An Approach to the Context-oriented Use Case Analysis

Kalinka Kaloyanova[1] and Neli Maneva[2]
[1]*Faculty of Mathematics and Informatics, Sofia University, 5 J. Bourchier Blvd., Sofia, Bulgaria*
[2]*Institute of Mathematics and Informatics, BAS, Acad. G. Bonchev Str., Bl.8, Sofia, Bulgaria*
kkaloyanova@fmi.uni-sofia.bg, neman@math.bas.bg

Keywords: Requirements Engineering, Use Case Analysis, Information Systems Development, Multiple-criteria Decision Making, Comparative Analysis.

Abstract: The paper describes our efforts to propose a feasible solution of a significant problem in information systems development – requirements engineering, based on use cases. The adjustable use case quality model is constructed and used within the Comparative Analysis method to support the decