ignored at all. Third, results of comparison may have deviations generated by shortcomings of the particular modelling formalism.

Method of normative comparison makes comparison of phenomena even more objective than method of conceptual analysis based comparison. The main idea is to choose "the best" way of social reality conceptualisation, consider it to be a norm, and compare all analysed phenomena to this norm.

If $A = \{a_i\}$ is the set of norm features, $B = \{b_i\}$ is the set of analysed phenomenon features, and $R \subset A \times B$ is the collation relation, then the features of the analysed phenomena can be identified as follows:

- **Incompleteness:** it is identified what are the features that are available in the norm, but the analysed phenomenon does not have them, that is: $\exists x \forall y((x \in A) \& (y \in B) \& \neg R(x, y))$;
- **Enrichment:** it is identified by which in norm available features it is possible to enrich the analysed phenomenon that does not have such features;
- **Overload:** it is identified what are the features that are available in the norm, but in the analysed phenomenon they are expressed as only one feature, that is: $\exists x_1,x_2,y ((x_1 \in A) \& (x_2 \in A) \& (y \in B) \& R(x_1,y) \& R(x_2, y))$;
- **Redundancy:** it is identified what are the features that the analysed phenomenon has, but in the norm they are expressed as only one feature, that is: $\exists x,y_1,y_2((x \in A) \& (y_1 \in B) \& (y_2 \in B) \& R(x,y_1) \& R(x, y_2))$;
- **Excess:** it is identified what are the features that are not available in the norm, but the analysed phenomenon has them, that is: $\exists x \forall y((x \in A) \& (y \in B) \& \neg R(x, y))$.

The method of normative analysis is used, for example, to do ontological analysis of specification languages. Two normative ontologies have been proposed – Bunge (see sections 3.6, 3.7) and Chisholm (see section 3.8).

In the *method of ontological categories based comparison* it is refused to use normative ontology. Instead of this it is proposed to use the comparative framework that classifies phenomena according to their ontological categories supported by a phenomenon in the selected system of ontologies. The analysed phenomena are evaluated using qualitative scale and the formula to calculate the overall evaluation of each phenomenon. This formula depends on the ontological categories that are supported by the particular phenomena.

The method of ontological categories is used, for example, to evaluate and compare information modelling techniques (see section 3.9).

Method of quality model based comparison has common comparison framework, based on quality model construction. Quality model is used to evaluate quality characteristics of the particular phenomena according to in the quality model specified quality goals. To evaluate quality characteristics qualitative or quantitative evaluation scale should be developed.

The most typical example of quality model is used in specification languages quality evaluation approach, based on quality model construction (see section 3.10).

3.5. Methodology of Comparative Analysis

To compare known approaches to evaluate quality of specification languages it is proposed to use the methodology that combines the following qualitative analysis methods:

- method of conceptual analysis based comparison;
- method of agreement and method of difference.

Table 1. The main components of analysed approaches

Approach	**Method (s)**	**Components**					
		Goals	**Criteria**	**Quality attributes**	**Metric**	**Scale**	**Framework**
Bunge ontology based approach	Method of normative comparison	*[Available| Partially available| Not available]*	*[Available| Partially available| Not available]*	*[Available| Partially available| Not available]*	*[Available| Partially available| Not available]*	*[Available| Partially available| Not available]*	*[Available| Partially available| Not available]*
Extension of Bunge ontology based approach							
Chisholm ontology based approach	Method of normative comparison						
	Method of agreement and method of difference						
Ontological categories based approach	Method of ontological categories based comparison						
Quality model based approach	Method of quality model based comparison						

Method of conceptual analysis based comparison is used to identify the main components (Table 1) of analysed approaches to evaluate quality of specification languages. The components of the approaches are separated according to the following criteria:

- possibility to consider the particular project context by stating quality evaluation goals;
- possibility to define quality evaluation criteria;

- possibility to separate and evaluate the particular quality attributes (quality characteristics or quality features that describe different aspects and/or abilities of evaluated specification language);
- availability of metric (various parameters or ways of looking at a process that is to be measured);
- possibility to use qualitative measurement scale;
- availability of the theoretically-grounded framework of evaluation.

Method of agreement and method of difference provides the possibility to compare different phenomena by their commonalities and differences. This is done by filling in the comparison table (Table 1) and comparing the components by availability of the particular component: is the component *Available*, *Not available* or *Partially available* in the particular approach.

3.6. Bunge Ontology Based Approach

3.6.1. Bunge's Ontology

Already in 1977 an attempt to evaluate and compare specification languages was made [PT77]. Semi-formal graphical specification languages were divided into two groups: dataflow-based languages and data structure-based ones. In other words, in both cases particular ontological assumptions about the nature of social reality were made. In the first case social reality was conceptualised using data store, data flow and data flow transforming process categories, in the second case data structure and structure creating process categories were used. So already comparing the first specification languages the main problem of such comparison can be noticed. The problem is how to compare specification languages, which use different ways of social reality categorisation. In other words, problems emerge comparing languages, which conceptualise modelled social reality in completely different ways.

The problem of social reality conceptualisation is very important. Different evaluation criteria can be used: clarity of conceptualisation (how much intellectual effort is necessary to understand a specification), expressive power of the system of categories (is it possible to specify all necessary aspects of social reality), semantic power of the system of categories (what is the size of specification), selective power of the system of categories (what is the degree of distinguishing details) and others. These criteria can conflict with each other, so everything depends on the priority that is set for them in the particular case. An attempt to evaluate different ways of social reality conceptualisation using various criteria is made in the method of conventional comparison. However, as it was mentioned in 3.4 this method is very subjective.

Subjectivity can be eliminated using some kind of standard or norm. Already in 1977 philosopher Mario Bunge has proposed ontology [Bun77], which can be used as standard or

normative one. In other words, quality of specification language can be evaluated comparing constructs of this language to some collection of “standard” constructs.

The main idea of Bunge’s ontology is the assumption that social reality can be conceptualised using category of “thing”. In other words, in Bunge’s ontology the real world is composed of **things**. The notion of thing is based on the concept of substantion proposed by Aristotle. According to this concept things are material, existing independently of the observer and saving their **identity** while changes are made. Things can be **simple** and **composite**, made up of simple things. Simple things are called components of some composite thing. Every thing has some particular **properties**. Outside or inside events, related to a thing, and laws, changing states of a thing, are called properties of this thing. Generally, properties of things are divided into four groups: intrinsic, mutual, emergent and hereditary properties. Group of intrinsic properties has properties of individual things, group of mutual properties includes properties of two or more things (for example, dependencies and relations between the things), group of emergent properties has properties of composite things, but not of their components, group of hereditary properties has properties of composite things, which they inherit from their components. Sum of the properties of a thing at some moment of time is called its **state**. **Class** is a "set of things that can be defined via their possessing a particular set of properties” [WZ96].

Things are particular objects in Bunge’s ontology. The main property of every thing is its existence. Existing thing is bounded to it surrounding things. So things have not only structure, but also environment. Environment of a thing is made up of things, to which this thing is bounded. Neither things nor systems can be analysed ignoring their environment. Relations of a thing with environment form external structure of this thing. It should be noticed, that relations belong to the group of mutual properties. So, intrinsic and mutual properties form internal and external structure of a thing accordingly. Relations can be bonding or non-bonding. Bonding relations have influence on states of bound things, while non-bonding relations do not have any influence on states. They connect things with frames of reference. For example, time and space relations are non-bonding. Space relations in Bunge’s ontology are treated as mutual properties of a thing. Space is considered to be a combination of things connected by relations. In other words, space-forming things are outlined, and space determines spatial properties of these things. Such concept of space is constructive enough, but is hardly compatible with everyday notion of space as abstract thing limited by some restrictions. In everyday life space category defines where things exist and events take place. According to Bunge every collection of things, for example, constellation of stars or alley of trees can be treated as space.

System is defined as composite thing, which components are connected by bonds of one or different types. System may at the same time be a component thing of a larger system. This

establishes a part-of relation over things. If components are connected not by bonds, but by other relations, then we have not system but aggregate. Ontology defines two classes of problems, connected to systems – analysis and synthesis. Analysis problem is that having a system and knowing its composition, environment and structure, it is necessary to determine behaviour of this system. It should be reminded, that the structure of a system is the combination of all its relations (internal and external), the composition has the things, from which a system is composed, and environment includes all things, to which the analysed thing is bounded. Synthesis problem is inversed: having behaviour of a system it is necessary to determine what kind of system can generate such behaviour. In other words, synthesis is interpreted as development of a system having its requirements specification, and analysis can be treated as evaluation of already developed system. From the notion of synthesis very important conclusion can be made – properties of a thing and a thing itself may be specified separately. Properties of things are defined by requirement specification, and a thing itself is defined by development specification. Things in Bunge's ontology are interpreted as material things having material properties, so separation of things and properties also can have another meaning: while properties of a thing change, the thing itself does not change. For example, requirements specification is transformed into development specification, and the latter into working system, but in all three cases we have the same thing (the same system).

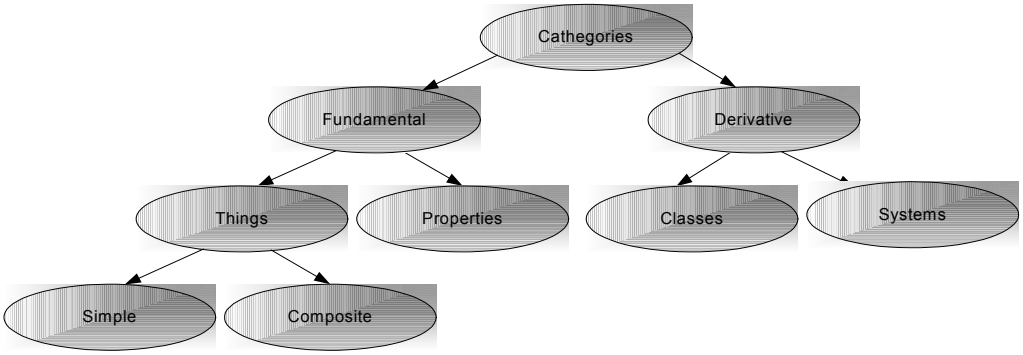


Figure 3. Bunge categories

Things, properties, systems and classes are meta-constructs, "*that is, fundamental elements in modelling systems*" [WSW99]. Things and properties can be considered as fundamental Bunge categories, while classes and systems are derived Bunge categories (Figure 3).

Bunge's ontology is extensive and it covers static structure as well as dynamic behaviour. It belongs to the group of integrated pluralism ontologies [Joh98]. In other words, variety and unity of social reality is announced at the same time. According to Bunge theses, social reality consists of multiple layers. The main features of such system of multiple layers are:

- every object (thing) of social reality belongs to at most one layer;

- system of multiple layers is evolving: new layers, properties and laws emerge, and some old properties are lost;
- all new layers are connected to old ones, and it makes their existence continuous;
- every layer is stable and autonomous, if it is analysed inside its boundaries;
- every event depends on laws, characterising the layer, to which it belongs, and the related layers.

Bunge epistemology can be derived from this system of multiple layers. According to this epistemology layers can represent knowledge about social reality. Every new science, in which social reality is analysed, has its specific objects of research and specific methods of research. However ideas from origin sciences are kept. Deeper understanding of knowledge is impossible without analysis of related layers. Every new science, in which social reality is analysed, is independent and stable at some extent. Every system or event can be described, explained and forecasted in terms of to it related layers, and it isn't necessary to analyse all system of multiple layers. This epistemology should be considered during construction of methodologies of social reality analysis, and methodologies that are used during creation of Information or Software Systems.

Bunge's ontology is widely used in different scientific areas, including informatics. For instance, UML specification language is based on Bunge's ontology. Using Bunge's ontology for Information or Software System, system is treated as the model of social reality. This model is constructed according to some meta-physical theory of social reality, i.e. to Bunge's ontology.

3.6.2. Bunge-Wand-Weber (BWW) models

In works [WSW99], [WW96], [WW89a], [WW89b], [WW90a], [WW90b], [WW91], [WW93], [WW95], [CW98] Bunge's ontology is used to evaluate specification languages (modelling grammars in the terminology of Wand and Weber) and specifications (scripts in the terminology of Wand and Weber). On the basis of Bunge's ontology normative method of comparison, called Bunge, Wand and Weber (BWW) set of models, is constructed.

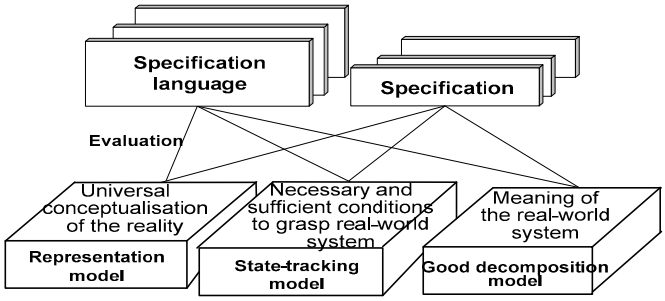


Figure 4. BWW models

Things and properties are the only fundamental meta-constructs of the BWW models that are specific to static structure. Hence, systems and classes are derived from, and only from, things and properties, which are therefore the basic static building blocks of the real world. Consequentially, the BWW models assume that systems and classes are "out there" in the real world. Applied to analysis of domain of discourse, this implies that these systems should be represented as validly and completely as possible, and that a discourse analysis method should ideally support exploration of relations within and between each of the meta-constructs as systematically as possible.

BWW family of models includes three models: representation model, state-tracking model (STM), and good decomposition model (GDM) (Figure 4).

STM and GDM are intended to use to characterise the specifications of Information or Software Systems. STM identifies the necessary and sufficient conditions that a specification must satisfy in order to grasp the real world system (organisation's social reality) it is supposed to describe. The good decomposition model focuses on the problem of communication of the meaning of the real world system to the users. It is supposed that specifications, which possess certain types of attributes, better communicate the meaning of real world [CLV02], [CW98].

Representation model is thought as a kind of conceptualisation that can be used to evaluate ontological completeness [CW98] and ontological clarity of a specification language. The ontological completeness is defined as the ability of the language and associated reasoning system to represent all phenomenon of interest in the domain of discourse. In other words, ontological completeness defines both expressive power and selective power of a specification language. A system is ontologically incomplete if it is not completely specified, so a system having, for example, only an ERA conceptual model would not be ontologically complete [CW98], because using only this language it is impossible to specify all aspects of a social reality in appropriate degree of detail. Ontological clarity is a degree, in which for specification language conceptualisation used categories can be compared to category system of representation model, or Bunge's ontology.

It is supposed that only essential aspects (so called deep structures) of Information or Software System should be specified. Essential aspects of a system are those that represent the meaning of the real world system. Technological and implementation aspects (so called surface structures) of a system are supposed to be not essential, because those aspects can be implemented automatically using appropriate CAiSE tools [CW98]. Consequently, we should specify not Information or Software System itself but describe the social reality [CW98].

To serve as a theoretical basis, on which the specification languages can be evaluated from the point of view of ontological completeness and ontological clarity, BWW representation

model should categorise all possible aspects of the social reality, and should allow checking whether the specification is at least potentially complete (logical or computational completeness). The claim, that the BWW representation model fulfils this requirement, is fundamental. To accept BWW approach one should suppose that there exists a universal conceptualisation of any social reality, which is language neutral and independent of any observer's interest in it [CW98].

BWW models where used to evaluate ontological completeness and ontological clarity of different specification languages [GR99], [GR00a], [GR00b], [RG99], [RG02], [WZ96], [JD01], [WW89a], [WW95], [Web97], [PW97], [WSW99]. The main evaluation criterion is based on assumption that specification language should be appropriate to specify all social reality's things, in which Information or Software System users are interested. In other cases specification language is considered to be not ontologically complete. Ontological completeness of a specification language is evaluated comparing constructions of BWW representation model to constructions of this language. It is required that every construction of BWW model could be compared to at least one construction of the evaluated specification language. If such comparison is not possible then the specification language is considered to be not ontologically complete or incomplete. Ontological clarity of the specification language is violated, if it is determined that:

- different constructions of BWW representation model can be compared to the same construction of evaluated specification language (this is called construct overload);
- one construction of BWW representation model can be compared to different constructions of evaluated specification language (this is called construct redundancy);
- BWW representation model has no constructions that can be compared to some construction of evaluated specification language (this is called construct excess).

So, in BWW models particular specification language is evaluated comparing constructions of BWW representation model to constructions of evaluated specification language. In other words, method of comparison with a norm is used. BWW model is proclaimed to be the norm, and other specification languages are evaluated comparing them to this norm.

BWW models have been applied for numerous purposes, including evaluation and analyses of:

- information Systems design methods [Wan88];
- Dataflow diagrams and ER diagrams [WW89],
- Systems decomposition [PW89],
- NIAM [WZ96], and
- Data quality dimensions [WW96].

3.6.3. Guizzardi Framework

In [Gui05] it is proposed to merge BWW models with in [Gur98], [Gur99] presented framework into one single evaluation framework. In [Gur98], [Gur99] the author presents a framework to formally evaluate the relation between the properties of a specification language representation system and the properties of the domain entities they represent. According to him, representations are more or less effective depending on the level of homomorphism between the algebras used to represent what he terms the representing and the represented world. In [Gur99] four properties are defined, which are required to hold for a homomorphic correlation to be an isomorphism: lucidity, soundness, laconicity, and completeness. In [Gui05] these properties are combined with in [WW89a], [WW95], [Web97], [PW97], [WSW99] proposed evaluation of specification languages by construct overload, construct redundancy, and construct excess. Guizzardi [Gui05] focuses our evaluation on the level of the system of representations by discussing the relation between a particular specification S (representing world in terms of Gurr) and a particular abstraction of a portion of reality, i.e., a particular model M (represented world in terms of Gurr). The original proposal of Wand & Weber is extended in the sense that by considering desirable properties of the mapping of individual diagrams onto what they represent, it is possible to account for desirable properties of the modelling languages used to produce these diagrams. The following properties form in [Gui05] proposed framework:

- **Lucidity and construct overload**. A specification S is called **lucid** with respect to a model M if a (representation) mapping from M to S is injective. A mapping between M and S is injective iff every entity in the specification S represents **at most** one (although perhaps none) entity of the model M.
- **Soundness and construct excess**. A specification S is called **sound** with respect to a model M if a (representation) mapping from M to S is surjective. A representation mapping from M to S is surjective iff the corresponding interpretation mapping from S and M is total, i.e. iff every entity in the specification S represents **at least** one entity of the model M (although perhaps several).
- **Laconicity and construct redundancy**. Specification S is called **laconic** with respect to a M if the interpretation mapping from S to M is injective, i.e. iff every entity in the model M is represented by **at most** one (although perhaps none) entity in the representation S.
- **Completeness**. A specification S is called **complete** with respect to a model M if an interpretation mapping from S to M is surjective. An interpretation mapping from S to M is surjective iff the corresponding representation mapping from M to S is total,

i.e., iff every entity in a model (instance of the domain conceptualisation) is represented by **at least** one (although perhaps many) entity in the representation S.

Let's discuss every property shortly.

The notion of lucidity at the level of individual diagrams is strongly related to the notion of **ontological clarity** at the language level as discussed in [Web97], [WW96], [WW89a]. In [Web97], the author states that the ontological clarity of a modelling grammar is undermined by what they call **construct overload**: "*construct overload occurs when a single grammatical construct can stand for two or more ontological constructs, The grammatical construct is overloaded because it is being used to do more than on job."*

The notions of lucidity and ontological clarity albeit related are not identical. We say that a language (system of representation) is non-lucid according to a conceptualisation if there is a construct of the language, which is nonlucid, i.e., a construct that when used in a specification of a model (instantiation of this conceptualisation) stands for more than one entity of the represented model. A construct can be overloaded in the language level, i.e. it can be used to represent different concepts, but every manifestation of this construct in individual specifications is used to represent only one of the possible concepts [Gui05]. Thus, in ideal case a specification language should not contain construct overload and every instance of a construct of this language should represent only one individual of the represented domain abstraction.

Unsoundness at the level of individual specifications is strongly related to unsoundness at language level, a property that is termed **construct excess** by Weber: *"construct excess occurs when a grammatical construct does not map onto an ontological construct.* [Web97].

Although construct excess can result in the creation of unsound specifications, soundness at the language level does not prohibit the creation of unsound specifications. According to [Web97], users of a specification language must be able to make a clear link between a construct and its interpretation in terms of domain concepts. Otherwise, they will be unable to articulate precisely the meaning of the specifications they generate using the language [Gui05]. Therefore, a specification language should not contain construct excess and every instance of its constructs must represent an individual in the domain.

The notion of laconicity in the level of individual specifications is related to the notion of **construct redundancy** in the language level in [Web97]: *"construct redundancy occurs when more than one grammatical construct can be used to represent the same ontological construct."*

Despite of being related, laconicity and construct excess are two different (even opposite) notions. On one hand, construct redundancy does not entail non-laconicity. For example, a language can have two different constructs to represent the same concept. However, in every situation the construct is used in particular specifications it only represents a single domain

element. On the other hand, the lack of construct redundancy in a language does not prevent the creation of non-laconic specifications in that language [Gui05]. In ideal case a specification language should not contain construct redundancy, and elements in the represented domain should be represented by at most one instance of the language constructs.

The notion of completeness at the level of individual specifications is related to the notion of **ontological expressiveness** and, more specifically, **completeness** at the language level, which is perhaps the most important property that should hold for a representation system. A specification language is said to be complete if every concept in a domain conceptualisation is covered by at least one construct of the language. Language incompleteness entails lack of expressivity, i.e., that there are phenomena in the considered domain (according to a domain conceptualisation) that cannot be represented by the language. An incomplete specification language is bound to produce incomplete specifications unless some existing construct is overloaded [Gui05]. Thus, a specification language should be complete with respect to a domain conceptualisation and every element in a domain abstraction (instance of this domain conceptualisation) must be represented by an element of a specification built using this language.

3.6.4. Strengths and Weaknesses of Bunge Ontology Based Approach

A method of determining whether an Information of Software System development environment is ontologically adequate is proposed in Bunge-Wand-Weber (BWW) models. Taking the view that information is a representation of social reality, the state of a system is considered to be a text, and the dynamics of the system is expressed by the dynamics of a text editor. This view allows making use of a generalised ontology developed by Bunge to get clear picture of the functions of Information or Software System, and therefore a set of criteria for ontological adequacy. The value of these results is that they validate a large body of existing Information and Software Systems. They also validate the basic approach used to construct them, although proposing some improvements [CW98].

BWW models also have a number of weaknesses. First of all, according to the methodology, proposed by Wand and Weber, it is required to accept that BWW ontology conceptualises all aspects of social reality, and that this methodology is constructed in such way, that it should be the most natural for every user, independently from his subjective thinking, interests and native language. Although this requirement is fundamental, in works [WSW99], [WW96], [WW89a], [WW89b], [WW90a], [WW90b], [WW91], [WW93], [WW95], [CW98] provided arguments are not sufficient to ground it. This was noticed and by other authors [GR99], [GR00a], [GR00b]. In their works it is stated, that possibly BWW ontology is overloaded by some unnecessary constructions, because in none of the specification languages that have been analysed corresponding constructions were met. Besides, some specification languages are used for

specification of only some aspects of Information or Software System. In practice they are combined with other specification languages, and the whole Information or Software System is specified using collection of several languages. Collection is constructed according to principle that specification languages that form it should overlap minimally from ontological point of view [WZ96]. Thus, it has no sense to require that all collection forming languages should be ontologically complete. On the other hand, ontology of BWW representation model clearly has no some necessary constructions, for instance, it has no constructions for specifying of business objectives, strategies, goals, or knowledge, which can be necessary for management modelling [GR00a].

Second important weakness of suggested methodology is that quality of a specification language is evaluated from two points of view – ontological completeness and ontological clarity. It is supposed that specifying Information or Software System only ontological aspects of social reality (deep structures) are essential. All other aspects are considered to be not essential, technical (surface structures). However, it should be noted that most frequently Software System implementation progresses though a number of intermediary steps. A step starts with a set of statements representing the system design so far achieved. This set of statements is considered as the specification for the current step. The resultant representation must admit dual interpretation: as a partial implementation of the software system under development and as specification for the next step. So, we have a chain of theories and models, each model being a theory for the next step. It is also important, that each specification (except the final implementation) is incomplete. Incompleteness reflects the fact that each level should provide design alternatives. Resultant representation may differ in quite important aspects and it is impossible to choose the alternative mechanically. The designer enriches each level of representation using his intuition, experience, and knowledge. He defines a number of internal objects that quietly differs from external objects described by domain theory. A software system has immediate access to its internal objects and, consequently, can restructure and modify those objects. In other words, software system can be designed to be self-structuring and self-modifying, however, the correct implementation of such system is highly intellectual task. So, the assumption adopted in BWW approach that surface structures are not essential, because those structures can be implemented automatically is weak-grounded [CW98].

However, an attempt to solve problems of completeness and quality in development of Information or Software Systems is made in BWW. Proposed categorisation of social reality is useful, when it is necessary to outline aspects, by which specification languages should be analysed and evaluated.

3.7. Extension of Bunge Ontology Based Approach

3.7.1. Extension of BWW Models

In [Opd97] it is proposed to extend category system of BWW representation model by two new meta-constructs – perspective and conception (Figure 5):

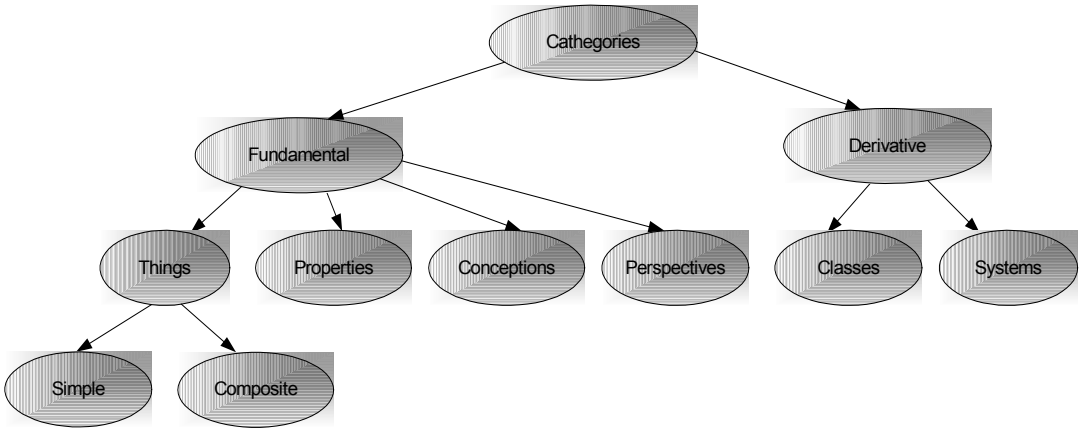


Figure 5. Extension of Bunge categories

Domain of discourse analysis involves numerous stakeholders and groups of stakeholders, including future users, general and information technologies managers, as well as the analytics themselves. Stakeholders or groups of stakeholders have different knowledge backgrounds, values and beliefs, and hence different perspectives on the domain of discourse. It should be noted that our world is shaped by our experience of it. We see different things, have different perspectives, and structure the world differently, depending on interests, background, education and culture. Complicating the picture further, individual stakeholders also take different roles in different contexts and at different times, making even individual perspectives situation-dependent.

Assuming a real world comprising only two fundamental meta-constructs, i.e., things with properties, according to the BWW model, a perspective can only shape the real world in two ways, by limiting

- the set of things in the problem domain that is part of the perspective, and
- the set of properties of these things that is part of the perspective.

Unsurprisingly, a perspective is therefore an excerpt of the problem domain. However, it also has a more profound effect on how that excerpt is conceptualised. As the BWW model defines a class as a set of things that possess a particular set of properties, and a perspective extracts only a subset of properties of each thing, it follows that the same thing may belong to different classes when perceived from different perspectives.

Perspective is an independent from context and moment of time view of the user to analysed social reality. In other words, this is a particular interpretation of social reality. According to Bunge's ontology social reality is constructed from particular properties having things. So perspective can be created either considering not all things of social reality or considering not all properties of these things.

When things are perceived from a perspective, they are therefore not perceived as the things themselves, but as **conceptions** of the things. Conception is a perception of a thing from some perspective, inheriting a subset of properties of that thing [Opd97]. In other words, when some thing is seen from some perspective, then we have not the thing itself but only some conception of it. A conception possesses a subset of the properties of the underlying thing, and is classified according to those properties. Therefore, the perspective does not comprise things and their properties, but a set of conceptions of things and their properties, and maybe also a set of class definitions.

Since different conceptions of the same thing may possess different properties, and mutual properties correspond to dependencies and relations between things, it follows that different conceptions of the same thing correspond to different dependencies and relations between conceptions. Hence, when perceived from different perspectives, the domain of discourse appears as different systems of conceptions and properties. From different perspectives seen social reality has different business aspects. In [Opd97] made propositions are based on assumptions of radical constructivism philosophy. These assumptions contain thoughts that social reality partially is mental construction of the observer. In other words, it is difficult to determine at what extent the notion of social reality that is at the disposal of an analytic is objective. That is why specifications of Information or Software Systems should be extended by descriptions of perspectives and conceptions of things.

So, four meta-constructs - things, properties, conceptions and perspectives - are essential to the understanding of multiple perspectives on a domain of discourse. While both perspectives and things are fundamental according to this line of reasoning, conceptions emerged when things with properties were perceived from a perspective. Hence, the corresponding meta-construct is derived. Nevertheless, these two meta-constructs should be considered equally important.

In [Opd97] it is proposed to use by new constructs extended Bunge's ontology to evaluate not only specification languages, but also methods of domain of discourse analysis methods. During analysis of social reality it is intended to understand it better by considering different perspectives, thus domain of discourse analysis methods as well as specification languages can be evaluated using category matrix of social reality conceptualisation (Table 2):

Table 2. Category matrix

	Things	**Properties**	**Conceptions**	**Perspectives**
Things				
Properties				
Conceptions				
Perspectives				

In Opdahl's [Opd97] opinion, quality of a specification language can be evaluated answering the following questions:

- What semantic quality categories can be expressed for a language?
- How easily can it be done?

Terms of conceptualisation categories are used to answer these questions. Thus, it is convenient to form questions in the cells of the category matrix. Each cell represents an analysis opportunity, which a particular method (or language) may or may not provide. According to [Opd97], each cell raises the question:

"*Whether the method given a particular element indicated by the cell's row identified by the user explicitly and systematically explores relevant semantic quality features of that element in relation to all the relevant identified and/or identifiable elements indicated by the cell's column* ".

For example:

1. For the "property" - column of the "thing" - row, this question is reduced to

"Whether the method (language), given a particular thing identified by the user, explicitly and systematically explores relevant semantic quality features of that thing in relation to all the relevant identified and/or identifiable properties [Opd97]".

2. For the "thing" - column of the "thing" - row, this question is reduced to

"Whether the method (language), given a particular thing identified by the user, explicitly and systematically explore all other relevant things in its context [Opd97]".

Such questionnaire should be prepared for every quality characteristic of in [Opd97] evaluated domain of discourse analysis method. Set of quality characteristics and its importance depends on the particular project, so both the set of questionnaires and in questionnaires presented questions also depend on the particular project. In [LSS94] it is stated that all quality characteristics can be generalised to two characteristics - completeness and validity, and quality can be analysed using only these characteristics. Thus, in [Opd97] quality is analysed using only these two characteristics. The main attention in [Opd97] is paid to domain of discourse analysis methods, leaving specification languages for future research. Four families of domain of discourse analysis methods are compared: structured, object-oriented, faceted, and viewpoints-based analysis. The intention behind such a comparison is to point out how and why existing

methods and method families differ in how they support multiple perspectives. The underlying assumption, which makes this effort worthwhile is that existing method families tend to focus analysis on distinct, but complementary, elements of multi-perspective problem domains, and that the quality of discourse analysis in many cases can be improved by integrating them.

3.7.2. Strengths and Weaknesses of the Extension of Bunge Ontology Based Approach

Proposed framework extends the basic static structure of the BWW model with two meta-constructs - conceptions and perspectives, which makes it applicable to the evaluation of languages from different perspectives. Thus, the step towards evaluation of a specification language according to project context is made. Besides, an attempt to propose qualitative scale is made, however the scale is very imprecise, thus further research is needed.

Practical usage of the methodology has shown that it has several weaknesses. One of them is so called transitive problem, connected with interrelations between in the questionnaires proposed questions. For instance, while creating questionnaires for evaluation of specification languages completeness it has become clear that some relations in the matrix (Table 2) implicate the other relations. For example, if it is possible to specify explicitly relations between thing and its conceptions and relations between conceptions and their properties, then it is possible to specify explicitly relations between thing and its properties. In other words, the questionnaire is not minimal, and the scope of the analysis can be broadened without any ground. Besides, it is still not clear how to evaluate quality characteristics of a specification language using only questionnaire.

The other problem is that really the same quality characteristics and their importance can be treated differently at different moments of time. For instance, specification consistency with no doubt is a good feature, but very often at the beginning of the project it can be even necessary to work with contradictory specification leaving elimination of contradictions for future. This is not considered in the proposed methodology.

The third problem is that although in [Opd97] proposed the methodology rejects the view of method (language) evaluation using only two characteristics - ontological completeness and ontological clarity, it still uses only one way of quality characteristics evaluation - ontological analysis.

The fourth problem is that although the proposed framework extends the basic static structure of the BWW model with conceptions and perspectives, which makes it applicable also to the evaluation of multi-perspective methods, in comparison to the BWW model, it has less developed and less firmly grounded foundation. Strengthening the foundation of the present work can be done by further underpinning and clarifying the present platform, or, preferably, by

aligning the present platform with an established and widely-accepted ontology, which takes into account multiple perspectives.

Finally, the main problem of the methodology is that as BWW models it is also normative. In other words, it is supposed that the best way of social reality conceptualisation exists, and all other ways should be compared to it.

However, in spite of all the weaknesses the proposed methodology it proposes to evaluate a specification language (or domain of discourse method) taking into account the context of the particular project. During such evaluation strengths and weaknesses of each specification language can be noticed, and its appropriateness for the particular project can be determined.

3.8. Chisholm Ontology Based Approach

3.8.1. Chisholm's Ontology

In [MKK98] it is proposed to evaluate specification languages using not Bunge, but Chisholm's ontology [Ch92], [Ch96]. According to Chisholm's ontology social reality is constructed from entities. In this case the meaning of the term "entity" is not the same as it has in informatics, for example, in data modelling. In Chisholm's ontology, **entities** describe all things, which make up the furniture of the world, with individuals representing significant substances within that world. Chisholm divides the world of entities into **contingent**, which don't have to exist, and **necessary** entities, which always exist. The taxonomy of Chisholm categories of social reality is shown in Figure 6.

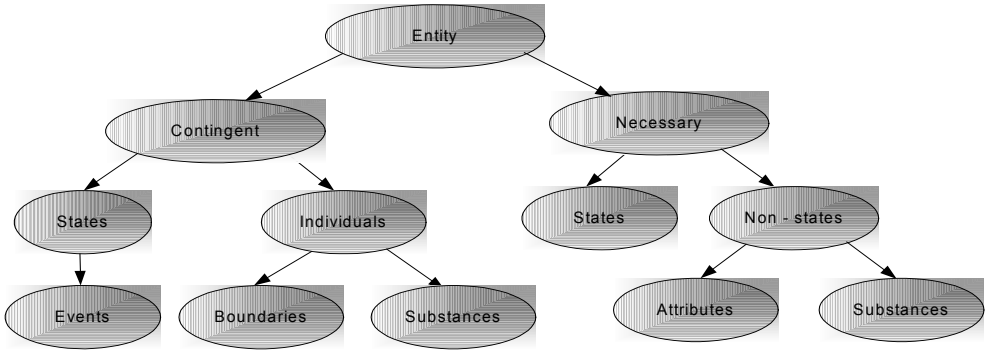


Figure 6. Chisholm categories

Contingent entities consist of **individuals**, and their **states**. Individuals can either be **substances**, or **boundaries**. Boundaries are spatial elements, which define the bounds of individual substances [MKK98]. Substances have much in common with entities (in the meaning they are used in informatics). Substances can encompass physical things as well as legendary or conceptual things. Necessary entities has no individuals, they encompass **attributes**, **necessary substances** and their **states**. Attributes and necessary substances are examples of enduring thing and do not pass away or come into being.

Entities of both types have states, bet only for contingent entities **events** are described as being a subcategory of states. Events are used for modelling dynamics of the system. Attributes of necessary entities and substances, which they represent, cannot be neither created nor deleted. Attributes are associated with entities by observers (analytics), according to their intentions and intuition. If an observer believes that an individual has a particular attribute, and that belief is shared, then that individual is said to *exemplify* that attribute. This is called the *primacy of intention*, because intent, or belief, is the fundamental notion in the ontology. Intentions and intuition are main Chisholm's ontology categories. In other words, it is not necessary that an entity has an attribute, but rather that someone believes that an entity has an attribute (an idea of subjectivism). Some attributes are exemplified, others are not, and some cannot be exemplified. We can represent both facts and propositions using attributes. Attributes may be simple or compound. Compound attributes are formed through disjunctions and conjunctions of compound or simple attributes. A conjunction of attributes conceptually entails, and exemplifies, each of the attributes in the conjunction simultaneously. A disjunction of attributes exemplifies one attribute or the other attribute but not both simultaneously [MKK98].

Relations exist between individuals. Relations are directed from one individual to another, which means that they have a concrete direction. Relations are represented by an attribute, which is an ordered pair of attributes (the attributes can be compound). Indeed, the attribute representing the relation is said to order things. An attribute relating one thing to another consists of an attribute, which uniquely describes the first individual, and an attribute, which uniquely describes the second individual. Moreover, one can have dyadic (two term), triadic (three term), tetradic (four term), and in general n-ary (n term) relations, but all of these relations can be represented as a series of dyadic relations [MKK98].

Classes and **sets** are also represented in Chisholm's ontology using attributes. There is a set of axioms, which establish the fundamental properties of sets and classes in Chisholm's ontology.

Chisholm's ontology also considers theology, perceptions, appearances, stories, times, and events [MKK98].

3.8.2. S. Milton, E. Kazmierczak and C. Keen Methodology

In [MK99], [MKK98] it is proposed to use Chisholm's ontology instead of Bunge's ontology, because, in authors' opinion, it has the potential to be a unifying framework for data models. Specification languages can be considered as modelling constructions of social reality. Two different analytics for the same fragment of social reality can create different specifications. One of them may use one tool, the other - another tool. Thus, in authors' opinion languages should be evaluated not from the viewpoint of its constructions, but from the point of its ability

to specify a variety of situations of social reality. For this aim it is proposed to use method of agreement and method of difference (see section 3.4). Using these methods, social cases are analysed to identify similarities and differences between them, and to establish causal links that exist in qualitative data. Further, the inferred causations may be extended to form limited generalisations. For this aim a set of qualitative features and a set of characteristics should be defined. Qualitative features are the elements of the situations, which the analytic believes to be important in identification of differences between cases. Characteristics contain the set of properties, which qualitative features may have in a particular situation. By determining agreements the analytic intends to identify common characteristics of the particular outcome and decide what qualitative features could cause that outcome. By determining differences the analytic intends to check if at different outcomes for the same quality feature different characteristics are observed. By application of the method of agreement and the method of difference it is analysed what commonalities and what differences are observed for a specification language comparing it to Chisholm's ontology. Qualitative agreement/difference scale of five levels is used to determine whether the specification language has a particular qualitative feature [MKK98]. Qualitative features are summarised in Table 3:

Table 3. Qualitative features

Qualitative feature	Qualitative summary of emphasis
Entity Support (ES)	Model supports significant perceivable or observable entities or things, which may be described in part through the perception of attributes or properties.
Attribute Support (AS)	Attributes must be modelled with one overriding requirement – attributes are displayed by entities (and perceived by observers), but must not be tightly coupled with that entity.
Attribute Construction (AC)	Attributes must be able to be formed through the combination of other attributes. These combinations may be conjunctive or disjunctive.
Class/Set Support (CS)	The membership of classes or sets is based on attributes. Importantly, classes can be described through attributes. This means that entities can be members of several classes simultaneously and can be moved freely between classes.
Relationship Support (RS)	Relations must be attributes (or provided in the same way as attributes). Additionally, they must be directional.

The first three in Table 3 presented features form fundamental (or core) features. All others are resultant (or consequential). The process of evaluation includes not only determining whether the language has some qualitative feature, but also at what extent this feature is supported. The evaluation of feature support is made using the following evaluation levels:

√ - Support for the feature.

X - No support for the feature.

* - Support given, but attributes are tightly coupled to entities.

+ - Support not given, but could be provided relatively easily.

∧ - Loose coupling between features is provided, but generally is ignored.

Evaluation levels are assigned to each pair "Qualitative feature – Language" (Table 4):

Table 4. Results of the evaluation

Qualitative feature	Language 1	Language 2	…	Language N
ES				
AS				
AC				
CS				
RS				

3.8.3. Strengths and Weaknesses of Chisholm Ontology Based Approach

Comparing to BWW models Milton, E. Kazmierczak and C. Keen is more suitable for specification of specific requirements of Information or Software Systems, because it has qualitative scale for evaluation of quality characteristics. However, this methodology is also normative, that is, it intends to construct "the best" way of social reality conceptualisation. Thus, it has the same weaknesses as BWW models (see section 3.6.2).

3.9. Ontological Categories Based Approach

3.9.1. Mylopolous Methodology

An attempt to eliminate weaknesses of normative methodologies is made in works [Myl98], [Gua98], where normative ontology is not used. Instead, it is proposed to compare analysed phenomena (for example, information modelling techniques, specification languages) by their ontological categories. In [Myl98] ontological analysis is done using complex ontological system that comprises static, dynamic, intentional and social ontologies. In general, it is proposed to evaluate quality of a specification language in three orthogonal dimensions: ontologies, abstraction mechanisms, and tools (Figure 7):

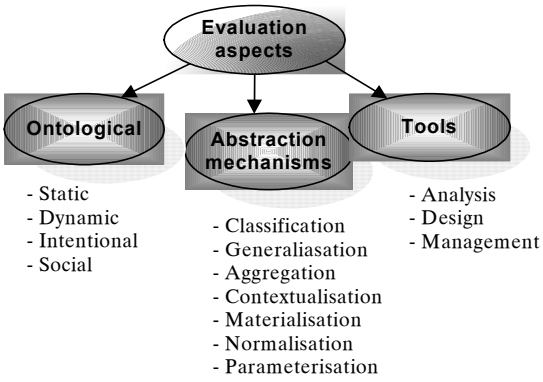


Figure 7. Evaluation aspects

Ontological aspect is described by several ontologies, which define the specified social reality. It is static ontology, dynamic ontology, intentional ontology, and social ontology.

Static ontology encompasses static aspects of an application, by describing the existing things, their attributes and interrelationships. Most conceptual models assume that the world is populated by entities, which are endowed with a unique and immutable identity, a lifetime, a set of attributes, and relationships to other entities [Myl98]. However, this way of social reality conceptualisation is neither universal nor minimal or the best in some other aspect. It can be used to conceptualise social reality in order to solve problems of a particular kind. Another kind of problems can be solved more successfully using another conceptualisation of social reality.

Dynamic ontology encompasses dynamic aspects of an application in terms of states, state transition and processes [Myl98]. However, this is also not the only way of changes conceptualisation. For instance, dynamics of social reality can be conceptualised using process, process step, goal, and agent categories.

Intentional ontology encompasses the world of agents that is things agents believe in, want, prove or disprove, and argue about [Myl98]. Different category systems can be used, for example, the system, made of the following categories: agent, issue, goal, subgoal of, supports, denies, and etc. It should be noticed, that only the newest specification languages operate by such kind of categories, while in artificial intelligence these categories are used already for several decades.

Social ontology covers social settings, permanent organisational structures or shifting networks of alliances and inter-dependencies. Traditionally, this ontology has been characterised in terms of categories such as actor, position, role, authority, commitment, etc. [Myl98]. These categories are used to determine roles of actors and strategic dependencies between the actors. Such dependencies can occur, for example, when one actor passes fulfilment of a particular task to the other actor.

Specification languages differ in what aspects of social reality (static, dynamic, intentional, social) can be specified using a particular language.

Ontologies describe specified social reality, while abstraction mechanisms of a specification language determine allowed ways of information organisation. The most important abstraction mechanisms are classification, generalisation, aggregation, contextualisation, materialisation, normalisation and parameterisation [Myl98]. Specification languages differ by supported abstraction mechanisms.

Tools are used to increase the productivity of Information or Software System specifier. They help to construct, analyse and evaluate specifications. There are three basic classes of tools:

analysis, design and management tools. For different specification languages different tools can be used or constructed.

Mylopolous proposes to evaluate specification languages depending on the ontologies that they are based on, abstraction mechanisms that they support, and tools that can be used or constructed for them. Using by the author proposed classification schemes all three aspects can be evaluated by particular scale. It is proposed to use the following qualitative scale: {excellent, good, OK, so-so, none}. The overall evaluation of a specification language is the combination of the evaluations it gets with respect to each dimension. Likewise, the overall evaluation for each dimension is the combination of the partial evaluations for each component of the dimension [Myl98]

When specification languages are evaluated the specification language, which is the most sufficient for any possible project is selected according to the following evaluation criteria:

- the type of Information or Software System (what kind of ontologies will become necessary);
- the degree of complexity of Information or Software System (what abstraction mechanisms should be supported);
- the scope of Information or Software System (what tools are the most important);
- the character of fulfilled tasks (what abstraction mechanisms and tools should be used).

3.9.2. Strengths and Weaknesses of Ontological Categories Based Approach

Mylopolous methodology also has some serious weaknesses. First of all, it is not clear whether proposed types of ontologies are really sufficient to conceptualise all possible aspects of social reality. For example, in [Gua98] ontology system is constructed in completely different way. Second, all category systems of a particular ontology are supposed to be equivalent, but it is not true. One of them is suitable for one purpose, others – for another purpose. Third, evaluation is done subjectively without any exact measurements, and the proposed scale is not strict, qualitative.

However, this methodology is a step forward comparing to normative comparison. Instead of comparison to a norm it is proposed to use evaluation criteria. Evaluation by the defined criteria is done using the proposed metric, which has a qualitative scale. Such evaluation is more objective than comparison to predefined norm or standard.

3.10. Quality Model Based Approach

3.10.1. Semiotic Framework

One of quality model-based methodologies is semiotic framework [Kr01a], [Kr01b], [Kr03], [KS03], [LSS94]. Semiotic framework is the first serious attempt to develop quality model for

specification languages. This model have been proposed originally by Sindre [Sin90], further extended by Seltveit [Sel94], and completed in works [Kr01a], [Kr01b], [Kr03], [Ks03], [LSS94]. Semiotic framework addresses quality of specification as well as the quality of specifying process [CLV02]. This framework includes a language quality model represented in a form of quality characteristics tree. The framework provides a systematic structure for evaluation of specification languages.

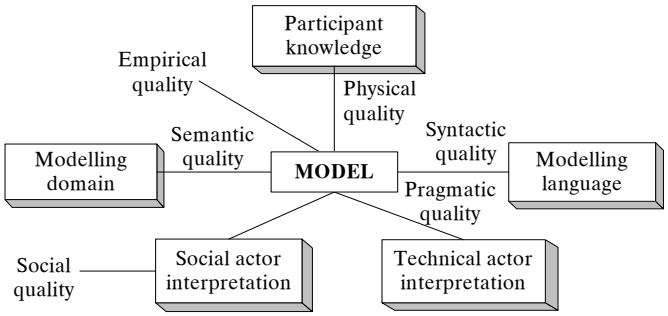


Figure 8. Semiotic framework scheme

Semiotic framework introduces the concepts of a language, a domain, and a model used to explain the meaning of quality attributes. The language is defined as a set of all syntactically correct statements, the domain as a set of all statements meaningful for the given domain of discourse, and the model as a set of statements included in the actual specification [CLV02]. Notion of language quality is based on quality of specifications, which can be written using that language. Specification language is of high quality if it allows writing high quality specifications. The semiotic approach (Figure 8) requires that a specification language should be appropriate to ensure physical, empirical, syntactic, semantic, pragmatic, and social quality of specifications written in this language [Kr01a], [Kr01b], [Kr03], [KS03].

Physical quality is associated with the form, in which a specification is represented (on disc, paper, etc.). The main feature that describes physical quality is the ability to store specifications in such terms that, on one hand, they should be saved from losses, and, on the other hand, they should be accessible for all interested persons.

Empirical quality determines the probability of errors in specifications and ergonomic features of specifications and of potentially available tools that are used to construct, analyse and evaluate specifications.

Syntactic quality can be seen as consistency of the model of social reality with the extension of a specification language, which tools are used to create specifications. A specification language is extended by supplementing it with the vocabulary of the particular fragment of social reality. A language should be constructed in such a way, that syntax analyser could recognise all syntactically invalid sentences, including those, which use language extending constructions.

Semantic quality can be seen as consistency of the model of social reality with specified fragment of that social reality. Model should be complete and valid. In other words, a specification language should allow creating complete and valid specifications.

Pragmatic quality can be seen as consistency of the model of social reality with mentality of that model analysing persons. That means, that a specification language should allow conceptualisation of social reality so as it will be conceptualised by the future users of Software or Information System that is specified.

Social quality is seen as the ability to achieve consistency of different points of view.

Quality model of a specification language has a hierarchical structure. The hierarchy is constructed from two levels (Figure 9).

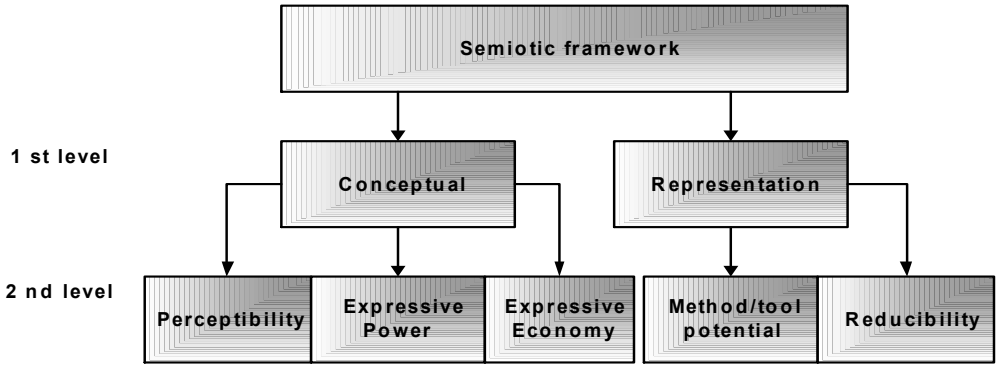


Figure 9. Semiotic framework

The top level of the language quality model distinguishes two groups of quality attributes: conceptual, describing the underlying conceptual basis of the language (i.e. constructs of a language), and representation attributes, describing the external representation of the language (i.e. visualisation of its constructs) [CLV02].

Further, each group is divided into five groups of second level attributes:

- **Perceptibility** is related to the audience (analytics, designers, etc) and describes how easy is for persons to understand the language (both constructs and their representation).
- **Expressive power** is related to the domain and describes what part of domain statements is expressible in the language.
- **Expressive economy** also is related to the audience and describes how effectively statements are expressible in the language both at the conceptual level and at the syntactical level (semantic power) [KS03].
- **Method/tool potential** characterises ability of specification language supporting tools creation.
- **Reducibility** describes the appropriateness of the language to deal with large and complex specifications [Sel94]

So, second level attributes characterise the language from three different points of view: domain, audience, and technology. Using proposed quality model one could evaluate domain appropriateness of the language (expressive power), audience appropriateness of the language (through perceptibility, expressive economy, and reducibility), technological appropriateness of the language (through method/tool potential) [CLV02].

The concept of expressive power in the semiotic approach is closely related to the concept of ontological completeness in the BWW approach. However, they are slightly different concepts, because it is required [KS03] that the measure of expressive power could be used not only to evaluate is any statement of the domain expressible in the language, but also to evaluate is it possible to express in the language any statement, which is not in the domain. It should be noted also that sometimes the term "expressive power" is used as synonymous to the term "expressive adequacy". According to Woods [Wo83], expressive adequacy "*has to do with the expressive power of the representation – that is, what it can say.*" Expressive adequacy has two aspects. The first characterises the ability of the language to distinguish details (selective power), and the second characterises the ability of the language to hide details (generalitive power) [CLV02]. The concept of audience appropriateness is closely related to ontological clarity. Specification is understandable for the user if the following conditions are held:

- conceptualisation of the language corresponds to the conceptualisation of the social reality accepted by the audience;
- the external representation of the language corresponds to the domain metaphor accepted by the audience;
- the language takes into account the psycho-physical characteristics of the audience: reasonable number of constructs, possibility to hide detail, uniformity, separation of concerns, etc. [CLV02].

Thus, the concept of audience appropriateness is wider than the concept of ontological clarity.

Technological appropriateness is thought as the appropriateness of the language to interpret it by software. It is supposed [Sin90] that the technological appropriateness requires that the language lend itself to automatic reasoning and that the reasoning must be fairly efficient to be of practical use. Technological appropriateness is closely related to the property of notational efficacy [Wo83]. Notational efficacy concerns the structure of a language and its external representation as well as the impact this structure has on the software that manipulates specification text. According to Woods [Wo83], "*notational efficacy can be subdivided into issues of computational efficiency and conceptual efficiency*". Conceptual efficiency supports

knowledge acquisition and computational efficiency supports reasoning and other computational algorithms [CLV02].

3.10.2. Strengths and Weaknesses of Quality Model Based Approach

Semiotic approach provides a systematic structure for evaluating specification languages. Its methodological basis includes:

- the idea to separate conceptual and representation issues of the language;
- the idea to compare and evaluate the specification languages on the basis of quality model;
- the idea to evaluate quality from domain, audience and technology points of view;
- the idea to use set-theoretical approach to explain the meaning of the quality attributes [CLV02].

The main strength of this methodology is that comparing to all other methodologies, which were analysed earlier in the dissertation, this methodology is very close to the goal of creation of objective specification languages evaluation methodology, because the framework of specification languages evaluation and comparison is proposed.

Although this approach is very advanced, it has some serious weaknesses. The proposed quality model is only sketched. It is not exhaustive and not homogeneous. It seems that the semiotic framework has been focused on the issues of specification quality and investigated the language quality model only occasionally [CLV02]. It is not clear how to measure and evaluate quality characteristics, and proposals of different authors to evaluate specification languages using appropriate ontologies are ignored. Finally, it is not clear how to evaluate specification language quality according to requirement of the particular project.

3.11. Comparative Analysis of Approaches

The results of comparative analysis are presented in Table 5:

Table 5. The results of comparative analysis

Approach	**Method (s)**	**Components**					
		Goals	**Criteria**	**Quality attributes**	**Metric**	**Scale**	**Framework**
Bunge ontology based approach	Method of normative comparison	*Not available*	*Not available*	*Not available*	*Not available*	*Not available*	*Partially available*
Extension of Bunge ontology based approach		*Partially available*	*Not available*	*Partially available*	*Not available*	*Available*	*Partially available*
Chisholm ontology based approach	Method of normative comparison	*Not available*	*Not available*	*Partially available*	*Not available*	*Available*	*Partially available*
	Method of agreement and method of difference						
Ontological categories based approach	Method of ontological	*Partially available*	*Available*	*Partially available*	*Not available*	*Available*	*Partially available*

Approach	Method (s)	Components					
		Goals	**Criteria**	**Quality attributes**	**Metric**	**Scale**	**Framework**
	categories based comparison						
Quality model based approach	Method of quality model based comparison	*Not available*	*Available*	*Partially available*	*Partially available*	*Partially available*	*Available*

It can be seen from Table 5 that an attempt to evaluate quality according to the particular project context is made in the extension of Bunge ontology based approach and in ontological categories based approach. The first approach proposes evaluation of languages from different perspectives, while the second approach includes intentional ontology that allows expressing of agents beliefs and goals. More or less objective quality evaluation criteria are proposed only in ontological categories based approach and in quality model based approach. The other approaches propose subjective evaluation by comparison to some standard or norm. An attempt to provide some kind of quality attributes of a specification language is made in all approaches (except for Bunge ontology based approach). However these attributes (also called characteristics or features) are neither well-defined, nor structured or organised in some way. Only quality model based approach has quality attributes that form two-level quality model. None of the approaches has quality metric that could be used to measure quality attributes, except for quality model based approach, which proposes set-theoretical approach to explain the meaning of the quality attributes. Some kind of qualitative scale is available in all approaches, however the scale is very imprecise, thus further research is needed. And, finally, evaluation framework is partially available in all compared approaches, however the most complete and theoretically-ground framework is proposed in quality model-based approach only.

3.12. Conclusions

First of all, all in scientific literature proposed approaches to evaluate quality of specification languages are incomplete, because they propose neither theoretical ground for separation and evaluation of quality characteristics, nor precise metrics for their measurement or scale for the interpretation and aggregation of the results of measurements. At the moment, no commonly accepted agreement exists about the collection of specification language internal quality characteristics, their names and their taxonomy. It is true for programming languages, too. An attempt to develop the taxonomy of quality characteristics and their measurement procedures is made in quality model based approach. However, this taxonomy is not enough systematic and exhaustive, and the proposed quality model itself is only sketched. Even the proposed quality characteristics are not precisely defined, and it is not proposed how exactly they should be

measured. Besides, language characteristics and supporting tools characteristics are considered together.

Another conclusion that follows from the results of comparative analysis is that none of known approaches pays attention to relativism of quality concept and dependency on the particular project requirements and peculiarities. No approach exists, in which quality in use is separated from internal quality, and which proposes exhaustive procedure to evaluate quality in use. Thus, we conclude that none of known approaches is mature enough theoretically and is not developed enough to be used in practice to evaluate quality of a specification language.

One more and the most important conclusion is that the framework to evaluate specification languages quality should be based on the appropriate quality model, because it ensures obtaining of objective evaluation results. This requires having an exhaustive taxonomy. In known approaches to evaluate quality of specification languages only first small steps are made in this direction. The most valuable results that can be used for the further research have been obtained in ontological categories based approach and quality model-based approach.

4. An Approach to Evaluate Quality of Specification Languages

4.1. Contribution to the Research

The main results of this section are published in papers [CG05a], [CG05b], [CG05c], [CG04], [Gas06b], [Gas04], [GC06], [GC05a], [GC05b]. The dissertation is the result of the join research, which has been carried out together with my advisor. Each idea and even each sentence in our common publications is the result of long and intensive discussions. So, it is not simple do define individual contribution of each of us. Nevertheless, it is possible to say who has originated and has elaborated the proposed ideas. A sketch of the proposed taxonomy of quality characteristics has been proposed by A. Čaplinskas and co-authors in [CLV02], and elaborated further in this dissertation (see sections 4.2.1, 4.2.2). The main idea how to evaluate elementary characteristics of the functionality has been proposed by me (see section 4.2.3) and, after long discussions and a number of suggestions made by my advisor have been elaborated by me. The main approach how to aggregate quality characteristics has been proposed by my advisor and elaborated in details by me (see section 4.2.4). The idea to evaluate quality in use on the basis of internal quality has been suggested by my advisor but most of concrete techniques including the idea to use fuzzy theory has been proposed and elaborated by me (see sections 4.3.2, 4.3.4, 4.3.5). The details of procedure for quality model construction (see section 4.3.3) also have been proposed by me.

4.2. Specification Languages Internal Quality Evaluation Framework

4.2.1. Taxonomy of Quality Characteristics

4.2.1.1. Main Groups of Quality Characteristics

A number of publications on the problems of specification languages quality evaluation [Kr01a], [Kr01b], [Kr03], [KS03], [LSS94], [Opd97], [Sel94], [Sin90], [WW89a], [WW93], [WW95] mention different quality characteristics that specification language should have. Some quality characteristics discussed in publications on the quality of programming languages and cognitive dimensions of information artefacts (see, for example, [PM96] also are closely related to the quality of specification languages. However, exhaustive set of quality characteristics up to date still has not been proposed. Extensive library research [CLV02] has brought to light that some important quality characteristics even have no names and some important groups of quality characteristics are not thought as coherent groups. Also in the field of programming languages, where research has already a long history, different researchers evaluate quality of programming

languages using slightly different quality characteristics and classify these characteristics in different ways. For example, some researchers argue that reliability of a language is influenced by both readability and writability, others – that it is influenced only by readability. In addition, characteristics of internal quality and characteristics of quality in use often are not entirely separated. On the basis of the results of conceptual analysis of wide spectrum of specification languages, including UML, Z, VDL, TROLL, and Alloy, we decided to organise quality characteristics of specification language in a way proposed by ISO/IEC 9126 standard [ISO91]. Although this standard addresses quality of software, its conceptual basis is significantly wider and can be applied to evaluate quality of languages, too [KM98], [QJ03]. Thus, we define internal quality through four lower-level sub-characteristics: functionality, reliability, usability, and efficiency (Figure 10). Further, we treat the concepts of functionality, reliability, usability, and efficiency in a different way than ISO/IEC 9126 standard and, consequently, refine these characteristics using other lower-level sub-characteristics. We do not include maintainability and portability, because these sub-characteristics are relevant rather to software tools supporting production of specifications than to specification language itself. The proposed taxonomy of quality characteristics and its graphical representation are presented accordingly in "APPENDIX 1. Taxonomy of Quality Characteristics" and "APPENDIX 2. Graphical Representation of the Taxonomy of Quality Characteristics".

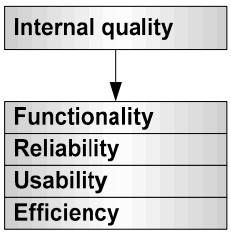


Figure 10. Sub-characteristics of internal quality

It should be noted that although the descriptive sub-characteristics of usability - ontology and paradigm (see "APPENDIX 1. Taxonomy of Quality Characteristics") – describe important properties of a language, they are not related directly to the internal quality of this language. However, evaluation of the quality in use often requires considering formalism beyond specification language and assessing of ontological aspects of this language. Besides, these characteristics sometimes may be operationalised, for example, a metric should be developed to measure the degree of procedureness of the language.

4.2.1.2. Linguistic System

It should be noted that sub-characteristics of internal quality can be considered from conceptual as well as from representational points of view [CM75], [Kle52], because any language has two most important aspects: its semantic and its syntax. Following the idea,

proposed by Krogstie and Sølvberg [KS03] and elaborated in [CLV02], we describe quality characteristics in terms of the linguistic system, defining a formal structure beyond the language, and in terms of the representation system, defining forms of representation of its constructs. Although we share Felleisen's viewpoint [Fel90] that analogy between formal system and language could be very useful, we argue, however, that for specification languages more useful is analogy with first-order languages.

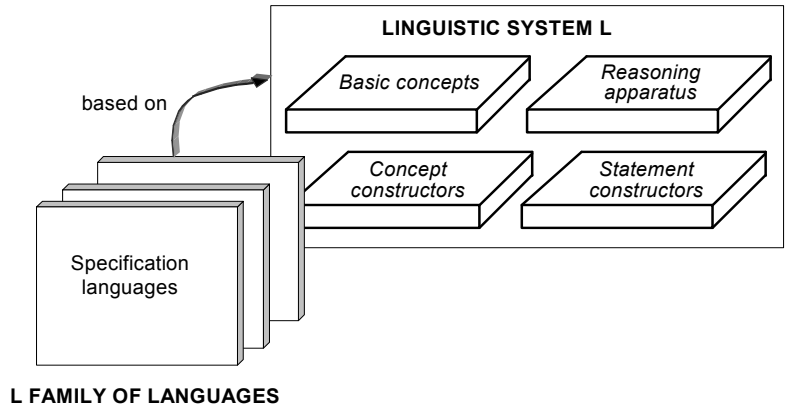


Figure 11. Linguistic system [CLV02]

We define the notion of the linguistic system defining a formal structure beyond the language as four-tuple (see Figure 11):

$$\Phi = <\alpha, \Sigma, \Xi, \Omega> \quad (1),$$

where

α is a nonempty set of basic concepts (primitive concepts),

Σ is a set of constructors used to construct composite concepts,

Ξ is a nonempty set of constructors used to construct statements,

Ω is a reasoning apparatus.

In this definition a concept is considered as some abstract meaning that can be represented in many different ways using syntactically different notations. It is also supposed that the notion of constructor comprises also the rules of correct application of constructors. So the set of composite concepts corresponds to the set of the terms of first-order language and the set of statements corresponds to the set of well-defined formulae.

It is supposed further that the set α is defined by the chosen ontology (more exactly, by the chosen conceptualisation) and provides a set of ontological primitives. So it answers the question: In what terms does a language describe the systems? The set Σ defines a kind of "algebra of concepts" [Nil00] on α. It provides the apparatus to define domain-oriented conceptual primitives and operators, including so-called epistemological (abstraction/structural) primitives, which allows constructing complex concepts from primitive ones. In some cases (e.g.

for domain-oriented specification languages) domain-oriented primitives can be built-in into linguistic system itself. It is supposed that in such cases domain-oriented primitives have the status of ontological primitives and are considered as elements of the set α. The set Ξ defines apparatus to express assertions (more exactly, axioms) about the properties of the system in question.

It should be noted that specification languages can be seen as modelling languages that are used to build an abstract model (a theory) of system in question or, in other words, to define a set of assertions about the required properties of an existing or a future system. Linguistic system describes modelling facilities and, like description logics, distinguishes two modelling levels, conceptual and assertional, although language itself cannot distinguish those levels explicitly. Employing conceptual level apparatus one can define conceptual primitives and create complex concepts. Assertional level apparatus allows formulating statements about the properties of instances of concepts.

It should also be noted that specification languages are used to describe systems as "black boxes" and to define their external properties; design and programming languages deal with the "transparent boxes" and are used to describe implementation details. The purpose of design language is to refine operational specification, and the purpose of programming language is to refine design specification. Therefore it is desirable that the transition from specification to design and further to implementation would be seamless. On the other hand, the development of an Information System starts usually with business modelling and is completed by the implementation of a number of applications (or Software Systems). Architecture of a system should be aligned with business goals and mission, and architectures of applications should be aligned with goals and mission of the system. Consequently, it is highly desirable that the transition from business level to information processing level and further to application requirements specifications would be seamless, too. Thus, in general case, it is desirable that a specification language would be suitable also to specify business systems as well as applications and would be applicable at all stages of development. From this we conclude that functionality should characterise the degree of potential appropriateness of a language to specify properties of any kind of system (i.e. business system, Information System, or Software System) and at any stage of development. Thus, further in the dissertation, we use the term system for business system, Information System, or Software System that is specified using the particular specification language. The higher is the value of functionality as an aggregate quality characteristic of a specification language the less is the expected frequency with which this language will not be applied to specify some system in question.

4.2.2. Functionality and Its Sub-Characteristics

4.2.2.1. The concept of Functionality

Functionality (from Latin *functio* meaning "to perform") means the degree to which the designed product will perform to meet its intended purpose. In the case of specification languages, functionality is understood as the set of features necessary to describe requirements of a future system. Despite the fact that functionality is one of the most important characteristics of internal quality of a specification language, only separate aspects of functionality, mostly expressiveness and ontological completeness, have been discussed in scientific literature.

A language is said to be more expressive than another one if it can express more concepts than the other. The concept of expressiveness has been studied formally already in logic [Kle52], [Tr73]. The main result is that the additional symbols of core logic can be eliminated (i.e. expressed), if there exist a translation from extended logic to its core that satisfies some given conditions.

The concept of expressiveness has been extensively studied also in the realm of programming languages. The earliest works (e.g. [CM75] in this field have been based on comparative schematology. In this context, some authors consider such concept as computable functions and define maximally expressive class of languages as those that are equivalent to Turing machine. However, as it is pointed out in [Fel90], "*Comparing the set of computable functions that a language can represent is useless, because the languages in question are usually universal; other measures do not exist.*" Other authors [Rey81], [SS76] studied expressiveness of imperative programming languages modelling common programming constructs in terms of a simple applicative language based on a lambda calculus. They demonstrated that a number of programming constructs can be modelled locally, without restructuring the whole program, and analysed to which extent a language can support the organisation of a problem. Steele and Sussman [SS76] pointed out that "*The emphasis not should be on eliminating "bad" language constructs, but on discovering or inventing helpful ones.*" Further, Felleisen [Fel90] suggested the following analogy between formal systems and programming languages: the set of phrases of a programming language corresponds to the expressions of a formal system, the set of programs corresponds to the set of well-defined formulae, and the set of terminating programs corresponds to the set of theorems. On the basis of this analogy and the works [Kle52], [Tr73] he made an attempt to develop a formal theory of expressiveness for programming language. In this approach a common language universe U that comprises all constructs of interest is constructed. A language L is less expressive than language L_1, if L_1 can express all the constructs from U, which L can express. Felleisen concluded that programs in less expressive languages exhibit

repeated occurrences of programming patterns and suggested that such programming style is harmful.

Ontological completeness is the ability of a specification language and associated reasoning system to represent all phenomena of interest in the domain of discourse [WW93] or, in other words, the ability to describe social reality at a certain level of granularity [CW98]. Ontological completeness of a number of specification languages (mostly diagrammatic languages) has been analysed in [CW98], [RG99], [WW93], [WW90b]. As a theoretical basis to evaluate ontological completeness in this approach Bunge-Wand-Weber (BWW) models (see section 3.6.2) have been used. Using BWW ontological completeness of the language in question is evaluated with regards to Bunge's ontology. So, BWW play the role of universe of ontological primitives. In this sense BWW approach is similar to Felleisen's approach. A language L is less expressive than language L_1, if L_1 can express all the ontological BWW primitives, which L can express. The shortcoming of both approaches is that they can be accepted only in the case when one shares objectivistic point of view that the proposed universes are universal indeed, that is represent everything that may be important for the user of programming or specification language, and those representations are language neutral and independent of any user's interest.

Another similar approach to evaluate ontological completeness has been proposed in S. Milton, E. Kazmierczak and C. Keen methodology (see section 3.8.2). In [MKK98] it is proposed to use Chisholm's ontology instead of Bunge's ontology and to evaluate ontological completeness not from the viewpoint of language constructs but from the point of its ability to specify a variety of situations.

Sølvberg and co-authors [Ks03], [LSS94] see this problem in a slightly different way. Discussing wider issues of appropriateness [Kr01a], [Kr01b], [Kr03] they analyse also the concept of expressive power that differs from the discussed concept of ontological completeness in the measure that could be used not only to evaluate is any statement in the domain expressible in the language, but also to evaluate whether it is impossible to express in the language any statement, which is not in the domain. In [Kr01a], [Kr01b], [Kr03], [Ks03], [LSS94] the concept of expressiveness is defined using set-theoretical approach. This definition, in principle, can be considered as an abstract measure, however it is unclear how one can operationalise this measure.

4.2.2.2. Main Structure of Functionality

Usually a language provides core facilities allowing specifying in sufficient detail any aspect of "typical" systems in the realm. The functionality of core facilities can be restricted in two different ways: by restricting the application area or by restricting the meaning of used concepts. A flexible language should provide also some apparatus allowing extending or adapting core

facilities in order to specify "untypical" systems. Thus, in addition to suitability (i.e. core facilities) the concept of functionality comprises also flexibility of a language. Consequently, we decompose functionality into two sub-characteristics: suitability and flexibility (Figure 12).

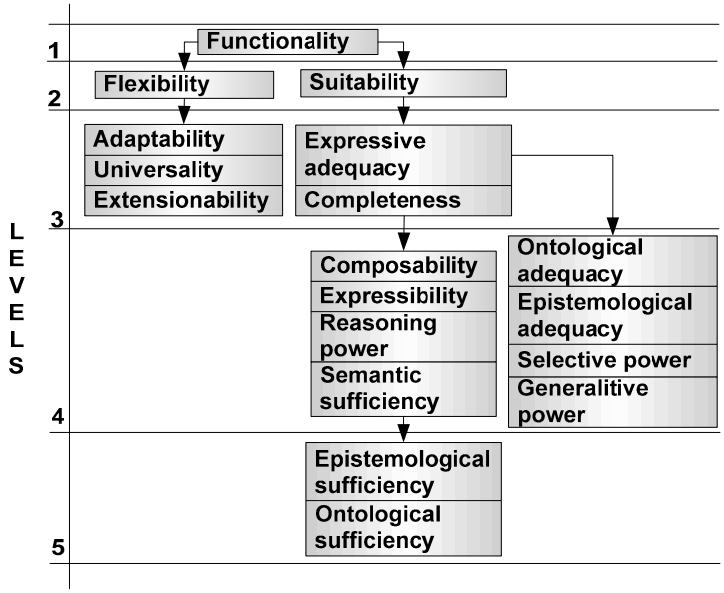


Figure 12. Sub-characteristics of functionality

Suitability, like functionality, is an aggregate characteristic. It is decomposed further into lower level characteristics (Figure 12) for two reasons: to understand better the nature of functionality and to facilitate the operationalisation of measures of functionality.

Suitability characterises how sophisticated statements about potential systems in a particular realm a specification language is able to express, and at what level of granularity it can be done. The measure of suitability is the expected frequency with which any statement about any system of a particular kind can be formulated in terms of a specification language. The expected frequency becomes equal to 1, if a language allows formulating statements about any property of a system in a given realm and expressing it with the needed degree of precision. Therefore the higher is suitability of a language the more sophisticated systems can be described and in the more details it can be done. Because of this dualistic nature of suitability, we split it further into two sub-characteristics: completeness and expressive adequacy (Figure 12).

4.2.2.3. Completeness

Completeness is the measure of the ability of a specification language to describe any system in question sufficiently and exhaustively. The value of this measure is the expected frequency with which any required property of a given system will be described in terms of evaluated specification language. However, completeness does not ensure that statements about these properties will be expressed in adequate terms or even that they will be expressed with sufficient

degree of precision. It has several aspects (Figure 12). One of the most important from them is semantic sufficiency.

Semantic sufficiency characterises the conceptual level of the linguistic system. It is the measure of the ability of a specification language to specify all "things" that might be necessary for analysis and design of any system in question. Semantic sufficiency is measured by the expected frequency with which any required concept will be expressed in terms of a specification language in question. In other words, semantic sufficiency answers the question: "In which degree sets α and Σ of the linguistic system beyond the language in question are sufficient for the needs of system engineer?" Consequently, we split further semantic adequacy into ontological sufficiency and epistemological sufficiency (Figure 12).

We use the term "ontological sufficiency" instead of the traditional term "ontological completeness", because ontological completeness characterises a specification language with regards to a chosen ontology. A language L is ontologically complete with regards to ontology O, if exists a total mapping f from a set of ontological constructs of α_O of this ontology to the set α_L of ontological primitives of the linguistic system beyond the language L. In other words, language L is ontologically complete, if exist a construct of L that can be used to represent each ontological construct of O. We aim, however, to define an ontology-independent measure of completeness. Ontological sufficiency is the expected frequency with which any system in question will be conceptualised successfully through categories provided by α_L. It answers the question: "In which degree the set α of the linguistic system beyond the language in question is sufficient for the needs of system engineer?"

Epistemological sufficiency characterises the ability of the linguistic system to express epistemological primitives or, in other words, the constructive power of "algebra of concepts" provided by Σ. Here we use the term "epistemology" in a very narrow sense, namely, to refer to concept structuring/abstraction machinery beyond a language L or, more exactly, to the facilities allowing to define conceptual primitives on the basis of ontological ones and to combine defined concepts further in order to form more complex concepts. Epistemological sufficiency can be measured by the expected frequency with which all required conceptual structures will be modelled successfully using language constructs. It ensures the ability to structure any given system in an adequate way and answers the question: "In which degree the set Σ of the linguistic system beyond the language in question is sufficient for the needs of system engineer?"

Semantic sufficiency does not characterise completeness of a specification language exhaustively, because it describes only conceptual level of the linguistic system beyond language. Assertional level can be described by two additional characteristics: expressibility and reasoning power.

Expressibility of a language is a measure of what it can be used to say. However, we use this term in more narrow sense. In our case, it describes the ability of a language to express statements about properties of instances of concepts. In other words, expressibility characterises the class of formulae expressible in language L and answers the question: "In which degree the set Ξ of the linguistic system beyond a language L is sufficient for the needs of system engineer?" The measure of the expressibility is the expected frequency with which any statement about the system in question will be expressed successfully using language constructs.

Reasoning power of the linguistic system beyond a language L is closely related to its expressibility. Reasoning power characterises the ability of reasoning apparatus Ω to derive new statements about properties of conceptual primitives and their compositions. Mainly, reasoning power is important in cases when a specification is used to reason about a system in question, for example, to prove some system properties. The measure of reasoning power is the expected frequency with which any property of the system that in principle can be derived from the stated basic assumptions will be proved using reasoning apparatus Ω.

One more characteristic of completeness is composability. It characterises the degree of possibility to compose language constructs and features together. Limited composability limits functionality, because not all meaningful compositions can be expressed using a language in question. In such cases composition cannot be realised, because the composition scheme adopted in the language does not support it, although composition is possible from logical perspective. So the measure of composability is the expected frequency with which any intuitively and logically correct composition will be realisable in the language. In terms of the linguistic system, it means that constructors of Σ and Ξ are free from any artificial restrictions on their applicability.

4.2.2.4. Expressive Adequacy

Completeness characterises how exhaustively any system in question can be described using a specification language L. However, it ensures neither that statements about the properties of this system will be expressed with sufficient degree of precision nor that they are described in adequate terms. Expressive adequacy (Figure 12) is a characteristic of internal quality of a language L that describes the ability of this language to specify the properties of a system in question in an adequate way. The measure of this characteristic is the expected frequency with which all statements about any system in question will be formulated in adequate terms and will be expressed with required degree of precision. Expressive adequacy has several aspects: ontological adequacy, epistemological adequacy, selective power, and generalitive power. Thus we split expressive power into four sub-characteristics (Figure 12).

Ontological commitments can be regarded as decisions about interpretation of statements in a given language. They can be defined not only by mapping to ontological primitives directly.

Even a language of symbolic level can be used to define ontological commitments. For example, they can be defined on the basis of set-theoretical formalism as in specification language Z. Ontological adequacy is the ability of the linguistic system to express its ontological commitments within this system itself. It focuses on the system-model link and ensures that primitives from the set α are linked directly to categories describing the system. The concern about ontological adequacy is that we can adequately capture peculiarities of the system through ontology O beyond the language L. The measure of ontological adequacy is the expected frequency with which that any system in question will be conceptualised adequately using ontological primitives provided by α. Thus ontological adequacy answers the question: "How well do specifications written in the language represent real-world phenomena?"

Epistemological adequacy is closely related to ontological adequacy. The term epistemological adequacy is used in many different senses, for example, as a degree to which the language is able to reflect all distinctions that are important. We define epistemological adequacy as the characteristic that describes the degree to which the linguistic system beyond the language is able to express epistemological primitives directly. In other words, epistemological adequacy ensures that constructors provided by Σ are linked directly to such epistemological schemes as generalisation, aggregation, and etc. The measure of epistemological adequacy is the expected frequency with which any useful epistemological primitive will have his counterpart in Σ.

The notion of selective power has been developed in the realm of relational databases. The measure of selective power of a query language is relational completeness. The language L is relationally complete if any relation derivable by means of relational calculus can be retrieved by means of this language. We argue the notion of expressive adequacy is applicable also to other languages, including specification languages. For example, we can say that the language of first order logic has limited selective power, because it cannot distinguish two elementary-equivalent structures [Rud04]. The measure of selective power of the specification language L is the expected frequency with which any two different concepts, any two different instances of a concept and any two properties of an instance of a concept can be described in the language L in a distinguishable way.

Finally, the fourth sub-characteristic of expressive adequacy is generalitive power. It characterises the ability of the language to describe system at different levels of granularity. Generalitive power allows suppressing irrelevant details while preserving essential properties of the system. The influence of generalitive power on internal quality of the language is ambiguous. The language that attempts to cover too many levels of granularity is likely to be overly complex. On the other hand, the language that supports only one level of granularity is likely to

be overly restricted. However, we suppose that measuring functionality other aspects of quality can be ignored (they are described by other characteristics) and define the measure of generalitive power as the expected frequency with which any system in question will be described at any required level of granularity.

4.2.2.5. Flexibility

Suitability characterises the power of built-in facilities of the language. Flexibility supplements suitability, because it in some sense characterises scope of applicability of the language. The more flexible is the language the easier is to adjust built-in facilities to situations that have been not provided in advance by language designers. Thus flexibility describes the extent to which the language can be adjusted to specify preliminary not intended properties. The measure of flexibility is the expected frequency with which the language will be applied to the whole spectrum of systems, namely, business systems (from the Information System design perspective), Information Systems, and Software Systems.

Flexibility can be achieved in different ways: by universalisation of the language constructs, by adaptability or by extensibility. So we split flexibility into three sub-characteristics: universality, adaptability, and extensibility (Figure 12).

Universality characterises degree of generality of ontological primitives beyond the language[1]. If the language is based on domain-oriented primitives, it has comparatively low degree of universality, because constructs of this language cannot be applied to systems in other realms. On the other hand, such ontological primitives as class or relation are universal and can be applied almost in any realm. Therefore the measure of universality is the expected frequency with which ontological primitives, provided by α, will be suitable to model concepts from the whole spectrum of systems in question. In other words, universality answers the question: “To which extent the language can be considered as a general-purpose language?”

Adaptability is in some degree is the characteristic that is opposite to universality. It describes the ability of the language to configure syntax and semantics to adapt it for arbitrary domain. Of course, adaptability is important for general-purpose languages only. Mechanisms of adaptability still are studied insufficiently. An example of such mechanism is tailoring that allows selecting some predefined constructs of general-purpose language and specialising these constructs in order to generate the language dedicated to some specific uses. In general, mechanisms of adaptability allow offering domain-specificity by the establishing domain-specific notation, constructs and abstractions on the basis of syntax and semantics of a general-purpose specification language. Therefore adaptability characterises both linguistic system and

1 It should be noted that composability also can be considered as some aspect of universality.

representational system beyond the language. The measure of adaptability is the expected frequency with which any required domain-specificity will be introduced using adaptability mechanisms provided by the language.

Extensibility is closely related to adaptability. It also allows introducing domain specificity by extending a general-purpose language with new features. However, we argue that adaptability and extensibility should be considered as separate characteristics, because they characterise presence of different mechanisms in the language and their power. Adaptability answers the question: "To which extent is the language reconfigurable?" Extensibility answers the question: "To which extent can the language be extended by new features?" A well-known mechanism of extensibility is so-called problem domain-specific profiles in UML that allow introducing some domain specificity into this language. The measure of extensibility is the expected frequency with which any required domain-specific features will be introduced using extensibility mechanisms provided by the language.

4.2.2.6. Operationalisation of the Measures of Functionality

The proposed approach provides that the values of all characteristics of internal quality are treated as expected frequencies. Indeed, expected frequencies in our approach are used as a kind of rating levels. Such approach allows calculating aggregated values of higher level characteristics from the values of lower level characteristics easily. However, characteristics of the lowest level, such as ontological and epistemological sufficiency, ontological and epistemological adequacy, selective and generalitive power, universality, adaptability, and extensibility should be measured directly. Ontological completeness is measured usually [RG99], [WW90b], [WW93] by comparing language constructs to categories of some chosen ontology. However, this approach cannot be used to measure neither ontological nor epistemological adequacy, because we aim to develop ontology-independent approach to evaluate internal quality of the specification language. For this aim we propose to use the library of evaluation test examples (see section 4.2.3).

4.2.2.7. Examples

Let's discuss some examples of the particular specification languages (UML, OCL, Alloy, and VDL-SL) comparison by their functionality and its high-level sub-characteristics.

Example 1 (Functionality)

Comparing UML, its constraint language OCL and Alloy, we conclude that:

- Comparing UML to Alloy, functionality of both languages is limited at some extent, because neither Alloy nor UML address the notion of time events directly.

- At some extent UML is more functional than Alloy, because UML defines sequence models, which can be time bound to individual actions. However, sequence models help to express only some temporal constraints.

■

Example 2 (Suitability)

From one hand, Alloy is more suitable than UML:

- UML constraints (without using OCL) are insufficient to express statements about system's properties.
- Alloy has a variety of constraints: definitions, invariants, and assertions.

From the other hand, in conjunction with OCL, UML is more complete and expressive than Alloy.

■

Example 3 (Completeness)

From one hand, UML/OCL is more complete than Alloy:

- UML/OCL includes sequences, bags, strings and numbers.
- Alloy has only sets.

From the other hand, in the relational subset UML is less complete. It has no transitive closure. An attempt to form closures using operations is not well-grounded.

■

Example 4 (Expressive adequacy)

Comparing UML, its constraint language OCL and Alloy, we conclude that expressive adequacy of UML/OCL is higher than of Alloy, because:

- Alloy is more abstract; its ontological primitives include only relations and atoms.
- UML has special aggregation and composition mechanisms, which are absent in Alloy.

■

Example 5 (Flexibility)

Comparing UML and VDL-SL we conclude that UML is more flexible than VDL-SL:

- Although UML is a general-purpose language, it contains extensibility mechanisms (stereotypes, tagged values, constraints) that can be used to tailor it to specific domains. UML extensions are expressed using profiles for different kinds of systems.
- In VDL-SL there is only possibility to extend data types with user-defined types (sets, maps, records, products).

■

4.2.3. Methodology to Evaluate Functionality Characteristics of Internal Quality

4.2.3.1. Short Description of the Proposed Approach

Elementary characteristics of internal quality are the lowest-level characteristics. Thus, we should evaluate these characteristics in some way. Best of all would be to measure elementary characteristics, but it is possible for some characteristics only. In general, the evaluation techniques depend on the kind of characteristics or, in other words, depend on the aspect of the language (functionality, reliability, efficiency or usability), which these characteristics should describe. However, it is not known how to measure these characteristics directly. For several reasons, it is a hard problem. Firstly, only several elementary characteristics can be measured directly. Secondly, internal quality is relative. Elementary characteristics of a specification language have different weights. Although the internal quality does not depend on any particular context, the global context should be taken into account. It means that the weights of characteristics (their importance) depend on the properties of the current population of Software Systems. Some characteristics are necessary for most of the population, while the others are used very rarely. Thus, when the structure of the population of Software Systems changes the degree of languages quality changes too. Finally, the evaluation results should be assessed and interpreted in a correct way. All three mentioned issues are complex and intricate.

In this dissertation it is proposed how to evaluate the characteristics describing the functionality of the language L. In the proposed approach the values of elementary characteristics are described as expected frequency with which corresponding feature of a language L will be successfully used to specify the current population of Software Systems. However, it is impossible to calculate this frequency directly, because it is impossible to have sufficient statistics about requirements of all theoretically possible Software Systems or even of the entire current population of Software Systems. So we propose to combine non-statistical sampling, domain engineering, and testing theory methods for this aim. The main idea is to develop a library of representative examples for each category of Software Systems and to use these examples as evaluation test examples to test the sufficiency of a specification language L for specification of requirements of any system from the current population of Software Systems. We propose to use a multi-stage sampling scheme (see Figure 13), which provides that all theoretically possible Software Systems should be first divided into categories of systems that meet principally different requirements, then from the population of really existing systems a sample for each category of Software Systems should be chosen via purposive judgment sampling. After this using domain analysis methods for each sample feature model should be developed. Further these models should be used to develop a number of representative examples

(evaluation test examples), which should be included into quality evaluation tests to test functionality, reliability, efficiency and usability of specification languages.

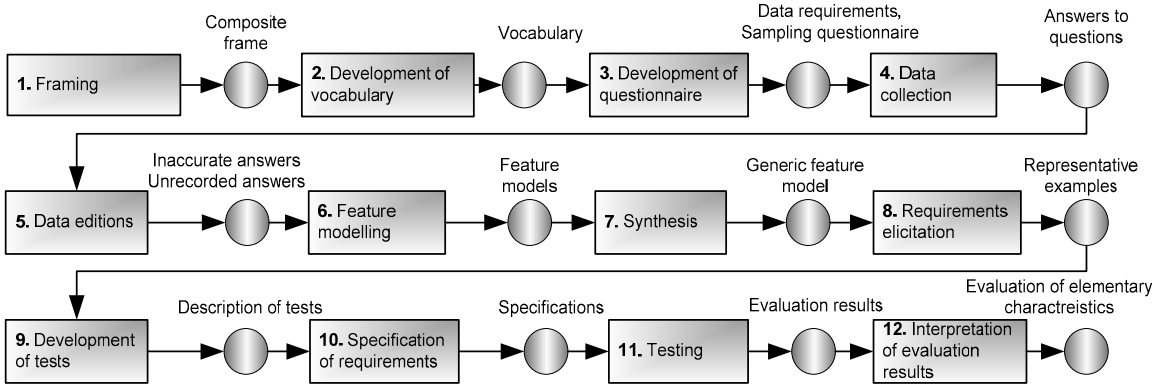


Figure 13. Methodogy to evaluate elementary characteristics of functionality

In Figure 13 presented methodology to evaluate elementary characteristics of functionality is described in detail in the remaining part of section 4.2.3.

It should be noted that although many components of the proposed methodology can be applied to evaluate elementary characteristics of reliability, efficiency, and usability of a specification language, for each of the mentioned cases the methodology as a whole should be elaborated and adapted separately.

4.2.3.2. Questions to be answered

To make the proposed approach practically usable, we must answer several significant questions:

- In which way to classify Software Systems that is what taxonomy of Software Systems is better to choose for this aim?
- What information should contain an evaluation test example and how to choose a sample and collect data that is necessary for development of such examples? How to evaluate relevance, reliability, and validity of the sample? What are the requirements for the collected data?
- How to analyse and interpret sampling results?
- In which way to describe and represent evaluation test examples in the library?
- How many evaluation test examples are needed to evaluate internal quality of a specification language?
- What should be the structure of evaluation test example? How to construct the suites of quality evaluation tests? What kind of testing methodology to apply? How to evaluate the test coverage?

- In which way to describe quality evaluation results and how to calculate expected frequency with which corresponding feature of a language L will be successfully used to specify the current population of Software Systems?
- What infrastructure is necessary for data collecting, sample developing, managing quality evaluation tests and suites of quality evaluation tests and performing tests?

The remaining part of section 4.2.3 aims to answer these questions.

4.2.3.3. Taxonomy of Software Systems

Different taxonomies of systems have been proposed in the literature. For example, Information Systems were classified into strategic-level, management-level, knowledge-level and operational-level systems [OB00]. The kind of taxonomy depends on its intended use. In our case the classification should be done in such a way that each class of the systems should be characterised by some group of requirements specific for this class only. Additionally, the subject of our considerations is rather Software Systems that can be considered as software constituent of Information Systems, because only properties of software should be specified using a specification language L. Our aim is to evaluate functionality of the language L, which is defined as the set of this language features intended to be used for specification of requirements. Taxonomy relevant to our aims has been proposed by Michael Jackson [Jac95], [Jac01].

Jackson suggests that the main classification criteria of Software Systems should be the kind of problems solvable by the system. He proposed problem-oriented methodology - problem frames - that is purposed to characterise systems of different classes. Key concepts of this methodology are world, phenomena, domains, and descriptions. Problems are all located in the world. World phenomenology includes entities, events, values, states, truths, and roles. A domain is a distinct part of the world or, in other words, a collection of related phenomena. Phenomena and their relationships constitute domain properties. Domains can share phenomena. The only way two domains can interact is by an interface of shared phenomena [Jac00]. System (*machine* in Jackson terms) is seen as a separate domain. Generally, there are four kinds of domains: systems (*machines* in Jackson terms), causal domains, lexical domains, and biddable domains. Essential properties of a causal domain are causal relationships. These relationships allow the system to cause and constrain events and state changes in the domain. The significance of the lexical domain is in the values and the truths and other relationships among them. The relationships here are not causal but definitional. Biddable domains represent users and operators, because their correct behaviour is described in instructions and they are bidden to follow the instructions.

Properties of problem domain describe facts about real world that are entirely beyond the control or influence of the system that should be built [Bra02]. These are knowledge that is

required to solve the problem and, in parallel to requirements, should be described using specification language. Problem frames model the problem domain as a set of inter-related subdomains, where a subdomain is any part of the problem domain that may be usefully singled out [Bra02]. The effects that the system should produce also should be described using specification language. Jackson terms this description as requirements. Problem frames model requirements, too. The system interacts with the real world through the interface of shared phenomena. Typically, shared phenomena are events and states. Problem frames are represented using special graphical notation. Such representation is called Problem Diagram (Figure 14).
Jackson identified seven elementary categories of the systems:

- **Transformation systems –** where the system must transform input data in a particular format into output data in a corresponding particular format.
- **Control systems** – where the system must control the behaviour of some part of the real world.
- **Commanded behaviour systems** – where the system must control the behaviour of some part of the real world in accordance with commands issued by an operator.
- **Workpiece systems –** where the system must perform directed operations upon objects that exist only within the system.
- **Connection systems –** where the system must maintain correspondence between subdomains that are not directly connected.
- **Information display systems –** where the system must maintain a continuous display of information about an autonomous dynamic real world.
- **Information answer systems** – where the system must handle requests for information about the problem domain.

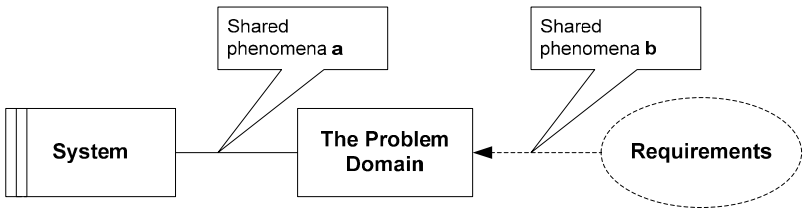


Figure 14. Problem Diagram

Whilst Jackson explicitly made no claim that he has identified all the elementary frames, as far as it is known, to date, only one additional frame – the simulator frame – has been proposed [BK03].

More complex systems can, generally, be described using composite frames composed of two or more interacting, elementary frames. To construct a multi-frame the system must be seen as composition of conceptual subsystems, each from which can be described by an elementary

frame. Some elementary frames may partly overlap, for example, they may share some subdomains of problem domain. Of course, the problem should be partitioned into elementary subproblems, too. On the other hand, it is possible to distinguish subtypes of the systems described by elementary frames. For this aim elementary frames must be enriched using so called variants. A variant typically adds additional subdomain to the problem domain and supplementary requirements. Jackson proposes [Jac01] four kinds of variants: description, operator, connection, and control variants. Description variant introduces a description lexical domain, a biddable operator domain is included into operator variant, a connection variant provides a connection domain between the system and the problem domain, with which it interfaces, and a control variant introduces no new domain, but it changes the control characteristics of interface phenomena.

4.2.3.4. Development of Evaluation Test Examples

In statistics sample is defined as a part or subset of some population taken to be representative of this population as a whole for some investigative purposes of research [Co77]. In the context of this dissertation the term population refers to all systems of the particular category of Software Systems. However, any real system has features of several elementary categories of Software Systems. So, we suppose that our population consists of the systems, in which features of the particular elementary category are mandatory. For example, many real portals can be classified as content management systems or as "chatting room" or even as workflow management systems. However, any portal first of all is a content management system and we may consider all portals as a population, which represents some subcategory of information answer systems.

The technique of selecting a suitable sample is called sampling. There are two basic kinds of sampling: statistical (probability) and non-statistical (non-probability). Statistical sampling is also called random sampling. Non-statistical sampling is any sampling technique, in which the probability of a population element being chosen is unknown. There are two basic kinds of non-probability sampling: accidental sampling and purposive sampling [Co77]. Patton defines purposive sampling as a "*sampling procedure that selects information rich cases for in-depth study*" [Pat90]. It means that during purposive sampling the sample is always intentionally selected according to the needs of the study. Judgment sampling is a kind of purposive sampling. It is a sampling technique, in which special expertise is used to choose representative population elements.

According to William M. Trochim

> "*The difference between non-probability and probability sampling is that non-probability sampling does not involve random selection and probability sampling*

does. Does that mean that non-probability samples aren't representative of the population? Not necessarily. But it does mean that non-probability samples cannot depend upon the rationale of probability theory. At least with a probabilistic sample, we know the odds or probability that we have represented the population well. We are able to estimate confidence intervals for the statistic. With non-probability samples, we may or may not represent the population well, and it will often be hard for us to know how well we've done so. In general, researchers prefer probabilistic or random sampling methods over non-probabilistic ones, and consider them to be more accurate and rigorous." [Tro02]

So, evaluation of relative sampling risk is a hard problem for any non-statistical sampling approach, including purposive judgment sampling. However, although the reliability of the sample cannot be measured, there are other ways to ensure acceptable reliability. It can be done requiring that the judgement about which element of population should be included into sample should be made by an expert in the field and that this judgement should be made using three basic criteria: representativeness of the element, value of the element, and its relative sampling risk. According to the first criterion, the system is representative enough to be chosen, if it conforms to the appropriate problem frame. It means that the selected system must be information rich or, in other words, all the information that is provided by the appropriate problem frame should be necessary in order to develop this system. The main tool to support evaluation of the representativeness is sampling questionnaire that should be developed for each category of Software Systems. According to the second criterion, the system is valuable enough to be chosen, if it still is up-to-date; if it is popular enough among users, and if its intended application area is important enough. According to the third criterion, the relative sampling risk to choose the system is acceptable, if it is expected that the requirements of this system can be in some way elicited anew more or less completely. Such expectations can be recognised as justified in case, when the system requirements specification is available, or in case, when it is possible to contact the system developers, or in case, when the system provides at least exhaustive helps, demos, the system itself is available, and its non-functional requirements can be derived from some formal or informal standards of its intended use and its using modes. In addition, data quality requirements for collected questionnaire data should be stated in such a way that the relative sampling risk would be minimised. Data quality requirements define the required degree of data validity, reliability, consistency, accuracy, completeness, and the level of detail. To be more precise, data validity requirements define the extent to which questionnaire data should conform real features of the system and its other properties, data reliability requirements define the degree to which questionnaire data should be free of errors, data

consistency requirements define terms and classifications that should be used answering to questionnaire questions, data accuracy requirements define the degree to which the questionnaire data correctly describe the system, and data completeness requirements define the degree to which questionnaire data should be exhaustive. Before starting data analysis, the expert should obtain sufficient, competent, and relevant evidence that questionnaire data meet the requirements. However, even for statistical sampling there is no effective statistical model for bringing together all these characteristics of quality into a single indicator. Additionally, except for very simple cases, there is no general statistical model for determining whether one particular set of quality characteristics provides higher overall quality than another. Thus, the only practical way to check that questionnaire data meets quality requirements is to apply manual data editing procedures. Another expert than the one who has collected data should edit data. He should detect unanswered questions, inaccurate answers, unrecorded answers and other violations of quality requirements.

Despite the difficulties in evaluation of relative sampling risk, there are two reasons to prefer purposive judgement sampling against statistical sampling. Firstly, it is impossible to define sampling frame, because, in general case, the size of the population is unknown. Of course, for any category of Software Systems the population is large and finite, but it is impossible to say, even approximately, how large it is. Secondly, it is practically impossible by random sampling to take into account the value of the chosen system, which is very important for our purposes.

For analysis and interpretation of sampling results we propose to use feature-oriented domain analysis methods. Feature-oriented domain analysis usually is defined as

> "*the process of identifying, collecting, organising, and representing the relevant information in a domain, based upon the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within a domain*" [KCH90].

The aim of domain analysis is to discover features of each system chosen for the sample and to produce feature-oriented model, which describes features of a generic system for the particular category of Software Systems. Because almost all real systems are described using several problem frames, this generic system reflects the properties of several categories of Software Systems, too. However, only some features are mandatory for all sampled systems and it is legitimately to suppose that the generic system may be used as a basis to design evaluation test example for the category of Software Systems that is characterised by these features. Of course, this test is not necessarily exhaustive and additional tests likely will be necessary. On the other hand, this test may be used for additional testing of categories, which are characterised by some optional features.

There are several feature-oriented domain analysis techniques. The essence of each technique is increasing understanding of the analysed systems by capturing the information in formal models. In our approach, the Feature-Oriented Domain Analysis (FODA) technique [KCH90] is used. The feature-oriented model, developed using this technique, describes common and variable properties of all sampled systems and the dependences between these properties.

Thus, we propose the following methodology for the development of evaluation test examples for a specification language:

1. **Framing**. Some category of Software Systems should be chosen and candidates for sampling should be identified, using criteria of representativeness, worthiness, and relative sampling risk.
2. **Development of vocabulary**. The different systems chosen for the sample may use different terminology and even be conceptualised in different ways. Thus, some conceptualisation should be chosen for the analysis, the basic terms used to model this system should be defined, and their equivalents in sampled systems should be listed. This vocabulary should describe concepts of chosen ontology and relations between these concepts. It is developed in a step-by-step manner, when analysis of the sampled systems progresses.
3. **Development of questionnaire**. The sampling questionnaire and data requirements should be developed. Both, questionnaire and data requirements, should be formulated in terms of vocabulary. Outside expert should validate completeness and correctness of the questionnaire and data requirements.
4. **Data collection**. Each sampled system should be analysed and all questions provided by the questionnaire should be answered. Decomposition should be used to support analysis. Each analysed multi-frame system should be decomposed into elementary frames and each elementary frame should be analysed autonomously.
5. **Data edition**. Inaccurate answers, unrecorded answers and other violations of quality requirements should be detected and corrected. Data edition should be done by some outside expert.
6. **Modelling**. Feature model should be developed for each sampled system. Modelling proceeds in bottom-up manner starting from the elementary frames.
7. **Synthesis**. Feature models of all sampled systems should be combined in order to produce the feature model of the generic system that is meant to be representative for the chosen category of Software Systems. The following rules should be applied to define features of the generic system:

- Features, which are identical for all sampled systems, are defined as mandatory for the generic system.
- Features, which differ for different subsets of sampled systems, however, can be generalised to one mandatory feature of the generic system, are defined as alternative subfeatures of this feature.
- All other features of sampled systems are defined as optional for the generic system.

8. **Requirements elicitation**. Functional and non-functional requirements for the generic system should be derived from the feature model and documented (see section 4.2.3.5). The requirements specification of the generic system should include requirements for all features provided by the feature model. Optional and alternative requirements should be annotated correspondingly. The main tool to support requirements derivation is refinement of features and parameterisation. Each feature should be refined into corresponding group of functional and non-functional requirements. The instances of entities mentioned in requirements should be replaced by corresponding variables. Composition rules (Table 6) should be included into the requirements specification.

Table 6. Feature table

ID	Feature	Description	Rationale	Type	Composition rules
<Hierarchical number>	<Feature name>	<Text>	<Text>	[Optional| Alternative| Mandatory]	[Mutually exclusive with: <feature names>| Requires: <feature names>]

We propose the tabular notation, feature tables (Table 6), which should be used to describe feature models. The main reason to use the tabular representation is that it is hard to understand and to analyse large feature models represented using the original FODA graphical notation. In addition, the tabular notation allows collecting together all information about the feature that in the original FODA approach is distributed among feature diagrams, feature definitions and rationale of features.

Feature table (Table 6) represents FODA feature hierarchy. In this table feature's ID is a unique identifier of the feature, which reflects all previous levels of the features hierarchy. The column "Type" describes the type of the feature. Common properties are addressed as mandatory or as alternative features and variable features are addressed as optional features. Composition rules define the semantics existing between features. They cannot be expressed in the feature diagram. There are two types of composition rules: mutual dependency (Requires) and mutual exclusion (Mutually exclusive with). All optional and alternative features that cannot be combined with the feature described in this row are listed after "Mutually exclusive with:"

statement. All optional and alternative features that must be necessarily combined with this feature are listed after "Requires:" statement [KCH90].

4.2.3.5. Refinement of the Feature Model

To be used as an evaluation test example, the feature model should be represented in the form of requirements specification. First of all requirements should be derived from the feature model. We propose to use feature refinement technique for this aim. The feature refinement is a step-by-step process that involves taking the terminal features from feature model and expanding it into software requirements, which may be expanded further into more detail requirements. In other words, it is an iterative process, in which more details are considered with each pass through the requirements specification. During this process, the requirements specification consists of two parts: existing text (everything written up to this point) and intended text (everything that is to be written in the following iterations). The structure of the existing part depends on the chosen standard. We suppose in this dissertation that requirements are structured on the basis of ISO 9126 standard [ISO91]. It means that refined requirements may be added to several structural parts of the specification at once.

The interface between the two parts is called backlog interface. It consists of all terms, which have already been used in the existing part, but which have not been defined yet. There are two kinds of definitions of a term: structural and functional. For example, if the term Web site has been mentioned in already written requirements, this term should be defined in a structural way, describing the required structure of this page, and in a functional way, describing the operations, which should be provided by the system for processing Web pages and for manipulation with them. Some terms require only structural definitions, some terms only functional definitions, and some should be defined in a structural as well as a functional way. All definitions should be in strong accordance with the sampling vocabulary, all should be expressed in the form of requirements, and all may include some constraints (non-functional requirements). In terms of Jackson [Jac95], [Jac01], structural definitions, constraints and even some functional definitions are not real requirements, because they describe interfaces or domain properties, but not the effects that the system should produce. Similar as in the object-oriented decomposition [Raj88], the terms in backlog interface address objects, functions, flows and constraining properties (e.g., "interface should be *convenient* for the user"). Thus, the backlog interface contains all functions, objects, events, roles, flows and properties mentioned in the existing part of the requirements specification.

A refinement iteration step consists of the following activities:

- Select a cluster of related terms from the current backlog interface. Definition of these terms will constitute a group of functional requirements and, possibly, related groups of

reliability, performance, usability or other non-functional requirements, which will be added to the existing part of the requirements specification.

- Define all terms in the cluster and add derived in such way requirements to the corresponding part of the existing requirements specification.
- Update the backlog interface, i.e., delete all terms defined in the step, and add all new terms, which have appeared in the step.

The refinement process proceeds until all terms are defined completely without considering any design decisions. Because the requirements specification describes a generic system, apart mandatory requirements, it should include requirements for all optional and alternative features provided by the feature model. The requirements also may be prioritised. In this dissertation we suppose that the MoSCoW list [Cle94] is used to annotate requirements priorities.

The requirements are presented in a tabular form (Table 7). The reason to use tables is the necessity to store the examples in the library of representative examples and to use tool support to manipulate with examples. For each requirement the table contains a unique identifier that reflects all previous levels of the requirements hierarchy, kind of requirement (F for functional requirements, R for reliability requirements, U for usability requirements, E for efficiency requirements, M for maintainability requirements, and P for portability requirements), statement of the requirement, requirement type (M for mandatory, O for optional, A for alternative), and requirement priority (M for must have this, S for should have this, C for could have this, and W for won't have now, but would like to have in the future). Each requirement statement should be as precise and unambiguous as possible, in an ideal case expressible in the first-order language of predicate calculus.

Table 7. Requirements table

ID	Kind	Requirement	Type	Priority
REQ - <hierarchical requirement number>	[F| R| U| E | M| P]	<Statement of the requirement>	[M| O| A]	[M| S| C| W]

Table 8. Portal requirements table

ID	Kind	Requirement	Type	Priority
REQ-1.	F	Common Web portal requirements	M	M
REQ-1.1	F	Web system must be created as Web portal **Pt**.	M	M
REQ-1.2	F	Web portal **Pt** should have three main groups of users ***Usr***: Visitor ***Vst***, Member ***Mem*** and Administrator ***Adm***.	M	S
…	…	…	…	…
REQ-2.	F	Content management	M	M
REQ-2.1	F	Portal **Pt** content items could be library items ***LbItm*** and/or document items ***DocItm***.	O	C
REQ-2.2	F	Library items ***LbItm*** should be reusable, unstructured pieces of content, typically images, static text, and banners.	O	S

…	…	…	…	…
REQ-3.	F	Collaboration requirements	O	C
REQ-3.1	F	Asynchronous collaboration requirements	O	C
REQ-3.1.1	F	Member ***Mem*** of the portal **Pt** could have possibility to participate in forums $\mathbf{Fr_1,\ldots, Fr_N}$, which are provided by the portal **Pt**.	A	C
REQ-3.1.2	F	Member ***Mem*** of the portal **Pt** could have possibility to participate in newsgroups $\mathbf{Nw_1,\ldots, Nw_N}$, which are provided by the portal **Pt**.	A	C
…	…	…	…	…

Table 8 describes example of portal requirements representation in tabular form. Because statements of requirements describe a generic system, all the concrete values should be replaced by the parameters. For some technical reasons we also use acronyms of entity names. In the Table 8, parameters are written in bold and acronyms in italic.

4.2.3.6. Evaluation Procedure

A *quality evaluation test* consists of a test identifier, a test example, test execution conditions, and expected results, which describe what this test allows to check. It is recommended that identifier includes information about internal quality characteristic (e.g., functionality) under testing, the category of systems, which has been used to produce the test example (e.g., IAS for information answer systems), and about the specification language under testing (e.g., UML). Test example is a representative set of functional and non-functional requirements, which should be specified using specification language L under testing. Test execution conditions define what specification language is tested (e.g., Z, UML, Alloy, etc.), what tools should be used to produce specification, in which way requirements should be represented using the language L (each requirement should be modelled separately, groups of related requirements should be modelled, entire requirements specification should be represented as a coherent model, etc.), how requirements should be modelled (only directly, usage of language flexibility mechanisms, except for the extensibility mechanisms, is allowed, etc.). Expected results describe elementary characteristics of internal quality under testing. The list of characteristics is derived from requirements specification using the following rules:

- if some requirement addresses an ontological primitive (e.g., concept) defined by sampling vocabulary (e.g., *user account*), then the characteristics "ontological sufficiency" and "ontological adequacy" should be added to the list;
- if some requirement addresses an epistemological primitive (e.g., specialisation of concept) defined by sampling questionnaire (e.g., "For every *registered user* an account should be provided."), then the characteristics "epistemological sufficiency" and "epistemological adequacy" should be added to the list;

- the characteristic "expressibility" should be always added to the list, because by definition any requirement is a statement about the system under the consideration (e.g., "For *every* registered user an account *should be* provided.");
- if some requirements are described not explicitly and should be derived from other requirements (e.g., "Every registered user should *inherit* the access rights *from default* user and the ability to change *inherited rights* should be provided."), then the characteristic "reasoning power" should be added to the list;
- if some requirement describes composition of different ontological categories (e.g., object state and task), defined by sampling vocabulary (e.g., "*Registered user* should be able to *unsubscribe* to newsletter he has *subscribed* to before."), then the characteristic "composability" should be added to the list;
- if some requirement includes some qualified expression (e.g., "The ability to find *all news on the specified subject* that were published *during specified time interval* should be provided."), then the characteristic "selective power" should be added to the list;
- if requirements describe system at different levels of granularity (i.e., requirement specification has hierarchical structure and requirements are refined step-by-step), then the characteristic "generalitive power" should be added to the list;
- if some requirement addresses a domain-specific ontological primitive defined by sampling vocabulary that cannot be mapped directly to the type system of the specification language under testing (e.g., "All users should have unique *e-mail addresses*."), then the characteristics "extensibility" and "adaptability" should be added to the list;
- if some requirement addresses a domain-independent ontological primitive defined in sampling vocabulary defined by sampling questionnaire (e.g., *class*), then the characteristic "universality" should be added to the list.

Figure 15 presents an example of the expected results of the test T-F-IAS-UML-01, which is developed to evaluate the ability of UML to describe information answer systems. Such tests may be used to evaluate the internal quality of a specification language for the current population of a particular category of Software Systems. To evaluate the internal quality for the whole current population of Software Systems, an appropriate suite of tests should be designed.

A suite of tests is a collection of tests developed to test some top-level group of the characteristics of internal quality (functionality, reliability, efficiency, and usability) for the particular specification language using the same testing infrastructure. It consists of evaluation tests, from which each is developed to test the different category of systems. In order to

minimise prior arrangement efforts, the suite should be developed for particular collection of tools required supporting the testing (i.e., for a particular testing infrastructure).

It is recommended that the identifier of a suite of tests include information about internal quality characteristic (e.g., functionality) under testing, about the specification language under testing (e.g., UML), and about for testing used tool (e.g., MagicDraw UML).

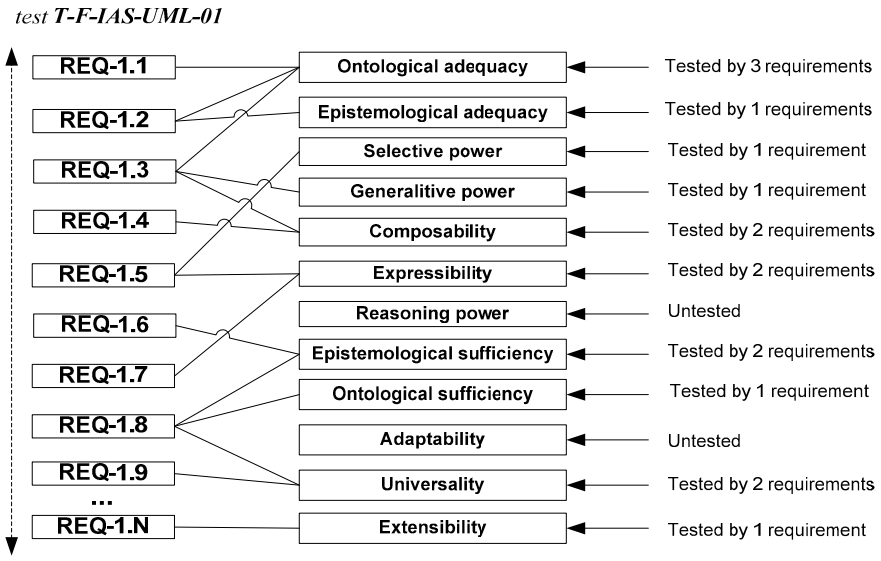


Figure 15. An example of the expected results of the test

Figure 16 presents an example of the coverage of elementary characteristics of the functionality of a specification language by the suite of evaluation tests TS-F-UML-MagicDraw, which is designed to test the functionality of UML using the CASE tool MagicDraw™[2]:

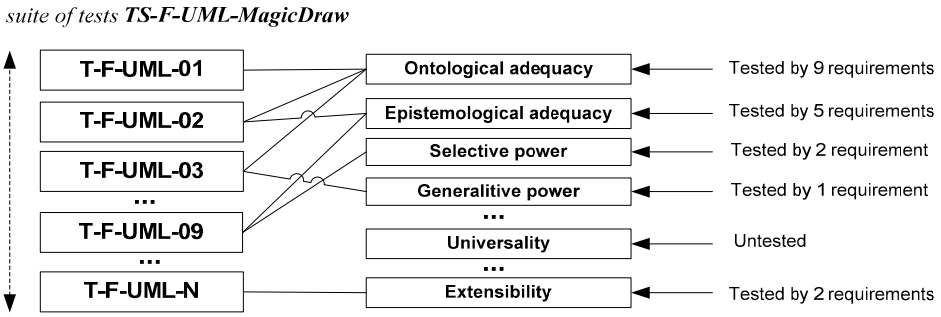


Figure 16. An example of the coverage of elementary characteristics by the suite of evaluation tests

In this example the number of requirements used to test the characteristics of functionality is calculated summarising requirements used for this aim in the tests of the suite. If the suite does not cover some characteristics, the sampling frames for each category of systems should be carefully examined. If some frames have been defined incorrectly, they should be extended to

[2] MagicDraw is the trademark of No Magic, Inc.

include additional systems. In the case, when all frames have been defined correctly, the uncovered characteristics should be eliminated from the further evaluation process, because they are insignificant for the current population of Software Systems. Although, at least in the ideal case, tools used to produce specifications to test some specification language cannot affect the evaluation results, it is reasonable to point out the tool for the suite explicitly, because appropriate testing infrastructure should be prepared. The suite of tests developed for a particular tool should cover all four top-level characteristics of the evaluated language and all should be included in the evaluation plan of this language. To avoid measurement errors, which can occur as a result of using a particular tool, it is recommended that the evaluation plan provide at least three groups of suites developed for different tools.

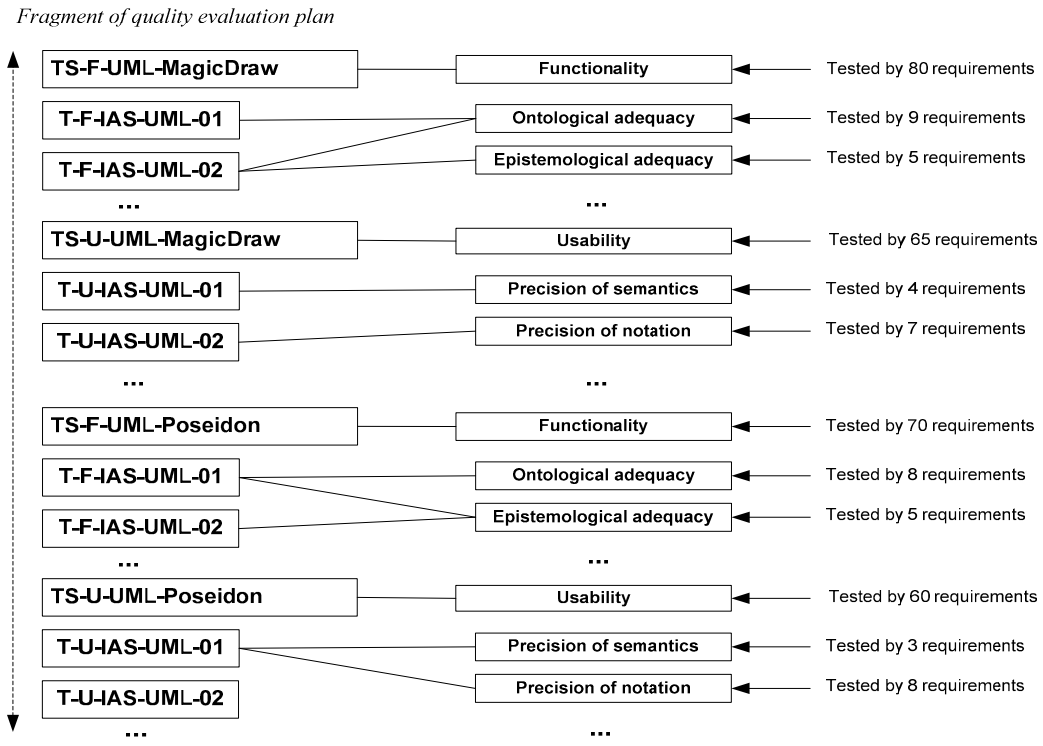


Figure 17. An example of the coverage of tests by suites of quality evaluation tests

Apart the set of suites of evaluation tests, the evaluation plan should provide time and financial constraints, evaluators and other necessary information. The coverage of tests by suites of quality evaluation tests for characteristics of internal quality is described by diagram (Figure 17) produced summarising expected results of all suites of evaluation tests provided by the plan.

4.2.3.7. Interpretation of Evaluation Results

Each elementary characteristic of functionality characterises corresponding feature of a language. In our approach, the values of these characteristics are described as expected frequency p(ξ) with which corresponding feature of a language L will be successfully used to

specify the current population of Software Systems. Thus, in order to define this frequency, the specification language evaluation results should be interpreted in an appropriate way.

Let us denote the feature of the language L described by characteristic ξ_i by $L(\xi_i)$, the expected frequency with which $L(\xi_i)$ will become necessary specifying the current population of Software Systems by $q(\xi_i)$, and the expected frequency with which $L(\xi_i)$ will be sufficient for this aim by $p(\xi_i)$. Then the value of the characteristic ξ_i is calculated as a production of expected frequencies $q(\xi_i)$ and $p(\xi_i)$.

For the characteristics of reliability, efficiency and usability $q(\xi_i)=1$. For the characteristics of functionality the value of $q(\xi_i)$ should be defined in two steps: firstly, the value of $q(\xi_i)$ should be defined for each test and after that the value of $q(\xi_i)$ should be calculated for the whole suite taking into account all evaluation tests included into this suite.

The value of $q(\xi_i)$ for a particular test should be defined taking into account test coverage and the types of requirements used to test the characteristic ξ_i. Any characteristic ξ_i is tested using a group $G(\xi_i)$ of $N(\xi_i)$ requirements. If for some evaluation test $G(\xi_i)$ is empty, then $q(\xi_i)=0$, because it means that ξ_i was not required to specify any requirement. In other cases the type of requirement used to test ξ_i should be examined:

- If at least one mandatory requirement or at least one group of alternative requirements belongs to the group $G(\xi_i)$, then $q(\xi_i)=1$.
- If the group $G(\xi_i)$ consists only of optional requirements, then, taking into account the whole feature model, the expected frequency $q_i(\xi_i)$ for each requirement r_i belonging to the $G(\xi_i)$ should be calculated, and after this $q(\xi_i)$ is defined as follows:

$$q(\xi_i) = 1 - \prod_{i=1}^{n_\xi} (1 - q_i(\xi_i)) \qquad (2).$$

In order to define expected frequency $q_i(\xi_i)$ for the r_i, it is necessary to start from the initial node of the feature model and proceed down the feature model up to the terminal feature that generates r_i. If this path is unique, then in each optional node the expected frequency of this node should be calculated as the ratio of the number of systems, which provide corresponding optional feature, to the total number of sampled systems, and the expected frequency for the r_i should be calculated multiplying all intermediate expected frequencies. If some intermediate node can be reached via several paths, then the expected frequencies should be calculated for each path (including the node itself), and the expected frequency for this node should be calculated in the same way as in formula (2). After this, it is necessary to take this node as the initial node and to proceed down further. So, in this way the expected frequencies for all terminal features are calculated. Requirements generated by these features inherit their expected frequencies.

To calculate the value of $q(\xi_i)$ for the whole suite taking into account all evaluation tests included into this suite the formula analogous to the formula (2) should be used once again.

The expected frequency $p(\xi_i)$ is defined examining testing results. It is defined as a ratio of the number $N_+(\xi_i)$ of requirements, which belong to the group $G(\xi_i)$ and has been successfully expressed in the language L, to the total number of requirements $N(\xi_i)$ in this group, i.e.:

$$p(\xi_i)=N_+(\xi_i)/N(\xi_i) \quad (3).$$

To calculate the value of $q(\xi_i)$ the group $G(\xi_i)$ of requirements should include requirements related to the particular quality characteristic tested by the particular quality evaluation test, while to calculate the value of $p(\xi_i)$ the group of requirements should include requirements related to the particular quality characteristic tested by the whole suite of quality evaluation tests.

4.2.3.8. Infrastructure

Infrastructure can be defined as the particular software tool(s) required supporting data collection, sampling, and testing of internal quality characteristics.

Sampling questionnaire data should be entered using the particular textual editor (e.g., Microsoft® Word[3].). Functional and non-functional requirements should be documented using the particular requirements management tool. Because refinement of features is an iterative process, in which more details are considered with each pass through the requirements specification, it is necessary to use the tool allowing quick view and modification of requirements. Two of the leading requirements management tools in today's market are Telelogic DOORS®[4] and IBM®[5] Rational®[6] RequisitePro[7]. RequisitePro is a requirements and use case management tool, which has requirements adding, organisation, tracking, and management capabilities, and supports Microsoft Word for requirement authoring and communication. Requisite Pro provides possibility to expose changes of requirements and omissions at multiple levels of detail. DOORS is a requirements management tool, having powerful capabilities for capturing, linking, structuring, analysing, and managing changes to requirements and their traceability. This multi-platform system ensures conformance to requirements and compliance with regulations and standards.

Specification of requirements should be done using the particular tools, which supports the evaluated specification language. In this dissertation we evaluate Z [Spi92], [Woo96] and UML

---

[3] Microsoft is a registered trademark of Microsoft Corporation.

[4] DOORS is a registered trademark of Telelogic Company.

[5] IBM is is a registered trademark of IBM Corporation.

[6] Rational Software is a registered trademark of IBM Corporation.

[7] RequisitePro is a registered trademark of IBM Corporation.

2.0 [OMG05] languages, thus as a testing infrastructure MagicDraw UML 10.0 [NM06] and Z/EVES 2.1 [Saa99] have been used. Using MagicDraw UML requirements are specified using diagrammatical UML 2.0 notation, and using Z/EVES 2.1 requirements are specified using textual Z notation. The specifications are saved as the libraries of files.

The particular testing tool should be used that should allow development of suites of quality evaluation tests for every testing infrastructure. The tool should provide possibility to derive from the requirements specification the list of the quality characteristics under testing and calculate the number of requirements, by which every derived characteristic is tested. The tools that could be used for this aim is Rational® TestManager or Test Input Adapter for Telelogic DOORS. TestManager is the central console for test activity management, execution and reporting. Built for extensibility, it supports everything from pure manual test approaches to various automated paradigms. TestManager is integrated with Rational® RequisitePro. This ensures seamless traceability between requirements and quality evaluation tests. Requirements are linked to quality evaluation tests, ensuring proper test coverage. In addition, suspicion analysis ensures that when requirements change, quality evaluation tests traced to the requirement are automatically flagged as possible candidates for modification. Test Input Adapter for Telelogic DOORS is developed for Telelogic DOORS, but it is also integrated with Rational® RequisitePro. This tool has similar capabilities as Rational® TestManager. It allows organisation and management of requirements testing process, easy modification and re-execution of quality evaluation tests, and ensures traceability between requirements and quality evaluation tests. Both Rational® TestManager and Test Input Adapter for Telelogic DOORS provide possibility to generate meaningful reports, including a series of predefined graphical and textual reports capturing the crucial aspects of quality of the specification language under testing and test completeness.

4.2.4. Techniques to Aggregate the Characteristics of Internal Quality

4.2.4.1. Short Description of the Proposed Techniques

The results of measurements of internal quality sub-characteristics should be aggregated to evaluate internal quality. Aggregation techniques are used for several purposes: to aggregate different sub-characteristics of internal quality; to aggregate results of several measurements of a particular quality sub-characteristic obtained using the same suite of quality evaluation tests; and to aggregate results of measurements of a particular quality sub-characteristic obtained using several different suites of quality evaluation tests.

In the first case the problem is to bring to light what kinds of dependencies can occur among quality sub-characteristics in the taxonomy of quality characteristics and to decide in which way to aggregate sub-characteristics in order to take into account all measurements in a proper way.

In last two cases the problem is to minimise possible deviations generated by shortcomings of the particular metric or by inaccuracy of the particular measurement. The dissertation considers this problem too but only in a very limited extent. Theoretical analysis of this problem is not provided, however, the heuristic, which is acceptable in most of practical situations, is proposed.

4.2.4.2. Preliminaries

Aggregation of information plays an important role in many fields. According to [Det00] the purpose of aggregation is "*the simultaneous use of different pieces of information (provided by several sources) in order to come to a conclusion or a decision*". More exactly, "*Aggregation refers to the process of combining values (numerical or non numerical) into a single one, so that the final result of aggregation takes into account in a given fashion all the individual aggregated values"* [GOY98]. A number of approaches, including rule-based approach, neuronal networks, fusion specific techniques, probabilistic approach, evidence theory, possibility theory, fuzzy set approach, and many other approaches have been proposed for this aim. However, all proposed approaches are based at some extent on the numerical aggregation. In other words, all approaches include aggregation of some numerical values. More generally, "*the aggregation operators are mathematical objects that have the function of reducing a set of numbers into a unique representative (or meaningful) number*" [Det00]. In theoretical considerations it is assumed usually that both aggregated values and result of aggregation belong to some finite interval, say interval [0,1], without any assumption about their nature. The choice of the interval [0,1] is not restrictive, because any interval can be transformed into this interval using a positive linear transformation ax+b, a>0 [GOY98]. Then an aggregation operator is defined as a function that assigns a real number y ($y \in [0,1]$) to any n-tuple ($x_1, x_2, \dots, x_n | x_i \in [0,1]$).

Fundamental properties of aggregation operators have been considered by [MT86], [Ovc98], [MK97], [Det00]. These properties can be divided into two groups: mathematical properties and behavioural properties. To be intuitively meaningful, an aggregation operator must satisfy at least the following axioms [Det00]:

$$\text{Aggreg}(x)=x \qquad \text{(identity)} \qquad (4)$$

$$\text{Aggreg}(0, \dots, 0) = 0,\ \text{Aggreg}(1, \dots, 1)=1 \qquad \text{(boundary conditions)} \qquad (5)$$

$$\text{Aggreg}(x_1, x_2, \dots, x_n) \le \text{Aggreg}(y_1, y_2, \dots, y_n) \qquad \text{(monotonicity)} \qquad (6)$$

$$\text{if} (x_1, x_2, \dots, x_n) \le (y_1, y_2, \dots, y_n)$$

These axioms define fundamental mathematical properties that are inherent to any aggregation operator.

It is proven [Det00] that any Aggreg operator is continuous with respect to each variable, associative, commutative, bisymmetric, and idempotent. However, these properties are non-

axiomatic and can be derived from the axioms (4), (5) and (6). Besides, it is proven that some specific classes of aggregation operators have such mathematical properties as a neutral element, an absorbent element (annihilator), compensation (Pareto property), counterbalancement and reinforcement. Finally, some aggregation operators can be stable with respect to linear changes of measurement scale and invariant for any bijection (e.g., in case when aggregation operator is defined as a projection).

Behavioural properties are specific for each class of aggregation operators. In other words, each class has individual behavioural properties. In general, there exist four behavioural properties:

- possibility to express weights of importance on the values that are aggregated,
- possibility to express relations between the values that are aggregated,
- possibility take into account the aims of aggregation, and
- possibility of an easy semantic interpretation (i.e. being able to relate the values that are aggregated to the behaviour implied by aggregation operator).

4.2.4.3. Aggregation Operators

There exist a great variety of aggregation operators. They can be roughly classified into several categories [GOY98], each of which possesses distinct behaviour or semantic: conjunctive, disjunctive, averaging (or compensative), non-compensative, and weighted operators, according to the way the values are aggregated.

Conjunctive operators are used to aggregate the values when these values are orthogonal. The term "conjunctive" accents that, in this case, aggregation is in some way analogous to the logical operator "*and*", because the resulting value is high if and only if all the aggregated values are high. For binary operator this requirement can be expressed in the following axiom [GOY98]:

$$\text{Aggreg}(1,a)=a,\ \forall a\in[0,1] \tag{7}$$

It means that for conjunctive operators 1 is a neutral element.

Important sub-category of conjunctive operators is so called triangular norms (t-norms). Triangular norms are often denoted by T. They have been introduced by Schweizer and Sklar [Sch60], [Sch83] to model distance in probabilistic spaces. t-norm is a symmetric, associative, non-decreasing for any argument mapping

$$T: [0,1]^n \rightarrow [0,1], \tag{8}$$

for which 1 is a neutral element. In aggregation theory t-norms are used to generalise Boolean logical operator "*and*" to multi-valued logic. For t-norms it is true that

$$T(x,y)\leq \min(x,y) \tag{9}$$

A t-norm is called strict, if it is strictly increasing for any argument. Well-known examples of t-norms are operator min(x,y) and product xy. Axiomatic of t-norms attempts to capture the basic properties of set intersection. It is one of the main advantages of t-norms. The main disadvantages are that t-norms generally do not satisfy criteria (idempotence, compensativeness, scale invariance, etc.), required for the aggregation of values that have different nature, for example, for the aggregation results of measurements obtained using different suites of quality evaluation tests.

Disjunctive operators are dual of conjunctive operators. They are used in cases when aggregation operator must have properties analogous to the logical operator "*or*" or, in other words, when the resulting value must be high if and only if at least one of aggregated values is high. It will be low if and only if all aggregated values are low. For binary operator this requirement can be expressed in the following axiom [GOY98]:

$$Aggreg(0,a)=a,\ \forall a\in[0,1] \tag{10}$$

It means that for disjunctive operators 0 is a neutral element.

Important sub-category of disjunctive operators is so called triangular conorms (t-conorms, s-norms). Triangular conorms are often denoted by $\perp$. t-conorm is a symmetric, associative, non-decreasing for any argument mapping

$$\perp: [0,1]\times[0,1]\rightarrow[0,1] \tag{11}$$

that satisfies the axiom (10). In aggregation theory t-conorms are used to generalise Boolean logical operator "*or*" to multi-valued logic. For t-conorms it is true that

$$\perp(x,y)\geq\max(x,y) \tag{12}$$

Using the construction

$$\perp(x,y):=1-T(1-x,1-y) \tag{13}$$

for any t-norm T dual t-conorm $\perp$ can be defined [Ful05]. t-norm and a t-conorm are dual, if they satisfy the DeMorgan law [Det00]:

$$\neg T(x, y)=\perp(\neg x, \neg y), \tag{14}$$

where negation usually is defined as a strong negation

$$\neg x=1-x \tag{15}$$

Well-known examples of t-conorms are operator max(x,y) and probabilistic addition x+y-xy. Axiomatic of t-conorms attempts to capture the basic properties of set union. It is one of the main advantages of t-conorms. The main disadvantage is that t-conorms, similarly as t-norms cannot be used to aggregate the values of different nature.

For details of basic analytical and algebraic properties of t-norms and t-conorms see [KMP02].

The third category of aggregation operators is compensative operators that are neither conjunctive nor disjunctive. They are compensative in the sense that low values are compensated by high values, and the result of combination is a medium value. Thus, compensative operators are averaging operators. They are monotonic, idempotent and are suitable for combining the values of different nature. Examples of compensative operators are mean operators[8], median and order statistics[9]. A family of mean operators formed by different extensions of arithmetic mean is called quasi-arithmetic means. It is defined [Kol30] as follows:

$$\text{Aggreg}_f(x_1, x_2, ..., x_n) = f^{-1}[\sum_{i=1}^{n} \frac{1}{n} f(x_i)] \quad (16)$$

where f is any continuous strictly monotonic function.

Conjunctive, disjunctive and compensative operators are the main categories of aggregation operators. However, there exist some aggregation operators, so called non-compensative operators, which do not belong to any of these categories. Usually, non-compensative operators are more or less of average type. However, they may extend beyond minimum and maximum operators. Examples of non-compensative operators are symmetric sum [Sil79] and compensatory operators [Zim80]. Symmetric sums are continuous, monotonic and commutative. Compensatory operators are a mix of t-norm and t-conorm. Such mixed operators provide a kind of compensation of each other by the values that are aggregated. They have rather limited properties: continuity, monotonicity, and associativity. Some non-compensative operators have a neutral element.

All discussed categories of aggregation operators are non-weighted or, in other words, treat all arguments as of the same importance. However, in many cases arguments are not of the same importance. So, in such cases, the weights of arguments must be introduced. Generally, weights in aggregation can be necessary either from qualitative or quantitative reasons. Qualitative reasons arise when the values that are aggregated have different importance. Quantitative reasons arise in cases when the input of aggregation operator has different frequencies or cardinalities. Introduction of weights extends non-weighted operators. For example, minimum and maximum

[8] Quasi-arithmetic means are idempotent, continuous, strictly monotonic, compensative and decomposable. They may have a neutral element. Arithmetic mean has the following additional property: it is stable with respect to linear changes of measurement scale.

[9] Median and statistics are idempotent, continuous, monotonic, compensative and stable with respect to linear changes of measurement scale. They may have a neutral element. In addition, medians are associative.

operators have been extended by Dubois and Prade [DP85] to weighted minimum and weighted maximum operators:

$$w\min_{w_1,\ldots,w_n}(x_1,x_2,\ldots,x_n)=\min_{i=1}^{n}[\max(1-w_i,x_i)] \tag{17}$$

$$w\max_{w_1,\ldots,w_n}(x_1,x_2,\ldots,x_n)=\max_{i=1}^{n}[\min(w_i,x_i)], \tag{18}$$

where weights $w_i \geq 0$ for all i= 1,2, …,n and $\max_{i=1}^{n}(w_i)=1$.

These operators can be generalised to weighted t-norm T_w and weighted t-conorm $\perp_w$ [Mor01]. Let $w=[w_1, w_2, \ldots, w_n]$, where $w_i \in [0,1]$, $1 \leq i \leq n$. Then $\forall x_i \in [0,1]$, $1 \leq i \leq n$

$$T_w(x_1, \ldots, x_n)=T(\perp(x_1, 1-w_1), \perp(x_2, 1-w_2), \ldots, \perp(x_n, 1-w_n)) \tag{19}$$

$$\perp_w(x_1, \ldots, x_n)=\perp(T(x_1, w_1), T(x_2, w_2), \ldots, T(x_n, w_n)) \tag{20}$$

However, axiomatic definition of t-norms does not allow weighted aggregation. In order to obtain a weighted extension of t-norms, some of the axiomatic requirements must be dropped. Weighted operators by definition are not commutative, so commutativity and associativity axioms are eliminated for them.

Example 6

Consider T=xy is the probabilistic multiplication, and $\perp=x+y-xy$ is the probabilistic addition. Then for n=2 the weighted t-norm and the weighted t-conorm will be the following:

$$T_w(x_i, y_i) = (1-w_1(1-x_1))(1-w_2(1-x_2)) \tag{21}$$

$$\perp_w(x_i, y_i) = w_1x_1+w_2x_2-w_1w_2x_1x_2 \tag{22}$$

■

The whole family of quasi-arithmetic means (16) is extended by:

$$\text{Aggreg}_f(x_1, x_2, \ldots, x_n) = f^{-1}[\sum_{i=1}^{n} w_i f(x_i)], \tag{23}$$

where w_i are the weights of the values x_i.

Example 7

Consider the arithmetic mean is extended to the weighted mean in the following way:

$$\text{Aggreg}(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i x_i, \tag{24}$$

where the weights are normalised so that $\sum_{i=1}^{n} w_i = 1$.

■

As a natural framework for the inclusion of the behavioural properties of aggregation operators the family of ordered weighted operators (OWA) has been proposed [Yag88]. OWA

operators are idempotent, monotonic and commutative. They are important, because they integrate both conjunctive and disjunctive behaviour. The OWA family is defined as follows:

$$\mathrm{Aggreg}_{\mathrm{OWA}}(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} w_i \mathrm{order}(i, x_1, x_2, ..., x_n), \tag{25}$$

where function order: $\{1,2, \ldots, n\} \times [0,1]^n \rightarrow [0,1]$ defines ordering of the values $x_1, x_2, \ldots, x_n$ in increasing order[10]. Thus, in OWA operator the weight is associated with a particular ordered position of aggregate but not with a particular value [Ful96]. The main advantage of OWA operators is their versatility [Det00], [GOY98]. In case, when $w_1=1$ and all other weights are equal to zero, OWA is equivalent to maximum operator; in case, when only $w_n=1$ and all other weights are equal to zero, it is equivalent to minimum operator. Arithmetic mean also is a particular case of OWA operator. It is obtained when all weights are equal to 1/n.

Further generalisation of average operators is so-called fuzzy integrals. It is evident that discretisation of traditional integral (including Lebesgue integral) of a function is a particular case of averaging operator, because it represents the average value of this function [GOY98]. [Det00], [Sug74] [0] and other authors extended the notion of Lebesgue integrals by introducing the fuzzy measure that is considered as the weight of importance of a set of the values that are aggregated. It is defined as a mapping

$$m : P(V) \rightarrow [0,1], \tag{26}$$

where $V=\{v_1, v_2, \ldots, v_n\}$ is a set of the values that are aggregated, and P(V) denotes the power set of V. In addition, the function m must satisfy boundary conditions and monotonicity axiom:

$$m(\varnothing) = 0, m(V) = 1; \tag{27}$$

$$\forall A, B \in P(V)((A \subset B) \Rightarrow (m(A) \leq m(B))) \tag{28}$$

Lebesgue integrals with fuzzy measure are called fuzzy integrals. Discrete fuzzy integrals can be seen as particular cases of averaging operators, too. There are two kinds of fuzzy integrals: Sugeno integrals and Choquet integrals. The discrete Choquet integral of values $x_1, x_2, \ldots, x_n$ for the values $V=\{v_1,v_2, \ldots, v_n\}$ that are aggregated with respect to a fuzzy measure m is defined as follows:

$$\begin{aligned} &\mathrm{Aggreg}_{\mathrm{Choquet}}(x_1, x_2, ..., x_n) = \\ &\sum_{i=1}^{n} (\mathrm{order}(i, x_1, x_2, ..., x_n) - \mathrm{order}(i-1, x_1, x_2, ..., x_n)) m(\{x_i, x_{i+1}, ..., x_n\}), \end{aligned} \tag{29}$$

where it is supposed that $\mathrm{order}(0, x_1, x_2, \ldots, x_n)=0$.

[10] Using substitution n-i+1 it is possible to order the values in decreasing order.

In the case when fuzzy measure $m_0=m(\varnothing)=0$, and $m_i=m(S)=i$ for any $S\subseteq\{x_1, x_2, \dots, x_n\}$ of the cardinality i, the Choquet integral corresponds to the OWA operator:

$$\mathrm{Aggreg}_{\mathrm{OWA}}(x_1, x_2, ..., x_n) = \sum_{i=1}^{n}(m_i - m_{i-1})\mathrm{order}(i, x_1, x_2, ..., x_n) \quad (30)$$

And vice versa, each OWA operator corresponds to the Choquet integral with the measure

$m_k = \sum_{i=k}^{n} w_i$.

The discrete Sugeno integral of values $x_1, x_2, \dots, x_n$ for the values $V=\{v_1, v_2, \dots, v_n\}$ that are aggregated with respect to a fuzzy measure m is defined as follows:

$$\mathrm{Aggreg}_{\mathrm{Sugeno}} = \max_{i=1}^{n}(\min(\mathrm{order}(i, x_1, x_2, ..., x_n)m(x_i, x_{i+1}, ..., x_n))), \quad (31)$$

where it is supposed that $\mathrm{order}(0, x_1, x_2, \dots, x_n)=0$.

Sugeno integrals generalise the weighted minimum and the weighted maximum. In the case, when fuzzy measure $m_0=m(\varnothing)=0$, and $m_i=m(S)=i$ for any $S\subseteq\{x_1, x_2, \dots, x_n\}$ of the cardinality i, the Sugeno integral corresponds to the weighted maximum:

$$\mathrm{Aggreg}_{\max} = \max_{i=1}^{n}(\min(\mathrm{order}(i, x_1, x_2, ..., x_n)m_i)) \quad (32)$$

In the case, when fuzzy measure $m_0=m(\varnothing)=0$, and $m_i=m(S)=i$ for any $S\subseteq\{x_1, x_2, \dots, x_n\}$ of the cardinality i, the Sugeno integral corresponds to the weighted minimum:

$$\mathrm{Aggreg}_{\min} = \min_{i=1}^{n}(\max(\mathrm{order}(i, x_1, x_2, ..., x_n)m_i)) \quad (33)$$

Choquet integrals and Sugeno integrals have been widely studied in literature and compared to each other (see, for example [BHS96]. These integrals are complementary in the sense that Choquet integrals can be seen as a kind of average, while Sugeno integrals can be seen as a kind of median. Thus, each integral is intended to be used for different purposes.

4.2.4.4. Aggregation of Internal Quality Characteristics

Let us consider further how to apply surveyed above aggregation theory in order to solve the problem of aggregation of quality characteristics of a specification language.

It is highly desirable that internal quality of a language will be measured using some numerical scale. It is also preferable to use a scale of cardinal type. On the other hand, internal quality of a language cannot be measured in any standard units, like meters or amperes, of some measurement system. In addition, internal quality characterises the potential applicability of a language only but says nothing about its quality in use or about the degree to which this language satisfies the requirements of a particular project. These are the main reasons to define internal quality of a language through the expected frequency with which this language will satisfy the requirements of any possible project regardless of its complexity or size. If, for example, quality

of some specification language is characterised by the value 0,8, it means that this language will be acceptable approximately for 4/5 of all imaginable projects.

The quality is described in a hierarchical way: it is decomposed into characteristics and sub-characteristics of several levels. Each characteristic (sub-characteristic) describes some feature of a language. Characteristics of the lowest level are described by the expected frequencies with which corresponding feature of a language will be sufficient for any possible project. So, the notion of expected frequency allows expressing partiality that is natural for specification languages, because almost always a language supports any particular feature at some limited extent only. Expected frequencies can be evaluated, for example, using in this dissertation proposed suites of quality evaluation tests (see section 4.2.3).

The values of the characteristics of all higher levels can be calculated using some aggregation techniques. Let us consider this problem in detail.

Let us discuss now the techniques to aggregate measurements of sub-characteristics.

There are several cases and for each of them different aggregation method should be used:

1. All sub-characteristics $\xi_1, \xi_2, \dots, \xi_n$ of the characteristic ξ are orthogonal and for $\forall i \in [1,n](q(\xi_i)=1)$.
2. All sub-characteristics $\xi_1, \xi_2, \dots, \xi_n$ of the characteristic ξ are orthogonal and there exists such subset $\xi' \subseteq \{\xi_1, \xi_2, \dots, \xi_n\}$ that $\xi'=\{\xi_k | q(\xi_k)<1\}$.
3. There exists some main sub-characteristic of the characteristicξ, say ξ_1, in the sense that $q(\xi_1)=1$. For all other sub-characteristics $\xi_2, \dots, \xi_n$ is true that $q(\xi_i)<1$ for $i \in [2,n]$ and $L(\xi_2)$ suppl $L(\xi_1)$, $L(\xi_3)$ suppl $L(\xi_2)$, … , $L(\xi_n)$ suppl $L(\xi_{n-1})$, where $L(\xi_k)$ suppl $L(\xi_l)$ means that $L(\xi_k)$ is supplementary (additional) for $L(\xi_l)$.
4. All sub-characteristics $\xi_1, \xi_2, \dots, \xi_n$ of the characteristic ξ are alternative, $q(\xi_i)<1$ for any $i \in [1,n]$ and is true that:

$$\sum_{i=1}^{n} q(\xi_i) = 1 \qquad (34)$$

The expected frequencies $p(\xi_1)$, $p(\xi_2)$,…, $p(\xi_n)$ can be compared to compound probabilities. Thus, methods to aggregate sub-characteristics were constructed by analogy to in probability theory used methods to combine probabilities [Pfe79]: multiplication rule and addition rule. We will apply the multiplication rule to aggregate orthogonal sub-characteristics or, in other words, in cases when features described by these sub-characteristics are used for different aims. Addition rule will be applied to aggregate supplemental and alternative sub-characteristics or, in other words, when features described by these sub-characteristics are used for the same aim and compensate each other or are alternative to each other. Let us consider now how to apply multiplication and addition rules for aggregation of quality sub-characteristics in each of cases

mentioned above. We will also check for each case that the proposed aggregation technique is an aggregation operator and that it indeed has the properties required in this particular case to aggregate sub-characteristics in a proper way.

In the first case all sub-characteristics are orthogonal. If two sub-characteristics ξ_1 and ξ_2 are orthogonal it means that $L(\xi_1)$ and $L(\xi_2)$ are independent, used for different aims. So, in the first case all characteristics are independent and will become necessary (the expected frequency with which $L(\xi)$ will become necessary specifying the current population of Software Systems is equal to 1) for any possible project. Consequently, in this case the multiplication rule can be applied directly to aggregate the sub-characteristics.

$$p(\xi) = \prod_{i=1}^{n} p(\xi_i) \tag{35}$$

In order to illustrate the proposed aggregation techniques, we use geometric probability. As sample space (i.e. the set of all possible values the events may assume) we use unitary square. Event space (i.e. the subset of the sample space consisting of events that represent a successful outcome) is represented by the corresponding area in the unitary square. Further we use Venn diagrams to illustrate aggregation techniques for aggregation of some small number of sub-characteristics. Figure 18 represents the result of aggregation of $p(\xi_1)$ and $p(\xi_2)$ by the area $p(\xi_1)p(\xi_2)$:

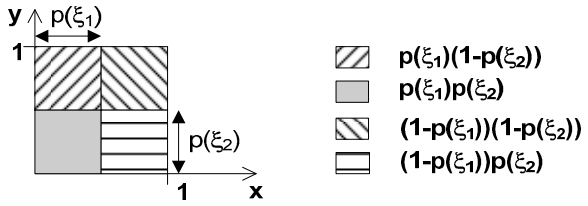


Figure 18. Aggregation of orthogonal sub-characteristics quality (case 1)

Statement 1. *The formula (35) defines a t-norm*.

Proof. It is evident that operator $\prod_{i=1}^{n} p(\xi_i)$ is aggregation operator. Satisfaction of axioms (4)-(6) follows immediately from the properties of multiplication rule:

- for n=1, the formula (35) defines identity (i.e. $p(\xi) = p(\xi_1)$);
- boundary conditions are satisfied: because (if $p(\xi_i) = 0$ for any i then $p(\xi) = 0$); if $p(\xi_i) = 1$ for any i then $p(\xi) = 1$);
- multiplication is monotonic in the interval [0,1].

It is also evident that multiplication rule is symmetric, associative and that for multiplication rule 1 is the neutral element. Consequently, (35) defines a t-norm.

■

Thus, the aggregation technique defined by formula (35) really can be applied to aggregate orthogonal sub-characteristics in a proper way, since t-norms are conjunctive aggregation operators generalising Boolean logical operator "and" to multi-valued logic.

Example 8

Internal quality of a specification language L is defined through four characteristics: functionality ξ_1, reliability ξ_2, usability ξ_3, and efficiency ξ_4. Measures of these characteristics are expected frequencies $p(\xi_1)$, $p(\xi_2)$, $p(\xi_3)$ and $p(\xi_4)$ correspondingly. All four characteristics are orthogonal to each other (i.e. describes the features of a language that are used for different purposes) and the expected frequencies $q(\xi_1)$, $q(\xi_2)$, $q(\xi_3)$ and $q(\xi_4)$ that they will become necessary for any project P are equal to 1. Thus, internal quality of the language L is described by the expected frequency $p(\xi)=p(\xi_1)p(\xi_2)p(\xi_3)p(\xi_4)$.

■

In the second case, all sub-characteristics ξ_1, ξ_2, … , ξ_n of the characteristic ξ are also orthogonal, but there exists at least one sub-characteristic ξ_i, for which $q(\xi_i)<1$. In this case multiplication rule cannot be applied directly. It must be changed in the way to meet the requirement that any feature $L(\xi)$ must be sufficient for any particular project at the same degree as its sub-features $L(\xi_i)$ are sufficient for this project in case, when these sub-features are required for this project at all. Of course, $L(\xi_i)$ is sufficient also for projects, for which it is not required. It means that multiplication rule must be modified to cover also the projects, for which $L(\xi)$ is not required:

$$p(\xi) = \prod_{i=1}^{n} (1 - q(\xi_i)(1 - p(\xi_i))) \qquad (36)$$

Figure 19 represents the result of aggregation of one sub-characteristic, which is evaluated as $p(\xi_1)$ and for which $q(\xi_1)<1$. In this case aggregated characteristic is expressed by the area $p(\xi_1)q(\xi_1)$ (for the case that characteristic is indeed sufficient for any possible project) and areas $p(\xi_1)(1-q(\xi_1))$ and $(1-p(\xi_1))(1-q(\xi_1))$ (for the case, when the characteristic is not required for this project at all). Although only one characteristic is illustrated in Figure 19, this principle can be generalised for any number of quality sub-characteristics.

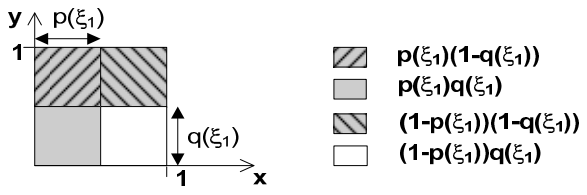


Figure 19. Aggregation of orthogonal sub-characteristics quality (case 2)

Statement 2. *The formula (36) defines a weighted t-norm*.

Proof.

Rewrite formula (36) in the form

$p(\xi)=(1-q(\xi_1)(1-p(\xi_1))(1-q(\xi_2)(1-p(\xi_2)) \dots (1-q(\xi_n)(1-p(\xi_n))$.

Denote $T=xy$ and $\perp= x+y-xy$. Let us use now proof by induction.

For n=2

$p(\xi)=(1-q(\xi_1)(1-p(\xi_1)))(1-q(\xi_2)(1-p(\xi_2)))=$

$=(1-q(\xi_1)+q(\xi_1)\, p(\xi_1))(1-q(\xi_2)+q(\xi_2)\, p(\xi_2))=$

$=(1+p(\xi_1)-p(\xi_1)-q(\xi_1)+q(\xi_1)p(\xi_1))(1+p(\xi_2)-p(\xi_2)-q(\xi_2)+q(\xi_2)p(\xi_2))=$

$=(p(\xi_1)+1-q(\xi_1)-p(\xi_1)(1-q(\xi_1))(p(\xi_2)+1-q(\xi_2)-p(\xi_2)(1-q(\xi_2))=$

$=\perp(p(\xi_1), 1-q(\xi_1))\perp(p(\xi_2), 1-q(\xi_2))=$

$=T(\perp(p(\xi_1), 1-q(\xi_1)), \perp(p(\xi_2), 1-q(\xi_2)))$.

Thus, for n=2 the **Statement 2** is true.

Now suppose that the **Statement 2** is true for n=i. It means that for n=i

$p(\xi) = (1-q(\xi_1)(1-p(\xi_1))(1-q(\xi_2)(1-p(\xi_2)) \dots (1-q(\xi_i)(1-p(\xi_i))$

defines weighted t-norm

$T(\perp(p(\xi_1), 1-q(\xi_1)), \perp(p(\xi_2), 1-q(\xi_2)), \dots , \perp(p(\xi_i), 1-q(\xi_i)))$.

For i=i+1

$p(\xi) = T(\perp(p(\xi_1), 1-q(\xi_1)), \perp(p(\xi_2), 1-q(\xi_2)), \dots , \perp(p(\xi_i), 1-q(\xi_i)))(1-q(\xi_{i+1})$

$(1-p(\xi_{i+1}))=(1-q(\xi_1)(1-p(\xi_1))(1-q(\xi_2)(1-p(\xi_2)) \dots (1-q(\xi_i)(1-p(\xi_i))(1-q(\xi_{i+1})$

$(1-p(\xi_{i+1})) = T(\perp(p(\xi_1), 1-q(\xi_1)), \perp(p(\xi_2), 1-q(\xi_2)), \dots , \perp(p(\xi_i), 1-q(\xi_i)), \perp(p(\xi_{i+1}),$

$1-q(\xi_{i+1})))$.

Consequently, (36) defines weighted t-norm $T(\perp(p(\xi_1), 1-q(\xi_1)), \perp(p(\xi_2), 1-q(\xi_2)), \dots ,$

$\perp(p(\xi_n), 1-q(\xi_n)))$.

■

Thus, the aggregation technique defined by formula (36) really can be applied to aggregate orthogonal sub-characteristics of different importance in a proper way, since weighted t-norms are theoretically sound extension of t-norms.

Example 9

Semantic sufficiency ξ of a specification language L characterises the conceptual level of the linguistic system Φ_L beyond this language. It is the measure of the ability of language L to specify all "things" that might be necessary for analysis and design of any possible project P.

Semantic sufficiency ξ is decomposed into ontological sufficiency ξ_1 and epistemological sufficiency ξ_2. Ontological sufficiency ξ_1 is characterised through the expected frequency $p(\xi_1)$ with which any project P will be conceptualised successfully through categories α_L provided by the linguistic system Φ_L. Epistemological sufficiency ξ_2 characterises the ability of the linguistic system Φ_L to express epistemological primitives or, in other words, characterises the constructive power of "algebra of concepts" provided by Φ_L. ξ_2 is characterised through the expected frequency $p(\xi_2)$ with which all required conceptual structures will be modelled using constructs of language L. The expected frequency $q(\xi_1)$ with which ξ_1 will become necessary for any project P is equal to 1, and the expected frequency $q(\xi_2)$ with which ξ_2 will become necessary for any project P is less than 1. Thus the measure of semantic sufficiency ξ is described by the expected frequency $p(\xi)=(1-q(\xi_1)(1-p(\xi_1)))(1-q(\xi_2)(1-p(\xi_1)))=p(\xi_1)(1-q(\xi_2)(1-p(\xi_1)))$.

■

In the third case sub-characteristics are supplemental to the main sub-characteristic ξ_1 or, in other words, all features $L(\xi_1)$, $L(\xi_2)$, … , $L(\xi_n)$ are used for the same purpose, and supplemental features are necessary only when previous features are insufficient for this purpose. In this case addition rule can be applied, however, it must be modified appropriately, because $q(\xi)$ can be expressed through $p(\xi)$ in the following way:

$q(\xi_1)=1$,

$q(\xi_2)=1-p(\xi_1)$,

$q(\xi_3)=1-p(\xi_1)-p(\xi_2)(1-p(\xi_1))$,

$q(\xi_4)=1-p(\xi_1)-p(\xi_2)-p(\xi_3)(1-p(\xi_1)-p(\xi_2)(1-p(\xi_1)))$,

…

$q(\xi_n)=1-p(\xi_1)-\ldots-p(\xi_{n-1})q(\xi_{n-1})$.

Thus,

$$p(\xi)=\sum_{i=1}^{n}p(\xi_i)q(\xi_i)=p(\xi_1)+p(\xi_2)q(\xi_2)+p(\xi_3)q(\xi_3)+\ldots+p(\xi_n)q(\xi_n)=$$
$$=\sum_{1\le i\le n}p(\xi_i)-\sum_{1\le j<k\le n}p(\xi_j)p(\xi_k)+\sum_{1\le j<k<s\le n}p(\xi_j)p(\xi_k)p(\xi_s)-\ldots \qquad (37)$$
$$+(-1)^{n-1}p(\xi_1)\ldots p(\xi_n)=\sum_{r=1}^{n}(-1)^{r-1}\sum_{1\le j<k<s<\ldots<r\le n}p(\xi_1)\ldots p(\xi_r)$$

This formula is a well-known inclusion-exclusion formula [Dur93]. Figure 20 represents the result of aggregation of $p(\xi_1)$, $p(\xi_2)$ and $p(\xi_3)$. In this case from the area $p(\xi_1)+p(\xi_2)+p(\xi_3)$ it is necessary to exclude the areas $p(\xi_1)p(\xi_3)$, $p(\xi_2)p(\xi_3)$, $p(\xi_1)p(\xi_2)$ and to include the area $p(\xi_1)p(\xi_2)p(\xi_3)$, because when $n>2$ the exclusion of the pairwise intersections is too severe, thus compensating inclusion is required.

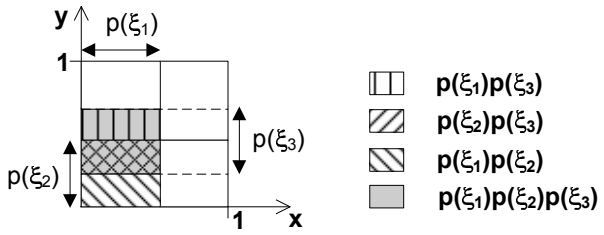


Figure 20. Aggregation of supplemental sub-characteristics quality (case 3)

Statement 3. *The formula (37) defines t-conorm*.

Proof.

Denote $\perp = x+y-xy$. Let us use now proof by induction.

For n=2

$p(\xi)=p(\xi_1)+p(\xi_2)-p(\xi_1)p(\xi_2)=\perp(p(\xi_1), p(\xi_2))$.

Thus, for n=2 the **Statement 3** is true.

Now suppose that the **Statement 3** is true for n=m. It means that for n=m

$p(\xi)=p(\xi_1)+p(\xi_2)-p(\xi_1)p(\xi_2)-\ldots+(-1)^{m-1}p(\xi_1)\ldots p(\xi_m)$

defines t-conorm

$\perp(p(\xi_1), p(\xi_2), ..., p(\xi_m))$.

For m=m+1

$$p(\xi) = \perp(p(\xi_1), p(\xi_2),..., p(\xi_m))(-1)^m p(\xi_1)...p(\xi_{m+1}) =$$
$$= \sum_{1\le i\le m}^{m} p(\xi_i) - \sum_{1\le j<k\le m}^{m} p(\xi_j)p(\xi_k) + \sum_{1\le j<k<s\le m}^{m} p(\xi_j)p(\xi_k)p(\xi_s) - ... + (-1)^{m-1} p(\xi_1)....p(\xi_m) +$$
$$+ p(\xi_m) - \sum_{1\le i\le m}^{m} p(\xi_i)p(\xi_{m+1}) + \sum_{1\le j<k\le m}^{m} p(\xi_j)p(\xi_k)p(\xi_{m+1}) - \sum_{1\le j<k<s\le m}^{m} p(\xi_j)p(\xi_k)p(\xi_s)p(\xi_{m+1}) +$$
$$+ ... + (-1)^{m+1} p(\xi_1)...p(\xi_m)p(\xi_{m+1}) =$$
$$= \sum_{1\le i\le m+1}^{m} p(\xi_i) - \sum_{1\le j<k\le m+1}^{m} p(\xi_j)p(\xi_k) + \sum_{1\le j<k<s\le m+1}^{m} p(\xi_j)p(\xi_k)p(\xi_s) - ... + (-1)^m p(\xi_1)...p(\xi_{m+1}) =$$
$$= \perp(p(\xi_1), p(\xi_2),..., p(\xi_{m+1})).$$

Consequently, (37) defines t-conorm $\perp(p(\xi_1), p(\xi_2), ..., p(\xi_n))$.

■

Thus, the aggregation technique defined by formula (37) really can be applied to aggregate in a proper way such non-orthogonal sub-characteristics, for which it is true that at least one sub-characteristic will become necessary for any possible project with the expected frequency 1, since t-conorms generalise Boolean logical operator "*or*" to multi-valued logic.

Example 10

Functionality ξ of a specification language L is defined as the existence of means required specifying functional and non-functional properties of the subject system. Functionality has two sub-characteristics: suitability ξ_1 and flexibility ξ_2. The sub-characteristic ξ_1 is the main sub-characteristic. It characterises how sophisticated statements about potential systems in a particular realm a specification language L is able to express, and at what level of granularity it can be done. The measure of ξ_1 is the expected frequency $p(\xi_1)$ with which any statement about any system of a particular kind can be formulated in terms of the language L. The expected frequency becomes equal to 1, if the language L allows formulating statements about any property of a system in a given realm and expressing it with the needed degree of precision. The sub-characteristic ξ_2 is supplemental. It describes the extent to which the language can be adjusted to specify preliminary not intended properties. The measure of ξ_2 is the expected frequency $p(\xi_2)$ with which the language L can be applied to the whole spectrum of systems, namely, business systems (from the Information Systems design perspective), Information Systems, and Software Systems. The expected frequency $q(\xi_2)=1-p(\xi_1)$, because ξ_2 becomes necessary only in cases, when $L(\xi_1)$ is not sufficient to specify the subject system. Thus, $p(\xi)=p(\xi_1)+p(\xi_2)q(\xi_2)=p(\xi_1)+p(\xi_2)(1-p(\xi_1))= p(\xi_1)+p(\xi_2)-p(\xi_1)p(\xi_2)$.

■

In the fourth case all sub-characteristics $\xi_1, \xi_2, \dots, \xi_n$, are alternative. It means that there exist no main sub-characteristic or, in other words, $q(\xi_i)<1$ for any $i\in[1,n]$, however, at least one of features $L(\xi_i)$ must be used. Thus, the condition (34) must be satisfied. In this case addition rule must be changed taking into account that sub-characteristics may duplicate each other and that the overlapping of alternative sub-characteristics should be eliminated:

$$p(\xi) = \sum_{i=1}^{n} p(\xi_i)q(\xi_i) - \prod_{i=1}^{n} p(\xi_i)q(\xi_i) \qquad (38)$$

Figure 21 represents the result of aggregation of $p(\xi_1)$ and $p(\xi_2)$. In this case from the area $p(\xi_1)q(\xi_1)+p(\xi_2)q(\xi_2)$ it is necessary to exclude the area $p(\xi_1)q(\xi_1)p(\xi_2)q(\xi_2)$.

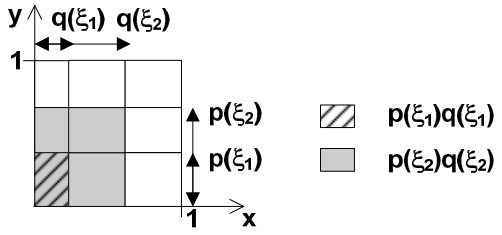


Figure 21. Aggregation of alternative sub-characteristics quality (case 4)

Statement 4. *The formula (38) defines a weighted t-conorm*.

Proof.

Rewrite formula (38) in the form

$p(\xi)=q(\xi_1)p(\xi_1)+q(\xi_2)p(\xi_2)+\ldots+q(\xi_n)p(\xi_n)-q(\xi_1)p(\xi_1)q(\xi_2)p(\xi_2)\ \ldots\ q(\xi_n)p(\xi_n))$

Denote $T=xy$ and $\perp=x+y-xy$. Let us use now proof by induction.

For n=2

$p(\xi)=q(\xi_1)p(\xi_1)+q(\xi_2)p(\xi_2)-q(\xi_1)p(\xi_1)q(\xi_2)p(\xi_2)=$

$=\perp(q(\xi_1)p(\xi_1), q(\xi_2)p(\xi_2))=\perp(T(p(\xi_1), q(\xi_1)), T(p(\xi_2), q(\xi_2))$.

Thus, for n=2 the **Statement 4** is true.

Now suppose that the **Statement 4** is true for n=i. It means that for n=i

$p(\xi)=q(\xi_1)p(\xi_1)+q(\xi_2)p(\xi_2)+\ldots+q(\xi_i)p(\xi_i)-q(\xi_1)p(\xi_1)q(\xi_2)p(\xi_2)\ \ldots\ q(\xi_i)p(\xi_i))$

defines weighted t-conorm

$\perp(T(p(\xi_1), q(\xi_1)), T(p(\xi_2), q(\xi_2)),\ldots,T(p(\xi_i), q(\xi_i)))$.

For i=i+1

$p(\xi)=\perp(T(p(\xi_1), q(\xi_1)), T(p(\xi_2), q(\xi_2)),\ldots,T(p(\xi_i), q(\xi_i)))\ q(\xi_1)p(\xi_1)+$

$+q(\xi_2)p(\xi_2)+\ldots+q(\xi_i)p(\xi_i)+q(\xi_{i+1})p(\xi_{i+1})-q(\xi_1)p(\xi_1)q(\xi_2)p(\xi_2)\ldots$

$\ldots q(\xi_i)p(\xi_i)q(\xi_{i+1})p(\xi_{i+1})=\perp(T(p(\xi_1), q(\xi_1)), T(p(\xi_2), q(\xi_2)),\ldots,T(p(\xi_i), q(\xi_i)),$

$T(p(\xi_{i+1}), q(\xi_{i+1}))$.

Consequently, (38) defines weighted t-conorm $\perp(T(p(\xi_1), q(\xi_1)), T(p(\xi_2), q(\xi_2)), ...,$

$T(p(\xi_n), q(\xi_n)))$.

■

Thus, the aggregation technique defined by formula (38) really can be applied to aggregate in a proper way non-orthogonal sub-characteristics of different importance, since weighted t-conorms are theoretically sound extension of t-conorms.

Example 11

Flexibility ξ of the language L has three sub-characteristics: universality ξ_1, adaptability ξ_2, and extensibility ξ_3. These characteristics are alternative. Sub-characteristic ξ_1 characterises the degree of generality of ontological primitives beyond the language L. The measure of ξ_1 is the expected frequency $p(\xi_1)$ with which ontological primitives, provided by categories α_L of the linguistic system Φ_L, is suitable to model concepts from the whole spectrum of subject systems. Sub-characteristic ξ_2 describes the ability of the language L to configure syntax and semantics to adapt it for arbitrary domain. The measure of ξ_2 is the expected frequency $p(\xi_2)$ with which any

required specific of domain can be introduced using adaptability mechanisms provided by the language L. ξ_3 is closely related to ξ_2. Although the features $L(\xi_2)$ and $L(\xi_1)$ in some sense are opposite to each other, we can consider them as alternative, because it is possible to implement flexibility through universality as well as through adaptability. Sub-characteristic ξ_3 is the characteristics of the language that bears on its ability to define new constructs. The measure of extensibility ξ_3 is the expected frequency $p(\xi_3)$ with which any required domain-specific features can be introduced using extensibility mechanisms provided by the language L. Both ξ_2 and ξ_3 are important for general-purpose languages only. If the expected frequency with which ξ_1 will become necessary for any project P is $q(\xi_1)$, the expected frequency with which ξ_2 will become necessary for any project P is $q(\xi_2)$ and the expected frequency with which ξ_3 will become necessary for any project P is $q(\xi_3)$, then flexibility ξ of the language L can be described by the expected frequency $p(\xi)=p(\xi_1)q(\xi_1)+p(\xi_2)q(\xi_2)+p(\xi_3)q(\xi_3)-p(\xi_1)q(\xi_1)p(\xi_2)q(\xi_2)p(\xi_3)q(\xi_3)$.

■

4.2.4.5. Aggregation of Measurements of Internal Quality Characteristics

Usually, it is very difficult or even impossible to develop precise metrics to measure the characteristics of internal quality of a specification language. One way to measure quality characteristics is to use suites of quality evaluation tests (see section 4.2.3). However, the notion of representativity cannot be defined strictly and precisely. It is defined on an empirical basis. So, in order to be more precise, any sub-characteristic ξ_i of the internal quality of a language L should be measured using a number of different metrics (suites of quality evaluation tests) and results of different measurements should be aggregated. For example, three different suites of quality evaluation tests can be used for this aim. Ideally, all measurements should produce almost identical results, because all quality evaluation tests should be debugged using a number of most popular specification languages (Z, UML, etc.). However, when a new specification language is evaluated it may occur that it accents a feature that is secondary only in most popular languages and for this reason has been underestimated in some suites of quality evaluation tests. Thus, in some cases an unacceptable dispersion of measurement values may appear. Unacceptable dispersion can also be generated by inaccuracy of measurements, because quality evaluation tests are specified manually and, consequently, measurement results can be impacted by the errors of the particular specifier. There is no way to determine the source of dispersion. It means that in both situations the same approach should be used to process unacceptable dispersion.

If the dispersion produces obviously meaningless results, for example, out of the interval [0,1], the only way to solve this problem is to analyse all quality evaluation tests, to discover

reasons of such behaviour and to improve appropriate suite of quality evaluation tests or to perform more accurate measurements. However, it is very expensive solution and can be accepted in some exceptional cases only. In most of the cases, it is preferable to minimise the dispersion in some way or, in other words, to use appropriate weights for aggregation of measurement values. Partially, it is because the taxonomy of characteristics encompasses fifty-seven characteristics and has five hierarchical levels and, therefore, the impact of one or even several characteristics to the final result of aggregation is very small.

Different approaches can be used to minimise possible deviations of measurement results. Most of methods of smoothing over measurement errors are based on the assumption that measurement errors are random errors and that measurements obey a normal, or Gaussian, distribution. It means that the measurements should be distributed about the mean in such a manner that more measurements lie close to the mean than lie far away from it. We also accept this assumption. However, we cannot use so called Q-tests, empirical schemes that are used in statistics for testing inaccurate values (so called outliers) in small data sets, for example, Grubbs Test [EPA92], Dixon Test [EPA96], or Discordance Test [EPA96], because in our case not only the deviations from Gaussian distribution, but also the distribution of values at some extent is unacceptable. In addition, the reliability of Q-tests for data sets including only three values is very low, although theoretically all mentioned above tests can be applied for such data sets, too. Aggregation operators with static weights (for example, Choquet integral) also cannot be used, because they require analysis of each particular case to assign appropriate weights. So, we propose the following heuristic, in which weights for aggregation of measurement values are determined dynamically:

1. to calculate the arithmetic mean

$$M(x_1, x_2, ..., x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i = \sum_{i=1}^{n}\left(\frac{1}{n} \cdot x_i\right) \tag{39}$$

   and the standard deviation

$$d = \sqrt{\frac{\sum_{i=1}^{n} (x_i - M(x_1, x_2, ..., x_n))^2}{n}} \tag{40}$$

   for the set of measurements $\{x_1, x_2, \ldots, x_n\}$ of the characteristic ξ;
2. to substitute the values that are out of interval [M-d;M+d] by the value M-d or M+d correspondingly;
3. to calculate the new arithmetic mean M_1 and to take it as the value of characteristic ξ.

This heuristic can be seen as a kind of combination of arithmetic and winsorised means. It allows dynamic determining of weights and minimising of dispersion. The main advantage of the

proposed heuristic is that it can be used in all cases when the dispersion of the values is not obviously meaningless, including cases when two measurements are equally distant from the mean (in this case the heuristic rejects both the highest and the lowest values) and even cases when the dispersion is at acceptable extent.

Example 12

Ontological sufficiency ξ of a language L was measured using metrics $\mu=\{m_1, m_2, m_3\}$. Measurements resulted in three different results: $m_1(\xi)=0,1$; $m_2(\xi)=0,2$; $m_3(\xi)=0,9$.

So, the aggregation of measurements should be done in the following way:

1. Firstly, the arithmetic mean of measurements

$$M = \frac{1}{3}(0,1 + 0,2 + 0,9) = 0,4$$

and the standard deviation

$$d = \sqrt{\frac{1}{3}\left((0,1 - 0,4)^2 + (0,2 - 0,4)^2 + (0,9 - 0,4)^2\right)} = 0,3559$$

are calculated by the formulas (39) and (40).

2. Secondly, the value 0,9 that is out of the interval

[M-d; M+d]=[0,0441; 0,7559]

is substituted by the value 0,7559.

3. Finally, the new arithmetic mean is calculated

$$M_1 = \frac{1}{2}(0,1 + 0,2 + 0,7559) = 0,35197$$

is calculated and the value 0,35197 is taken as the value of the characteristic ξ.

■

Example 13

Let three different measurements of the epistemological sufficiency ξ of a language L using metric m have given results with small dispersion: $m_1(\xi)=0,81$; $m_2(\xi)=0,82$; $m_3(\xi)=0,89$. Using the proposed heuristic to aggregate measurements, we obtain the following results:

1. The arithmetic mean

$$M = \frac{1}{3}(0,81 + 0,82 + 0,89) = 0,84$$

and the standard deviation

$$d = \sqrt{\frac{1}{3}\left((0{,}81 - 0{,}84)^2 + (0{,}82 - 0{,}84)^2 + (0{,}89 - 0{,}84)^2\right)} = 0{,}0356$$

are calculated by the formulas (39) and (40).

2. The value 0,89 that is out of the interval [M-d; M+d]=[0,8044; 0,8756] is replaced by the value 0,8756.

3. New arithmetic mean

$$M_1 = \frac{1}{3}(0{,}81 + 0{,}82 + 0{,}8756) = 0{,}8352$$

is calculated and the value 0,8352 is taken as the value of characteristic ξ.

■

The presented examples demonstrate that the heuristic produces acceptable results in the case with large dispersion as well as in case with small dispersion.

4.3. Specification Languages Quality in Use Evaluation Framework

4.3.1. On Construction of a Quality Model

There is a number of works, were an attempt to construct quality model is made. One of them, semiotic framework [Sin90], [Sel94], [Kr01a], [Kr01b], [Kr03], [KS03], [LSS94], was already discussed in 3.10. Let's discuss the other works, which do not address the quality of specification languages directly, but has a strong impact on the research in this field. First of all, ISO/IEC 9126 standard [ISO91] should be mentioned among them. Although this standard addresses the quality of software, its conceptual basis is significantly wider and can be applied to many other fields. For example, the ISO/IEC 9126 standard has been taken as a baseline for QStudio®[11] [QJ03], which specifies quality concepts for Java®[12] language. This approach defines so-called *Quality Attribute Tree*, which indeed is ISO/IEC 9126 quality model extended by additional sub-characteristics. The *Quality Attribute Tree* is not a proper tree, because many of quality sub-characteristics at once refine several characteristics. The "many to many" relationship introduces no difficulties in using the "tree" for quality assessment, because the quality is assessed top down by evaluating the characteristics first and then the sub-characteristics and metrics. So, using stepwise refinement techniques, the notion of code quality is expressed in terms of quality sub-characteristics and mapped further onto programming

11 QStudio is a registered trademark of QA Systems BV, The Netherlands.

12 Java is a registered trademark of Sun Microsystems.

constructs. In this way quality metrics are attached to the sub-characteristics. The advantage of such approach is the possibility to assess by measurement the quality on multiple levels of detail. Another example is EAGLES/ISO methodology [KM98] that aims to evaluate the quality of natural language processing systems. It supposes that evaluation expresses what some object is worth to somebody. Quality model is constructed according to two different perspectives. The first perspective (object-based perspective) is who likes it. The second perspective (user-based perspective) is what they like. EAGLES augments the ISO/IEC 9126 approach in the sense that it deals with the formulation of stated or implied needs, which are the primary input to the quality requirement definition. The project aims at producing an evaluation package, from which different elements can be taken and combined in different ways to reflect the needs of any particular user. In EAGLES, quality requirements definition is based on the union of the implied needs of classes of users, appropriate metrics are selected and measurements are carried out, but any user is left to construct his preferred rating level definition and assessment criteria definition [KM98]. ISO/IEC 9126 also has been used as a basis to develop quality characteristics trees for software components [SB98] and for ERP systems [BBC98].

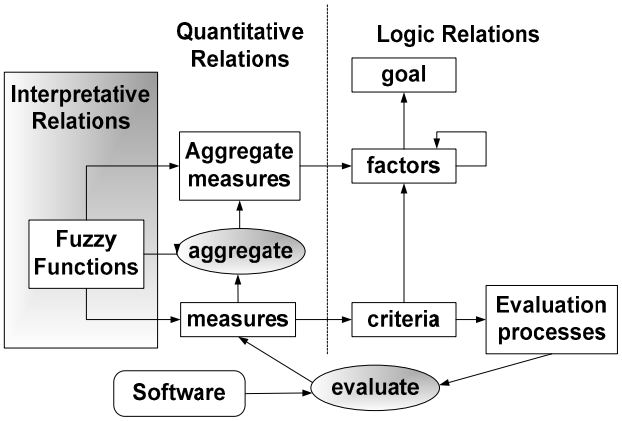


Figure 22. Fuzzy model

One more important approach, called the fuzzy model (see Figure 22) for software quality evaluation (FMSQE), has been proposed by Belchior and developed further by his colleagues [BXR96]. They proposed a hierarchical quality model based on four main concepts: goals, factors, criteria, and evaluation processes. Goals represent the general properties that a product should possess. Goals are decomposed in factors; factors can be further decomposed in sub-factors. Factors and sub-factors define different users' perspectives about the quality. Factors (sub-factors) should be decomposed in measurable quality characteristics, called criteria. For each criterion one or more alternative evaluation processes, describing a measurement methodology, should be established. In order to obtain the values of factors, both numerical and qualitative measurement results must be aggregated. Measures and aggregate measures are

related by quantitative relations. Obtained measures are interpreted using a set of fuzzy functions. Fuzzy functions support the aggregation of measures expressed in different units and are used as a suitable interpretation mechanism able to deal, at the same time, with qualitative measures and numerical data. The proposed approach provides: a membership function mapping the desired quality criterion; a method to calculate the membership function for the aggregate quality; and a final membership function for the whole product.

4.3.2. Quality Model

There exists no comprehensive definition of quality. Quality ever depends on context. Quality of a specification language is also relative. It depends on the requirements of the particular project. Following the ISO 8402 [ISO94] definition of quality, the quality of a specification language can be defined as "*the totality of features and characteristics of this language that bear on its ability to satisfy stated or implied project's needs*". The language that is excellent for one project may be unacceptable for some other projects, because each project has its specific priorities. Usually the definition of the term "quality" is formalised by a quality model. Quality models are used in many areas, however, often these models are defined imprecisely, only in the form of a quality characteristics tree and, may be, associated metrics. Although some approaches (e.g., ISO/IEC 9126 [ISO91]) address quality in use, quality requirements usually are not seen as a part of a quality model. They should be defined separately in terms of a quality characteristics tree. It is embarrassing for users, because such requirements are low-level requirements even in the case, when they are formulated using abstraction levels provided by quality characteristics tree. We propose context-oriented quality model (see Figure 23) that includes quality requirements and allows formulating those requirements in the form of high-level quality goals.

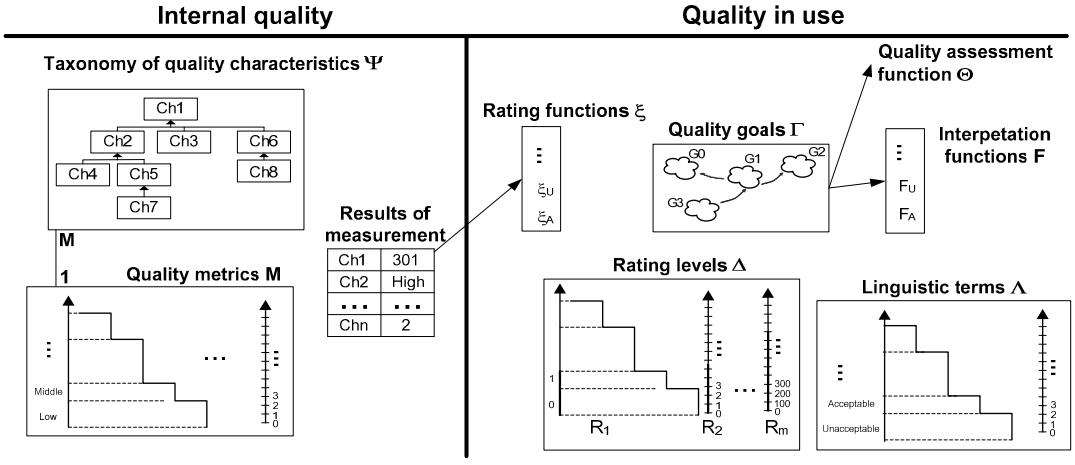


Figure 23. Context-oriented quality model

In [CLV02] the main idea how to define context-oriented quality model that includes quality requirements and allows formulating these requirements in the form of high-level quality goals is proposed. In this dissertation we elaborate this idea further. We define context-oriented quality model as the following nine-tuple:

$$Q=<\Gamma, \Psi, \Theta, \Delta, \Lambda, F, M, \mu, \xi> \quad (41),$$

where

$\Gamma=<V, A, G, f, l>$ is a weighted AND/OR digraph describing quality goal interdependencies, termed as "goal graph":

$V=\{v_i | 1\le i\le N\}$ is a nonempty set of graph vertices (quality goals), $V_1\subset V$ is a set of vertices that have no incoming arrows (terminal goals);

$A\subset VxV$ is a nonempty set of ordered pairs of vertices (graph's arrows), $A_1\subset V_1x\Psi$ is a nonempty set of ordered pairs that relates terminal goals to the characteristics of internal quality;

$G=\{g_k | 1\le k\le N_1\}$ is a nonempty set of weights;

$f: A\cup A_1\Rightarrow G$ is a weighting function that maps graph's arrows to weights;

$l: V\Rightarrow\{AND, OR\}$ is a labelling function that marks vertices with the labels "AND" or "OR".

Ψ is taxonomy of quality characteristics with the set of leaf nodes Ψ_1;

$\Delta = \{\delta_i | 0\le i\le N_2\}$ is a nonempty set of rating levels;

$\Lambda = \{\lambda_i | 0\le i\le N_2\}$ is a nonempty set of linguistic terms, used to interpret rating levels;

Θ is a quality assessment function used to evaluate quality goals and defined in the following way:

$$\Theta(v_i)=\begin{cases}\sum_{j=1}^{N_3}\delta_j * g_j, & \text{if } v_i \text{ has the label "AND"}\\ \max_{1\le j\le N_3}(\delta_j * g_j), & \text{if } v_i \text{ has the label "OR"}\end{cases} \quad (42),$$

where

$v_i\in V$ is a vertex;

δ_j is an evaluation value that signs vertices adjacent to the vertex v_i;

$g_j\in G$ is a weight that signs incoming arrow $a_j\in A\cup A_1$ of the vertex v_i;

N_3 is the number of incoming arrows for the vertex v_i (goals of the lowest levels are evaluated on the basis of rating levels of appropriate quality characteristics).

$F = \{F_\lambda | \lambda\in\Lambda\}$ is a nonempty set of interpretation functions used to interpret obtained measure of the top-level quality goal v_0. Functions $F_\lambda:\{\Theta(v) | v\in V\}\Rightarrow[0,1]$ are fuzzy functions that are defined following [BXR96].

$$F_{\lambda_i}(\Theta(v)) = \begin{cases} \dfrac{\Theta(v) - \delta_{i-1}}{\delta_i - \delta_{i-1}}, \text{ when } \delta_{i-1} \le \Theta(v) \le \delta_i \\ \dfrac{\delta_{i+1} - \Theta(v)}{\delta_{i+1} - \delta_i}, \text{ when } \delta_i < \Theta(v) \le \delta_{i+1} \\ \text{undefined, when } \Theta(v) < \delta_{i-1} \text{ or } \delta_{i+1} < \Theta(v) \end{cases} \quad (43),$$

where

$\{\delta_{i-1}, \delta_i, \delta_{i+1}\} \in \Delta$ are rating levels;

v is a vertex (the function is calculated for the vertex v_0).

$M = \{m_j \mid 1 \le j \le N_4\}$ - a nonempty set of quality metrics;

$\mu: \Psi_1 \Rightarrow M$ is one-to-many mapping that relates quality metrics to quality characteristics;

$\xi = \{\xi_\lambda \mid \lambda \in \Lambda\}$ is a nonempty set of rating functions for the set of measured values (scores), where functions $\xi_\lambda: \Psi_2 \Rightarrow [0,1]$ are defined in analogical way as in (43):

$$\xi_{\lambda_i}(m_j(x)) = \begin{cases} \dfrac{(N_2 - 1)m_j(x) - \delta_{i-1}}{\delta_i - \delta_{i-1}}, \text{ when } \dfrac{\delta_{i-1}}{N_2 - 1} \le m_j(x) \le \dfrac{\delta_i}{N_2 - 1} \\ \dfrac{\delta_{i+1} - (N_2 - 1)m_j(x)}{\delta_{i+1} - \delta_i}, \text{ when } \dfrac{\delta_i}{N_2 - 1} < m_j(x) \le \dfrac{\delta_{i+1}}{N_2 - 1} \\ \text{undefined, when } m_j(x) < \delta_{i-1} \text{ or } \delta_{i+1} < m_j(x) \end{cases} \quad (44),$$

where

$\{\delta_{i-1}, \delta_i, \delta_{i+1}\} \in \Delta$ are rating levels;

N_2 is the number of rating levels;

$m_j(x)$ – the value of quality characteristic $x \in \Psi_1$ measured using metric m_j;

$\Psi_2 = \{m(x) \mid m \in M, x \in \Psi_1, \mu(x) = m\}$.

A quality model for the particular project is designed in seven steps (Figure 24).

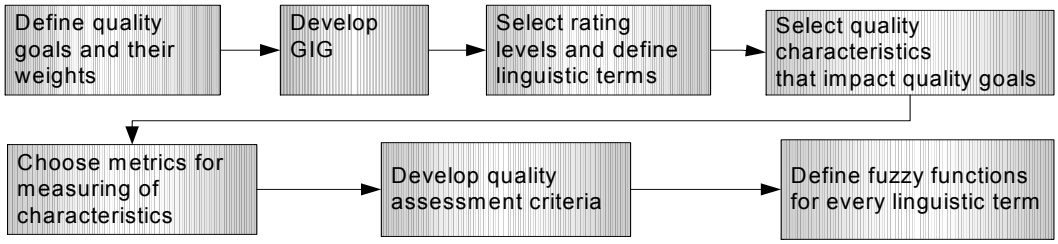


Figure 24. Quality model construction

The taxonomy of quality characteristics Ψ is the same for any context-oriented quality model, because it defines internal quality of the language and does not depend on the particular project. In general case, characteristics of internal quality can be measured using several different metrics $m_j \in M$. In the case of different measurements, their values should be aggregated. We suppose that, evaluating quality in use, internal quality has been evaluated already (an approach to evaluate internal quality is proposed in 4.2).

Because for the evaluation of quality in use not all quality characteristics may be important, in the particular quality model only some part of the taxonomy Ψ may be used. Values of to this part of the taxonomy Ψ belonging quality characteristics are mapped to appropriate rating values using (fuzzy) rating functions $\xi_\lambda \in \xi$.

Let us consider the example of a simple quality model Q_1 designed for project P_1 in order to choose the most appropriate specification language.

Example 14

1. Goal interdependencies graph Γ is presented in Figure 25.

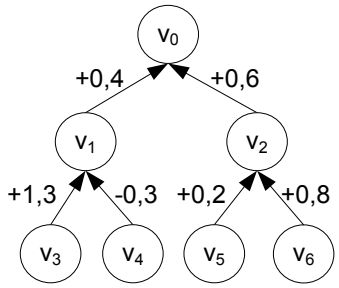


Figure 25. The weighted digraph describing interdependencies between quality goals

The set of weights is defined as G={-0.3, +0.2, +0.4, +0.6, +0.8, +1.3}.

Weighting function f is defined as follows:

$f(v_3,v_1)$=+1.3; $f(v_4,v_1)$=-0.3; $f(v_1,v_0)$=+0.4;

$f(v_5,v_2)$=+0.2; $f(v_6,v_2)$=+0.8; $f(v_2,v_0)$=+0.6.

2. The part of the taxonomy of quality characteristics Ψ_1 (Figure 26) has 3 leaf nodes Ch_1, Ch_2, Ch_3.

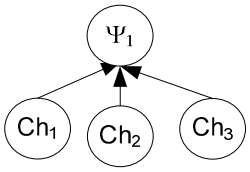


Figure 26. The part of the taxonomy of quality characteristics Ψ_1

3. The set of rating levels is defined as Δ = {0, 1, 2, 3}.

4. The set of linguistic terms is defined as Λ = {U, A, G, VG}, where U is *unacceptable*, A is *acceptable*, G is *good* and VG is *very good*.

5. According to (42) quality assessment function Θ is calculated in the following way:

$\Theta(v_1)$= 1.3 $\Theta(v_3)$ – 0.3 $\Theta(v_4)$;

$\Theta(v_2)$= 0.2 $\Theta(v_5)$ + 0.8 $\Theta(v_6)$;

$\Theta(v_0)$= 0.4 $\Theta(v_1)$ + 0.6 $\Theta(v_2)$;

$\Theta(v_3)$= 1.5 $\xi(Ch_1)$ – 0.5 $\xi(Ch_2)$;

$\Theta(v_4)= 0.2\ \xi(Ch_2) + 0.8\ \xi(Ch_3)$;

$\Theta(v_5)= 0.3\ \xi(Ch_1) + 0.8\ \xi(Ch_2) - 0.1\ \xi(Ch_3)$;

$\Theta(v_6)= 0.8\ \xi(Ch_1) + 0.2\ \xi(Ch_3)$.

6. The set of quality metrics is defined as $\mu = \{m_1, m_2\}$. Metrics are related to quality characteristics in the following way:

$\mu=\{(m_1,Ch_1), (m_2,Ch_2), (m_2,Ch_3)\}$

7. According to (44) rating functions ξ_λ are calculated in the following way:

$$\xi_U(m_j(x)) = -3m_j(x) + 1, \quad \text{when} \quad 0 < m_j(x) \le 1/3$$

$$\xi_A(m_j(x)) = \begin{cases} 3m_j(x), & \text{when } 0 \le m_j(x) \le 1/3 \\ -3m_j(x) + 2, & \text{when } 1/3 < m_j(x) \le 2/3 \end{cases}$$

$$\xi_G(m_j(x)) = \begin{cases} 3m_j(x) - 1, & \text{when } 1/3 \le m_j(x) \le 2/3 \\ -3m_j(x) + 3, & \text{when } 2/3 < m_j(x) \le 1 \end{cases}$$

$$\xi_{VG}(m_j(\xi_i)) = 3m_j(\xi_i) - 2, \quad \text{when} \quad 2/3 \le m_j(\xi_i) \le 1$$

8. According to (43), for $\Lambda = \{U, A, G, VG\}$ interpretation functions F_λ in the point v_0 are calculated in the following way:

$$F_U(\Theta(v_0)) = -\Theta(v_0) + 1, \quad \text{whe n} \quad 0 < \Theta(v_0) \le 1$$

$$F_A(\Theta(v_0)) = \begin{cases} \Theta(v_0), & \text{when } 0 \le \Theta(v_0) \le 1 \\ -\Theta(v_0) + 2, & \text{when } 1 < \Theta(v_0) \le 2 \end{cases}$$

$$F_G(\Theta(v_0)) = \begin{cases} \Theta(v_0) - 1, & \text{when } 1 \le \Theta(v_0) \le 2 \\ -\Theta(v_0) + 3, & \text{when } 2 < \Theta(v_0) \le 3 \end{cases}$$

$$F_{VG}(\Theta(v_0)) = -\Theta(v_0) - 2, \quad \text{whe n} \quad 2 \le \Theta(v_0) \le 3$$

■

4.3.3. Quality Evaluation Procedure

The proposed user-oriented quality model suggests appropriate quality evaluation procedure. This procedure includes five steps (Figure 27).

1. Characteristics of internal quality, which are relevant to the quality goals V, are identified and the part Ψ_1 of the taxonomy Ψ is constructed.
2. Rating values for characteristics of Ψ_1 are calculated using rating functions from ξ.
3. The rating values for terminal quality goals are calculated using quality assessment function Θ.
4. The rating values are propagated through the goal interdependencies graph Γ using quality assessment function Θ.

5. The rating value of the top-level quality goal $\Theta(v_0)$ is interpreted using interpretation functions F_λ $(\Theta(v_0))$.

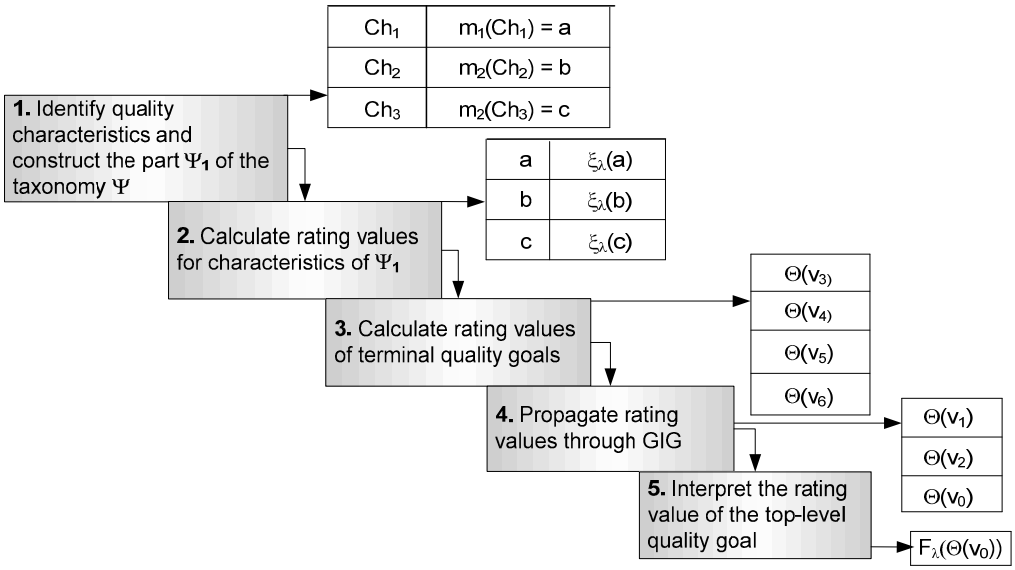


Figure 27. Quality evaluation procedure

Example 15

The evaluation of a specification language L_1 according to quality evaluation procedure provided by the quality model Q_1 presented in **Example 14** gives the following results:

1. $m_1(Ch_1) = 0{,}17$; $m_2(Ch_2) = 0{,}47$; $m_2(Ch_3) = 0{,}27$[13].

2. $m_1(Ch_1)$:

$0<m_1(Ch_1)\leq 1/3$, $0\leq m_1(Ch_1)\leq 1/3$

$\xi_U(m_1(Ch_1)) = -3*0{,}17+1=0{,}49$; $\xi_A(m_1(Ch_1)) = 3*0{,}17 = 0{,}51$

(Ch_1) is 49 % *Unacceptable* and 51 % *Acceptable.* Thus, the rating level is: 0*0,49+1*0,51=0,51 (0 and 1 are corresponding rating levels).

$m_2(Ch_2)$:

$1/3< m_2(Ch_2)\leq 2/3$, $1/3\leq m_2(Ch_2)\leq 2/3$

$\xi_A(m_2(Ch_2)) = -3*0{,}47+2 = 0{,}59$; $\xi_G(m_2(Ch_2)) = 3*0{,}47-1 = 0{,}41$

Ch_2 is 59 % *Acceptable* and 41 % *Good*. Thus, the rating level is: 1*0,59+2*0,41=1,41.

$m_2(Ch_3)$:

$0< m_2(Ch_3)\leq 1/3$, $0\leq m_2(Ch_3)\leq 1/3$

$\xi_U(m_2(Ch_3)) = -3*0{,}27+1 = 0{,}19$; $\xi_A(m_2(Ch_3)) = 3*0{,}27=0{,}81$

[13] We assume that these values were obtained after application of in 4.2 proposed approach to evaluate internal quality.

Ch_2 is 19 % *Unacceptable* and 81 % *Acceptable*. Thus, the rating level is: 0*0,19+1*0,81=0,81.

3. $\Theta(v_3)= 0,06$; $\Theta(v_4)= 0,93$; $\Theta(v_5)= 1,2$; $\Theta(v_6)= 0,57$;

4. $\Theta(v_1)= -0,2$; $\Theta(v_2)= 0,696$; $\Theta(v_0)= 0,338$

5. $F_U(\Theta(v_0)) = -0,338+1=0,662$; $F_A(\Theta(v_0))=0,338$

Thus, the quality of evaluated specification language L_1 for the project P_1 is 66% Unacceptable and 34% Acceptable.

■

4.3.4. Quality Goals

In this dissertation proposed approach to evaluate internal quality of a specification language, high-level quality requirements are formulated in the form of quality goals. The user's treatment of quality goals and interdependencies between goals is described using goal interdependency graph (GIG) Γ. The idea of GIG is borrowed from [CNM99], where similar graphs, namely, soft-goal interdependency graphs (SIG), are used to define non-functional software requirements. The advantage of GIG, comparing to the taxonomy of quality characteristics, is that the user can start from business goals, formulate high-level quality requirements and derive further detailed quality requirements, formulated in the terms of low-level quality characteristics. For example, top management may aim to spend money on staff training, minimise the amount of efforts needed to produce specifications, and produce specifications readable by domain experts. In other words, management is looking for a specification language, which would be simple enough to learn in short terms by not skilled staff, would be efficient and would have high degree of audience appropriateness. GIG allows defining goal interdependences and priorities, helps to expose implicit interdependencies and to refine requirements up to low-level quality characteristics. By refinement, for each goal a set of sub-goals to satisfy this goal is introduced. Parent goal may be satisfied by all of its sub-goals or by any of them. In addition, some sub-goals can contribute positively towards a particular goal and negatively towards other goals.

Although GIGs are very similar to SIGs, they are used for different purposes and in different way. The main problem investigated in [CNM99] is software design problem. SIGs are used in top-down manner with the aim to refine non-functional requirements, including quality requirements, to choose design decisions, which accomplish those requirements in the possibly best way, and to evaluate the impact of chosen decisions in bottom-up way. Our task is to choose such specification language, which satisfies quality requirements in the best way. So we need to choose not the design decisions, but the most appropriate metrics and measurement procedures (see section 4.2) and, further, to propagate evaluation results through the GIG in the bottom-up manner.

4.3.5. Quality Assessment Function

Each quality goal is evaluated using quality assessment function Θ. This function is used to propagate rating values of quality characteristics through the GIG from bottom towards the top of the graph. Measured or aggregated values of quality characteristics should be mapped to rating values, because, in general case, they are incomparable, expressed in different dimensions. Rating values of the lowest GIG level are used as intermediate to calculate quality assessment function of the highest GIG level. Quality assessment function depends on goals priorities and, consequently, is designed in such way that to evaluate the impact of quality characteristics of the chosen language on the quality goals. Because the arrows are signed by negative and positive priorities, the calculated value may be not exactly integer. It means that we propagate towards top of the graph not the rating levels itself, but some values that belong to intervals $[v_1, v_2]$, where v_1 and v_2 are adjacent rating levels.

For "AND" vertices the value of quality assessment function is calculated taking into account the weights of all incoming arcs. In this case, positive as well as negative weights are allowed, and it is required that the sum of weights should be equal to 1. For "OR" vertices the value of the Θ is the maximal one from the propagated values. In this case, only positive weights are allowed, and it is required that any weight do not exceed 1.

4.4. Conclusions

First of all, this dissertation continues research on the evaluation of the quality of specification languages, which has begun in [CLV02]. Internal quality of a specification language L describes the quality that is independent from any context of use. Because of the imprecise nature of quality characteristics, it is reasonable to define such quality as the expected frequency with which the language L will satisfy the requirements of any imaginable project P. In an analogous way can be defined also all characteristics of internal quality, including elementary ones.

Second, because characteristics of internal quality of the language L form a large hierarchical structure F, in order to evaluate internal quality it is necessary to aggregate sub-characteristics through the whole structure F. Thus, techniques of aggregation depend on the kind of dependences among characteristics that are aggregated. There exist four kinds of such dependencies: characteristics are orthogonal (independent) and all are required for any project; characteristics are orthogonal but not all are required for any project; characteristics supplement the one that is required for any project; and none of the characteristics is required for any project. In the first case the characteristics can be aggregated properly using a kind of t-norm, in the second case using a kind of weighted t-norm, in the third case using a kind of t-conorm, and in

the fourth case using a kind of weighted t-conorm. In order to minimise possible deviations generated by shortcomings of the particular metric (suite of quality evaluation tests) or by inaccuracy of the particular measurement, the mean with weights that are determined dynamically should be calculated. For this aim the dissertation proposes the heuristic, which can be seen as a kind of combination of arithmetic and winsorised means.

Third, the dissertation elaborates in [CLV02] proposed quality model and proposes a systematic evaluation of quality in use on the basis of measurements of internal quality. Such approach can be used to evaluate the quality of the specification language in the context of the particular project, i.e. considering the high-level quality requirements formulated by the users of the particular system in the form of quality goals.

Fourth, the main characteristic of internal quality – functionality– is analysed in the dissertation, and it is concluded that evaluation of the elementary characteristics of functionality is a hard and complicated task, which requires relatively high time and labour overheads. Many and long-time efforts are needed to do sampling, develop suites of tests, test language and interpret obtained results. Besides, some difficulties of theoretical character should be overcome. The theory of problem frames is relatively new and still not sufficiently elaborated. There are no any well-grounded methods for framing and sampling particular category of systems. Too little is known how to eliminate the impact of human factor to results of evaluation procedure. Thus, the systematic evaluation clarifies deep internal structure of each evaluated language as well as of specification languages in general and provides valuable experience that could be used during construction of new specification languages. We hope the proposed approach to evaluate internal quality will contribute both to the research on evaluation of the quality of existing specification languages and to the development of new ones.

5. Experimental evaluation of UML and Z languages functionality

Detail description of experimental evaluation of UML and Z specification languages functionality with UML and Z specifications (created using MagicDraw UML 10.0 [NM06] and Z/EVES 2.1 [Saa99] tools) is presented in technical report [Gas06a].

5.1. Contribution to the Experimental Evaluation

The experimental evaluation has been carried out by me. First of all, I have chosen the category of systems under experiment and found the particular systems of this category. Second, using methodology to evaluate functionality characteristics of internal quality I have developed the requirements specification for generic Web portal. Third, I have specified these requirements in UML and Z specification languages. Fourth, I have developed suites of quality evaluation tests to test functionality characteristics of UML and Z languages and used these test to evaluate elementary characteristics of functionality. Finally, I have used aggregation techniques to aggregate elementary characteristics of functionality and obtain the evaluation of functionality for UML and Z languages.

5.2. Aim and Objectives

The experimental evaluation aims to demonstrate how to use in the dissertation proposed specification languages internal quality evaluation framework to evaluate functionality of UML and Z specification languages independently from any context of use.

The tasks that should be carried out during the experimental evaluation are the following:

- to choose the category of Software Systems and describe its multi-frame system;
- to develop sampling vocabulary and sampling questionnaire;
- to create feature model;
- to develop evaluation test examples by refinement of feature model into requirements specification;
- to develop quality evaluation tests for MagicDraw UML 10.0 [NM06] testing infrastructure;
- to develop quality evaluation tests for Z/EVES 2.1 [Saa99] testing infrastructure;
- to apply developed quality evaluation tests to test elementary characteristics of functionality for UML and Z specification languages;
- to interpret the results of elementary characteristics evaluation;
- to aggregate the values of elementary characteristics and calculate the value of functionality for UML and Z specification languages;

- to make conclusions about functionality characteristics of UML and Z specification languages.

5.3. Subject of the Experimental Evaluation

The subject of the experimental evaluation is Z [Spi92], [Woo96] and UML 2.0 [OMG05] specification languages. The category of Software Systems that have been chosen for the experimental evaluation is Web portal. The reason of such choice is that it is a multi-frame system that can be described using two different information answer frames (for content management and for search) and two different connection frames (for chatting and workflow management).

5.4. Evaluation of the Functionality of UML and Z Languages

5.4.1. Framing

Most of Web portals contain content management, search, collaboration and groupware, task management and workflow functionality. Thus, a generic Web portal may be described by a multi-frame system, which contains four elementary frames (see Figure 28). The frames for content management and search are interconnected one with the other and with all other frames by Content Repository domain that is the subject of the enquiries, used to store/retrieve content items, and by Member domain that is the enquirer, who searches/publishes the content items of the portal. The frames for chatting and workflow management are interconnected one with the other and with all other frames by Member domain that is connected to all other domains of these two frames, because portal members use the provided connection to exchange messages and tasks with one another.

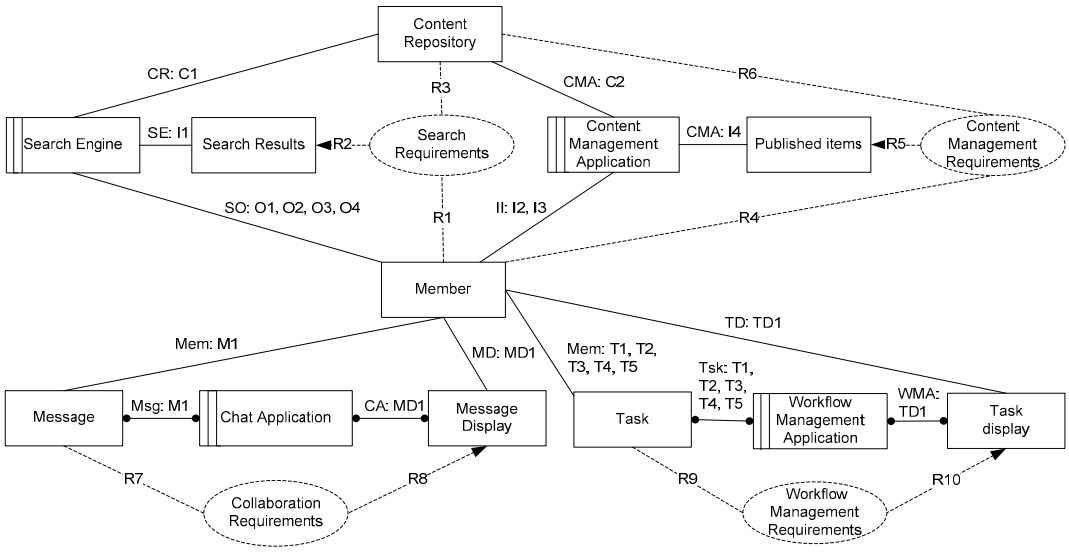


Figure 28. Multi-frame system for Web portal

In the Figure 28 used denotations for shared phenomena (events) are the following:

- R1 – Requirements for entered search options.
- R2 – Requirements that constrain displaying of found search results.
- R3 – Requirements for searching of content items in the repository by the path, used in the Portal Information Architecture.
- O1 – Search keywords are entered.
- O2 – Ordering options are entered.
- O3 – Date options are entered.
- O4 – Scope options are entered.
- I1 – Found search results are displayed.
- C1 – The path that leads to the searched for content items and consists of hierarchical categories of Portal Information Architecture.
- R4 – Requirements for content items under creation.
- R5 – Requirements that constrain publishing of content items and their display.
- R6 – Requirements for saving of content items in consistency with Portal Information Architecture.
- I2 – Library item is entered.
- I3 – Document item is entered.
- I4 – Published items are displayed.
- C2 – The path that leads to published items and consists of hierarchical categories of Portal Information Architecture.
- R7, R8 – Requirements for correspondence between by the particular portal member typed message and for the other portal member displayed message. R8 requirement constrains displaying of the messages in the course of chat.
- M1 – The message is typed.
- MD1 – The message is displayed.
- R9, R10 – Requirements for correspondence between by the particular portal member assigned/processed (accepted, rejected or suspended) task and for the other portal member displayed task. R8 requirement constrains displaying of the tasks in the tasks list.
- T1 – The task is assigned.
- T2 – The task is accepted.
- T3 – The task is rejected.
- T4 – The task is suspended.
- TD1 – Assigned/processed task is displayed.

Three Web portals have been chosen for the experiment:

- Developer Portal of Beunited organisation (http://www.beunited.org/),
- Indiana Learning Portal (http://www.ihets.org/progserv/education/ilportal/),
- HKU Portal of the University of Honkong (http://www.hku.hk/cc/).

The reasons for choosing these portals are the following:

- **Representativeness.** All of them conform to in Figure 28 represented multi-frame system.
- **Value.** They are up-to-date, popular enough among users, and their application area is important enough:
  - *Developer Portal* of Beunited organisation is a community that puts an emphasis on people-to-people communication. It is actively used by the users, because it can incorporate all types of people in the community, developers and end-users alike, rather than being solely developer-specific.
  - *Indiana Learning Portal* makes collaborative or community workspace available in Indiana and around the world, it provides access to corporate training, training programs, college classes and programs, professional development, public broadcasting, and other educational opportunities.
  - *HKU Portal* is designed to enable convenient and effective communications among University members, who can access on-line library resources, web-based email, on-line courses, personal records, administrative data, teaching and learning resources, research information and other on-line services provided by various departments with a single sign-in.
- **Relative sampling risk.** Developer Portal and Indiana Learning Portal have requirements specification documents [Req03], [ILP03]. HKU portal's user guide (http://www.hku.hk/cc/faq/portal/) and demo version is available on the Web.

5.4.2. Sampling Vocabulary

Sampling vocabulary that describes the main concepts of the chosen conceptualisation is presented in Table 9:

Table 9. Sampling vocabulary

Term	Definition	Equivalents		
		Developer Portal	Indiana Learning Portal	HKU Portal
Access rights	Permissions or privileges granted to portal members by portal administrator.			
Action	The action, which is done to the content item: "Check in" or "Check out".			

Term	Definition	Equivalents		
		Developer Portal	Indiana Learning Portal	HKU Portal
Administrator	The type of the role, which has possibility to access, create and modify all resources of the portal, i.e. to perform content, user, and search management.			
Advanced search	Search for the particular information by advanced search options, which allow limiting the scope of search. Advanced search allows making search more precise by using up to three search terms at once and combining search terms using Boolean operators AND, OR, NOT.		Targeted search	
Advanced search options	Option of combining a variety of terms to help to construct a more detailed search for particular portal content.			Guided search options
Asynchronous collaboration	Collaboration at different time (asynchronous).		Off-line collaboration	
Attachment	A computer file, which is sent along with messages in forums (newsgroups).			
Ban	Barring access to all portal chat rooms or to the particular chat room for a specific portal member either indefinitely or for a specific time period. Typically, a ban is levied by the Administrator as punishment for typing messages with unprintable words.			
Category	Hierarchical unit that function like directories (folders) and correspond to the Portal Information Architecture.	Section	Folder	Folder
Chat room group	A group of chat rooms, which are available for the particular role having users of the portal.			
Check in	Saving of created, modified or deleted piece of content. In case of creation/modification the new version is assigned to the content item.			
Check out	Reserving of the particular piece of content by the particular portal member for creation, modification or deletion.			
Content creator	The type of the role, which provides possibility to create content.			
Content item	Items, stored in the portal Content Repository.			
Content manager	The type of the role, which provides possibility to create and publish content, and approve or reject the content that the other portal members have created.	Administrator		Administrator
Content publisher	The type of the role, which provides possibility to publish content.			
Content repository	Facility that contains the content managed, displayed or just used by a content application such as a Content Management system.			
Date options	Options that allow filtering data according to dates (possibility to show all data or only the data that belongs to the particular interval of dates).		Date settings	
Document item	Structured pieces of content, typically stored in the form of files, such as .DOC or .PDF.			
Folder	A place in the portal Content Repository, where the results of simple or advanced search are saved.			Directory
Forum	Online collaboration through discussion		Discussion board	

Term	Definition	Equivalents		
		Developer Portal	Indiana Learning Portal	HKU Portal
	group, where portal members can exchange messages.			
Group task	A task that is assigned to the group of portal members.			
Individual task	A task that is assigned to individual portal member.			
Instant messaging	A form of real-time communication between two or more people based on typed text or by participation in voice or video conference.			
Item kind	The kind of library items: image, static text, banner or video.			
Item type	The type of content item: library item or document item.			
Library item	Reusable, unstructured pieces of content.			
Member	The type of the role, which is assigned to registered users of the portal. Members have access to all main resources of the portal.			
Member profile	A user account that contains user membership term and personal user information (such as name, e-mail, nickname, address, etc.).	Profile		
Message	A text that is typed by the particular member and displayed to all participants of the collaboration (forum, newsgroups, public chat room) or only to one participant of the collaboration (private chat room, instant messaging).			
Newsgroup	Collaboration through bulletin boards, where portal members can put and read messages on-line or download and save messages for off-line reading.		Discussion group	Bulletin board
Newsletter	Asynchronously distributed publication generally about one main topic that is of interest to its subscribers.			
Nickname	An alternate name someone uses or others use to refer to that person instead of using that person's real or complete name.			
Notification	E-mail indication that the particular event that took place in the portal requires member's attention.			
Ordering options	Options that allow ordering of search results in the particular way.		Sorting settings	
Poll	An inquiry into public opinion conducted by interviewing portal members.			
Private chat room	A virtual room, where two people can communicate pear-to-pear in real time while on the Internet.	Private chat	Private chat	
Public chat room	A virtual room, where people can communicate in real time while on the Internet.		Chat room	
Role	A relationship that person has to a project or portal. Usually a role has the set of access rights.	Activity		
Saved search	Simple or advanced search with possibility to save search results in folders, and later extract of saved search results from the particular folder.		Custom search	
Scope options	Options that allow limiting scope of search by providing the particular topics, which are relevant for the member.		Scope settings	

Term	Definition	Equivalents		
		Developer Portal	Indiana Learning Portal	HKU Portal
Search keywords	Keywords or key phrases that some search engines will use to search for particular portal content.			
Simple search	Search for the particular information by entered search keywords. It includes search using only one search term. The scope of simple search is „all portal content items“.			Basic search
Status	The status of the content item: “Approved” or “Not approved” (if approval is necessary).			
Synchronous collaboration	Collaboration in real-time (concurrently).	Real-time communication	On-line collaboration	
Task	A unit of work within a workflow. Workflows are composed of multiple tasks which can be executed serially, in parallel, or on a conditional basis. Examples of tasks include tasks that are assigned by project manager to in the particular projects participating developers, different learning tasks assigned to students by their tutors (lectors), etc.			
Task action	The action, which is done to the task: “Send for approval”, “Approve”, “Reject.”			
Task manager	Portal member, who assigns tasks to the other members and administers the processing of tasks (approves or rejects by portal members accepted, rejected or suspended tasks). Task managers in different portals are different (for example, project managers, tutors (lectors), etc.).	Project manager	Lector, tutor	Lector, tutor
Task priority	The importance of task (“High”, “Low”, “Medium”).			
Task status	The status of the task, which depends on the states of the processed task: „Accepted“, „Rejected”, „Suspended”, „Completed”.			
Topic	The subject matter of a conversation or discussion, which takes place in forum or newsgroup, or the subject of to portal members distributed newsletter.			
Topic post	Topics and their replies.	Post		Post
Topic reply	Reply to the particular topic.			
Topic subject	The name of the topic.			
Visitor	Not registered user of the portal.			
Workflow	Automation of a business process, in whole or part, during which documents, information or tasks are passed from one member to another for action, according to a set of procedural rules.			

5.4.3. Sampling Questionnaire

5.4.3.1. Data Requirements

Data in the sampling questionnaire should meet the following requirements:

1) Data validity requirements:

- Questionnaire data should express which components (Figure 28) are mandatory for the particular portal and which are optional and, thus, are not present in that portal.

- Questionnaire data should show how the main components of the particular portal interact one with the other.
- Questionnaire should contain questions about the features of the main components of the particular portal.
- Questionnaire data should show interactions and dependencies between different features of the particular portal.
- Questionnaire data should express which features are mandatory for the particular portal and which are optional and, thus, are not present in that portal.

2) Data reliability requirements:
   - Questionnaire should not be too large; it should be as short as possible – only essential questions should be included into it.
   - If the question has no any answering conditions, then it should be answered.
   - If it is explicitly stated in the question that it should be answered only if the particular condition holds (for example, if answer to the previous question is „Yes"), then it should not be required to have answers to such questions.
   - In order to avoid inaccurate answers because of invalid interpretation of questions, all the questions should be as unambiguous as possible.
   - Questions should not have technical terms or acronyms that are not defined in sampling vocabulary.
   - Every question should be formulated in such a way that it should cover all the possible answers.
   - In order to make questions easier to answer questions on the same topic should be grouped together.
3) Data consistency requirements:
   - For every question the type of answer should be defined: multiple choice, numeric answer, free text answer.
   - Every multiple choice answer should use the particular rating scale, agreement scale or enumeration of possible values.
   - If the particular question should be answered using the particular rating or agreement scale, then only one rating or agreement value from the available set of values should be chosen.
   - If the particular question should be answered using enumeration of possible values, then one or several possible values may be chosen.

- If the particular question should be answered using free text, then the entered text should not have terms or acronyms that are not defined in sampling vocabulary.
- If the particular question should be answered using numerical value, then this value should be meaningful (positive and not too large or too small).

4) Data accuracy requirements:
   - Questions should be ordered according to necessity of portal components: first the questions for mandatory components, then the questions for optional components.
   - Every question should be formulated in such a way that is should accurately express what feature (es) of the portal it aims to clarify and to what portal component this feature belongs.
   - If the particular question should be answered using enumeration of possible values, then the set of values should be finite (there should be no values as „others", „etc.", and so on).
5) Data completeness requirements:
   - Questions should cover all the main mandatory and optional components of the portal.
   - For every portal component it should be possible from questions and answers to them to clarify all its mandatory and optional features.
6) Level of detail requirement: Questionnaire data should have the appropriate level of detail, which, on the one hand, should be high enough to include all the essential features of the portal, and, from the other hand, not too high, because according to data reliability requirements too large questionnaire is the source of errors and inaccuracies

5.4.3.2. Questionnaire

For Web portal developed sampling questionnaire that meets data validity, data reliability, data consistency, data accuracy, data completeness, and level of detail requirements, is presented in Table 10:

Table 10. Sampling questionnaire

Nr.	Question	Answer
1.	What does content management include?	• Creation of portal content • Publishing of portal content • Deletion of portal content • Changing of portal content • Filtering of portal content • Searching for portal content
2.	What are the main roles that are defined for portal members to perform content management?	• Content manager • Content creator • Content publisher
3.	Should every role that is defined for content management have its own set of access rights?	• Yes • No

Nr.	Question	Answer
4.	Is content approval procedure used in the portal?	• Yes • No
5.	What are the access rights of the content manager?	• Create content • Publish content • Approve or reject by content creators created content
6.	For which roles approval of created content before its publishing is necessary?	• Content creators • Content publishers
7.	What are the access rights of the content creator?	• Only to create content • Create and publish content
8.	What are the access rights of the content publisher?	• Create content • Publish content.
9.	What kind of items is it possible to create?	• Library items • Document items
10.	What kind of library items is it possible to create?	• Images • Static text • Banners • Video
11.	Does portal has other kinds of library items?	• No • Yes
12.	If the answer to question 11 is "Yes", then what kinds of other library items are available?	
13.	In which formats is it possible to store document items?	• .DOC • .PDF
14.	Does Portal Information Architecture is organised in the form of the hierarchy of categories, in which library and document items are saved?	• Yes • No
15.	Is check in-check out procedure used in the portal?	• Yes • No
16.	Is it possible to change the items that are checked out?	• Yes • No, only one user should be able to work with the particular content item at the same time
17.	What are the attributes of every created item?	• Menu item • Item type • Item kind • Item name • Size (Kb) • Last modification date • Item author name
18.	Does portal has other attributes of created content items?	• No • Yes
19.	If the answer to question 18 is "Yes", then what other attributes of content items are available?	
20.	Which information is it necessary to indicate during creation of the particular content item?	• Menu item • Item type • Item kind
21.	Which information is generated automatically during creation of the particular content item?	• Item name • Size (Kb) • Last modification date • Item author name
Questions 22-24 should be answered only if answer to question 15 is "Yes".		
22.	Which information should be available if check in-check out procedure is used?	• Version of the item • Item action
23.	Does the version of content item changes,	• Yes

Nr.	Question	Answer
	when the item is checked in?	• No
24.	What are the available actions?	• Check in • Check out
Questions 25-26 should be answered only if answer to question 4 is "Yes".		
25.	Which information should be available if approval procedure is used?	• Item status • Other information
26.	What are the available statuses of items?	• Approved • Not approved
27.	Is it possible for content manager to change/delete all portal items?	• Yes • No, content manager can delete/change only the items that he has created
28.	Is it possible for content creator/content publisher to change/delete all portal items?	• Yes • No, content creator/content publisher can delete/change only the items that he has created
29.	For which roles is it possible to search for content items?	• For content manager only • For content manager and content publisher • For content manager, content creator and content publisher
30.	What kinds of content search is it possible to perform?	• Simple search • Advanced search
31.	What are the possible search conditions?	• Search keywords • Advanced search options
32.	For which roles is it possible to filter published content items?	• For content manager only • For content manager and content publisher
33.	By which criteria is it possible to filter published content items?	• Item type • Item kind • Last modification date • Version of the item
34.	For which roles is it possible to preview published content items?	• For content manager only • For content manager and content publisher
35.	What are the main roles that are defined for portal users to perform search?	• Visitor • Member
36.	Should every role that is defined for search have its own set of access rights?	• Yes • No
37.	What are the access rights to perform search for a Visitor?	• Perform simple search • Perform advanced search
38.	What are the access rights to perform search for a Member?	• Perform simple search • Perform advanced search
39.	What are the search options for simple search?	• Search keywords • Other
40.	If the answer to question 39 is "Other", then what kind of other simple search options are available?	
41.	How is it possible to limit the scope of simple search?	• By entering another search keywords • In another way
42.	If the answer to question 42 is "In another way", then what is the other way of limiting simple search scope?	
43.	What kinds of options are provided by the advanced search?	• Date options • Ordering options • Scope options
44.	Does portal has other kinds of search options?	• No • Yes
45.	If the answer to question 44 is "Yes", then what kinds of other search options are available?	
46.	How is it possible to limit the scope of	• By entering another advanced search options

Nr.	Question	Answer
	advanced search?	• In another way
47.	If the answer to question 46 is "In another way", then what is the other way of limiting advanced search scope?	
48.	How saved search results are saved?	• There is no possibility to save search results • Search results are saved in folders (directories)
49.	Is the number of possible to create folders limited?	• Yes, it is possible to create no more than **N** folders • No, member can create as much folders as he wishes
50.	Is it possible to create folders with the same names?	• No, folder name should be unique • Yes
51.	Is it possible to create any number of folders	• Yes • No, the number is restricted
52.	Is it possible to save both simple search results and advanced search results?	• No, only simple search results • No, only advanced search results • Yes, both types of results
53.	Are the results of simple search and the results of advanced search saved in separate folders?	• Yes, there are simple search folders and advanced search folders • No, results of different kinds of search can be saved in one folder
54.	Is it possible to run saved search?	• Yes • No
55.	Does the member have possibility to view/delete only the folders that he has created or also the folders of the other members?	• Yes, the member has possibility to view/delete only the folders that he has created • No, the member can view/delete the folders of the other members
56.	Does portal provide collaboration service?	• Yes • No
Questions 57-90 should be answered only if answer to question 56 is "Yes".		
57.	What kinds of collaboration are available in the portal?	• Synchronous collaboration • Asynchronous collaboration
58.	What are the main roles that are defined for portal users to collaborate?	• Visitor • Member
59.	Should every role that is defined for collaboration have its own set of access rights?	• Yes • No
60.	What are the access rights to collaborate for a Visitor?	• To collaborate synchronously • To collaborate asynchronously
61.	What are the access rights to collaborate for a Member?	• To collaborate synchronously • To collaborate asynchronously
62.	If in question 57 "Asynchronous collaboration" is marked, then what kind of asynchronous collaboration means does the portal have?	• Forum • Newsgroup
63.	Is newsletter available in the portal?	• Yes • No
Questions 64-79 should be answered only if in question 62 "Forum" is marked.		
64.	Is it possible to view the profile information of forum participants?	• Yes, only for members of the portal • No
65.	How is it possible to filter topic posts?	• View the posts in selected forum • View the posts in selected topic
66.	What possibilities to work with topics are provided for portal member?	• Creation of topics • Posting replies to topics • Deletion of topics/replies to topics
67.	What information is it necessary to indicate	• Subject

Nr.	Question	Answer
	during creation of the particular topic?	• Message • Notification type
68.	What information is optional during creation of topic?	• Poll information • Attachment • Topic options
69.	Which information is generated automatically during creation of the particular topic?	• Topic author • Topic post date and time
70.	Is the number of poll options restricted?	• Yes • No
71.	If in question 68 "Poll information" is marked, then what do the results of voting process include?	• Total number of votes • The list of poll options with corresponding percents of votes
72.	What information is it necessary to indicate during creation of topic reply?	• Subject • Message • Notification type
73.	What information is optional during creation of topic reply?	• Attachment • Reply options
74.	Which information is generated automatically during creation of the particular topic reply?	• Topic reply author • Topic reply post date and time
75.	Is the length of topic/topic reply message restricted?	• Yes • No
76.	What kinds of notifications can be sent to topic author and to other topic participants (who have posted replies to the topic)?	• Notification about new topic reply to topic author • Notification about new topic reply to other topic participants
77.	Is it possible for members to change/delete all topics/topic replies?	• Yes • No, content manager can delete/change only the topics/topic replies that he has created
78.	Are all topic replies deleted automatically after deletion of topic?	• Yes • No
79.	If in question 57 "Synchronous collaboration" is marked, then what kind of synchronous collaboration means does the portal have?	• Chat rooms • Instant messaging
Questions 80-90 should be answered only if in question 79 "Chat rooms" is marked.		
80.	What are the possible kinds of chat rooms?	• Public chat rooms • Private chat rooms
81.	Is access to chat rooms restricted in some way?	• Yes, visitors are allowed to enter limited set of chat rooms • No, both members and visitors can enter any chat rooms
82.	Is single sign-in principle used?	• Yes, for members • No, members as visitors should enter their nickname
83.	How visitors enter chat rooms?	• Enter nickname before entering every chat room • Enter nickname one time, before entering the first chat room
84.	Is the number of chat room participants limited?	• Yes • No
85.	What kind of information is it possible to send in private chat rooms?	• Only messages (typed text) • Only files • Files and messages
86.	Is it possible to view the profile information of chat room participants?	• Yes, only for members of the portal • No
87.	If in question 80 "Private chat room" is marked, then what kind of information is it possible to send in public chat rooms?	• Only messages (typed text) • Only files • Files and messages
88.	Is the number of at one time opened public or	• Yes

Nr.	Question	Answer
	private chat rooms limited?	• No
89.	Is it possible to type messages with unprintable words in public or private chat room?	• Yes • No, the member, who did so, will be banned
90.	Is for member set ban unset automatically, when the particular period of time ends?	• Yes • No, portal Administrator should unset the ban
91.	Does portal provide workflow management service?	• Yes • No
Questions 92-120 should be answered only if answer to question 91 is "Yes".		
92.	What are the main services provided by the workflow management?	• Creation and assignment of tasks • Viewing of tasks • Processing of tasks • Changing of tasks • Deletion of tasks • Approval or rejection of tasks • Filtering and grouping of tasks • Searching for tasks • Viewing the history of task modifications
93.	What are the main roles that are defined for portal users to participate in workflow?	• Workflow manager • Member
94.	Should every role that is defined for workflow have its own set of access rights?	• Yes • No
95.	What are the access rights to participate in workflow for a workflow manager?	• To create and assign/reassign tasks to portal members • To view tasks • To change tasks • To delete tasks • To approve or reject tasks • To filter and group tasks • To search for tasks • To view the history of task modifications
96.	What are the access rights to participate in workflow for a member?	• To view assigned tasks • To process assigned tasks
97.	How workflow manager can assign tasks to portal members?	• To the group of portal members • To the particular portal member
98.	Is it possible for workflow manager to view all tasks?	• Yes, workflow manager has possibility to view all tasks that he has assigned to portal members • No
99.	Is it possible for portal member to view all tasks?	• Yes, the member can view tasks of the other members • No, the member has possibility to view only to him assigned tasks
100.	What are the main attributes of the assigned tasks?	• Task name • Resource name (name of the member or member group the task is assigned to) • Task assignment date • Task description • Task type (group or individual task) • Task priority • Task last modification date • Task status • Task action • Task completion date • Information date • Task deletion date • Comment

Nr.	Question	Answer
101.	Does portal has other attributes of assigned tasks?	• No • Yes
102.	If the answer to question 101 is "Yes", then what other attributes of assigned tasks are available?	
103.	Which information is it necessary to indicate during creation and assignment of the particular task?	• Task name • Resource name (name of the member or member group the task is assigned to) • Task assignment date • Task description • Task type (group or individual task) • Task priority • Task last modification date
104.	Which task attributes are filled during processing and approval of tasks?	• Task status • Task action
105.	What are the possible statuses of processed tasks?	• Accepted • Rejected • Suspended • Completed
106.	What are the possible dependencies between statuses of tasks?	• Suspended tasks later can be accepted or rejected • Accepted or rejected tasks become completed • Other dependency
107.	If in question 106 "Other dependency" is marked, then what kind of dependency is it?	
108.	What are the possible actions for processed tasks?	• Check in • Check out • Send for approval • Approve • Reject
109.	When is the action "Check out" set?	• When member is viewing or processing the task • When member sets the particular task status
110.	When is the action "Check in" set?	• When member is viewing or processing the task • When member sets the particular task status
111.	When is the action "Send for approval" set?	• After member sets the status of the particular task to "Accepted" or "Rejected" • After member sets the status of the particular task to "Suspended"
112.	When is the action "Approve" set?	• When workflow manager approves the particular task • When workflow manager rejects the particular task
113.	When is the action "Reject" set?	• When workflow manager approves the particular task • When workflow manager rejects the particular task
114.	Is it possible to assign by workflow manager rejected task to the other portal member or member group?	• Yes • No
115.	Is it possible to change/delete all tasks?	• Yes, but only workflow manager can change/delete all tasks that he has assigned • No
116.	Do several workflow managers can change/delete the same task at the same time?	• Yes • No, only one workflow manager is able to work with the particular task at the same time
117.	When are notifications to workflow manager sent?	• After approval or rejection of tasks by portal members

Nr.	Question	Answer
		• After suspending of tasks by portal members
118.	When are notifications to portal members sent?	• After workflow manager accepts or rejects tasks, which were sent to him for approval • After workflow manager deletes/changes the tasks
119.	By which criteria is it possible to filter tasks?	• Member group • Particular member • Task status • Deleted tasks • Tasks pending manager's approval (action "Send for approval") • Overdue tasks • Completed tasks
120.	What does the history of task modification include?	• Date and time (task deletion date in case of task deletion, information date in case of task creation, last modification date in case of task modification) • Member (resource name), who have changed the status of task • Was the task approved or moved on in the workflow (reassigned to the other member) • Was the task suspended for later processing

In Table 10 presented questionnaire has been used to analyse the features all for experiment chosen Web portals.

5.4.4. Feature Models

5.4.4.1. Feature Model for Developer Portal

Feature model for Developer Portal is presented in tabular form (Table 11):

Table 11. Developer Portal feature table

Level	Feature	Description	Rationale	Type	Composition rules
1	**Content management feature**	**A service that allows creation, management, and publishing of portal content**		**Mandatory**	
1.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
1.1.1	**Content manager role**	**A service that allows management of portal content**		**Mandatory**	
1.1.1.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.1.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.1.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.1.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.1.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content creation*
1.1.1.2.1	Item filtering	A service that allows portal members to filter published		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
		items by the particular filtering criteria.			
1.1.1.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.1.1.3	*Content approval*	*A service that allows content manager to approve item creation, changing or deletion, done by portal members.*		*Mandatory*	
1.1.1.4	*Content search*	*A service that allows content manager to search for necessary content items*		*Mandatory*	
1.1.1.4.1	Simple search	A service that includes search for content items by entered search keywords.		Mandatory	
1.1.2	**Content creator role**	**A service that allows creation of portal content**		**Mandatory**	
1.1.2.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.2.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.2.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.2.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3	**Content publisher role**	**A service that allows publishing of portal content**		**Mandatory**	
1.1.3.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.3.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.3.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.3.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content approval*
1.1.3.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.3.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.2	**Category**	**A service that allows storing of portal content items in hierarchical categories**		**Mandatory**	
1.2.1	Item	A service that allows storing of content in the form of content units.		Mandatory	
1.2.1.1	Library item	A service that allows creation of reusable, unstructured pieces of content.	If it is necessary to store parts of documents for re-use	Optional	
1.2.1.1.1	Image	A service that allows creation of a visual representation of an object, scene, person, abstraction, etc. produced on a Web	If images should be reused in many documents	Optional	

Level	Feature	Description	Rationale	Type	Composition rules
		page.			
1.2.1.1.2	Static text	A service that allows creation of a text that doesn't change its position on a Web page.	If static text should be reused in many documents	Optional	
1.2.1.2	Item name	A service that allows assignment of names to content items.		Mandatory	
1.2.1.3	Menu item	A service that allows identification of portal menu item, to which the particular content item belongs.		Mandatory	
1.2.1.4	Last modification date	A service that allows identification of the date of last content item modification.		Mandatory	
1.2.1.5	Version of the item	A service that allows generation of content item version.	If it is necessary to version content items.	Optional	
1.2.1.6	Action	A service that allows performing items check in-check out procedure.	If check in-check out procedure is used.	Optional	
1.2.1.6.1	Check in	A service that allows making changes to the particular content item and automatically generate its new version.		Optional	
1.2.1.6.2	Check out	A service that allows reserving of the particular content item by the particular user (the other users can not work with that item).		Optional	
1.2.1.7	Status	A service that allows management of content items publishing.	If the procedure of content items approval is used.	Optional	
1.2.1.7.1	Approved	A service that allows approval of publishing of content items.		Optional	
1.2.1.7.2	Not approved	A service that allows disapproval of publishing of content items.		Optional	
1.2.1.8	Item author name	A service that allows identification of the name of the user, who has created the particular content item.	If it is necessary to have the name of the user, who has created the particular content item.	Optional	
2	**Search feature**	**A service that allows finding for necessary information by entering search keywords or search options.**		**Mandatory**	
2.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
2.1.1	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
2.1.1.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.1.1.1	Search keywords	A service that allows performing search for portal content by the particular		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
		keywords or key phrases.			
2.1.2	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
2.1.2.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.2.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2.2	*Advanced search*	*A service that includes search by advanced search options, which allow filtering and/or ordering of search results.*		*Mandatory*	
2.1.2.2.1	Date options	A service that allows filtering of search results according to the interval of dates.	If it is necessary to filter search results according to the interval of dates.	Optional	
2.1.2.2.2	Ordering options	A service that allows ordering of search results in the particular way.	If it is necessary to order search results in the particular way.	Optional	
2.1.2.3	*Saved search*	*A service that allows saving search results in folders, and later extract of saved search results from the particular folder.*		*Mandatory*	
2.1.2.3.1	Advanced search folder	A service that allows saving of advanced search results in the particular folder.		Mandatory	Requires: Advanced search results
2.2	**Search results**	**A service that allows viewing the list of simple or advanced search results.**		**Mandatory**	
2.2.1	*Simple search results*	*A service that allows viewing the list of simple search results.*		*Mandatory*	*Requires: Simple search*
2.2.2	*Advanced search results*	*A service that allows viewing the list of advanced search results.*		*Mandatory*	*Requires: Advanced search*
3	**Collaboration feature**	**An online service that provides means of communication between portal users.**		**Mandatory**	
3.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
3.1.1	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
3.1.1.1	*Asynchronous collaboration*	*A service that allows collaboration at different (asynchronous) time.*		*Mandatory*	
3.1.1.1.1	Forum	A service that provides online collaboration through discussion group, where users can exchange messages.		Mandatory	Requires: Message
3.1.1.1.1.1	Topic post	A service that allows posting topics and replies to them.		Mandatory	
3.1.1.1.1.1.1	Topic	A service that allows posting topics.		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
3.1.1.1.1.1.1.1	Subject	A service that allows naming of topics.		Mandatory	
3.1.1.1.1.1.1.2	Message	A service that allows creation of messages that could be seen to the participants of not-real time conversation.		Mandatory	
3.1.1.1.1.1.1.3	Notification type	A service that allows determining if e-mail notifications should be sent to the author of the topic, when replies to it are posted.		Mandatory	
3.1.1.1.1.1.1.4	Topic author	A service that allows identification of the name of the user, who has created the particular topic.		Mandatory	
3.1.1.1.1.1.1.5	Topic post date and time	A service that allows identification of the date and time of topic creation.		Mandatory	
3.1.1.2.1.1.1.6	Attachment	A service that allows sending computer files along with messages.	If it is necessary to send a computer file.	Optional	
3.1.1.1.1.1.2	Topic reply	A service that allows posting replies to the particular topic.		Mandatory	
3.1.1.1.1.1.2.1	Subject	A service that allows naming of topic replies.		Mandatory	
3.1.1.1.1.1.2.2	Message	A service that allows creation of messages that could be seen to the participants of not-real time conversation.		Mandatory	
3.1.1.1.1.1.2.3	Notification type	A service that allows determining if e-mail notifications should be sent to the author of the topic, when replies to it are posted.		Mandatory	
3.1.1.1.1.1.2.4	Reply author	A service that allows identification of the name of the user, who has created the particular reply to topic.		Mandatory	
3.1.1.1.1.1.2.5	Attachment	A service that allows sending computer files along with messages.	If it is necessary to send a computer file.	Optional	
4	**Workflow management feature**	**A service that includes assignment of tasks, tasks management and processing.**	**If automated management of tasks is required.**	**Optional**	
4.1	**Role**	**A service that has the set of access rights to participate in portal workflow.**		**Mandatory**	
4.1.1	**Task manager role**	**A service that has the set of access rights for task manager to participate in portal workflow.**		**Mandatory**	
4.1.1.1	Task creation	A service that allows creation of tasks and their assignment to portal members.		Mandatory	
4.1.1.2	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.3	Task changing	A service that allows changing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.4	Task deletion	A service that allows deletion of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.5	Task approval	A service that allows approval of by portal members processed tasks.		Mandatory	Requires: Task processing

Level	Feature	Description	Rationale	Type	Composition rules
4.1.2	**Member role**	**A service that has the set of access rights for portal member to participate in portal workflow.**		**Mandatory**	
4.1.2.1	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.2.2	Task processing	A service that allows processing (acceptation, rejection, suspending, etc.) of to the member assigned tasks.		Mandatory	
4.1.2.2.1	Task acceptance	A service that allows acceptance of by the member fulfilled tasks.	If task needs to be accepted.	Optional	
4.1.2.2.2	Task rejection	A service that allows rejection of tasks, which the member was unable to fulfil.	If task needs to be rejected.	Optional	
4.1.2.2.3	Task suspending	A service that allows suspending of tasks, which the member could not fulfil because some kind of delay.	If task needs to be suspended.	Optional	
4.2	**Task**	**A service that allows storing of to portal members assigned tasks.**		**Mandatory**	
4.2.1	Task name	A service that allows registration of task name.		Mandatory	
4.2.2	Resource name	A service that allows registration of member or member group name the task is assigned to.		Mandatory	
4.2.3	Task assignment date	A service that allows registration of task assignment date.		Mandatory	
4.2.4	Task description	A service that allows registration of task description.		Mandatory	
4.2.5	Task priority	A service that allows registration of task priority (importance of the particular task).	If importance of tasks is required.	Optional	
4.2.6	Task last modification date	A service that allows registration of task last modification date.		Mandatory	Requires: Task creation, Task changing
4.2.7	Task status	A service that allows setting of task status (“Accepted”, “Rejected”, “Suspended”, “Completed”).	If task is processed.	Mandatory	
4.2.8	Task action	A service that allows setting of task action (“Check out”, “Check in”, “Send for approval”, “Approve”, “Reject”)		Mandatory	
4.2.9	Task completion date	A service that allows registration of task completion date.	If task is completed.	Mandatory	Requires: Task status
4.2.10	Information date	A service that allows identification of task creation date.		Mandatory	Requires: Task creation
4.2.11	Task deletion date	A service that allows registration of task deletion date.	If task is deleted.	Mandatory	Requires: Task deletion
4.2.12	Comment	A service that allows registration of the comment about the particular task.	If task is rejected or suspended by the portal member.	Mandatory	Requires: Task status

5.4.4.2. Feature Model for Indiana Learning Portal

Feature model for Indiana Learning Portal is presented in tabular form (Table 12):

Table 12. Indiana Learning Portal feature table

Level	Feature	Description	Rationale	Type	Composition rules
1	**Content management feature**	**A service that allows creation, management, and publishing of portal content**		**Mandatory**	
1.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
1.1.1	**Content manager role**	**A service that allows management of portal content**		**Mandatory**	
1.1.1.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.1.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.1.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.1.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.1.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content creation*
1.1.1.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.1.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.1.1.3	*Content approval*	*A service that allows content manager to approve item creation, changing or deletion, done by portal members.*		*Mandatory*	
1.1.1.4	*Content search*	*A service that allows content manager to search for necessary content items*		*Mandatory*	
1.1.1.4.1	Simple search	A service that includes search for content items by entered search keywords.		Mandatory	
1.1.1.4.2	Advanced search	A service that includes search for content items by advanced search options, which allow filtering and/or ordering of search results.		Mandatory	
1.1.2	**Content creator role**	**A service that allows creation of portal content**		**Mandatory**	
1.1.2.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.2.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.2.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.2.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3	**Content publisher role**	**A service that allows publishing of portal**		**Mandatory**	

Level	Feature	Description	Rationale	Type	Composition rules
		content			
1.1.3.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.3.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.3.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.3.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content approval*
1.1.3.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.3.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.2	**Category**	**A service that allows storing of portal content items in hierarchical categories**		**Mandatory**	
1.2.1	Item	A service that allows storing of content in the form of content units.		Mandatory	
1.2.1.1	Library item	A service that allows creation of reusable, unstructured pieces of content.	If it is necessary to store parts of documents for re-use	Optional	
1.2.1.1.1	Image	A service that allows creation of a visual representation of an object, scene, person, abstraction, etc. produced on a Web page.	If images should be reused in many documents	Optional	
1.2.1.1.2	Static text	A service that allows creation of a text that doesn't change its position on a Web page.	If static text should be reused in many documents	Optional	
1.2.1.1.3	Banner	A service that allows creation of graphic image (static, animated, or rich media) that is used for the purpose of advertisement.	If banners should be reused in many documents	Optional	
1.2.1.1.4	Video	A service that allows creation of video (e.g. video clip).	If videos should be reused in many documents	Optional	
1.2.1.2	Document item	A service that allows creation of structured pieces of content, typically stored in the form of files (.DOC, .PDF, etc.).	If it is necessary to have documents for re-use	Optional	
1.2.1.2.1	File .DOC	A service that allows creation of structured pieces of content, typically stored in the form of .DOC files.	If it is necessary to store document items in .DOC files	Optional	
1.2.1.2.2	File .PDF	A service that allows creation of structured pieces of content, typically stored in the form of .PDF files.	If it is necessary to store document items in .PDF files	Optional	
1.2.1.3	Item name	A service that allows assignment of names to content items.		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
1.2.1.4	Size	A service that allows identification of the size of content items.	If it is necessary to have the size of files	Optional	
1.2.1.5	Menu item	A service that allows identification of portal menu item, to which the particular content item belongs.		Mandatory	
1.2.1.6	Last modification date	A service that allows identification of the date of last content item modification.		Mandatory	
1.2.1.7	Version of the item	A service that allows generation of content item version.	If it is necessary to version content items.	Optional	
1.2.1.8	Item author name	A service that allows identification of the name of the user, who has created the particular content item.	If it is necessary to have the name of the user, who has created the particular content item.	Optional	
2	**Search feature**	**A service that allows finding for necessary information by entering search keywords or search options.**		**Mandatory**	
2.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
2.1.1	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
2.1.1.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.1.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
2.1.2.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.2.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2.2	*Advanced search*	*A service that includes search by advanced search options, which allow filtering and/or ordering of search results.*	*If it is necessary to filter/order search results.*	*Mandatory*	
2.1.2.2.1	Scope options	A service that allows limiting scope of search by providing the particular topics, which are relevant for the member.	If it is necessary to limit the scope of search.	Optional	
2.1.2.3	*Saved search*	*A service that allows saving search results in folders, and later extract of saved search results from the particular folder.*	*If it is necessary to save search results for later review.*	*Mandatory*	

Level	Feature	Description	Rationale	Type	Composition rules
2.1.2.3.1	Simple search folder	A service that allows saving of simple search results in the particular folder.	If it is necessary to save simple search results for later review.	Optional	Requires: Simple search results
2.1.2.3.2	Advanced search folder	A service that allows saving of advanced search results in the particular folder.	If it is necessary to save advanced search results for later review.	Optional	Requires: Advanced search results
2.2	**Search results**	**A service that allows viewing the list of simple or advanced search results.**		**Mandatory**	
2.2.1	*Simple search results*	*A service that allows viewing the list of simple search results.*		*Mandatory*	*Requires: Simple search*
2.2.2	*Advanced search results*	*A service that allows viewing the list of advanced search results.*		*Mandatory*	*Requires: Advanced search*
3	**Collaboration feature**	**An online service that provides means of communication between portal users.**		**Mandatory**	
3.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
3.1.1	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
3.1.1.1	*Synchronous collaboration*	*A service that allows collaboration in real-time (concurrently).*		*Mandatory*	
3.1.1.1.1	Chat room	A service that allows real-time communication through Internet by exchange of messages.		Mandatory	
3.1.1.1.1.1	Public chat room	A service that allows communication in real time while on the Internet.		Mandatory	
3.1.1.1.1.1.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.1.1.2	Private chat room	A service that allows communication pear-to-pear in real time while on the Internet.		Mandatory	
3.1.1.1.1.2.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.1.1.2.2	File	A service that allows exchange of files between participants of pear-to-pear conversation.	If files sending/receiving is required.	Optional	
3.1.1.1.2	Instant messaging	A service that allows real-time communication through Internet by typing messages or by participation in on-line voice or video conference	If real-time communication by participation in on-line voice or video conference is required.	Optional	
3.1.1.2	*Asynchronous collaboration*	*A service that allows collaboration at different (asynchronous) time.*		*Mandatory*	
3.1.1.2.1	Newsgroup	A service that provides collaboration through		Mandatory	Requires: Message

Level	Feature	Description	Rationale	Type	Composition rules
		bulletin boards, where users can put and read messages on-line or download and save messages for off-line reading.			
3.1.2	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
3.1.2.1	*Synchronous collaboration*	*A service that allows collaboration in real-time (concurrently).*		*Mandatory*	
3.1.1.2.1	Chat room	A service that allows real-time communication through Internet by exchange of messages.		Mandatory	
3.1.1.2.1.1	Public chat room	A service that allows communication in real time while on the Internet.		Mandatory	
3.1.1.2.1.1.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.2.1.2	Private chat room	A service that allows communication pear-to-pear in real time while on the Internet.		Mandatory	
3.1.1.2.1.2.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.2.1.2.2	File	A service that allows exchange of files between participants of pear-to-pear conversation.	If files sending/receiving is required.	Optional	
3.1.1.2.2	Instant messaging	A service that allows real-time communication through Internet by typing messages or by participation in on-line voice or video conference	If real-time communication by participation in on-line voice or video conference is required.	Optional	
4	**Workflow management feature**	**A service that includes assignment of tasks, tasks management and processing.**	**If automated management of tasks is required.**	**Optional**	
4.1	**Role**	**A service that has the set of access rights to participate in portal workflow.**		**Mandatory**	
4.1.1	**Task manager role**	**A service that has the set of access rights for task manager to participate in portal workflow.**		**Mandatory**	
4.1.1.1	Task creation	A service that allows creation of tasks and their assignment to portal members.		Mandatory	
4.1.1.2	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.3	Task changing	A service that allows changing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.4	Task deletion	A service that allows deletion of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.5	Task approval	A service that allows approval of by portal members processed tasks.		Mandatory	Requires: Task processing

Level	Feature	Description	Rationale	Type	Composition rules
4.1.1.6	Task analysis	A service that allows analysis of tasks, including filtering, grouping of tasks, search for tasks, viewing the history of tasks.	If it is necessary to analyse tasks.	Optional	
4.1.1.6.1	Task filtering	A service that allows filtering of tasks by the particular filtering options.	If filtering of tasks is required.	Optional	
4.1.1.6.2	Task grouping	A service that allows grouping of tasks by the particular grouping options.	If grouping of tasks is required.	Optional	
4.1.1.6.3	Task search	A service that allows search for tasks by the particular search options.	If search for tasks is required.	Optional	
4.1.1.6.4	Task history viewing	A service that allows viewing of task modification history.	If viewing of task history is required.	Optional	
4.1.2	**Member role**	**A service that has the set of access rights for portal member to participate in portal workflow.**		**Mandatory**	
4.1.2.1	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.2.2	Task processing	A service that allows processing (acceptation, rejection, suspending, etc.) of to the member assigned tasks.		Mandatory	
4.1.2.2.1	Task acceptance	A service that allows acceptance of by the member fulfilled tasks.	If task needs to be accepted.	Optional	
4.1.2.2.2	Task rejection	A service that allows rejection of tasks, which the member was unable to fulfil.	If task needs to be rejected.	Optional	
4.1.2.2.3	Task suspending	A service that allows suspending of tasks, which the member could not fulfil because some kind of delay.	If task needs to be suspended.	Optional	
4.2	**Task**	**A service that allows storing of to portal members assigned tasks.**		**Mandatory**	
4.2.1	Task name	A service that allows registration of task name.		Mandatory	
4.2.2	Resource name	A service that allows registration of member or member group name the task is assigned to.		Mandatory	
4.2.3	Task assignment date	A service that allows registration of task assignment date.		Mandatory	
4.2.4	Task description	A service that allows registration of task description.		Mandatory	
4.2.5	Task type	A service that allows assignment of individual or group tasks.		Optional	
4.2.5.1	Group task	A service that allows assignment of tasks to the particular group of portal members.	If it is required to assign the same task to the group of members.	Optional	
4.2.5.2	Individual task	A service that allows assignment of tasks to the particular member.	If it is required to assign the task to the particular member personally.	Optional	

Level	Feature	Description	Rationale	Type	Composition rules
4.2.6	Task last modification date	A service that allows registration of task last modification date.		Mandatory	Requires: Task creation, Task changing
4.2.7	Task status	A service that allows setting of task status (“Accepted”, “Rejected”, “Suspended”, “Completed”).	If task is processed.	Mandatory	
4.2.8	Task action	A service that allows setting of task action (“Check out”, “Check in”, “Send for approval”, “Approve”, “Reject”)		Mandatory	
4.2.9	Task completion date	A service that allows registration of task completion date.	If task is completed.	Mandatory	Requires: Task status
4.2.10	Information date	A service that allows identification of task creation date.		Mandatory	Requires: Task creation
4.2.11	Task deletion date	A service that allows registration of task deletion date.	If task is deleted.	Mandatory	Requires: Task deletion
4.2.13	Comment	A service that allows registration of the comment about the particular task.	If task is rejected or suspended by the portal member.	Mandatory	Requires: Task status

5.4.4.3. Feature Model for HKU Portal

Feature model for HKU Portal is presented in tabular form (Table 13):

Table 13. HKU Portal feature table

Level	Feature	Description	Rationale	Type	Composition rules
1	**Content management feature**	**A service that allows creation, management, and publishing of portal content**		**Mandatory**	
1.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
1.1.1	**Content manager role**	**A service that allows management of portal content**		**Mandatory**	
1.1.1.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.1.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.1.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.1.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.1.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content creation*
1.1.1.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.1.2.2	Item preview	A service that allows preview of published content		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
		items			
1.1.1.3	*Content approval*	*A service that allows content manager to approve item creation, changing or deletion, done by portal members.*		*Mandatory*	
1.1.1.4	*Content search*	*A service that allows content manager to search for necessary content items*		*Mandatory*	
1.1.1.4.1	Simple search	A service that includes search for content items by entered search keywords.		Mandatory	
1.1.1.4.2	Advanced search	A service that includes search for content items by advanced search options, which allow filtering and/or ordering of search results.	.	Mandatory	
1.1.2	**Content creator role**	**A service that allows creation of portal content**		**Mandatory**	
1.1.2.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.2.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.2.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.2.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3	**Content publisher role**	**A service that allows publishing of portal content**		**Mandatory**	
1.1.3.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.3.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.3.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.3.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content approval*
1.1.3.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.3.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.2	**Category**	**A service that allows storing of portal content items in hierarchical categories**		**Mandatory**	
1.2.1	Item	A service that allows storing of content in the form of content units.		Mandatory	
1.2.1.1	Document item	A service that allows creation of structured pieces of content, typically stored in the form of files (.DOC, .PDF, etc.).	If it is necessary to have documents for re-use	Optional	
1.2.1.1.1	File .DOC	A service that allows creation of structured pieces of content, typically stored in	If it is necessary to store document items	Optional	

Level	Feature	Description	Rationale	Type	Composition rules
		the form of .DOC files.	in .DOC files		
1.2.1.1.2	File .PDF	A service that allows creation of structured pieces of content, typically stored in the form of .PDF files.	If it is necessary to store document items in .PDF files	Optional	
1.2.1.2	Item name	A service that allows assignment of names to content items.		Mandatory	
1.2.1.3	Menu item	A service that allows identification of portal menu item, to which the particular content item belongs.		Mandatory	
1.2.1.4	Last modification date	A service that allows identification of the date of last content item modification.		Mandatory	
1.2.1.5	Status	A service that allows management of content items publishing.	If the procedure of content items approval is used.	Optional	
1.2.1.5.1	Approved	A service that allows approval of publishing of content items.		Optional	
1.2.1.5.2	Not approved	A service that allows disapproval of publishing of content items.		Optional	
1.2.1.6	Item author name	A service that allows identification of the name of the user, who has created the particular content item.	If it is necessary to have the name of the user, who has created the particular content item.	Optional	
2	**Search feature**	**A service that allows finding for necessary information by entering search keywords or search options.**		**Mandatory**	
2.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
2.1.1	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
2.1.1.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.1.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
2.1.2.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.2.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2.2	*Advanced search*	*A service that includes search by advanced search options, which allow filtering and/or ordering of search*		*Mandatory*	

Level	Feature	Description	Rationale	Type	Composition rules
		results.			
2.1.2.2.1	Date options	A service that allows filtering of search results according to the interval of dates.	If it is necessary to filter search results according to the interval of dates.	Optional	
2.1.2.2.2	Ordering options	A service that allows ordering of search results in the particular way.	If it is necessary to order search results in the particular way.	Optional	
2.1.2.2.3	Scope options	A service that allows limiting scope of search by providing the particular topics, which are relevant for the member.	If it is necessary to limit the scope of search.	Optional	
2.2	**Search results**	**A service that allows viewing the list of simple or advanced search results.**		**Mandatory**	
2.2.1	*Simple search results*	*A service that allows viewing the list of simple search results.*		*Mandatory*	*Requires: Simple search*
2.2.2	*Advanced search results*	*A service that allows viewing the list of advanced search results.*		*Mandatory*	*Requires: Advanced search*
3	**Collaboration feature**	**An online service that provides means of communication between portal users.**		**Mandatory**	
3.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
3.1.1	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
3.1.1.1	*Synchronous collaboration*	*A service that allows collaboration in real-time (concurrently).*		*Mandatory*	
3.1.1.1.1	Instant messaging	A service that allows real-time communication through Internet by typing messages or by participation in on-line voice or video conference		Mandatory	
3.1.1.2	*Asynchronous collaboration*	*A service that allows collaboration at different (asynchronous) time.*		*Mandatory*	
3.1.1.2.1	Forum	A service that provides online collaboration through discussion group, where users can exchange messages.		Mandatory	Requires: Message
3.1.1.2.1.1	Topic post	A service that allows posting topics and replies to them.		Mandatory	
3.1.1.2.1.1.1	Topic	A service that allows posting topics.		Mandatory	
3.1.1.2.1.1.1.1	Subject	A service that allows naming of topics.		Mandatory	
3.1.1.2.1.1.1.2	Message	A service that allows creation of messages that could be seen to the participants of not-real time conversation.		Mandatory	
3.1.1.2.1.1.1.3	Notification type	A service that allows determining if e-mail notifications should be sent to the author of the topic, when replies to it are posted.		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
3.1.1.2.1.1.1.4	Topic author	A service that allows identification of the name of the user, who has created the particular topic.		Mandatory	
3.1.1.2.1.1.1.5	Topic post date and time	A service that allows identification of the date and time of topic creation.		Mandatory	
3.1.1.2.1.1.1.6	Poll	A service that allows voting to the particular question that is raised in the topic.	If some question that requires the opinion of conversation participants is raised in the topic.	Optional	
3.1.1.2.1.1.1.7	Attachment	A service that allows sending computer files along with messages.	If it is necessary to send a computer file.	Optional	
3.1.1.2.1.1.1.8	Topic options	A service that allows setting of different topic reply options, such some restrictions to allowable number of replies to topic, etc.	If topic has some restrictions that should be checked.	Optional	
3.1.1.2.1.1.2	Topic reply	A service that allows posting replies to the particular topic.		Mandatory	
3.1.1.2.1.1.2.1	Subject	A service that allows naming of topic replies.		Mandatory	
3.1.1.2.1.1.2.2	Message	A service that allows creation of messages that could be seen to the participants of not-real time conversation.		Mandatory	
3.1.1.2.1.1.2.3	Notification type	A service that allows determining if e-mail notifications should be sent to the author of the topic, when replies to it are posted.		Mandatory	
3.1.1.2.1.1.2.4	Reply author	A service that allows identification of the name of the user, who has created the particular reply to topic.		Mandatory	
3.1.1.2.1.1.2.5	Attachment	A service that allows sending computer files along with messages.	If it is necessary to send a computer file.	Optional	
3.1.1.2.1.1.2.6	Reply options	A service that allows setting of different topic reply options, such as restrictions to length of every topic reply, etc.	If reply to topic has some restrictions that should be checked.	Optional	
3.1.1.2.2	Newsletter	A service that provides periodic sending of publications on a specific topic by e-mail only to those members, who have subscribed to that service.		Mandatory	
3.1.2	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
3.1.2.1	*Synchronous collaboration*	*A service that allows collaboration in real-time (concurrently).*	*If real-time communication is required.*	*Optional*	
3.1.1.2.1	Instant messaging	A service that allows real-time communication through Internet by typing messages or by participation in on-line		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
		voice or video conference			
4	**Workflow management feature**	**A service that includes assignment of tasks, tasks management and processing.**	**If automated management of tasks is required.**	**Optional**	
4.1	**Role**	**A service that has the set of access rights to participate in portal workflow.**		**Mandatory**	
4.1.1	**Task manager role**	**A service that has the set of access rights for task manager to participate in portal workflow.**		**Mandatory**	
4.1.1.1	Task creation	A service that allows creation of tasks and their assignment to portal members.		Mandatory	
4.1.1.2	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.3	Task changing	A service that allows changing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.4	Task deletion	A service that allows deletion of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.5	Task approval	A service that allows approval of by portal members processed tasks.		Mandatory	Requires: Task processing
4.1.2	**Member role**	**A service that has the set of access rights for portal member to participate in portal workflow.**		**Mandatory**	
4.1.2.1	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.2.2	Task processing	A service that allows processing (acceptation, rejection, suspending, etc.) of to the member assigned tasks.		Mandatory	
4.1.2.2.1	Task acceptance	A service that allows acceptance of by the member fulfilled tasks.	If task needs to be accepted.	Optional	
4.1.2.2.2	Task rejection	A service that allows rejection of tasks, which the member was unable to fulfil.	If task needs to be rejected.	Optional	
4.1.2.2.3	Task suspending	A service that allows suspending of tasks, which the member could not fulfil because some kind of delay.	If task needs to be suspended.	Optional	
4.2	**Task**	**A service that allows storing of to portal members assigned tasks.**		**Mandatory**	
4.2.1	Task name	A service that allows registration of task name.		Mandatory	
4.2.2	Resource name	A service that allows registration of member or member group name the task is assigned to.		Mandatory	
4.2.3	Task assignment date	A service that allows registration of task assignment date.		Mandatory	
4.2.4	Task description	A service that allows registration of task description.		Mandatory	
4.2.5	Task last modification date	A service that allows registration of task last modification date.		Mandatory	Requires: Task creation, Task changing
4.2.6	Task status	A service that allows setting of task status (“Accepted”,	If task is processed.	Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
		"Rejected", "Suspended", "Completed").			
4.2.7	Task action	A service that allows setting of task action ("Check out", "Check in", "Send for approval", "Approve", "Reject")		Mandatory	
4.2.8	Task completion date	A service that allows registration of task completion date.	If task is completed.	Mandatory	Requires: Task status
4.2.9	Information date	A service that allows identification of task creation date.		Mandatory	Requires: Task creation
4.2.10	Task deletion date	A service that allows registration of task deletion date.	If task is deleted.	Mandatory	Requires: Task deletion
4.2.11	Comment	A service that allows registration of the comment about the particular task.	If task is rejected or suspended by the portal member.	Mandatory	Requires: Task status

5.4.4.4. Generic feature model

In 5.4.4.1, 5.4.4.2, 5.4.4.3 presented feature models have been combined into generic feature model using in 4.2.3.4 proposed synthesis method (Table 14):

Table 14. Generic feature table

Level	Feature	Description	Rationale	Type	Composition rules
1	**Content management feature**	**A service that allows creation, management, and publishing of portal content**		**Mandatory**	
1.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
1.1.1	**Content manager role**	**A service that allows management of portal content**		**Mandatory**	
1.1.1.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.1.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.1.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.1.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.1.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content creation*
1.1.1.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.1.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.1.1.3	*Content approval*	*A service that allows content manager to approve item creation, changing or*		*Mandatory*	

Level	Feature	Description	Rationale	Type	Composition rules
		deletion, done by portal members.			
1.1.1.4	*Content search*	*A service that allows content manager to search for necessary content items*		*Mandatory*	
1.1.1.4.1	Simple search	A service that includes search for content items by entered search keywords.		Mandatory	
1.1.1.4.2	Advanced search	A service that includes search for content items by advanced search options, which allow filtering and/or ordering of search results.	If it is necessary to filter/order search results.	Optional	
1.1.2	**Content creator role**	**A service that allows creation of portal content**		**Mandatory**	
1.1.2.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.2.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.2.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.2.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3	**Content publisher role**	**A service that allows publishing of portal content**		**Mandatory**	
1.1.3.1	*Content creation*	*A service that allows creation of portal content.*		*Mandatory*	
1.1.3.1.1	Item creation	A service that allows creation of content items.		Mandatory	
1.1.3.1.2	Item changing	A service that allows portal members to change the items they have created.		Mandatory	Requires: Item creation
1.1.3.1.3	Item deletion	A service that allows portal members to delete the items they have created.		Mandatory	Requires: Item creation
1.1.3.2	*Content publishing*	*A service that allows publishing of portal content.*		*Mandatory*	*Requires: Content approval*
1.1.3.2.1	Item filtering	A service that allows portal members to filter published items by the particular filtering criteria.		Mandatory	
1.1.3.2.2	Item preview	A service that allows preview of published content items		Mandatory	
1.2	**Category**	**A service that allows storing of portal content items in hierarchical categories**		**Mandatory**	
1.2.1	Item	A service that allows storing of content in the form of content units.		Mandatory	
1.2.1.1	Library item	A service that allows creation of reusable, unstructured pieces of content.	If it is necessary to store parts of documents for re-use	Optional	
1.2.1.1.1	Image	A service that allows creation of a visual representation of an object, scene, person, abstraction, etc. produced on a Web page.	If images should be reused in many documents	Optional	
1.2.1.1.2	Static text	A service that allows creation of a text that doesn't	If static text should be reused	Optional	

Level	Feature	Description	Rationale	Type	Composition rules
		change its position on a Web page.	in many documents		
1.2.1.1.3	Banner	A service that allows creation of graphic image (static, animated, or rich media) that is used for the purpose of advertisement.	If banners should be reused in many documents	Optional	
1.2.1.1.4	Video	A service that allows creation of video (e.g. video clip).	If videos should be reused in many documents	Optional	
1.2.1.2	Document item	A service that allows creation of structured pieces of content, typically stored in the form of files (.DOC, .PDF, etc.).	If it is necessary to have documents for re-use	Optional	
1.2.1.2.1	File .DOC	A service that allows creation of structured pieces of content, typically stored in the form of .DOC files.	If it is necessary to store document items in .DOC files	Optional	
1.2.1.2.2	File .PDF	A service that allows creation of structured pieces of content, typically stored in the form of .PDF files.	If it is necessary to store document items in .PDF files	Optional	
1.2.1.3	Item name	A service that allows assignment of names to content items.		Mandatory	
1.2.1.4	Size	A service that allows identification of the size of content items.	If it is necessary to have the size of files	Optional	
1.2.1.5	Menu item	A service that allows identification of portal menu item, to which the particular content item belongs.		Mandatory	
1.2.1.6	Last modification date	A service that allows identification of the date of last content item modification.		Mandatory	
1.2.1.7	Version of the item	A service that allows generation of content item version.	If it is necessary to version content items.	Optional	
1.2.1.8	Action	A service that allows performing items check in-check out procedure.	If check in-check out procedure is used.	Optional	
1.2.1.8.1	Check in	A service that allows making changes to the particular content item and automatically generate its new version.		Optional	
1.2.1.8.2	Check out	A service that allows reserving of the particular content item by the particular user (the other users can not work with that item).		Optional	
1.2.1.9	Status	A service that allows management of content items publishing.	If the procedure of content items approval is used.	Optional	
1.2.1.9.1	Approved	A service that allows approval of publishing of content items.		Optional	
1.2.1.9.2	Not approved	A service that allows disapproval of publishing of content items.		Optional	
1.2.1.10	Item author name	A service that allows identification of the name of the user, who has created the particular content item.	If it is necessary to have the name of the user, who has	Optional	

Level	Feature	Description	Rationale	Type	Composition rules
			created the particular content item.		
2	**Search feature**	**A service that allows finding for necessary information by entering search keywords or search options.**		**Mandatory**	
2.1	**Role**	**A service that has the set of access rights to portal content**		**Mandatory**	
2.1.1	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
2.1.1.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.1.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
2.1.2.1	*Simple search*	*A service that includes search by entered search keywords.*		*Mandatory*	
2.1.2.1.1	Search keywords	A service that allows performing search for portal content by the particular keywords or key phrases.		Mandatory	
2.1.2.2	*Advanced search*	*A service that includes search by advanced search options, which allow filtering and/or ordering of search results.*	*If it is necessary to filter/order search results.*	*Optional*	
2.1.2.2.1	Date options	A service that allows filtering of search results according to the interval of dates.	If it is necessary to filter search results according to the interval of dates.	Optional	
2.1.2.2.2	Ordering options	A service that allows ordering of search results in the particular way.	If it is necessary to order search results in the particular way.	Optional	
2.1.2.2.3	Scope options	A service that allows limiting scope of search by providing the particular topics, which are relevant for the member.	If it is necessary to limit the scope of search.	Optional	
2.1.2.3	*Saved search*	*A service that allows saving search results in folders, and later extract of saved search results from the particular folder.*	*If it is necessary to save search results for later review.*	*Optional*	
2.1.2.3.1	Simple search folder	A service that allows saving of simple search results in the particular folder.	If it is necessary to save simple search results for later review.	Optional	Requires: Simple search results
2.1.2.3.2	Advanced search folder	A service that allows saving of advanced search results in the particular folder.	If it is necessary to save advanced search results for later review.	Optional	Requires: Advanced search results

Level	Feature	Description	Rationale	Type	Composition rules
2.2	**Search results**	**A service that allows viewing the list of simple or advanced search results.**		**Mandatory**	
2.2.1	*Simple search results*	*A service that allows viewing the list of simple search results.*		*Mandatory*	*Requires: Simple search*
2.2.2	*Advanced search results*	*A service that allows viewing the list of advanced search results.*		*Mandatory*	*Requires: Advanced search*
3	**Collaboration feature**	**An online service that provides means of communication between portal users.**	**If communication between portal users is required.**	**Optional**	
3.1	**Role**	**A service that has the set of access rights to collaborate in the portal.**		**Mandatory**	
3.1.1	**Member role**	**A service that has the set of access rights for portal member to search for portal content.**		**Mandatory**	
3.1.1.1	*Synchronous collaboration*	*A service that allows collaboration in real-time (concurrently).*	*If real-time communication is required.*	*Optional*	
3.1.1.1.1	Chat room	A service that allows real-time communication through Internet by exchange of messages.	If real-time communication by exchange of typed messages is required.	Alternative	
3.1.1.1.1.1	Public chat room	A service that allows communication in real time while on the Internet.		Mandatory	
3.1.1.1.1.1.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.1.1.2	Private chat room	A service that allows communication pear-to-pear in real time while on the Internet.	If communication pear-to-pear is required.	Optional	
3.1.1.1.1.2.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.1.1.2.2	File	A service that allows exchange of files between participants of pear-to-pear conversation.	If files sending/receiving is required.	Optional	
3.1.1.1.2	Instant messaging	A service that allows real-time communication through Internet by typing messages or by participation in on-line voice or video conference	If real-time communication by participation in on-line voice or video conference is required.	Optional	
3.1.1.2	*Asynchronous collaboration*	*A service that allows collaboration at different (asynchronous) time.*	*If communication not in real-time is required.*	*Optional*	
3.1.1.2.1	Forum	A service that provides online collaboration through discussion group, where users can exchange messages.	If service that allows exchanging messages on-line is required.	Alternative	Requires: Message
3.1.1.2.1.1	Topic post	A service that allows posting topics and replies to them.		Mandatory	
3.1.1.2.1.1.1	Topic	A service that allows posting		Mandatory	

Level	Feature	Description	Rationale	Type	Composition rules
		topics.			
3.1.1.2.1.1.1.1	Subject	A service that allows naming of topics.		Mandatory	
3.1.1.2.1.1.1.2	Message	A service that allows creation of messages that could be seen to the participants of not-real time conversation.		Mandatory	
3.1.1.2.1.1.1.3	Notification type	A service that allows determining if e-mail notifications should be sent to the author of the topic, when replies to it are posted.		Mandatory	
3.1.1.2.1.1.1.4	Topic author	A service that allows identification of the name of the user, who has created the particular topic.		Mandatory	
3.1.1.2.1.1.1.5	Topic post date and time	A service that allows identification of the date and time of topic creation.		Mandatory	
3.1.1.2.1.1.1.6	Poll	A service that allows voting to the particular question that is raised in the topic.	If some question that requires the opinion of conversation participants is raised in the topic.	Optional	
3.1.1.2.1.1.1.7	Attachment	A service that allows sending computer files along with messages.	If it is necessary to send a computer file.	Optional	
3.1.1.2.1.1.1.8	Topic options	A service that allows setting of different topic reply options, such some restrictions to allowable number of replies to topic, etc.	If topic has some restrictions that should be checked.	Optional	
3.1.1.2.1.1.2	Topic reply	A service that allows posting replies to the particular topic.		Mandatory	
3.1.1.2.1.1.2.1	Subject	A service that allows naming of topic replies.		Mandatory	
3.1.1.2.1.1.2.2	Message	A service that allows creation of messages that could be seen to the participants of not-real time conversation.		Mandatory	
3.1.1.2.1.1.2.3	Notification type	A service that allows determining if e-mail notifications should be sent to the author of the topic, when replies to it are posted.		Mandatory	
3.1.1.2.1.1.2.4	Reply author	A service that allows identification of the name of the user, who has created the particular reply to topic.		Mandatory	
3.1.1.2.1.1.2.5	Attachment	A service that allows sending computer files along with messages.	If it is necessary to send a computer file.	Optional	
3.1.1.2.1.1.2.6	Reply options	A service that allows setting of different topic reply options, such as restrictions to length of every topic reply, etc.	If reply to topic has some restrictions that should be checked.	Optional	
3.1.1.2.2	Newsgroup	A service that provides collaboration through bulletin boards, where users can put and read messages on-line or download and	If service that allows reading messages off-line is required.	Alternative	Requires: Message

Level	Feature	Description	Rationale	Type	Composition rules
		save messages for off-line reading.			
3.1.1.2.3	Newsletter	A service that provides periodic sending of publications on a specific topic by e-mail only to those members, who have subscribed to that service.	If service that provides sending of publications on a specific topic by e-mail is required.	Optional	
3.1.2	**Visitor role**	**A service that has the set of access rights for unregistered user (visitor) to search for portal content.**		**Mandatory**	
3.1.2.1	*Synchronous collaboration*	*A service that allows collaboration in real-time (concurrently).*	*If real-time communication is required.*	*Optional*	
3.1.1.2.1	Chat room	A service that allows real-time communication through Internet by exchange of messages.	If real-time communication by exchange of messages is required.	Alternative	
3.1.1.2.1.1	Public chat room	A service that allows communication in real time while on the Internet.		Mandatory	
3.1.1.2.1.1.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.2.1.2	Private chat room	A service that allows communication pear-to-pear in real time while on the Internet.	If communication pear-to-pear is required.	Optional	
3.1.1.2.1.2.1	Message	A service that allows exchange of messages between the participants of the on-line conversation.		Mandatory	
3.1.1.2.1.2.2	File	A service that allows exchange of files between participants of pear-to-pear conversation.	If files sending/receiving is required.	Optional	
3.1.1.2.2	Instant messaging	A service that allows real-time communication through Internet by typing messages or by participation in on-line voice or video conference	If real-time communication by participation in on-line voice or video conference is required.	Optional	
4	**Workflow management feature**	**A service that includes assignment of tasks, tasks management and processing.**	**If automated management of tasks is required.**	**Optional**	
4.1	**Role**	**A service that has the set of access rights to participate in portal workflow.**		**Mandatory**	
4.1.1	**Task manager role**	**A service that has the set of access rights for task manager to participate in portal workflow.**		**Mandatory**	
4.1.1.1	Task creation	A service that allows creation of tasks and their assignment to portal members.		Mandatory	
4.1.1.2	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.3	Task changing	A service that allows changing of assigned tasks.		Mandatory	Requires: Task creation

Level	Feature	Description	Rationale	Type	Composition rules
4.1.1.4	Task deletion	A service that allows deletion of assigned tasks.		Mandatory	Requires: Task creation
4.1.1.5	Task approval	A service that allows approval of by portal members processed tasks.		Mandatory	Requires: Task processing
4.1.1.6	Task analysis	A service that allows analysis of tasks, including filtering, grouping of tasks, search for tasks, viewing the history of tasks.	If it is necessary to analyse tasks.	Optional	
4.1.1.6.1	Task filtering	A service that allows filtering of tasks by the particular filtering options.	If filtering of tasks is required.	Optional	
4.1.1.6.2	Task grouping	A service that allows grouping of tasks by the particular grouping options.	If grouping of tasks is required.	Optional	
4.1.1.6.3	Task search	A service that allows search for tasks by the particular search options.	If search for tasks is required.	Optional	
4.1.1.6.4	Task history viewing	A service that allows viewing of task modification history.	If viewing of task history is required.	Optional	
4.1.2	**Member role**	**A service that has the set of access rights for portal member to participate in portal workflow.**		**Mandatory**	
4.1.2.1	Task viewing	A service that allows viewing of assigned tasks.		Mandatory	Requires: Task creation
4.1.2.2	Task processing	A service that allows processing (acceptation, rejection, suspending, etc.) of to the member assigned tasks.		Mandatory	
4.1.2.2.1	Task acceptance	A service that allows acceptance of by the member fulfilled tasks.	If task needs to be accepted.	Optional	
4.1.2.2.2	Task rejection	A service that allows rejection of tasks, which the member was unable to fulfil.	If task needs to be rejected.	Optional	
4.1.2.2.3	Task suspending	A service that allows suspending of tasks, which the member could not fulfil because some kind of delay.	If task needs to be suspended.	Optional	
4.2	**Task**	**A service that allows storing of to portal members assigned tasks.**		**Mandatory**	
4.2.1	Task name	A service that allows registration of task name.		Mandatory	
4.2.2	Resource name	A service that allows registration of member or member group name the task is assigned to.		Mandatory	
4.2.3	Task assignment date	A service that allows registration of task assignment date.		Mandatory	
4.2.4	Task description	A service that allows registration of task description.		Mandatory	
4.2.5	Task type	A service that allows assignment of individual or group tasks.		Optional	
4.2.5.1	Group task	A service that allows assignment of tasks to the particular group of portal members.	If it is required to assign the same task to the group of members.	Optional	
4.2.5.2	Individual task	A service that allows	If it is required	Optional	

Level	Feature	Description	Rationale	Type	Composition rules
		assignment of tasks to the particular member.	to assign the task to the particular member personally.		
4.2.6	Task priority	A service that allows registration of task priority (importance of the particular task).	If importance of tasks is required.	Optional	
4.2.7	Task last modification date	A service that allows registration of task last modification date.		Mandatory	Requires: Task creation, Task changing
4.2.8	Task status	A service that allows setting of task status ("Accepted", "Rejected", "Suspended", "Completed").	If task is processed.	Mandatory	
4.2.9	Task action	A service that allows setting of task action ("Check out", "Check in", "Send for approval", "Approve", "Reject")		Mandatory	
4.2.10	Task completion date	A service that allows registration of task completion date.	If task is completed.	Mandatory	Requires: Task status
4.2.11	Information date	A service that allows identification of task creation date.		Mandatory	Requires: Task creation
4.2.12	Task deletion date	A service that allows registration of task deletion date.	If task is deleted.	Mandatory	Requires: Task deletion
4.2.13	Comment	A service that allows registration of the comment about the particular task.	If task is rejected or suspended by the portal member.	Mandatory	Requires: Task status

5.4.5. Web Portal Requirements (Evaluation Test Examples)

Web portal functional requirements have been derived from in 5.4.4.4 presented feature model. Four groups of requirements have been separated: content management, search, collaboration, and workflow requirements. These requirements have been specified using UML 2.0 and Z specification languages. UML 2.0 specification has been created using MagicDraw UML 10.0 [NM06], and Z specification has been created using Z/EVES 2.1 [Saa99].

5.4.5.1. Requirements for Content Management

Content management requirements are presented in Table 15:

Table 15. Content management requirements table

ID	Kind	Requirement	Type	Priority
REQ-1	**F**	**Content management requirements**	**M**	**M**
REQ-1.1	F	Content management should include: • Creation of portal **Pt** content; • Publishing of portal **Pt** content; • Deletion of portal **Pt** content; • Changing of portal **Pt** content; • Filtering of portal **Pt** content; • Searching for portal **Pt** content.	M	S

ID	Kind	Requirement	Type	Priority
REQ-1.2	F	The main roles **Rl** that should be defined for content management are: • Content manager ***CM***; • Content creator ***CC***; • Content publisher ***CP***.	M	S
REQ-1.3	F	All these roles having users should be members of the portal **Pt**.	M	S
REQ-1.4	F	Every role **Rl** should have the set of access rights $\mathbf{Rt_1, \ldots, Rt_N}$ that defines permissions and privileges granted to portal members.	M	S
REQ-1.5	F	Content manager ***CM*** should be able to manage content of the portal by: • Creating portal **Pt** content; • Publishing portal **Pt** content; • Approval or rejection of the content that content creators have created.	M	S
REQ-1.6	F	Content creator ***CC*** should be able only to create drafts of portal **Pt** content items **Itm** that should be approved by the content manager ***CM*** before publishing.	M	S
REQ-1.7	F	Content publisher ***CP*** should be able to: • Create portal **Pt** content; • Publish portal **Pt** content.	M	S
REQ-1.8	F	Portal **Pt** content items ***CntItm*** could be classified into library items ***LbItm*** and document items ***DocItm***.	O	C
REQ-1.9	F	Library items ***LbItm*** could be reusable, unstructured pieces of content.	O	C
REQ-1.9.1	F	Library items ***LbItm*** could be classified into images, static text, banners, and video.	O	C
REQ-1.10	F	Document items ***DocItm*** could be structured pieces of content.	O	C
REQ-1.10.1	F	It could be possible to store document items in .DOC and .PDF formats.	O	C
REQ-1.11	F	Portal information architecture should be organised in the form of hierarchy of categories $\mathbf{Cat_1, \ldots, Cat_N}$, in which library items ***LbItm*** and document items ***DocItm*** should be saved.	M	S
REQ-1.12	F	Portal **Pt** could have content check in-check out procedure.	O	C
REQ-1.12.1	F	Check in-check-out procedure should be used to ensure that at the same moment of time only one member ***Mem*** should be able to create/change/delete the same content item ***CntItm***.	O	C
REQ-1.13	F	Every created library item ***LbItm*** or document item ***DocItm*** should have: • Menu item, • Item type (document item or library item), • Item kind (image, text, banner, video), • Item name, • Size (Kb), • Last modification date, • Item author name.	M	S
REQ-1.13.1	F	During creation of the particular content item the following attributes should be indicated: • Item name, • Menu item, • Item type, • Item kind.	M	S
REQ-1.13.2	F	During creation of the particular content item its name should be generated.	M	S
REQ-1.13.3	F	During creation of the particular content item the following attributes should be generated automatically: • Version of the item, • Size (Kb), • Last modification date, • Item author name,	O	C

ID	Kind	Requirement	Type	Priority
		• Item status, • Item action.		
REQ-1.13.4	F	Version of the item and item action could be generated automatically after that item is checked in.	O	C
REQ-1.13.5	F	The available actions could be "Check in" or "Check out".	O	C
REQ-1.13.6	F	The available statuses of items could be "Approved" or "Not approved".	O	C
REQ-1.14	F	Content manager ***CM*** should be able to change/delete all portal items.	M	S
REQ-1.15	F	Content creator ***CC*** and content publisher ***CP*** should be able to change/delete only the items that he has created.	M	S
REQ-1.16	F	Content manager ***CM*** should be able to search for particular content items of the portal **Pt**.	M	S
REQ-1.16.1	F	It should be possible to perform two kinds of content search: • Simple search, • Advanced search.	M	S
REQ-1.16.2	F	To search for the content of the portal **Pt** content manager ***CM*** could enter search keywords $\mathbf{Kw_1,\ldots, Kw_N}$ and advanced search options $\mathbf{Opt_1, \ldots, Opt_N}$.	M	S
REQ-1.17	F	Content manager ***CM*** should be able to filter published content items of the portal **Pt**.	M	S
REQ-1.17.1	F	It should be possible to filter published content items ***CntItm*** by the following filtering criteria: • Item type, • Item kind, • Last modification date, • Version of the item.	M	S
REQ-1.18	F	Content manager ***CM*** and content publisher ***CP*** should be able to preview published content items of the portal **Pt**.	M	S

5.4.5.2. Requirements for Search

Search requirements are presented in Table 16:

Table 16. Search requirements table

ID	Kind	Requirement	Type	Priority
REQ-2	**F**	**Search requirements**	**M**	**M**
REQ-2.1	**F**	**Visitor search requirements**	**M**	**M**
REQ-2.1.1	F	Visitors ***Vst*** of the portal **Pt** should have possibility to perform only simple search that has the scope "All" (search the entire portal **Pt**).	M	S
REQ-2.1.2	F	The functionality of simple search should include searching in portal **Pt** database for certain keywords $\mathbf{Kw_1, \ldots., Kw_N}$ and returning the list of results.	M	S
REQ-2.1.2.1	F	If the visitor ***Vst*** has entered such search keywords $\mathbf{Kw_1, \ldots., Kw_N}$ that no results are found in the database, then the visitor ***Vst*** should be informed that search was not successful.	M	S
REQ-2.1.2.2	F	If total number of search results is too large, then the visitor ***Vst*** should be proposed to limit the scope of search.	M	S
REQ-2.1.2.3	F	The visitor ***Vst*** should be able limit the scope of search by entering another keywords $\mathbf{Kw_1, \ldots., Kw_N}$.	M	S
REQ-2.1.3	F	The top of search results page $\mathbf{P_{REZ}}$ should contain: • Search keywords $\mathbf{Kw_1, \ldots., Kw_N}$ and • The interval of results (from $\mathbf{N_1}$ to $\mathbf{N_2}$) that are shown on one page.	M	S
REQ-2.1.4	F	The number of returned results that are shown on one page should be not greater than **N**.	M	S
REQ-2.1.4.1	F	The returned results should be the numbered list of links $\mathbf{L_1, ..., L_N}$	M	S

ID	Kind	Requirement	Type	Priority
		for pages $\mathbf{P_1, \ldots, P_N}$ that form part of the portal **Pt**.		
REQ-2.1.4.2	F	Search results should be prioritised depending on the importance of each result, but not purely on the basis of how many search keywords the found pages $\mathbf{P_1, \ldots, P_N}$ contain.	M	S
REQ-2.1.4.3	F	To view the results that don't fit on the first page the visitor ***Vst*** should open the Nth search results page $\mathbf{P_{REZ}}$, where he can view the next **N** results.	M	S
REQ-2.2	**F**	**Member search requirements**	**M**	**M**
REQ-2.2.1	F	Member ***Mem*** of the portal **Pt** should have possibility to perform two kinds of search: • Simple search, • Advanced search.	M	S
REQ-2.2.2	F	Members ***Mem*** of the portal **Pt** should be able to target their search by using advanced search options $\mathbf{Opt_1, \ldots, Opt_N}$ for narrowing the results.	M	S
REQ-2.2.2.1	F	The portal **Pt** could provide several kinds of advanced options: • Ordering options $\mathbf{Oopt_1, \ldots, Oopt_N}$, • Date options $\mathbf{Dopt_1, \ldots, Dopt_N}$, • Scope options $\mathbf{Sopt_1, \ldots, Sopt_N}$.	O	C
REQ-2.2.2.2	F	Ordering options $\mathbf{OOpt_1, \ldots, OOpt_N}$ should allow ordering of search results in member ***Mem*** preferred way.	O	C
REQ-2.2.2.3	F	Date options $\mathbf{DOpt_1, \ldots, DOpt_N}$ should allow filtering of data according to dates (to show all data or only the data that belongs to some date interval).	O	C
REQ-2.2.2.4	F	Scope options $\mathbf{SOpt_1, \ldots, SOpt_N}$ should allow limiting scope of search (it should be possible to provide particular topics that are relevant for the member ***Mem***).	O	C
REQ-2.2.2.5	F	If the member ***Mem*** has entered such advanced search options $\mathbf{Opt_1, \ldots, Opt_N}$ that no results are found, then the member ***Mem*** should be informed that search was not successful.	M	S
REQ-2.2.2.6	F	The scope of advanced search may be limited by entering another advanced search scope options $\mathbf{SOpt_1, \ldots, SOpt_N}$.	M	S
REQ-2.2.3	F	The top of search results page $\mathbf{P_{REZ}}$ should contain: • Search keywords $\mathbf{Kw_1, \ldots, Kw_N}$, • Ordering options $\mathbf{OOpt_1, \ldots, OOpt_N}$, • Date options $\mathbf{DOpt_1, \ldots, DOpt_N}$, • Scope options $\mathbf{SOpt_1, \ldots, SOpt_N}$, • The interval of results (from $\mathbf{N_1}$ to $\mathbf{N_2}$) that are shown on one page.	M	S
REQ-2.2.4	F	The other requirements for advanced search should be the same as REQ-2.1.4 and it comprising requirements.	M	S
REQ-2.2.5	F	Member ***Mem*** of the portal **Pt** could have possibility to save the results of simple search or the results of advanced search.	O	C
REQ-2.2.5.1	F	The results of simple or advanced search could be saved in folders $\mathbf{F_1, \ldots, F_M}$.	O	C
REQ-2.2.5.2	F	To save the results of simple or advanced search the member ***Mem*** while being on search results page $\mathbf{P_{REZ}}$ should open saved search page $\mathbf{P_{SS}}$.	M	S
REQ-2.2.5.3	F	Saved search page should contain the list of by the member ***Mem*** saved folders $\mathbf{F_1, \ldots, F_M}$.	M	S
REQ-2.2.5.4	F	To save the results of simple or advanced search the member ***Mem*** should enter the name of folder $\mathbf{F_i}$.	M	S
REQ-2.2.5.5	F	By the member ***Mem*** entered folder $\mathbf{F_i}$ must be unique, that is such that no two folders can be identical.	M	M
REQ-2.2.5.6	F	If the folder $\mathbf{F_i}$ is unique then the total number of folders **M** that have been created by the member ***Mem*** should be counted.	M	S
REQ-2.2.5.7	F	New saved search folder $\mathbf{F_i}$ should be created only if the entered folder $\mathbf{F_i}$ is unique and number of member ***Mem*** folders is not greater than **M**.	M	S
REQ-2.2.5.8	F	If member folder $\mathbf{F_i}$ is not unique, that is the folder with such	M	S

ID	Kind	Requirement	Type	Priority
		name already exists in content repository **Cr**, then the folder $\mathbf{F_i}$ should not be created and the member ***Mem*** should be informed that another folder name should be entered.		
REQ-2.2.5.9	F	If the number of member ***Mem*** folders is greater than **M** then before creation of the new folder $\mathbf{F_i}$ the member ***Mem*** should be informed that it is necessary to delete some folder $\mathbf{F_k}$ that was created earlier.	M	S
REQ-2.2.6	F	Member ***Mem*** could have possibility to run his saved searches.	O	C
REQ-2.2.6.1	F	To run saved search member ***Mem*** should open saved search page $\mathbf{P_{SS}}$ and select the folder $\mathbf{F_i}$ that he prefers to run.	M	S
REQ-2.2.6.2	F	If the folder $\mathbf{F_i}$ contains simple search then search results page $\mathbf{P_{REZ}}$ should be opened, which should have all fields that are mentioned in requirement REQ-2.1.3.	M	S
REQ-2.2.6.3	F	If the folder $\mathbf{F_i}$ contains advanced search then search, results page $\mathbf{P_{REZ}}$ should be opened, which should have all fields that are mentioned in requirement REQ-2.2.3.	M	S
REQ-2.2.6.4	F	The member ***Mem*** should have possibility to see the other results that don't fit on the first page (see REQ-2.1.4.3).	M	S
REQ-2.2.6.5	F	The member ***Mem*** should have possibility to delete any of the folders $\mathbf{F_1, \ldots, F_M}$ that he has created.	M	S

5.4.5.3. Requirements for Collaboration

Collaboration requirements are presented in Table 17:

Table 17. Collaboration requirements table

ID	Kind	Requirement	Type	Priority
REQ-3.	**F**	**Collaboration requirements**	**O**	**C**
REQ-3.1	**F**	**Member collaboration requirements**	**O**	**C**
REQ-3.1.1	**F**	Member ***Mem*** of the portal **Pt** could have possibility to collaborate asynchronously (not in real time).	O	C
REQ-3.1.1.1	F	Member ***Mem*** of the portal **Pt** could have possibility to participate in forums $\mathbf{Fr_1, \ldots, Fr_N}$, which are provided by the portal **Pt**.	A	C
REQ-3.1.1.2	F	Member ***Mem*** of the portal **Pt** could have possibility to participate in newsgroups $\mathbf{Nw_1, \ldots, Nw_N}$, which are provided by the portal **Pt**.	A	C
REQ-3.1.1.3	F	Member ***Mem*** of the portal **Pt** could have possibility to subscribe to newsletters $\mathbf{Nwl_1, \ldots, Nwl_N}$, which are provided by the portal **Pt**.	O	C
REQ-3.1.1.1.1	F	A forum ***Fr*** should be an online notice board, where members ***Mem*** should be able to post topics $\mathbf{Top_1, \ldots, Top_N}$ or replies to selected topics $\mathbf{Re_1, \ldots, Re_N}$, vote to questions that can be set in topics $\mathbf{Top_1, \ldots, Top_N}$.	M	S
REQ-3.1.1.1.2	F	To access necessary forum ***Fr*** and its particular topic ***Top*** member ***Mem*** should open asynchronous collaboration page $\mathbf{P_{AC}}$ and select the particular forum ***Fr*** and topic ***Top***.	M	S
REQ-3.1.1.1.3	F	Asynchronous collaboration page $\mathbf{P_{AC}}$ should have: • User name of members that have logged on, • Total number of portal **Pt** members.	M	S
REQ-3.1.1.1.4	F	On asynchronous collaboration page $\mathbf{P_{AC}}$ displayed list of available forums should contain: • Forum name, • Number of topics, • Number of posts, • Last post author, • Last post date and time.	M	S
REQ-3.1.1.1.5	F	On the forum page $\mathbf{P_F}$ displayed list of topics should	M	S

ID	Kind	Requirement	Type	Priority
		contain: • Topic subject (topic name), • Number of replies, • Last post author, • Last post date and time.		
REQ-3.1.1.1.6	F	On the topic page $\mathbf{P_{TP}}$ displayed topics should contain: • Subject, • Message, • Post date and time, • Topic author information (user name, membership start date, number of member posts).	M	S
REQ-3.1.1.1.7	F	The portal **Pt** should provide possibility to view topic author profile ***Prf*** information.	M	S
REQ-3.1.1.1.8	F	Member ***Mem*** should be able to view only selected posts that are: • In selected forum ***Fr***, • In selected topic ***Top***.	M	S
REQ-3.1.1.1.9	F	Member ***Mem*** of the portal **Pt** should have possibility to create new topics.	M	S
REQ-3.1.1.1.9.1	F	To create new topic member ***Mem*** should enter topic info: • Subject, • Message, • Notification type (notification type should be filled to determine if e-mail notifications should be sent to the author of the topic ***Top***, when replies to the topic are posted).	M	S
REQ-3.1.1.1.9.2	F	Topic message should be not shorter than **S** symbols.	M	S
REQ-3.1.1.1.9.3	F	The other topic information that could be set for the new topic is optional: • Poll information (poll question, poll options, poll timeout), • Attachment, • Topic options.	O	C
REQ-3.1.1.1.9.4	F	The number of poll options should be not less than **R** and not greater than **K**.	M	S
REQ-3.1.1.1.9.5	F	If entered topic information is valid and number of poll options is allowable, then topic author name, topic post date, post time should be generated and new topic ***Top*** should be created.	M	S
REQ-3.1.1.1.9.6	F	If topic message is shorter than **S** symbols then the member should be informed that he should enter a message of valid length.	M	S
REQ-3.1.1.1.9.7	F	If the number of poll options is less than **R** or not greater than **K**, then member should be informed to correct poll information.	M	S
REQ-3.1.1.1.10	F	To post reply ***Re*** to the particular topic ***Top*** member ***Mem*** should select the necessary topic and provide the following information: • Subject, • Message, • Notification type.	M	S
REQ-3.1.1.1.10.1	F	The other topic reply information that can be set for the new topic reply should be optional: • Attachment, • Reply options.	M	S
REQ-3.1.1.1.10.2	F	If entered topic reply information is valid, then topic reply author name, topic reply post date, post time should be generated and new topic reply ***Re*** should be created.	M	S
REQ-3.1.1.1.10.3	F	If reply message is shorter than **S** symbols then the member should be informed that he should enter a message of valid	M	S

ID	Kind	Requirement	Type	Priority
		length.		
REQ-3.1.1.1.10.4	F	After reply ***Re*** is created portal **Pt** should determine: • If it is necessary to send notification about new topic reply to topic author; • If it is necessary to send notification about new topic reply to other topic participants (who have posted replies to the topic).	M	S
REQ-3.1.1.1.11	F	If necessary, notifications should be sent to topic author and/or to the other topic participants.	M	S
REQ-3.1.1.1.12	F	Member ***Mem*** should have possibility to delete the topics/topics replies that he has created.	M	S
REQ-3.1.1.1.12.1	F	When topic ***Top*** is deleted all replies to this topic should be deleted automatically.	M	S
REQ-3.1.1.1.13	F	If topic has poll options, then member ***Mem*** should have possibility to vote.	M	S
REQ-3.1.1.1.13.1	F	Voting process should include answering to poll question by selection of preferred poll option.	M	S
REQ-3.1.1.1.13.2	F	After poll option is selected the portal **Pt** should provide possibility to see the results of voting that should include total number of votes and the list of poll options with corresponding percents of votes.	M	S
REQ-3.1.2	F	Member ***Mem*** of the portal **Pt** could have possibility to collaborate synchronously (in real time).	O	C
REQ-3.1.2.1	F	Portal **Pt** could have a set of virtual public chat rooms $\mathbf{CR_1, \ldots, CR_N}$.	A	C
REQ-3.1.2.2	F	Portal **Pt** could have a set of instant messaging services $\mathbf{IM_1, \ldots, IM_N}$.	A	C
REQ-3.1.2.1.1.	F	Every virtual public chat room ***CR*** should have a particular topic ***Top***.	M	S
REQ-3.1.2.1.2	F	To enter necessary chat room ***CR*** the member ***Mem*** should open synchronous collaboration page $\mathbf{P_{SC}}$ and select the particular chat room ***CR***.	M	S
REQ-3.1.2.1.3	F	Portal **Pt** members should belong to members chat room group $\mathbf{GR_M}$, that is member ***Mem*** should have possibility to enter any chat room ***CR*** that is provided by the portal **Pt**.	M	S
REQ-3.1.2.1.3.1	F	Member ***Mem*** nickname should be determined by user name that he has entered during login to the portal **Pt**, thus to enter the particular chart room ***CR*** with member rights member shouldn't need to log in for the second time.	M	S
REQ-3.1.2.1.3.2	F	Member ***Mem*** should be allowed to enter the selected chat room ***CR*** only if: • That member ***Mem*** nickname is not banned, • Total number of chat room participants is not greater than **R**.	M	S
REQ-3.1.2.1.3.3	F	In case of successful entering the chat room ***CR*** the portal **Pt** should open public chat room page $\mathbf{P_{CR}}$, which should have: • Topic $\mathbf{Top_i}$, • List of available chat rooms $\mathbf{CR_1, \ldots, CR_N}$, • List of chat room participants (first members and then visitors) $\mathbf{Usr_1, \ldots, Usr_N}$.	M	S
REQ-3.1.2.1.4	F	After clicking on any member ***Mem*** that is in the list of chat room participants it should be possible to see his profile **Prf** information.	M	S
REQ-3.1.2.1.5	F	Member ***Mem*** of the portal **Pt** could have possibility to chat in private chat rooms $\mathbf{PCR_1, \ldots, PCR_N}$ pear-to-pear.	O	C
REQ-3.1.2.1.5.1	F	To start chatting pear-to-pear member ***Mem*** should be able to select the member ***Mem*** he wants to chat and the portal **Pt** should open private chat room page $\mathbf{P_{PCR}}$.	M	S
REQ-3.1.2.1.5.2	F	Private chat room page $\mathbf{P_{PCR}}$ should be seen only to two	M	S

ID	Kind	Requirement	Type	Priority
		members of conversation, other participants of the chat room ***CR*** should not see that page.		
REQ-3.1.2.1.6	F	Every virtual chat room (public or private) ***CR*** should allow members ***Mem*** to chat with all chat room participants by typing messages $\mathbf{Msg_1, \ldots, Msg_N}$ to each other in real time, creating an online conversation.	M	S
REQ-3.1.2.1.6.1	F	Messages $\mathbf{Msg_1, \ldots, Msg_N}$ should appear on an area of the public (private) chat room page $\mathbf{P_{CR}}$ next to the member's ***Mem*** nickname.	M	S
REQ-3.1.2.1.6.2		In private chat rooms $\mathbf{PCR_1, \ldots, PCR_N}$ members ***Mem*** could have possibility not only exchange messages, but send/receive files $\mathbf{Fl_1, \ldots, Fl_N}$.	O	C
REQ-3.1.2.1.7	F	The portal **Pt** should allow member ***Mem*** to open not more than **N** public or private chat rooms at one time.	M	S
REQ-3.1.2.1.7.1	F	If the member ***Mem*** has already opened **L** chat rooms and attempts to open one more chat room, then he should be informed that number of allowed to enter chat rooms have exceeded.	M	S
REQ-3.1.2.1.8	F	If the member ***Mem*** in public or in private chat room types a message **Msg** that is unprintable, then the portal **Pt** should automatically set ban ***Ban*** for such user.	M	S
REQ-3.1.2.1.8.1	F	Ban should have start date and end date that means that during that period the member ***Mem*** should not be able to enter none of portal **Pt** chat rooms $\mathbf{CR_1, \ldots, CR_N}$.	M	S
REQ-3.1.2.1.8.2	F	When ban for ***Mem*** ends, it is unset automatically and the member should be allowed to enter the chat rooms $\mathbf{CR_1, \ldots, CR_N}$, which are available for portal members.	M	S
REQ-3.2	**F**	**Visitor collaboration requirements**	**O**	**C**
REQ-3.2.1	**F**	Visitor ***Vst*** of the portal **Pt** could have possibility to collaborate synchronously (in real time).	O	C
REQ-3.2.1.1	F	Visitor ***Vst*** should be allowed to enter only the chat rooms that belong to the particular group $\mathbf{GR_i}$ (Visitor chat rooms).	M	S
REQ-3.2.1.2	F	Visitor ***Vst*** of the portal **Pt** should have possibility to enter limited number **R** of chat rooms	M	S
REQ-3.2.1.3	F	To enter necessary chat room $\mathbf{CR_i}$ the visitor ***Vst*** should open synchronous collaboration page $\mathbf{P_{SC}}$ and enter his nickname.	M	S
REQ-3.2.1.4	F	If the nickname is unique, then the list of available chat rooms $\mathbf{CR_1, \ldots, CR_N}$ should be displayed on the synchronous collaboration page $\mathbf{P_{SC}}$.	M	S
REQ-3.2.1.5	F	In case of successful entering the chat room $\mathbf{CR_i}$ the portal **Pt** should open chat room page $\mathbf{P_{CR}}$, which should have: • Topic $\mathbf{Top_i}$, • List of chat room participants (first members and then visitors) $\mathbf{Usr_1, \ldots, Usr_N}$, • Filtered list of available chat rooms $\mathbf{CR_1, \ldots, CR_N}$.	M	S
REQ-3.2.1.6	F	Messages $\mathbf{Msg_1, \ldots, Msg_N}$ should appear on an area of the public chat room page $\mathbf{P_{CR}}$ next to the visitor's ***Vst*** nickname.	M	S
REQ-3.2.1.7	F	The number of opened chat rooms $\mathbf{CR_i}$ should not be greater than **L**.	M	S
REQ-3.2.1.7.1	F	If the visitor ***Vst*** have already opened **L** chat rooms and attempts to open one more chat room, then he should be informed that number of allowed to enter chat rooms have exceeded.	M	S

5.4.5.4. Requirements for Workflow

Workflow requirements are presented in Table 18:

Table 18. Workflow requirements table

ID	Kind	Requirement	Type	Priority
REQ-4	**F**	**Collaboration requirements**	**O**	**C**
REQ-4.1	F	Workflow management could include: • Creation and assignment of tasks ***Tsk***, • Viewing of tasks ***Tsk***, • Processing of tasks ***Tsk***, • Changing of tasks ***Tsk***, • Deletion of tasks ***Tsk***, • Approval or rejection of tasks ***Tsk***, • Filtering and grouping of tasks ***Tsk***, • Searching for tasks ***Tsk***, • Viewing the history of task ***Tsk*** modifications.	O	C
REQ-4.1.1	F	The main roles **Rl** that should be defined for workflow are: • Workflow manager ***WM***, • Portal member ***Mem***.	M	S
REQ-4.1.2	F	Workflow manager ***WM*** should be able to manage workflow process by: • Creation and assignment/reassignment of tasks ***Tsk*** to portal members, • Viewing of tasks ***Tsk***, • Changing of tasks ***Tsk***, • Deletion of tasks ***Tsk***, • Approval or rejection of tasks ***Tsk***, • Filtering and grouping of tasks ***Tsk***, • Searching for tasks ***Tsk***, • Viewing the history of task ***Tsk*** modifications.	M	S
REQ-4.2	F	Every task ***Tsk*** should have: • Task name, • Resource name (member ***Mem*** name or member group name $\boldsymbol{G_M}$, to which the task is assigned to), • Task assignment date, • Task description, • Task type (group or individual task), • Task priority, • Task last modification date, • Task status (“Accepted”, “Rejected”, “Suspended”, “Completed”), • Task action (“Check out”, “Check in”, “Send for approval”, “Approve”, “Reject”), • Task completion date, • Information date, • Comment.	M	S
REQ-4.2.1	F	During creation of the particular task and assignment it to the portal member the following attributes should be indicated: • Task name, • Resource name (member ***Mem*** name or member group name $\boldsymbol{G_M}$, to which the task is assigned to), • Task assignment date, • Task description, • Task type (group or individual task), • Task priority, • Task last modification date.	M	S
REQ-4.2.1.1	F	Task priority could be registered if it is necessary to indicate the importance of the particular task.	O	C

ID	Kind	Requirement	Type	Priority
REQ-4.3	F	It could be possible to assign the particular task ***Tsk*** to the particular ***Mem*** of the portal **Pt** or to the group of portal **Pt** members G_M.	O	C
REQ-4.4	F	Workflow manager ***WM*** should have possibility to view all tasks $\mathbf{Task_1, \ldots, Task_N}$ that he has assigned to portal members.	M	S
REQ-4.4.1	F	Every portal member ***Mem*** should have possibility to view only to him assigned tasks $\mathbf{Task_1, \ldots, Task_N}$.	M	S
REQ-4.5	F	Portal member ***Mem*** should be able to process to him assigned tasks by their acceptation, rejection or suspending.	M	S
REQ-4.5.1	F	Task ***Tsk*** accepting should mean that the member has performed the task that was assigned to him.	M	S
REQ-4.5.2	F	Task ***Tsk*** rejecting should mean that the member could not accomplish the task. In this case comment with the rejection reason should be provided.	M	S
REQ-4.5.3	F	Task ***Tsk*** suspending should mean that the member could not perform the task because of some kind of delay. In this case comment with the reason of delay should be provided.	M	S
REQ-4.5.3.1	F	Suspended tasks further should change their status to "Rejected" or "Accepted" depending on the member's ability to perform the task.	M	S
REQ-4.6	F	Accepted or rejected tasks should be approved by workflow manager ***WM***, thus notifications should be sent to him. After notification is sent, the action should become "Send for approval".	M	S
REQ-4.6.1	F	After workflow manager ***WM*** approves or rejects the particular task by setting their status to "Approve" or "Reject" correspondingly, notifications should be sent to the particular member or to all member that belong to the particular group.	M	S
REQ-4.6.1.1	F	After notifications to the members are sent the status of the approved or rejected task should change to "Completed".	M	S
REQ-4.6.2	F	If the task was rejected by workflow manager ***WM,*** this should mean that the task is not assigned to that member/member group any more.	M	S
REQ-4.6.3	F	If necessary, workflow manager ***WM*** should be able to assign rejected tasks to the other portal member ***Mem*** or member group G_M.	M	S
REQ-4.7	F	Workflow manager ***WM*** should have possibility to delete tasks $\mathbf{Task_1, \ldots, Task_N}$, which have lost their actuality for some reasons or to change tasks $\mathbf{Task_1, \ldots, Task_N}$ that need to be modified.	M	S
REQ-4.7.1	F	At the same moment of time only one workflow manager ***WM*** should be able to change/delete the same task ***Tsk***.	M	S
REQ-4.7.1.1	F	It should be possible to change/delete task with the status "Check in".	M	S
REQ-4.7.1.2	F	It should not be possible to change/delete task with the status "Check out".	M	S
REQ-4.7.2	F	After workflow manager ***WM*** deletes not actual tasks or makes changes, notifications to the members, to whom these tasks were assigned or to all member that belong to the particular group, should be sent.	M	S
REQ-4.8	F	Workflow manager ***WM*** could have possibility to analyse tasks by task filtering, grouping, search for particular tasks $\mathbf{Task_1, \ldots, Task_N}$, and viewing the history of task modifications.	O	C
REQ-4.8.1	F	Filtering could be done by the following filtering options $\mathbf{Fopt_1, .., Fopt_N}$: • Member group, • Particular member, • Task status, • Tasks pending manager's approval (action "Send for approval"), • Overdue tasks,	O	C

ID	Kind	Requirement	Type	Priority
		• Completed tasks.		
REQ-4.8.2	F	Grouping of tasks could be done by selection of grouping options **$Gopt_1$, …, $Gopt_N$**.	O	C
REQ-4.8.3	F	Search for particular tasks could be done by performing two kinds of search: • Simple search, • Advanced search.	O	C
REQ-4.8.3.1	F	To search for the tasks workflow manager ***WM*** should enter search keywords **Kw_1,…, Kw_N** and advanced search options **Opt_1, …, Opt_N**.	M	S
REQ-4.9	F	Workflow manager ***WM*** could have possibility to view the history of task modifications during the particular period of time (Start date, End date).	O	C
REQ-4.9.1	F	The history typically could include: • Date and time (information date in case of task creation, last modification date in case of task modification), • Member (resource name), who have changed the status of task, • If the task was approved or moved on in the workflow (reassigned to the other member), • If the task was suspended for later processing.	O	C

5.4.6. Description of Quality Evaluation Tests

This section provides descriptions of UML and Z quality evaluation tests and suites of tests. For every test it has been defined which sub-characteristics of functionality will be tested by that test and by specification of which requirements these characteristics will be tested. Quality evaluation tests have been combined into suites of tests.

5.4.6.1. Description of UML Evaluation Tests

Descriptions of UML evaluation tests, which have been developed for content management and search components of Web portal, are presented in Table 19, Table 20:

Table 19. Description of UML evaluation test T-F-IAS-UML-01

Test identifier	T-F-IAS-UML-01	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Requirements**	**Should be tested by [14]:**
Ontological sufficiency Ontological adequacy	REQ-1.1, REQ-1.2, REQ-1.3, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.9, REQ-1.10, REQ-1.11, REQ-1.12, REQ-1.13, REQ-1.13.1, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.18, REQ-2.1.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-	54

[14] Number of requirements that belong to the group G_ξ of $N(\xi)$ to test the characteristic ξ.

Test identifier	T-F-IAS-UML-01	
	2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	
Epistemological sufficiency Epistemological adequacy	REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-2.1.2, REQ-2.1.3, REQ-2.1.4.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.3, REQ-2.2.5.1, REQ-2.2.5.4	15
Expressibility	REQ-1.4, REQ-1.12.1, REQ-1.13, REQ-1.13.4, REQ-1.14, REQ-1.15, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.2.2.5, REQ-2.2.5, REQ-2.2.5.6, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6.2, REQ-2.2.6.3	15
Reasoning power	REQ-1.3, REQ-1.4, REQ-1.12.1, REQ-1.15, REQ-2.1.1, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.2.1, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5.8, REQ-2.2.5.9	12
Composability	REQ-1.5, REQ-1.15, REQ-1.12.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.2.2, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3	12
Selective power	REQ-1.16.2, REQ-1.17, REQ-2.1.2.1, REQ-2.1.2.3, REQ-2.2.2, REQ-2.2.2.6	6
Generalitive power	REQ-1.9, REQ-1.9.1, REQ-1.10, REQ-1.10.1, REQ-1.12, REQ-1.12.1, REQ-1.13, REQ-1.13.1, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.13.5, REQ-1.13.6, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.17.1, REQ-2.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	50

Table 20. Description of UML evaluation test T-F-IAS-UML-02

Test identifier	T-F-IAS-UML-02	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Requirements**	**Should be tested by:**
Extensibility	REQ-1.10.1, REQ-1.12.1, REQ-1.13, REQ-1.13.1, REQ-1.13.5, REQ-1.13.6, REQ-1.15, REQ-2.1.2, REQ-2.1.3, REQ-2.2.2, REQ-2.2.3	11
Adaptability	REQ-1.10.1, REQ-1.13.5, REQ-1.13.6, REQ-1.16.2, REQ-2.1.2, REQ-2.1.2.3, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6	13
Universality	REQ-1.1, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.1, REQ-2.1.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.6	23

Descriptions of UML evaluation tests, which have been developed for collaboration and workflow components of Web portal, are presented in Table 21, Table 22:

Table 21. Description of UML evaluation test T-F-CS-UML-01

Test identifier	T-F-CS-UML-01	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Requirements**	**Should be tested by:**
Ontological sufficiency Ontological adequacy	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.3, REQ-4.4, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	85
Epistemological sufficiency Epistemological adequacy	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.2.1, REQ-3.1.2.1.3, REQ-3.1.2.1.6, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1.1, REQ-4.3, REQ-4.4.1, REQ-4.6.2, REQ-4.6.3	16
Expressibility	REQ-3.1.1.1.10.4, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13.2, REQ-3.1.2.1.3, REQ-3.1.2.1.4, REQ-3.1.2.1.6, REQ-3.1.2.1.8.2, REQ-3.2.1.5, REQ-4.2, REQ-4.3, REQ-4.4.1, REQ-4.6.3, REQ-4.7.1, REQ-4.7.2, REQ-4.8.3.1	15
Reasoning power	REQ-3.1.1, REQ-3.1.2, REQ-3.2.1, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10.3, REQ-3.1.2.1.3.2, REQ-3.1.2.1.5, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-4.1.1, REQ-4.7.1, REQ-4.3, REQ-4.6.1	14
Composability	REQ-3.1.1.1.2, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.5, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.2.1, REQ-4.3, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.5, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1	33
Selective power	REQ-3.1.1.1.8, REQ-3.1.2.1.5.2, REQ-3.1.2.1.3, REQ-3.2.1.1, REQ-4.3, REQ-4.4.1, REQ-4.6.1, REQ-4.6.3, REQ-4.7.2, REQ-4.8.1, REQ-4.8.3.1, REQ-4.9	12
Generalitive power	REQ-3.1, REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.8, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-	87

Test identifier	T-F-CS-UML-01	
	3.1.1.1.10.4, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.5.2, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1. REQ-4.6.2, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	

Table 22. Description of UML evaluation test T-F-CS-UML-02

Test identifier	T-F-CS-UML-02	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Requirements**	**Should be tested by:**
Extensibility	REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.6, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.3, REQ-3.1.1.1.13.2, REQ-3.1.1.1.8, REQ-3.1.1.1.9.3, REQ-3.1.1.1.10.1, REQ-3.1.2.1.5.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.7.1.2, REQ-4.8.1	13
Adaptability	REQ-3.1.1.1, REQ-3.1.1.2, REQ-3.1.1.3, REQ-3.1.1.1.9.3, REQ-3.1.1.1.10.1, REQ-3.1.2.1, REQ-3.1.2.2, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3.1	10
Universality	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.2.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.6, REQ-3.1.2.1.6.2, REQ-3.1.2.1.8, REQ-3.2.1.1, REQ-3.2.1.3, REQ-3.2.1.6, REQ-4.2, REQ-4.2.1, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.6.3	22

Descriptions of UML evaluation tests have been combined into description of suite of quality evaluation tests TS-F-UML-MagicDraw (Table 23). The number of requirements, which should be used to test the characteristics of functionality, has been calculated summarising requirements used for this aim in the tests of the suite:

Table 23. Description of suite of UML evaluation tests TS-F-UML-MagicDraw

Test suite identifier	TS-F-UML-MagicDraw	
Quality characteristic	**Tests**	**Should be tested by:**
Ontological sufficiency	T-F-IAS-UML-01, T-F-CS-UML-01	139
Ontological adequacy	T-F-IAS-UML-01, T-F-CS-UML-01	139
Epistemological sufficiency	T-F-IAS-UML-01, T-F-CS-UML-01	31
Epistemological adequacy	T-F-IAS-UML-01, T-F-CS-UML-01	31
Expressibility	T-F-IAS-UML-01, T-F-CS-UML-01	30
Reasoning power	T-F-IAS-UML-01, T-F-CS-UML-01	26
Composability	T-F-IAS-UML-01, T-F-CS-UML-01	45
Selective power	T-F-IAS-UML-01, T-F-CS-UML-01	18

Generalitive power	T-F-IAS-UML-01, T-F-CS-UML-01	137
Extensibility	T-F-IAS-UML-02, T-F-CS-UML-02	24
Adaptability	T-F-IAS-UML-02, T-F-CS-UML-02	23
Universality	T-F-IAS-UML-02, T-F-CS-UML-02	45

5.4.6.2. Description of Z Evaluation Tests

Descriptions of Z evaluation tests, which have been developed for content management and search components of Web portal, are presented in Table 24, Table 25:

Table 24. Description of Z evaluation test T-F-IAS-Z-01

Test identifier	T-F-IAS-Z-01	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Requirements**	**Should be tested by:**
Ontological sufficiency Ontological adequacy	REQ-1.1, REQ-1.2, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.12, REQ-1.13, REQ-1.13.1, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-16.2, REQ-1.17, REQ-1.18, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	54
Epistemological sufficiency Epistemological adequacy	REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.11, REQ-2.1.2, REQ-2.1.3, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.3, REQ-2.2.5.1, REQ-2.2.5.4	11
Expressibility	REQ-1.1, REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-1.12, REQ-1.12.1, REQ-1.13, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.13.5, REQ-1.13.6, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-16.2, REQ-1.17, REQ-1.17.1, REQ-1.18, REQ-2.1.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	63
Reasoning power	REQ-1.1, REQ-1.3, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.12.1, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.1, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.4, REQ-2.1.4.2, REQ-2.2.1, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6	25
Composability	REQ-1.1, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.1, REQ-2.2.1, REQ-2.2.5, REQ-2.2.6	14
Selective power	REQ-1.16.2, REQ-1.17, REQ-1.17.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.2.2, REQ-2.2.2.6, REQ-2.2.5.3, REQ-2.2.6.1	11
Generalitive power	REQ-1.9, REQ-1.9.1, REQ-1.10, REQ-1.10.1, REQ-1.12, REQ-	48

Test identifier	T-F-IAS-Z-01	
	1.12.1, REQ-1.13, REQ-1.13.1, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.13.5, REQ-1.13.6, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.17.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	

Table 25. Description of Z evaluation test T-F-IAS-Z-02

Test identifier	T-F-IAS-Z-02	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Requirements**	**Should be tested by:**
Extensibility	REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.12.1, REQ-1.13, REQ-1.13.1, REQ-1.13.5, REQ-1.13.6	8
Adaptability	REQ-1.16.2, REQ-2.1.2, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.5, REQ-2.2.2.6	6
Universality	REQ-1.1, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.1, REQ-2.2.1, REQ-2.2.5, REQ-2.2.6	14

Descriptions of Z evaluation tests, which have been developed for collaboration and workflow components of Web portal, are presented in Table 26, Table 27:

Table 26. Description of Z evaluation test T-F-CS-Z-01

Test identifier	T-F-CS-Z-01	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Requirements**	**Should be tested by:**
Ontological sufficiency Ontological adequacy	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.5.2, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7,	87

Test identifier	T-F-CS-Z-01	
	REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	
Epistemological sufficiency Epistemological adequacy	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.3.3, REQ-3.1.2.1.6, REQ-3.1.2.1.6.2, REQ-3.1.2.1.8, REQ-3.2.1.5, REQ-3.2.1.6, REQ-4.2, REQ-4.3, REQ-4.5.2, REQ-4.5.3	14
Expressibility	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.8, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.2, REQ-4.5.3, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	83
Reasoning power	REQ-3.1.1, REQ-3.1.2, REQ-3.1.1.1.2, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.9, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.3, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.2, REQ-3.1.2.1.2, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.5, REQ-4.1, REQ-4.1.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.6.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	49
Composability	REQ-3.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.9, REQ-3.1.1.1.10, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.2, REQ-3.1.2.1.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5.1, REQ-3.1.2.1.8, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.5, REQ-4.1, REQ-4.1.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.6.1, REQ-4.6.3, REQ-4.7, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	39
Selective power	REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.8, REQ-3.1.1.1.10, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.2.1.2, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.4, REQ-3.2.1.5, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.6, REQ-4.6.1, REQ-4.7, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3.1, REQ-4.9	34
Generalitive power	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.8, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-	84

Test identifier	T-F-CS-Z-01	
	3.1.1.1.10.4, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.5.2, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1. REQ-4.6.2, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	

Table 27. Description of Z evaluation test T-F-CS-Z-02

Test identifier	T-F-CS-Z-02	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Requirements**	**Should be tested by:**
Extensibility	REQ-3.1.1.1.9, REQ-3.1.1.1.9.3, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.2.1.3, REQ-3.1.2.1.8, REQ-3.2.1.1, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1	10
Adaptability	REQ-3.1.1.1, REQ-3.1.1.2, REQ-3.1.1.3, REQ-3.1.2.1, REQ-3.1.2.2, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3.1	8
Universality	REQ-3.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.9, REQ-3.1.1.1.10, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.2, REQ-3.1.2.1.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5.1, REQ-3.1.2.1.8, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.5, REQ-4.1, REQ-4.1.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.6.1, REQ-4.6.3, REQ-4.7, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	39

Descriptions of Z evaluation tests have been combined into description of suite of quality evaluation tests TS-F-Z-EVES (Table 28). The number of requirements, which should be used to test the characteristics of functionality, has been calculated summarising requirements used for this aim in the tests of the suite:

Table 28. Description of suite of Z evaluation tests TS-F-Z-ZEVES

Test suite identifier	TS-F-Z-ZEVES	
Quality characteristic	**Tests**	**Should be tested by:**
Ontological sufficiency	T-F-IAS-Z-01, T-F-CS-Z-01	141
Ontological adequacy	T-F-IAS-Z-01, T-F-CS-Z-01	141
Epistemological sufficiency	T-F-IAS-Z-01, T-F-CS-Z-01	25
Epistemological adequacy	T-F-IAS-Z-01, T-F-CS-Z-01	25
Expressibility	T-F-IAS-Z-01, T-F-CS-Z-01	146
Reasoning power	T-F-IAS-Z-01, T-F-CS-Z-01	74
Composability	T-F-IAS-Z-01, T-F-CS-Z-01	53

Selective power	T-F-IAS-Z-01, T-F-CS-Z-01	45
Generalitive power	T-F-IAS-Z-01, T-F-CS-Z-01	132
Extensibility	T-F-IAS-Z-02, T-F-CS-Z-02	18
Adaptability	T-F-IAS-Z-02, T-F-CS-Z-02	14
Universality	T-F-IAS-Z-02, T-F-CS-Z-02	53

5.4.7. Testing

This section provides the results of UML and Z testing. In every test for every sub-characteristic of functionality successfully specified requirements are provided.

5.4.7.1. UML Testing

The results of UML testing by tests T-F-IAS-UML-01, T-F-IAS-UML-02, T-F-CS-UML-01, T-F-CS-UML-02 are presented in Table 29, Table 30, Table 31, Table 32:

Table 29. UML evaluation test T-F-IAS-UML-01

Test identifier	T-F-IAS-UML-01	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Specified requirements**	**Tested by [15]:**
Ontological sufficiency	REQ-1.1, REQ-1.2, REQ-1.3, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.9, REQ-1.10, REQ-1.11, REQ-1.12, REQ-1.13, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.18, REQ-2.1.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	53
Ontological adequacy	REQ-1.1, REQ-1.2, REQ-1.3, REQ-1.5, REQ-1.7, REQ-1.9, REQ-1.10, REQ-1.11, REQ-1.12, REQ-1.13, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.18, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	49
Epistemological sufficiency	REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-2.1.3, REQ-2.1.4.1, REQ-2.2.2, REQ-2.2.2.1,	14

[15] Number of to the group G_ξ of $N(\xi)$ belonging requirements, which were tested by the characteristic ξ. In other words, it is the number of successfully specified requirements. It shows how many requirements from the requirements that belong to the group G_ξ (see 5.4.6) were specified using the particular specification language (UML or Z).

Test identifier	T-F-IAS-UML-01	
	REQ-2.2.3, REQ-2.2.5.1, REQ-2.2.5.4	
Epistemological adequacy	REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-2.1.3, REQ-2.1.4.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.3, REQ-2.2.5.1, REQ-2.2.5.4	14
Expressibility	REQ-1.4, REQ-1.13, REQ-1.14, REQ-1.15, REQ-2.1.2.2, REQ-2.2.2.5, REQ-2.2.5, REQ-2.2.5.6, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6.2, REQ-2.2.6.3	12
Reasoning power	REQ-1.3, REQ-1.4, REQ-1.15, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.2.1, REQ-2.2.2.6, REQ-2.2.5.8, REQ-2.2.5.9	9
Composability	REQ-1.5, REQ-1.15, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.2.2, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3	11
Selective power	REQ-1.16.2, REQ-1.17, REQ-2.1.2.1	3
Generalitive power	REQ-1.9, REQ-1.9.1, REQ-1.10, REQ-1.10.1, REQ-1.12, REQ-1.12.1, REQ-1.13, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.13.5, REQ-1.13.6, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.17.1, REQ-2.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.3, REQ-2.2, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	45

Table 30. UML evaluation test T-F-IAS-UML-02

Test identifier	T-F-IAS-UML-02	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Specified requirements**	**Tested by:**
Extensibility	REQ-1.10.1, REQ-1.13, REQ-1.13.5, REQ-1.13.6, REQ-1.15, REQ-2.1.2, REQ-2.1.3, REQ-2.2.2, REQ-2.2.3	9
Adaptability	REQ-1.10.1, REQ-1.13.5, REQ-1.13.6, REQ-1.16.2, REQ-2.1.2, REQ-2.1.2.3, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.6	10
Universality	REQ-1.1, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.6	22

Table 31. UML evaluation test T-F-CS-UML-01

Test identifier	T-F-CS-UML-01	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Specified requirements**	**Tested by:**
Ontological sufficiency	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-	82

Test identifier	T-F-CS-UML-01	
	3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1.1, REQ-4.3, REQ-4.4, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	
Ontological adequacy	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.4, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	77
Epistemological sufficiency	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.2.1, REQ-3.1.2.1.3, REQ-3.1.2.1.6, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1.1, REQ-4.3, REQ-4.4.1, REQ-4.6.2, REQ-4.6.3	16
Epistemological adequacy	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.2.1, REQ-3.1.2.1.3, REQ-3.1.2.1.6, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.2.1.1, REQ-4.3, REQ-4.4.1, REQ-4.6.2, REQ-4.6.3	15
Expressibility	REQ-3.1.1.1.10.4, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13.2, REQ-3.1.2.1.3, REQ-3.1.2.1.4, REQ-3.1.2.1.6, REQ-3.2.1.5, REQ-4.2, REQ-4.3, REQ-4.6.3, REQ-4.7.1, REQ-4.7.2, REQ-4.8.3.1	13
Reasoning power	REQ-3.1.1, REQ-3.1.2, REQ-3.2.1, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10.3, REQ-3.1.2.1.3.2, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-4.1.1, REQ-4.7.1, REQ-4.3, REQ-4.6.1	13
Composability	REQ-3.1.1.1.2, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.3, REQ-4.5, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1	28
Selective power	REQ-3.1.1.1.8, REQ-3.1.2.1.5.2, REQ-3.1.2.1.3, REQ-3.2.1.1, REQ-4.3, REQ-4.4.1, REQ-4.6.1, REQ-4.6.3, REQ-4.7.2, REQ-4.8.1, REQ-4.9	11
Generalitive power	REQ-3.1, REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-	83

Test identifier	T-F-CS-UML-01	
	3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5.1, REQ-3.1.2.1.5.2, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1. REQ-4.6.2, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	

Table 32. UML evaluation test T-F-CS-UML-02

Test identifier	T-F-CS-UML-02	
Test execution conditions	**1.** Tested language: UML. **2.** Tool to produce specifications: MagicDraw UML 10.0. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Specified requirements**	**Tested by:**
Extensibility	REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.6, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.3, REQ-3.1.1.1.13.2, REQ-3.1.1.1.8, REQ-3.1.1.1.9.3, REQ-3.1.1.1.10.1, REQ-3.1.2.1.5.2, REQ-4.2, REQ-4.2.1, REQ-4.7.1.2, REQ-4.8.1	14
Adaptability	REQ-3.1.1.1, REQ-3.1.1.2, REQ-3.1.1.3, REQ-3.1.1.1.9.3, REQ-3.1.1.1.10.1, REQ-3.1.2.1, REQ-3.1.2.2, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3.1	10
Universality	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.2.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.6, REQ-3.1.2.1.6.2, REQ-3.1.2.1.8, REQ-3.2.1.1, REQ-3.2.1.3, REQ-3.2.1.6, REQ-4.2, REQ-4.2.1, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.6.3	18

UML evaluation tests have been combined into suite of quality evaluation tests TS-F-UML-MagicDraw (Table 33), and for every sub-characteristic of functionality total number of successfully specified requirements have been calculated by summarisation of the results of UML testing:

Table 33. Suite of UML evaluation tests TS-F-UML-MagicDraw

Test suite identifier	TS-F-UML-MagicDraw	
Quality characteristic	**Tests**	**Tested by:**
Ontological sufficiency	T-F-IAS-UML-01, T-F-CS-UML-01	135
Ontological adequacy	T-F-IAS-UML-01, T-F-CS-UML-01	126
Epistemological sufficiency	T-F-IAS-UML-01, T-F-CS-UML-01	30
Epistemological adequacy	T-F-IAS-UML-01, T-F-CS-UML-01	29
Expressibility	T-F-IAS-UML-01, T-F-CS-UML-01	25
Reasoning power	T-F-IAS-UML-01, T-F-CS-UML-01	22
Composability	T-F-IAS-UML-01, T-F-CS-UML-01	39
Selective power	T-F-IAS-UML-01, T-F-CS-UML-01	14

Generalitive power	T-F-IAS-UML-01, T-F-CS-UML-01	128
Extensibility	T-F-IAS-UML-02, T-F-CS-UML-02	23
Adaptability	T-F-IAS-UML-02, T-F-CS-UML-02	20
Universality	T-F-IAS-UML-02, T-F-CS-UML-02	40

5.4.7.2. Z Testing

The results of Z testing by tests T-F-IAS-Z-01, T-F-IAS-Z-02, T-F-CS-Z-01, T-F-CS-Z-02 are presented in Table 34, Table 35, Table 36, Table 37:

Table 34. Z evaluation test T-F-IAS-Z-01

Test identifier	T-F-IAS-Z-01	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Specified requirements**	**Tested by:**
Ontological sufficiency	REQ-1.1, REQ-1.2, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.12, REQ-1.13, REQ-1.13.2, REQ-1.13.3, REQ-1.13.4, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-16.2, REQ-1.17, REQ-1.18, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	53
Ontological adequacy	REQ-1.1, REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.13, REQ-1.13.3, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.6, REQ-2.2.2.6.5	36
Epistemological sufficiency	REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.11, REQ-2.1.2, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.3, REQ-2.2.5.1, REQ-2.2.5.4	10
Epistemological adequacy	REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.8, REQ-1.11, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.5.1	8
Expressibility	REQ-1.1, REQ-1.2, REQ-1.3, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.11, REQ-1.12, REQ-1.12.1, REQ-1.13.4, REQ-1.13.5, REQ-1.13.6, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-16.2, REQ-1.17, REQ-1.17.1, REQ-1.18, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.2, REQ-2.1.4.3, REQ-2.2.1, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.6, REQ-2.2.3, REQ-2.2.4, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	55
Reasoning power	REQ-1.1, REQ-1.3, REQ-1.4, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.1.1, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.4, REQ-2.2.1, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6	23
Composability	REQ-1.1, REQ-1.5, REQ-1.6, REQ-1.7, REQ-1.14, REQ-1.15,	13

Test identifier	T-F-IAS-Z-01	
	REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.2.1, REQ-2.2.5, REQ-2.2.6	
Selective power	REQ-1.16.2, REQ-1.17, REQ-1.17.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.2.2, REQ-2.2.2.6, REQ-2.2.5.3, REQ-2.2.6.1	11
Generalitive power	REQ-1.9, REQ-1.9.1, REQ-1.10, REQ-1.10.1, REQ-1.12, REQ-1.12.1, REQ-1.13, REQ-1.13.3, REQ-1.13.4, REQ-1.13.5, REQ-1.13.6, REQ-1.16, REQ-1.16.1, REQ-1.16.2, REQ-1.17.1, REQ-2.1.2, REQ-2.1.2.1, REQ-2.1.2.2, REQ-2.1.2.3, REQ-2.1.4, REQ-2.1.4.1, REQ-2.1.4.3, REQ-2.2.2, REQ-2.2.2.1, REQ-2.2.2.2, REQ-2.2.2.3, REQ-2.2.2.4, REQ-2.2.2.5, REQ-2.2.2.6, REQ-2.2.5, REQ-2.2.5.1, REQ-2.2.5.2, REQ-2.2.5.3, REQ-2.2.5.4, REQ-2.2.5.5, REQ-2.2.5.6, REQ-2.2.5.7, REQ-2.2.5.8, REQ-2.2.5.9, REQ-2.2.6, REQ-2.2.6.1, REQ-2.2.6.2, REQ-2.2.6.3, REQ-2.2.6.4, REQ-2.2.6.5	45

Table 35. Z evaluation test T-F-IAS-Z-02

Test identifier	T-F-IAS-Z-02	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: content management requirements, search requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Specified requirements**	**Tested by:**
Extensibility	REQ-1.8, REQ-1.9.1, REQ-1.10.1, REQ-1.12.1, REQ-1.13.5, REQ-1.13.6	6
Adaptability	REQ-2.2.2.1, REQ-2.2.2.5	2
Universality	REQ-1.1, REQ-1.5, REQ-1.7, REQ-1.14, REQ-1.15, REQ-1.16, REQ-1.16.1, REQ-1.17, REQ-1.18, REQ-2.2.1, REQ-2.2.5, REQ-2.2.6	12

Table 36. Z evaluation test T-F-CS-Z-01

Test identifier	T-F-CS-Z-01	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled directly (without using flexibility mechanisms).	
Expected results	Elementary characteristics under testing: ontological sufficiency, ontological adequacy, epistemological sufficiency, epistemological adequacy, expressibility, reasoning power, composability, selective power, generalitive power.	
Quality characteristic	**Specified requirements**	**Tested by:**
Ontological sufficiency	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-	81

Test identifier	T-F-CS-Z-01	
	4.3, REQ-4.4, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	
Ontological adequacy	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.5, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-4.1, REQ-4.1.2, REQ-4.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.3, REQ-4.9	51
Epistemological sufficiency	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.2, REQ-3.1.2.1.8, REQ-3.2.1.5, REQ-3.2.1.6, REQ-4.2, REQ-4.3, REQ-4.5.2, REQ-4.5.3	13
Epistemological adequacy	REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.2, REQ-3.2.1.5, REQ-3.2.1.6, REQ-4.2, REQ-4.3	10
Expressibility	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.7, REQ-3.1.1.1.8, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.1, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.2, REQ-4.5.3, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	77
Reasoning power	REQ-3.1.1, REQ-3.1.2, REQ-3.1.1.1.2, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.9, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.3, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.2, REQ-3.1.2.1.2, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.5.1, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.5, REQ-4.1, REQ-4.1.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.6.1, REQ-4.6.3, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	47
Composability	REQ-3.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.9, REQ-3.1.1.1.10, REQ-3.1.1.1.11, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.2, REQ-3.1.2.1.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5.1, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.5, REQ-4.1, REQ-4.1.2, REQ-4.3, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.6.1, REQ-4.6.3, REQ-4.7, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	38
Selective power	REQ-3.1.1.1.3, REQ-3.1.1.1.4, REQ-3.1.1.1.5, REQ-3.1.1.1.6, REQ-3.1.1.1.7, REQ-3.1.1.1.8, REQ-3.1.1.1.10, REQ-3.1.1.1.10.4, REQ-3.1.1.1.11, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2,	32

Test identifier	T-F-CS-Z-01	
	REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.2.1.2, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.4, REQ-3.2.1.5, REQ-4.3, REQ-4.4.1, REQ-4.6, REQ-4.6.1, REQ-4.7, REQ-4.7.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3.1, REQ-4.9	
Generalitive power	REQ-3.1.1, REQ-3.1.1.1, REQ-3.1.1.1.1, REQ-3.1.1.1.2, REQ-3.1.1.1.7, REQ-3.1.1.1.9, REQ-3.1.1.1.9.1, REQ-3.1.1.1.9.2, REQ-3.1.1.1.9.3, REQ-3.1.1.1.9.4, REQ-3.1.1.1.9.5, REQ-3.1.1.1.9.6, REQ-3.1.1.1.9.7, REQ-3.1.1.1.10, REQ-3.1.1.1.10.2, REQ-3.1.1.1.10.3, REQ-3.1.1.1.10.4, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.1.1.13.1, REQ-3.1.1.1.13.2, REQ-3.1.2, REQ-3.1.2.1, REQ-3.1.2.1.1, REQ-3.1.2.1.2, REQ-3.1.2.1.3, REQ-3.1.2.1.3.1, REQ-3.1.2.1.3.2, REQ-3.1.2.1.3.3, REQ-3.1.2.1.4, REQ-3.1.2.1.5, REQ-3.1.2.1.5.1, REQ-3.1.2.1.5.2, REQ-3.1.2.1.6, REQ-3.1.2.1.6.1, REQ-3.1.2.1.6.2, REQ-3.1.2.1.7, REQ-3.1.2.1.7.1, REQ-3.1.2.1.8, REQ-3.1.2.1.8.1, REQ-3.1.2.1.8.2, REQ-3.2.1, REQ-3.2.1.1, REQ-3.2.1.2, REQ-3.2.1.3, REQ-3.2.1.4, REQ-3.2.1.5, REQ-3.2.1.6, REQ-3.2.1.7, REQ-3.2.1.7.1, REQ-4.1.1, REQ-4.1.2, REQ-4.2, REQ-4.2.1, REQ-4.2.1.1, REQ-4.4, REQ-4.4.1, REQ-4.5, REQ-4.5.1, REQ-4.5.2, REQ-4.5.3, REQ-4.5.3.1, REQ-4.6, REQ-4.6.1, REQ-4.6.1.1. REQ-4.6.2, REQ-4.7, REQ-4.7.1, REQ-4.7.1.1, REQ-4.7.1.2, REQ-4.8, REQ-4.8.1, REQ-4.8.2, REQ-4.8.3, REQ-4.8.3.1, REQ-4.9, REQ-4.9.1	78

Table 37. Z evaluation test T-F-CS-Z-02

Test identifier	T-F-CS-Z-02	
Test execution conditions	**1.** Tested language: Z. **2.** Tool to produce specifications: Z/EVES 2.1. **3.** Requirements should be modelled by groups: collaboration requirements, workflow requirements. **4.** Requirements should be modelled using language flexibility mechanisms.	
Expected results	Elementary characteristics under testing: extensibility, adaptability, universality.	
Quality characteristic	**Specified requirements**	**Tested by:**
Extensibility	REQ-3.1.1.1.9, REQ-3.1.1.1.10, REQ-3.1.2.1.3, REQ-3.1.2.1.8, REQ-3.2.1.1, REQ-4.2	6
Adaptability	REQ-3.1.1.1, REQ-3.1.2.1, REQ-4.8.1, REQ-4.8.2	4
Universality	REQ-3.1.1, REQ-3.1.1.1.9, REQ-3.1.1.1.10, REQ-3.1.1.1.12, REQ-3.1.1.1.12.1, REQ-3.1.1.1.13, REQ-3.1.2, REQ-3.2.1, REQ-4.1, REQ-4.1.2, REQ-4.7, REQ-4.8, REQ-4.8.1, REQ-4.8.3, REQ-4.9	15

Z evaluation tests have been combined into suite of quality evaluation tests TS-F-Z-EVES (Table 38), and for every sub-characteristic of functionality total number of successfully specified requirements have been calculated by summarisation of the results of Z testing:

Table 38. Suite of Z evaluation tests TS-F-Z-ZEVES

Test suite identifier	TS-F-Z-ZEVES	
Quality characteristic	**Tests**	**Tested by:**
Ontological sufficiency	T-F-IAS-Z-01, T-F-CS-Z-01	134
Ontological adequacy	T-F-IAS-Z-01, T-F-CS-Z-01	87
Epistemological sufficiency	T-F-IAS-Z-01, T-F-CS-Z-01	23
Epistemological adequacy	T-F-IAS-Z-01, T-F-CS-Z-01	18
Expressibility	T-F-IAS-Z-01, T-F-CS-Z-01	132
Reasoning power	T-F-IAS-Z-01, T-F-CS-Z-01	70
Composability	T-F-IAS-Z-01, T-F-CS-Z-01	51
Selective power	T-F-IAS-Z-01, T-F-CS-Z-01	43
Generalitive power	T-F-IAS-Z-01, T-F-CS-Z-01	123

Extensibility	T-F-IAS-Z-02, T-F-CS-Z-02	12
Adaptability	T-F-IAS-Z-02, T-F-CS-Z-02	6
Universality	T-F-IAS-Z-02, T-F-CS-Z-02	27

5.4.8. Interpretation of the Results of UML and Z Functionality Testing

5.4.8.1. Evaluation of Elementary Characteristics of Functionality

It can be seen from the quality evaluation tests that for all characteristics of functionality the frequency $q(\xi_i)$ with which the feature $L(\xi_i)$ of the language L described by characteristic ξ_i will become necessary specifying the population of Web portals is equal to 1, because for all characteristics at least one requirement in the group $G(\xi_i)$ is mandatory or alternative. The only characteristic, for which $q(\xi_i)$ should be calculated is extensibility of Z specification language (the test T-F-IAS-Z-02), because for extensibility all to the group $G(\xi_i)$ belonging requirements are optional. Let‘s demonstrate how the expected frequency $q(\xi_i)$ has been calculated. First of all, expected frequency $q_i(\xi_i)$ for every requirement r_i, which belong to the group $G(\xi_i)$ has been calculated. In order to define $q_i(\xi_i)$ for the r_i, it is necessary to start from the initial node of the feature model and proceed down the feature model up to the terminal feature that generates r_i (see 4.2.3.7). For example, requirement REQ-1.8 (Portal **Pt** content items ***CntItm*** could be classified into library items ***LbItm*** and document items ***DocItm***.) has been derived from the features, which are presented in Table 39.

Table 39. Features for requirement REQ-1.8

Level in generic feature model	**Feature**	**Type**	**Developer Portal** (see 5.4.4.1)	**ILP portal** (see 5.4.4.2)	**HKU portal** (see 5.4.4.3)
…	…	…	…	…	…
1.2.1	Item	Mandatory	Yes	Yes	Yes
1.2.1.1	Library item	Optional	Yes	Yes	No
1.2.1.1.1	Image	Optional	Yes	Yes	No
1.2.1.1.2	Static text	Optional	Yes	Yes	No
1.2.1.1.3	Banner	Optional	No	Yes	No
1.2.1.1.4	Video	Optional	No	Yes	No
1.2.1.2	Document item	Optional	No	Yes	Yes
1.2.1.2.1	File .DOC	Optional	No	Yes	Yes
1.2.1.2.2	File .PDF	Optional	No	Yes	Yes
…	…	…		…	…

It is necessary to start from feature 1.2.1.2.2 and proceed until terminal feature 1.2.1, which generates requirement REQ-1.8. Features 1.2.1.1.3 and 1.2.1.1.4 are present in 1 from 3 Web portals, while all the other features (1.2.1.1, 1.2.1.1.1, 1.2.1.1.2, 1.2.1.2, 1.2.1.2.1, 1.2.1.2.2) are present in 2 from 3 Web portals. Feature 1.2.1 is mandatory, thus its expected frequency, with which this feature will become necessary specifying generic Web portal is equal to 1. Thus, $q_i(\xi_i)$ = 1/3*1/3*2/3*2/3*2/3*2/3*2/3*2/3*1 = 0,009.

Expected frequencies $q_i(\xi_i)$ for all the other to the group $G(\xi_i)$ belonging requirements have been calculated in the analogous way, that is analysing the corresponding features of every Web portal under experiment and identification of expected frequencies for every feature (see 4.2.3.7):

- for requirement REQ-1.9.1 $q_i(\xi_i) = 1/3*1/3*2/3*2/3*2/3 = 0{,}0313$;
- for requirement REQ-1.10.1 $q_i(\xi_i) = 2/3*2/3*2/3 = 0{,}2904$;
- for requirement REQ-1.12.1 $q_i(\xi_i) = 2/3*2/3*2/3*1 = 0{,}2904$;
- for requirement REQ-1.13.5 $q_i(\xi_i) = 1/3*1/3*1/3 = 0{,}0359$;
- for requirement REQ-1.13.6 $q_i(\xi_i) = 3/3*3/3*3/3 = 0{,}2904$.

Expected frequencies $q(\xi_i)$ for extensibility of Z specification language has been calculated by the formula (2):

$$q(\xi_i) = 1 - \prod_{i=1}^{n_\xi} (1 - q_i(\xi_i)) = 1 - (1 - 0{,}009)(1 - 0{,}0313)(1 - 0{,}2904)(1 - 0{,}2904)(1 - 0{,}0359)(1 - 0{,}2904) = 0{,}1315$$

To calculate the value of $q(\xi_i)$ for extensibility of Z specification language for the whole suite of evaluation tests taking into account all tests included into this suite the formula analogous to the formula (2) has been used once again:

$$q(\xi_i) = 1 - \prod_{i=1}^{n_\xi} (1 - q_i(\xi_i)) = 1 - (1 - 1)(1 - 0{,}1315)(1 - 1)(1 - 1) = 1.$$

The results of the evaluation of the expected frequency $q(\xi_i)$ with which the feature $L(\xi_i)$ of the language L described by characteristic ξ_i will become necessary specifying the population of Web portals are presented in Table 40.

Table 40. Results of $q(\xi_i)$ evaluation for suites of quality evaluation test

Elementary characteristic, (ξ)	**Z evaluation for suite TS-TS-F-Z-ZEVES, $q(\xi_i)$**	**UML evaluation for suite TS-F-UML-MagicDraw, $q(\xi_i)$**
Ontological sufficiency	1	1
Epistemological sufficiency	1	1
Expressibility	1	1
Reasoning power	1	1
Composability	1	1
Ontological adequacy	1	1
Epistemological adequacy	1	1
Selective power	1	1
Generalitive power	1	1
Universality	1	1
Adaptability	1	1
Extensibility	1	1

The expected frequency $p(\xi_i)$ with which $L(\xi_i)$ will be sufficient specifying the current population of Software Systems has been calculated for the whole suite of quality evaluation

tests by the formula (3). For example, the value of $p(\xi_i)$ for UML ontological sufficiency ξ_i has been calculated comparing total number $N_+(\xi_i)$ of successfully specified to the group $G(\xi_i)$ belonging requirements (see the suite TS-F-UML-MagicDraw) to the total number of requirements $N(\xi_i)$, which were planned to test for this characteristic and were indicated in the descriptions of the corresponding quality evaluation tests (see 5.4.6).

The results of the evaluation of the expected frequency $p(\xi_i)$ with which the feature $L(\xi_i)$ of the language L described by characteristic ξ_i will be sufficient specifying the population of Web portals are presented in (Table 41):

Table 41. Results of $p(\xi_i)$ evaluation for suites of quality evaluation test

Elementary characteristic, (ξ)	**UML evaluation for suite TS-F-UML-MagicDraw, $p(\xi_i)$**	**Z evaluation for suite TS-TS-F-Z-ZEVES, $p(\xi_i)$**
Ontological sufficiency	135/139=0,9712	134/141=0,9504
Ontological adequacy	126/139=0,9065	87/141=0,617
Epistemological sufficiency	30/31=0,9677	23/25=0,92
Epistemological adequacy	29/31=0,9355	18/25=0,72
Expressibility	25/30=0,8333	132/146=0,9041
Reasoning power	22/26=0,8462	70/74=0,9459
Composability	39/45=0,8666	51/53=0,9623
Selective power	14/18=0,7777	43/45=0,9555
Generalitive power	128/137=0,9343	123/132=0,9318
Extensibility	23/24=0,9583	12/18=0,6666
Adaptability	20/23=0,8696	6/14=0,4286
Universality	40/45=0,8888	27/53=0,5094

The evaluation for every elementary characteristic (expected frequency $p(\xi)$ with which $L(\xi_i)$ will become necessary specifying the current population of Software Systems) has been calculated as a production of expected frequencies $q(\xi_i)$ and $p(\xi_i)$ (Table 42):

Table 42. Results of elementary characteristics evaluation

Elementary characteristic, (ξ)	**Evaluation for UML, $p(\xi)=p(\xi_i)*q(\xi_i)$**	**Evaluation for Z, $p(\xi)=p(\xi_i)*q(\xi_i)$**
Ontological sufficiency	0,9712	0,9504
Ontological adequacy	0,9065	0,617
Epistemological sufficiency	0,9677	0,92
Epistemological adequacy	0,9355	0,72
Expressibility	0,8333	0,9041
Reasoning power	0,8462	0,9459
Composability	0,8666	0,9623
Selective power	0,7777	0,9555
Generalitive power	0,9343	0,9318
Extensibility	0,9583	0,6666
Adaptability	0,8696	0,4286
Universality	0,8888	0,5094

5.4.8.2. Aggregation of Elementary Characteristics

The values of high-level functionality characteristics have been calculated using in 4.2.4.4 developed aggregation techniques. The high-level functionality characteristics are: semantic sufficiency, completeness, expressive adequacy, suitability, and flexibility.

Semantic sufficiency has been calculated using technique to aggregate orthogonal sub-characteristics ξ_1 (ontological sufficiency) and ξ_2 (epistemological sufficiency) of different importance (see formula (36)): $p(\xi) = (1-q(\xi_1)(1-p(\xi_1)))(1-q(\xi_2)(1-p(\xi_2)))$. It was considered that the expected frequency $q(\xi_1)$ with which ξ_1 will become necessary for any project P is equal to 1 and the expected frequency $q(\xi_2)$ with which ξ_2 will become necessary for any project P is 0,8.

Completeness has been calculated using technique to aggregate orthogonal sub-characteristics ξ_1 (semantic sufficiency), ξ_2 (expressibility), ξ_3 (reasoning power) and ξ_4 (composability) of different importance (see formula (36)): $p(\xi) = (1-q(\xi_1)(1-p(\xi_1)))(1-q(\xi_2)(1-p(\xi_2)))(1-q(\xi_3)(1-p(\xi_3)))(1-q(\xi_4)(1-p(\xi_4)))$. It was considered that the expected frequencies $q(\xi_1)$ and $q(\xi_2)$ with which ξ_1 and ξ_2 accordingly will become necessary for any project P are equal to 1, the expected frequency $q(\xi_3)$ with which ξ_3 will become necessary for any project P is 0,7, and the expected frequency $q(\xi_4)$ with which ξ_4 will become necessary for any project P is also 0,7.

Expressive adequacy has been calculated using technique to aggregate orthogonal sub-characteristics ξ_1 (ontological adequacy), ξ_2 (epistemological adequacy), ξ_3 (selective power) and ξ_4 (generalitive power) of different importance (see formula (36)): $p(\xi) = (1-q(\xi_1)(1-p(\xi_1)))(1-q(\xi_2)(1-p(\xi_2)))(1-q(\xi_3)(1-p(\xi_3)))(1-q(\xi_4)(1-p(\xi_4)))$. It was considered that the expected frequencies $q(\xi_1)$ and $q(\xi_2)$ with which ξ_1 and ξ_2 accordingly will become necessary for any project P are equal to 1, the expected frequency $q(\xi_3)$ with which ξ_3 will become necessary for any project P is 0,6, and expected frequency $q(\xi_4)$ with which ξ_4 will become necessary for any project P is 0,5.

Suitability has been calculated using technique to aggregate orthogonal sub-characteristics ξ_1 (completeness) and ξ_2 (expressive adequacy) (see formula (35)): $p(\xi) = p(\xi_1)p(\xi_2)$.

Flexibility has been calculated using technique to aggregate non-orthogonal alternative sub-characteristics ξ_1 (universality), ξ_2 (adaptability) and ξ_3 (extensibility) (see formula (38)): $p(\xi) = p(\xi_1)q(\xi_1)+p(\xi_2)q(\xi_2)+p(\xi_3)q(\xi_3)-p(\xi_1)q(\xi_1)p(\xi_2)q(\xi_2)p(\xi_3)q(\xi_3)$. It was considered consider that the expected frequency $q(\xi_1)$ with which ξ_1 will become necessary for any project P is 0,5, the expected frequency $q(\xi_2)$ with which ξ_2 will become necessary for any project P is 0,6, and the expected frequency $q(\xi_3)$ with which ξ_3 will become necessary for any project P is 0,8.

And, finally, functionality has been calculated using technique to aggregate non-orthogonal supplementary sub-characteristics ξ_1 (suitability) and ξ_2 (flexibility) (see formula (37)): $p(\xi) = p(\xi_1)+p(\xi_2)q(\xi_2) = p(\xi_1)+p(\xi_2)(1-p(\xi_1)) = p(\xi_1)+p(\xi_2)-p(\xi_1)p(\xi_2)$.

Table 43 has the results of all functionality characteristics evaluation, and the overall evaluation of UML and Z functionality.

Table 43. Results of functionality evaluation

Level	Characteristic	Elementary/ Aggregated	Aggregation technique	Evaluation for UML	Evaluation for Z
1.	**Functionality**	**Aggregated**	Technique to aggregate non-orthogonal supplementary sub-characteristics	**0,9502**	**0,7149**
1.1	**Suitability**	**Aggregated**	Technique to aggregate orthogonal sub-characteristics of the same importance	**0,4531**	**0,3146**
1.1.1	**Completeness**	**Aggregated**	Technique to aggregate orthogonal sub-characteristics of different importance	**0,6378**	**0,7534**
1.1.1.1	**Semantic sufficiency**	**Aggregated**	Technique to aggregate orthogonal sub-characteristics of different importance	**0,9461**	**0,8896**
1.1.1.1.1	Ontological sufficiency	Elementary	-	0,9712	0,9504
1.1.1.1.2	Epistemological sufficiency	Elementary	-	0,9677	0,92
1.1.1.2	Expressibility	Elementary	-	0,8333	0,9041
1.1.1.3	Reasoning power	Elementary	-	0,8462	0,9459
1.1.1.4	Composability	Elementary	-	0,8666	0,9623
1.1.2	**Expressive adequacy**	**Aggregated**	Technique to aggregate orthogonal sub-characteristics of different importance	**0,7104**	**0,4176**
1.1.2.1	Ontological adequacy	Elementary	-	0,9065	0,617
1.1.2.2	Epistemological adequacy	Elementary	-	0,9355	0,72
1.1.2.3	Selective power	Elementary	-	0,7777	0,9555
1.1.2.4	Generalitive power	Elementary	-	0,9343	0,9318
1.2	**Flexibility**	**Aggregated**	Technique to aggregate non-orthogonal alternative sub-characteristics	**0,9089**	**0,5841**
1.2.1	Universality	Elementary	-	0,8888	0,5094
1.2.2	Adaptability	Elementary	-	0,8696	0,4286
1.2.3	Extensibility	Elementary	-	0,9583	0,6666

Thus, the experiment demonstrates that, although the ontological and epistemological adequacies of the Z language are relatively low, it has high enough semantic sufficiency. The flexibility of the Z language is very low, but it is partly compensated by its relatively high completeness. On the other hand, although UML 2.0 has high enough ontological and

epistemological adequacies its expressive adequacy is not very high, because of the relatively low selective power. The composability of UML 2.0 and its reasoning power also are lower than of the Z language. UML 2.0 has higher extensibility than of the Z language. On the other hand, Z language is more expressible and allows expressing more classes of formulas than UML 2.0.

5.5. Conclusions

First of all, carried out experiment demonstrates that in the dissertation proposed framework to evaluate internal quality can be used to evaluate functional characteristics of specification languages (for example, UML and Z). From the evaluation of UML and Z functionality it can be concluded that both these languages can be successfully used to specify functional requirements of generic Web portal. Although the evaluation of functionality for Z is lower, than for UML, mathematical background of Z makes it more complete language. That means that it can be successfully used to specify all the main components of the linguistic system (concepts, composite concepts, statements, and reasoning apparatus) (see section 4.2.1.2), because expressibility of Z allows expressing more classes of formulas than UML, high semantic sufficiency and high composability of Z means that it can be used to specify most of in the requirements present basic concepts and to construct composite concepts, high reasoning power of Z means that is can be successfully used to derive new statements about properties of concepts and their compositions. However, Z has limited expressive adequacy, and, thus, in this language created specifications are very hard to read for the users who are unfamiliar with Z mathematical background. This is one of the main reasons why Z is not widely used in practice. UML graphical notation, which has separate constructs for almost every concept, composite concept and statement, makes specifications visual and usable even for the users, who do not know all the peculiarities of this specification language. Besides, UML is much more flexible language, its universality, adaptability and extensibility mechanisms allow expressing concepts, composite concepts or statements, which could not be specified directly, using existing language constructs.

Another conclusion that follows from the results of experimental evaluation is that even evaluation of the particular aspects of the particular specification language allows identification of shortcomings and strengths of the current specification languages, and preparing of recommendations how to use the particular language in more appropriate way as well as how to improve it. Thus, we conclude that a long-time research program for evaluation of specification languages quality is purposeful, because such program will significantly contribute to the entire theory of specification, and even programming languages.

6. Conclusions

1. All in scientific literature proposed approaches to evaluate quality of specification languages are incomplete, because they propose neither theoretical ground for separation and evaluation of quality characteristics, nor precise metrics for their measurement or scale for the interpretation and aggregation of the results of measurements. At the moment, no commonly accepted agreement exists about the collection of specification language internal quality characteristics, their names and their taxonomy. It is true for programming languages, too. An attempt to develop the taxonomy of quality characteristics and their measurement procedures is made in quality model based approach. However, this taxonomy is not enough systematic and exhaustive, and the proposed quality model itself is only sketched. Even the proposed quality characteristics are not precisely defined, and it is not proposed how exactly they should be measured. Besides, language characteristics and supporting tools characteristics are considered together.
2. It follows from the results of comparative analysis is that none of known approaches pays attention to relativism of quality concept and dependency on the particular project requirements and peculiarities. No approach exists, in which quality in use is separated from internal quality, and which proposes exhaustive procedure to evaluate quality in use. Thus, we conclude that none of known approaches is mature enough theoretically and is not developed enough to be used in practice to evaluate quality of a specification language.
3. The framework to evaluate specification languages quality should be based on the appropriate quality model, because it ensures obtaining of objective evaluation results. This requires having an exhaustive taxonomy. In known approaches to evaluate quality of specification languages only first small steps are made in this direction. The most valuable results that can be used for the further research have been obtained in ontological categories based approach and quality model-based approach.
4. Internal quality of a specification language L describes the quality that is independent from any context of use. Because of the imprecise nature of quality characteristics, it is reasonable to define such quality as the expected frequency with which the language L will satisfy the requirements of any imaginable project P. In an analogous way can be defined also all characteristics of internal quality, including elementary ones.
5. Because characteristics of internal quality of the language L form a large hierarchical structure F, in order to evaluate internal quality it is necessary to aggregate sub-characteristics through the whole structure F. Thus, techniques of aggregation depend

on the kind of dependences among characteristics that are aggregated. There exist four kinds of such dependencies: characteristics are orthogonal (independent) and all are required for any project; characteristics are orthogonal but not all are required for any project; characteristics supplement the one that is required for any project; and none of the characteristics is required for any project. In the first case the characteristics can be aggregated properly using a kind of t-norm, in the second case using a kind of weighted t-norm, in the third case using a kind of t-conorm, and in the fourth case using a kind of weighted t-conorm. In order to minimise possible deviations generated by shortcomings of the particular metric (suite of quality evaluation tests) or by inaccuracy of the particular measurement, the mean with weights that are determined dynamically should be calculated. For this aim the dissertation proposes the heuristic, which can be seen as a kind of combination of arithmetic and winsorised means.

6. The dissertation proposes a systematic evaluation of quality in use on the basis of measurements of internal quality. Such approach can be used to evaluate the quality of the specification language in the context of the particular project, i.e. considering the high-level quality requirements formulated by the users of the particular system in the form of quality goals.
7. The main characteristic of internal quality – functionality– is analysed in the dissertation, and it is concluded that evaluation of the elementary characteristics of functionality is a hard and complicated task, which requires relatively high time and labour overheads. Many and long-time efforts are needed to do sampling, develop suites of tests, test language and interpret obtained results. Besides, some difficulties of theoretical character should be overcome. The theory of problem frames is relatively new and still not sufficiently elaborated. There are no any well-grounded methods for framing and sampling particular category of systems. Too little is known how to eliminate the impact of human factor to results of evaluation procedure. Thus, the systematic evaluation clarifies deep internal structure of each evaluated language as well as of specification languages in general and provides valuable experience that could be used during construction of new specification languages. We hope the proposed approach to evaluate internal quality will contribute both to the research on evaluation of the quality of existing specification languages and to the development of new ones.
8. Carried out experiment demonstrates that in the dissertation proposed framework to evaluate internal quality can be used to evaluate functional characteristics of specification languages (for example, UML and Z). From the evaluation of UML and Z

functionality it can be concluded that both these languages can be successfully used to specify functional requirements of generic Web portal.

9. It can be concluded from the results of experimental evaluation that even evaluation of the particular aspects of the particular specification language allows identification of shortcomings and strengths of the current specification languages, and preparing of recommendations how to use the particular language in more appropriate way as well as how to improve it. Thus, we conclude that a long-time research program for evaluation of specification languages quality is purposeful, because such program will significantly contribute to the entire theory of specification, and even programming languages.

Bibliography

[BXR96] **A. D. Belchior, G. Xexéo, A. R. C. da Rocha** (1996). Evaluating software quality requirements using fuzzy theory. In *Proceedings of International Conference on IS Analysis and Synthesis „(ISAS) 96"*. July 22-26, Orlando, USA.

[BHS96] **M.J. Bolanos, L.D. Hernandez, A. Salmeron** (1996). Numerical Experimentation and Comparison of Fuzzy Integrals. *Mathware & Soft Computing*, **3(5)**, 309-319.

[BBC98] **P. Botella, X. Burgués, J.P. Carvallo, X. Franch, J.A. Pastor, C. Quer** (1998). Towards a Quality Model for the Selection of ERP Systems. In A. Cechich, M. Piattini, A. Vallecillo (Eds.), *Component-Based Software Quality: Methods and Techniques*. *Lecture Notes in Computer Science*, **2693**, Springer-Verlag, 225-245.

[Bra02] **I.K. Bray** (2002). *An Introduction to Requirements Engineering*. Addison-Wesley.

[BK03] **I.K. Bray, K. Cox** (2003). The Simulator; Another, Elementary Problem Frame ? In *Pre-Proceedings of the 9th International Workshop on Requirements Engineering – Foundation For Software Quality, In conjunction with CAiSE'03*. Klagenfurt/Velden, Austria. pp. 101-104. Available at: http://crinfo.univ-paris1.fr/REFSQ/03/papers/REFSQ03-PreProceedings.pdf, [Accessed September, 2006].

[Bun77] **M. Bunge** (1977). *Treatise on Basic Philosophy*, Vol. 3: Ontology I: The Furniture of the World, Reidel, Boston.

[CG05a] **A. Caplinskas, J. Gasperovic** (2005). Techniques to aggregate the characteristics of internal quality of an IS specification language. *Informatica*, **16(4)**, 519-540. Available at: http://www.vtex.lt/informatica/Contents.htm, [Accessed September, 2006].

[CG05b] **A. Caplinskas, J., Gasperovic** (2005). Functionality of information systems specification language: concept, evaluation methodology, and evaluation problems. In O. Vasilecas, A. Caplinskas, W. Wojtkowski, W.G. Wojtkowsky, S. Wrycza, and J. Zupancic (Eds.), *Information Systems Development. Advances in Theory, Practice and Education*. Kluwer Plenum Publishers. pp. 341-351. Available at: http://www.springer.com/sgw/cda/frontpage/0,11855,5-170-22-45084311-detailsPage%253Dppmmedia%257CaboutThisBook%257CaboutThisBook,00.html, [Accessed September, 2006].

[CG05c] **A. Caplinskas, J. Gasperovic** (2005). An Approach to Evaluate Quality in Use of IS Specification Language. In J. Barzdins, and A. Caplinskas (Eds.), *Frontiers in Artificial Intelligence and Applications*, Vol. 118. IOS Press, Amsterdam, pp. 152-166. Available at: http://www.iospress.nl/, [Accessed September, 2006].

[CG04] **A. Caplinskas, J., Gasperovic** (2004). A taxonomy of characteristics to evaluate specification languages. *Computer Science and Information Technologies (Acta Universitatis Latviensis)*, Vol. 672, pp. 321-336.

[CLV02] **A. Caplinskas, A. Lupeikiene, O. Vasilecas** (2002). A framework to analyse and evaluate Information Systems specification languages. *Lecture Notes in Computer Science*, **2435**, 248-262.

[CM75] **A.K. Chandra, Z. Manna** (1975). The power of programming features. *Journal of Programming Languages*,**1**, 219-232.

[Ch92] **R. M. Chisholm** (1992). The basic ontological categories. In K. Muligan (Ed.), *Language, Truth, and Ontology*, Kluwer Academic Publishers.

[Ch96] **R.M. Chisholm** (1996). *A Realistic Theory of Categories: An Essay on Ontology*. Cambridge University Press.

[CNM99] **L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos** (1999). *Non-functional requirements in software engineering*. Kluwer Academic Publishers.

[Cle94] **D. Clegg, B. Richard** (1994). *Case Method Fast-Track. A RAD Approach*. Addison Wesley.

[Co77] **G. Cochran**. (1977). *Sampling techniques*. 3rd ed. John Wiley and Sons.

[CW98] **R.M. Colomb, R. Weber** (1998). Completeness and Quality of Ontology for an IS. In N. Guarino (Ed.), *Proceedings of the conference Formal Ontology in IS „FOIS'98"*. Trento, Italy, 6-8 June 1998. IOS Press, Amsterdam. pp. 207-217.

[Det00] **M. Detyniecki** (2000) *Mathematical Aggregation Operators and their Application to Video Querying*. Phd. thesis, Universite Pierre & Marie Curie.

[DP85] **D. Dubois, H. Prade** (1985). A Review of Fuzzy Set Aggregation Connectives. *Information Sciences*, **36**, 85-121.

[Dur93] **R. Durrett** (1993). *The Essentials of Probability (Statistics)*. Duxbury Press.

[EPA92] **Environmental Protection Agency** (1992). *Statistical Training Course for Ground-Water Monitoring Data Analysis*, EPA/530-R-93-003, Office of Solid Waste, Washington, DC.

[EPA96] **Environmental Protection Agency** (1996). *Guidance for Data Quality Assessment*, EPA/600-R-96-084, Office of Research and Development, Washington, DC.

[Fel90] **M. Felleisen** (1990). On the expressive power of programming languages. In N. Jones (Ed.), *Proceedings of the 3rd European Symposium on Programming „ESOP '90"*. Copenhagen, Denmark, Vol. 432. Springer-Verlag, New York, pp. 134 – 151. Available at: http://citeseer.ist.psu.edu/cache/papers/cs/633/ftp:zSzzSzftp.cs.indiana.eduzSzpubzSzscheme-repositoryzSzdoczSzpubszSzexpress.pdf/felleisen90expressive.pdf/, [Accessed September, 2006].

[Ful96] **R. Fullér** (1996). OWA operators in decision making. In C. Carlsson (Ed.), *Exploring the Limits of Support Systems*, TUCS General Publications, No. 3, Turku Center for Computer Science, Åbo, pp. 85-104.

[Ful05] **R. Fullér** (2005). *Neural Fuzzy Systems*. Lectures. Abo Academy.

[Gas06a] **J. Gasperovič** (2006). *Evaluation of the functionality of UML and Z languages*. Technical Report IMI-SED-06-01, Software Engineering Department, Institute of Mathematics and Informatics, Vilnius, Lithuania, 242 p. Available at: http://www.mii.lt/files/IMI_sed_06_01_Gasperovic.pdf, [Accessed November, 2006]

[Gas06b] **J. Gasperovič** (2006). Development of representative examples to evaluate IS specification language functionality characteristics (in Lithuanian). The paper is submitted for publication to the "*Lithuanian Mathematical Journal"*.

[Gas04] **J. Gasperovič** (2004). Specification language quality evaluation procedure. *Lithuanian Mathematical Journal*, **44 (spec. no.)**, 271-275 (in Lithuanian).

[GC06] **J. Gasperovic, A. Caplinskas** (2006). Methodology to Evaluate the Functionality of Specification Languages. *Informatica*, **17(3)**, 309-330.

[GC05a] **J. Gasperovič, A. Čaplinskas** (2005). Specification languages quality evaluation problems. *Information sciences*, **34**, 268-271 (in Lithuanian).

[GC05b] **J. Gasperovič, A. Čaplinskas** (2005). Specific in aggregation of characteristics describing internal quality of IS specification language. *Lithuanian Mathematical Journal*, **45 (spec. no.)**, 133-138 (in Lithuanian).

[GC03a] **J. Gasperovič, A. Čaplinskas** (2003). Information systems specification languages quality evaluation. In *Proceedings of the Conference* "*Information Technologies 2003*", Kaunas, Technology, p. VI-1 - VI-10 (in Lithuanian).

[GC03b] **J. Gasperovič, A. Čaplinskas** (2003). Comparative analysis methods in informatics. *Lithuanian Mathematical Journal*, **43 (spec. no.)**, 223-227 (in Lithuanian).

[GC03c] **J. Gasperovič, A. Čaplinskas** (2003). Information systems specification languages comparative analysis methods. *Information sciences*, **26**, 104-107 (in Lithuanian).

[GOY98] **M. Grabisch, S.A. Orlovski, R.R. Yager** (1998). Fuzzy aggregation of numerical preferences. In R. Slowinski. (Ed.), *The Handbook of Fuzzy Sets Series,* Vol. 4: Fuzzy Sets in Decision Analysis, Operations Research and Statistics. Kluwer Academic Press, pp. 31-68.

[GR99] **P. Green, M. Rosemann** (1999). An Ontological Analysis of Integrated Process Modeling. In M. Jarke, A. Oberweis (Eds.), Advanced IS Engineering. *Lecture Notes in Computer Science*, **1626**, Springer-Verlag, Berlin Heidelberg New York, 225–240.

[GR00a] **P. Green, M. Rosemann** (2000). Integrated Process Modelling: An Ontological Evaluation, *IS,* **25 (2)**, 73-87.

[GR00b] **P. Green**. **M. Rosemann** (2000). Integrating multi-perspective views into ontological analysis. *Proceedings of the International Conference on IS „OCIS 2000"*, Brisbane, Australia, December 11-13.

[Gua98] **N. Guarino** (1998). Formal Ontology and IS. In N. Guarino (Ed.), *Proceedings of the conference Formal Ontology in IS "FOIS'98"*. Trento, Italy, 6-8 June 1998. IOS Press, Amsterdam. pp. 3-15.

[Gui05] **G. Guizzardi** (2005). *Ontological foundations for structural conceptual models*. Universal Press. 410 p.

[Gur99] **C.A. Gurr** (1999). Effective Diagrammatic Communication: Syntatic, Semantic and Pragmatic Issues. *Journal of Visual Languages and Computing*, **10**, 317-342.

[Gur98] **C.A. Gurr** (1998). On the isomorphism, or lack of it, of representations. In K. Mariot, B. Meyer (Eds.), *Theory of Visual Languages*, Chap. 10, Springer, Berlin.

[ILP03] **ILP Indiana Learning Portal Conceptual Functional Requirements**. (2003). Requirements specification document, Haverstick Consulting. Available at: http://www.ihets.org/progserv/education/ilportal/documents/2/DELIVERABLE_Conceptual_Functional_Requirements_v1.0.pdf, [Accessed September, 2006]

[ISO94] **ISO 8402**. (1994) *Quality management and quality assurance vocabulary*. Second edition, 1994-04-01.

[ISO99] **ISO/IEC 14598-1:1999** (1999). *Information technology - Software product evaluation - Part 1: General overview*. First edition, 1999-04-15.

[ISO91] **ISO/IEC 9126** (1991). *Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use*. First edition, 1991-12-15, reference number ISO/IEC 9126: 1991(E).

[Jac99] **D. Jackson** (1999). *Comparison of Object Modelling Notations: Alloy, UML and Z*. MIT Lab for Computer Science. Available at: http://geyer.lcs.mit.edu/~dnj/pubs/alloy~comparison.pdf, [Accessed August, 2006]

[Jac02] **D. Jackson** (2002). *Micromodels of Software: Lightweight Modelling and Analysis with Alloy*. MIT Lab for Computer Science. Available at: http://alloy.mit.edu/alloy2website/, [Accessed August, 2006]

[JW96] **D. Jackson, J. Wing** (1996). Lightweight formal methods. *IEEE Computer*, April 1996, 21-22. Available at: http://geyer.lcs.mit.edu/~dnj/pubs/ieee96-roundtable.html, [Accessed September, 2006]

[Jac95] **M. Jackson** (1995). *Software Requirements and Specifications*. Addison-Wesley.

[Jac01] **M. Jackson** (2001). *Problem Frames*. Addison-Wesley.

[Jac00] **M. Jackson** (2000). *Problem Analysis and Structure*. In T. Hoare, M. Broy and R. Steinbruggen (Eds.), *Engineering Theories of Software Construction*, *Proceedings of NATO Summer School*. IOS Press, Marktoberdorf, pp. 3-20.

[Joh98] **I. Johansson**. Pattern as an ontological category. In N. Guarino (Ed.), *Formal Ontology in IS,* IOS Press, pp.86-94

[JD01] **M. N. Johnstone, D.C. McDermid** (2001). Using ontological ideas to facilitate the comparison of requirements elicitation methods. *Proceedings of the Twelfth Australasian Conference on IS (CD)*. Coffs Harbour, NSW, Australia, December 5-7.

[JS80] **N.D. Jones, D.A. Schmidt** (1980). Compiler generation from denotational semantics, *Lecture Notes in Computer Science*, **94**, Springer Verlag.

[KCH90] **K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson** (1990). *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical Report CMU/SEI-90-TR-021, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

[KM98] **M. King, B. Maegaard** (1998). Issues in natural language systems evaluation. In *Proceedings of the First International Conference on Language Resources and Evaluation*, May 28-30, Granada, Spain, pp. 225-230. Availabe at: http://citeseer.nj.nec.com/514907.html, [Accessed September, 2006].

[Kle52] **S. C. Kleene** (1952). *Introduction to Metamathematics*, D. Van Nostrand Co., Inc., New York.

[KMP02] **E.P. Klement, R. Mesiar, E. Pap** (2002). *Triangular norms. Position paper I: Basic analytical and algebraic properties*. Technical Report FLLL–TR–0208, Johannes Kepler Universität Linz, Fuzzy Logic Laboratorium Linz-Hagenberg. Available at: http://www.flll.jku.at/research/technical_reports/flll-tr-0208.pdf, [Accessed September, 2006].

[Kol30] **A. Kolmogorov** (1930). Sur la notion de moyenne. Atti delle Reale Accademia Nazionale dei Lincei Mem. Cl. Sci. Mat. Mnatur. Sez. 12, pp. 323-343 (in French).

[Kr01a] **J. Krogstie** (2001). A semiotic approach to quality in requirements specifications. In K. Liu, R. J. Clarke, P. B. Andersen, R.K. Stamper (Eds.), *Proceedings of the IFIP TC8/WG8.1 Working Conference on Organizational Semiotics: Evolving a Science of IS*. July 23-25, Montreal, Quebec, Canada. Kluwer Academic Publishers. pp.231-249.

[Kr01b] **J. Krogstie** (2001). Using a semiotic framework to evaluate UML for the development of models of high quality. In K. Siau, T. A. Halpin (Eds.), *Unified Modelling Language: Systems Analysis, Design and Development Issues*. Idea Group Publishing, Hershey, PA, USA. pp. 89-106.

[Kr03] **J. Krogstie** (2003). Evaluating UML using a generic quality framework. In L. Favre (Ed.), *UML and the Unified Process*. Idea Group Publishing, Hershey, PA, USA. pp. 1-22.

[KS03] **J. Krogstie, A. Sølvberg** (2003). *Information systems engineering - Conceptual modeling in a quality perspective*. Trondheim, Norway, Kompendiumforlaget.

[LSS94] **O. I. Lindland, G. Sindre, A. Sølvberg** (1994). Understanding Quality in Conceptual modelling. *IEEE Software*, **11(2)**, 42-49.

[MT86] **G. Mayor, E. Trillas** (1986). On the representation of some Aggregation Functions. In *Proceedings of the XVIth IEEE-International Symposium on Multiple-Valued Logic*, pp. 110-114.

[MK97] **R. Mesiar, M. Komorníková** (1997). Aggregation operators. In *Proceeding of the XI Conference on Applied Mathematics „PRIM'96"*. Novi Sad, pp. 193-211.

[Mil02] **J.S. Mill** (2002). A System of Logic, University Press of the Pacific, Honolulu.

[MK99] **S. Milton, E Kazmierczak** (1999). Enriching the Ontological Foundations of Modelling in IS. In C. N. G. Dampney et al. (Eds.), *IS Foundations - Ontology, Semiotics and Practice*. Macquarie University, Sydney, Australia.

[MKK98] **S. Milton, E Kazmierczak**, **C. Keen** (1998). Comparing Data Modelling Frameworks Using Chisholm's Ontology. In: J.A. Bartoli (Ed.), *Proceedings of the 4th European Conference on IS "ECIS'98"*, June 1998, Aix-en-Provence, France, Euro-Arab Management School. pp. 260-272.

[Mor01] **C. Moraga** (2001). Neuro-fuzzy modeling between minimum and maximum. In *Proceeding of the Workshop on Computational Intelligence, Theory and Applications*. Press Technical University of Niš, Yugoslavia.

[Myl98] **J. Mylopoulos** (1998). Characterizing Information Modelling Techniques. In P. Bernus, K. Mertins, G. Schmidt (Eds.), *Handbook on Architectures of IS*. Springer, Berlin, pp. 17-57.

[Nil00] **J. F. Nilsson**. A conceptual space logic. In E. Kawaguchi et al. (Eds.), *Information Modelling and Knowledge Bases XI,* IOS Press/Ohmsha, Amsterdam, pp. 26-40. Available at: http://www.imm.dtu.dk/~jfn/publications/conceptspaces.ps, [Accessed September, 2006].

[NM06] **No Magic**, Inc. (2006). *MagicDraw™UML 11.5 User manual*. Available at: http://www.magicdraw.com/main.php?ts=navig&NMSESSID=500a881a8816599a88224a0248272c6a&cmd_show=1&menu=download_manual&NMSESSID=500a881a8816599a88224a0248272c6a, [Accessed September, 2006].

[OB00] **J.A. O'Brien** (2000). *Introduction to Information Systems: Essentials for the Internetworked Enterprise*, 9th ed. Irwin/McGraw-Hill, New York.

[OMG05] **OMG** (2005). Unified Modeling Language: Superstructure version 2.0. Document formal/05-07-04, Object Management Group. Avalaible at: www.omg.org/docs/formal/05-07-04.pdf, [Accessed September, 2006].

[Opd97] **A. L. Opdahl** (1997). Applying Semantic Quality Criteria to Multi-Perspective Problem Analysis Methods. In E. Dubois, A.L. Opdahl, K. Pohl (Eds.), *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality „REFSQ'97"*. June, Barcelona, Catalonia, Spain. pp.49-66.

[Ovc98] **S. Ovchinnikov** (1998). On robust aggregation procedures. In B. Bouchon-Meunier (Ed.), *Aggregation and Fusion of Imperfect Informatikon*, Physic-Verlag. pp. 3-10.

[PM96] **J. F. Pane, B. A. Myers** (1996). *Usability Issues in the Design of Novice Programming Systems*. Technical Report CMU-CS-96-132, Carnegie Mellon University, School of Computer Science, Pittsburgh. Available at: http://www-2.cs.cmu.edu/~pane/cmu-cs-96-132.html, [Accessed January, 2004].

[PW97] **J. Parsons, Y. Wand** (1997). Using Objects in Systems Analysis, *Communications of the ACM,* **40(12)**, 104-110.

[Pat90] **M. Q. Patton** (1990). *Qualitative evaluation and research methods*, 2nd ed. Sage Publications, Newbury Park, CA.

[PW89] **D. Paulson and Y. Wand** (1989). *An automated approach to IS decomposition*. Technical Report. Working paper 89-MIS-001, Management Systems Division, Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, British Columbia, Canada.

[PT77] **L.J. Peters, L.L. Trip**. Comparing Software Design Methodologies, *Datamation*, **23(11)**, 89-94.

[Pfe79] **P.E. Pfeiffer** (1979). *Concepts of Probability Theory*, 2nd edition, Dover Publications.

[QJ03] **QStudio® for Java** (2003). *The Software Health Tool for Java*. QA Systems BV. The Netherlands. Available at: http://www.qa-systems.com/welcome.html, [Accessed January, 2004].

[Raj88] **V. Rajlich, J. Silva** (1988). Two object oriented decomposition methods. In *Proceedings of the 5th Washington Ada symposium on Ada*. Tyson's Corner, Virginia, USA. pp. 171 – 176.

[Req03] **Requirement Specification: Developer Portal** (2003). *The draft of the requirements specification document*, Beunited organisation. Available at: http://www.beunited.org/content/files/pubs/Dev_Portal_Spec.pdf, [Accessed September, 2006].

[Rey81] **J.C. Reynolds** (1981). The essence of Algol. In Jaco W. de Bakker and J. C. van Vliet (Eds.), *Algorithmic Languages*, North-Holland, Amsterdam, pp. 345-372.

[RG99] **M. Rosemann, P. Green** (1999). Enhancing the process of ontological analysis – the "Who cares" dimension. In C. N. G. Dampney et al. (Eds.), *Proceedings of the IS Foundations Workshop: Ontology, Semiotics and Practices*. Macquarie University, Sydney, Australia.

[RG02] **M. Rosemann, P. Green** (2002). Developing a Meta-Model for the Bunge-Wand-Weber Ontological Constructs. IS, **27**, 75-91.

[Rud04] **A. Rudys** (2004). Elementary-equivalence, Lecture 21. In M.Y. Vardi (Ed.), *Logic in Computer Sciences* (course material). Department of Computer Science, Rice University. Available at: http://www.cs.rice.edu/~vardi/comp409/2001/lec21.ps, [Accessed September, 2006].

[Saa99] **M. Saaltink** (1999). *The Z/EVES 2.0 User's Guide*. TR-99-5493-06a. ORA Canada. Available at: http://nexp.cs.pdx.edu/bart/omse/omse522-winter2002/nfp/sw/z-eves/99-5493-06a-users.pdf, [Accessed September, 2006].

[Sch60] **B. Schweizer, A. Sklar** (1960). Statistical metric spaces. *Pacific J. Math*, **10**, 313-334.

[Sch83] **B. Schweizer, A. Sklar** (1983). Probabilistic metric spaces, North-Holland.

[Sel94] **A. H. Seltveit** (1994). *Complexity Reduction in IS Modelling*. Phd. Thesis, Trondheim, Norway, IDT, NTH.

[Sil79] **W. Silvert** (1979). Symmetric summation: a class of operations on fuzzy sets. IEE Transactions on Systems, *Man and Cybernetics*, **9**, 659-667.

[SB98] **R. P. S. Simão, A.D. Belchior** (1998). Quality Characteristics for Software Components: Hierarchy and Quality Guides. In A. Cechich, M. Piattini, A. Vallecillo (Eds.), *Component-Based Software Quality: Methods and Techniques*. *Lecture Notes in Computer Science*, **2693**, Springer-Verlag, p 184-206.

[Sin90] **G. Sindre** (1990). *HICONS: A General Diagrammatic Framework for Hierarchical Modelling*. Phd. Thesis, Trondheim, Norway, IDT, NTH.

[Son92] **X. Song** (1992). *Comparing Software Design Methodologies through Process Modelling*. Technical Report ICS-92-48, Department of Information and Computer Science, University of California, Irvine.

[SO91] **X. Song X., L.J. Osterweil** (1991). Comparing design methodologies through process modelling. In M. Dowson (Ed.), *Proceedings of the 1st International Conference on Software Processing*. Los Alamtos, CA: IEEE Computer Society. pp.29-44.

[SO92] **X. Song X., L.J. Osterweil** (1992). A Framework for Classifying Parts of Software Design Methodologies. In R.W. Selby (Ed.), *Proceedings 2nd Irvine Software Symposium IRUS*. March 1992. pp. 49-68.

[Spi92] **J.M. Spivey** (1992). *The Z Specification Language*, 2nd. ed. Prentice-Hall.

[SS76] **G.L.Jr. Steele, G. J. Sussman**. Lambda (1976). *The ultimate imperatyve*. Memo 353, MIT AI Lab. Available at: ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-353.pdf, [Accessed September, 2006].

[Sug74] **M. Sugeno** (1974). *Theory of fuzzy integrals and its application*. Phd. Thesis, Tokyo, Institute of Technology.

[Tro02] **W. M. K. Trochim** (2002). *Research Methods Knowledge Base*. Cornell University. Available at: http://www.socialresearchmethods.net/kb/, [Accessed September, 2006].

[Tr73] **A.S. Troelstra** (1973). Metamathematical Thesis of the Intuitionistic Arithmetic and Analysis. *Lecture Notes in Mathematics*, **344**, Springer-Verlag.

[Wan88] **Y. Wand** (1988). An ontological foundation for IS design theory. In B. Pernici and A. A. Verrijn-Stuart (Ed.), *Proceedings of IFIP WG8.4 Working Conference on "Office IS: The Design Process"*. Elsevier Science, North-Holland.

[WSW99] **Y.Wand, V.C. Storey, R. Weber** (1999). An Ontological Analysis of the Relationship Construct in Conceptual Modelling. *ACM Transactions on Database Systems*, Vol. 24, No. 4, December 1999, 494-528

[WW96] **Y. Wand and R. Y. Wang** (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, **39(11)**, 86–95.

[WW89a] **Y. Wand, R. Weber** (1989). An ontological evaluation of systems analysis and design methods. In E.D. Falkenberg, P. Lindgreen (Eds.), *Information Systems Concepts: An In-depth Analysis*. IOS Press, North-Holland, Amsterdam. pp. 79-107.

[WW89b] **Y. Wand, R. Weber** (1989). A Model of Control and Audit Procedure Change in Evolving Data Processing Systems. *The Accounting Review,* **LXIV (1)**, 87–107

[WW90a] **Y. Wand, R. Weber** (1990). An Ontological Model of an IS. *IEEE Trans. Software Eng.,* **16(11)**, 1281-1291.

[WW90b] **Y. Wand, R. Weber** (1990). Mario Bunge's Ontology as a Formal Foundation for IS Concepts. In P. Weingartner, G.J.W. Dorn (Eds.)*, Studies on Mario Bunge's Treatise.* Rodopi, Atlanta, pp. 123-149.

[WW91] **Y. Wand, R. Weber** (1991). A Unified Model of Software and Data Decomposition. J. DeGross, I. Benbast, G. DeSanctis, C.M. Beath (Eds.), *Proceedings of the Twelfth International Conference on IS*. pp. 101-110.

[WW93] **Y. Wand, R. Weber** (1993). On the Ontological Expressiveness of IS Analysis and Design Grammars. *Journal of IS*, **3(4)**, 217-237.

[WW95] **Y. Wand, R. Weber** (1995). On the Deep Structure of Information Systems. *Information Systems Journal*, **5**, 203-223.

[Web97] **R. Weber** (1997). *Ontological Foundations of IS*, Coopers & Lybrand Accounting Research Methodology, Monograph No. 4, Melbourne.

[WZ96] **R. Weber, Y. Zhang**. An Analytical Evaluation of NIAM's Grammar for Conceptual Schema Diagrams. *IS Journal,* 6(2), 1996, p. 147-170.

[Woo96] **J. Woodcock, J. Davies** (1996). *Using Z: specification, refinement, and proof*. Prentice-Hall.

[Wo83] **W.A. Woods** (1983). What's important about knowledge representation. *Computer*, **16(10)**, 22-27.

[Yag88] **R. R. Yager** (1988). On ordered weighted averaging aggregation operators in multi-criteria decision-making. IEE Transactions on Systems, *Man and Cybernetics*, **18**, 183-190.

[Zim80] **H.J. Zimmermann, P. Zysno** (1980). Latent connectives in human decision making. *Fuzzy Sets and Systems*, **4**, 37-51.

The List of Publications

Papers in the scientific journals, included into the ISI Master Journal list:

1. **J. Gasperovic, A. Caplinskas** (2006). Methodology to Evaluate the Functionality of Specification Languages. *Informatica*, Vol. 17, Iss. 3. p. 309-330.
2. **A. Caplinskas, J. Gasperovic** (2005). Techniques to aggregate the characteristics of internal quality of an IS specification language. *Informatica*, Vol. 16, Iss. 4. p. 519-540. Available at: <http://www.vtex.lt/informatica/Contents.htm>.

Papers in other referred scientific journals:

3. **J. Gasperovič** (2006). Development of representative examples to evaluate IS specification language functionality characteristics. *Lithuanian Mathematical Journal*, 46 (spec. no.), p. 97-102 (in Lithuanian).
4. **J. Gasperovič, A. Čaplinskas** (2005). Specification languages quality evaluation problems. *Information sciences*, vol. 34, p. 268-271 (in Lithuanian).
5. **J. Gasperovič, A. Čaplinskas** (2005). Specific in aggregation of characteristics describing internal quality of IS specification language. *Lithuanian Mathematical Journal*, 45 (spec. no.), p. 133-138 (in Lithuanian).
6. **J. Gasperovič** (2004). Specification language quality evaluation procedure. *Lithuanian Mathematical Journal*, 44 (spec. no.), p.271-275 (in Lithuanian).
7. **J. Gasperovič, A. Čaplinskas** (2003). Comparative analysis methods in informatics. *Lithuanian Mathematical Journal*, 43 (spec. no.), p. 223-227 (in Lithuanian).
8. **J. Gasperovič, A. Čaplinskas** (2003). Information systems specification languages comparative analysis methods. *Information sciences*. vol. 26, p. 104-107 (in Lithuanian).

Papers in other reviewed periodical scientific publications:

9. **A. Caplinskas, J. Gasperovic** (2005). An approach to evaluate quality in use of IS specification language. *Frontiers in Artificial Intelligence and Applications*, Vol. 118, p. 152-166. Available at: <http://www.iospress.nl/>.
10. **A. Caplinskas, J., Gasperovic** (2004). A taxonomy of characteristics to evaluate specification languages. *Computer Science and Information Technologies (Acta Universitatis Latviensis)*, Vol. 672, p. 321-336.

Paper in international conference proceedings, included into the ISI Proceedings list:

11. **A. Caplinskas, J., Gasperovic** (2005). Functionality of IS specification language: concept, evaluation methodology, and evaluation problems. *IS Development*. *Advances in Theory, Practice and Education*, New York: Springer, p. 341-351. Available at: <http://www.springer.com/sgw/cda/frontpage/0,11855,5-170-22-45084311-detailsPage%253Dppmmedia%257CaboutThisBook%257CaboutThisBook,00.html>.

Papers in the proceedings of the local scientific conference:

12. **J. Gasperovič, A. Čaplinskas** (2003). Information systems specification languages quality evaluation. In *Proceedings of the Conference* "*Information Technologies 2003*", Kaunas: Technology, p. VI-1 - VI-10 (in Lithuanian).

APPENDIXES

APPENDIX 1. Taxonomy of Quality Characteristics

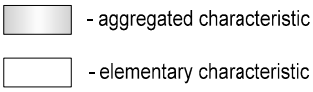
1	**Functionality**	Characteristics of a linguistic system that bears on the existence of the set of features required specifying properties of a system under consideration.
1.1	Suitability	Characteristics of a linguistic system that bears on ability to express statements about potential systems in a particular realm at a certain level of granularity.
1.1.1	Completeness	Characteristic of a linguistic system that bears on its ability to describe a system under consideration sufficiently and exhaustively.
1.1.1.1	Semantic sufficiency	Characteristics of a linguistic system that bears on its ability to specify all "things" that may be necessary for analysis or design of a system under consideration.
1.1.1.1.1	Ontological sufficiency	Characteristics of a linguistic system that bears on its ability to conceptualise (using ontological primitives provided by linguistic system) any system under consideration.
1.1.1.1.2	Epistemological sufficiency	Characteristics of a linguistic system that bears on its ability to express epistemological primitives, (i.e. to define conceptual primitives on the basis of a given set of ontological ones and to combine defined concepts further in order to form more complex concepts).
1.1.1.2	Expressibility	Characteristic of a linguistic system that bears on its ability to express any statement about any system under consideration.
1.1.1.3	Reasoning power	Characteristic of a linguistic system that bears on its ability to derive new statements about properties of system under consideration.
1.1.1.4	Composability	Characteristic of a linguistic system that bears on its ability to compose language constructs and features together.
1.1.2	Expressive adequacy	Characteristics of a linguistic system that bears on its ability to formulate statements about any system under consideration in adequate terms and express them with required degree of precision.
1.1.2.1	Ontological adequacy	Characteristic of a linguistic system that bears on its ability to express its ontological commitments within this system itself (i.e. use adequate terms).
1.1.2.2	Epistemological adequacy	Characteristic of a linguistic system that bears on its ability to express epistemological primitives directly (i.e. use such epistemological schemes as generalisation, aggregation, etc.)
1.1.2.3	Selective power	Characteristic of a linguistic system that bears on its ability to distinguish details with the needed degree of precision. (i.e. describe two different instances of a concept and two properties of an instance of a concept in a distinguishable way).
1.1.2.4	Generalitive power	Characteristic of a linguistic system that bears on its ability to describe system at different levels of granularity (i.e. suppress irrelevant details while preserving essential properties of the system).
1.2	Flexibility	Characteristics of a linguistic system that bears on the extent to which a language can be adjusted to specify preliminary not intended properties.
1.2.1	Universality	Characteristic of a linguistic system that bears on the ability to apply its constructs in different realms (i.e. real-world systems, IS, software systems, etc.).
1.2.2	Adaptability	Characteristic of a (general-purpose) language (both linguistic system and representation system) that bears on its ability to configure syntax and semantics to in order to adapt this language for arbitrary realm.

1.2.3	Extensibility	Characteristic of a (general-purpose) language (both linguistic system and representation system) that bears on its ability to extend this language with new features.
2	**Reliability**	Characteristics of a linguistic system that bear on its ability to ensure correctness of specifications.
2.1	Preventability of errors	Characteristic of a language (both linguistic system and representation system) that bears on the number of errors in the specification.
2.1.1	Semantic unambiguity	Characteristic of a linguistic system that bears on unambiguity of the concepts of language.
2.1.2	Notational unambiguity	Characteristic of a representation system that bears on unambiguity of the structure of language constructs.
2.1.3	Distinguishibility	Characteristic of a representation system that bears on absence of subtle distinctions in syntax, which may be overlooked or confused.
2.1.4	Simplicity	Characteristic of a language (both linguistic system and representation system) that bears on the comprehensibility of language.
2.1.4.1	Conceptual simplicity	Characteristic of a linguistic system that bears on the comprehensibility (including conceptual cleanliness) of the concepts defined by language.
2.1.4.2	Functional simplicity	Characteristic of a linguistic system that bears on a number of features of language.
2.1.4.3	Representational simplicity	Characteristic of a representation system that bears on the comprehensibility (including syntactical transparency) of the constructions of language.
2.1.5	Semantic power	Characteristic of a linguistic system (level of language) that bears on the semantic power of the concepts defined by this system.
2.1.6	Orthogonality	Characteristic of a linguistic system that bears on the degree in which the concepts of the language interfere with each other.
2.1.7	Uniformity	Characteristic of a language (both linguistic system and representation system) that bears on the degree of internal standardisation of the syntax (syntactical uniformity) and the semantic (semantic uniformity) of the construction of the language.
2.2	Verifiability	Characteristic of a linguistic system that bears on the ability (including executability) to check correctness of produced specifications.
3	**Efficiency**	Characteristics of a linguistic system and a representation system that bear on the amount of efforts needed to produce specification.
3.1	Expressive efficiency	Characteristic of a linguistic system that bears on the efforts necessary to express the statements about the phenomena.
3.2	Representational efficiency	Characteristic of a representation system that bears on efforts necessary to represent the statements about the phenomena.
3.3	Semantic power	Characteristic of a linguistic system (level of language) that bears on the semantic power of the concepts defined by this system.
3.4	Orthogonality	Characteristic of a linguistic system that bears on the degree in which the concepts of the language interfere with each other.
3.5	Permissiveness	Characteristic of a language (both linguistic system and representation system) that bears on its ability to express and represent things in several different ways.
3.6	Viscosity	Characteristic of a language (both linguistic system and representation system) that bears on the possibility to narrow the change impact on the produced specification.
3.7	Technological efficiency	Characteristics of a language (both linguistic system and representation system) that bear on the technological appropriateness of a language.
3.7.1	Manageability	Characteristics of a language (both linguistic system and

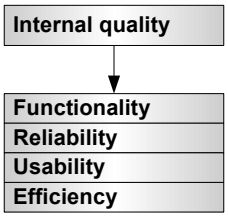
		representation system) that bear on appropriateness of the language to deal with large and complex specifications.
3.7.1.1	Decomposability	Characteristic of a language (both linguistic system and representation system) that bears on the possibility to divide the produced specification into relatively autonomous and independent subparts.
3.7.1.2	Genericity	Characteristic of a linguistic system that bears on the possibility to use generic types.
3.7.2	Processability	Characteristic of a language (both linguistic system and representation system) that bear on appropriateness of the language to manipulate specifications by software.
4	**Usability**	Characteristics of a linguistic system and a representation system that bear on the audiences' effort to understand and to learn a language.
4.1	Understandability	Characteristics of a linguistic system that bear on the audiences' effort to understand its conceptualisation.
4.1.1	Ontology	Characteristic of a linguistic system that describes language ontology.
4.1.2	Paradigm	Characteristic of a linguistic system that describes mathematical theory beyond the language.
4.1.3	Naturalness	Characteristic of a linguistic system that bears on the correspondence between the language ontology and the common sense.
4.1.4	Semantic unambiguity	See 2.1.1
4.1.5	Notational unambiguity	See 2.1.2
4.1.6	Distinguishibility	See 2.1.3
4.1.7	Uniformity	See 2.1.7
4.2	Learnability	Characteristics of a linguistic system and a representation system that bear on the audiences' effort for learning of language application.
4.2.1	Simplicity	See 2.1.4
4.2.2	Semantic power	See 3.3
4.2.3	Naturalness	See 4.1.3
4.2.4	Uniformity	See 2.1.7
4.2.5	Commonality	Characteristic of a language (both linguistic system and representation system) that make the language adhere to wide-accepted standards or conventions.

APPENDIX 2. Graphical Representation of the Taxonomy of Quality Characteristics

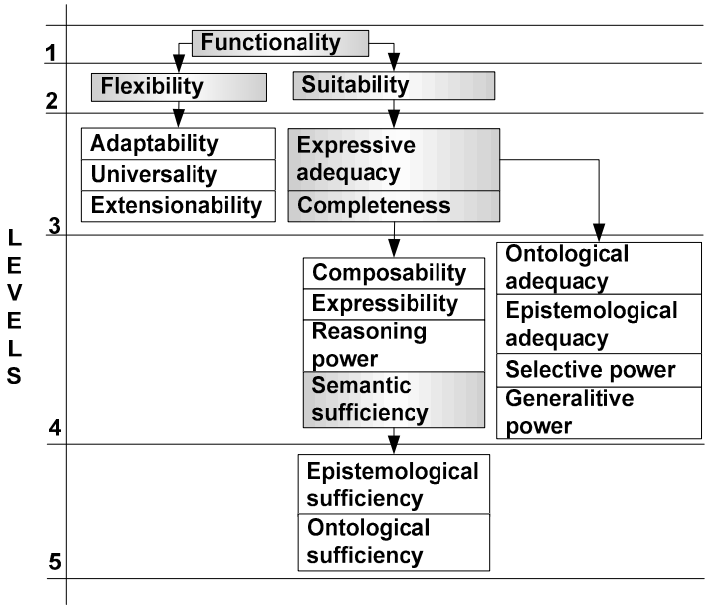
Denotation:



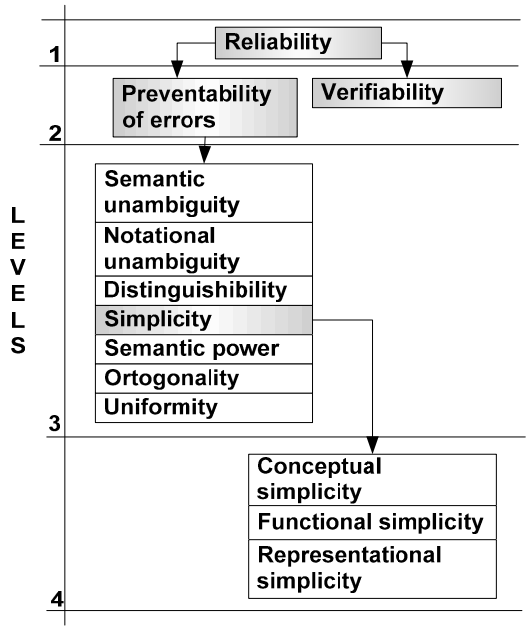
Internal quality and its sub-characteristics



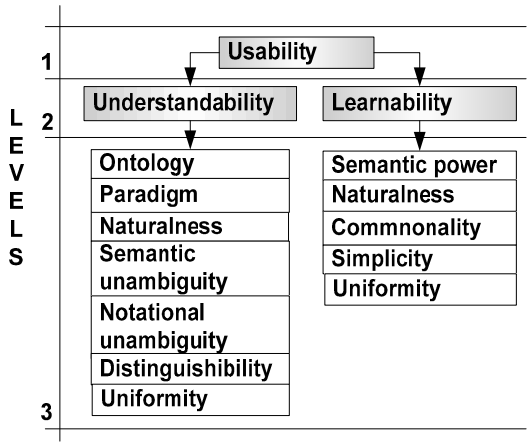
Functionality and its sub-characteristics



Reliability and its sub-characteristics



Usability and its sub-characteristics



Efficiency and its sub-characteristics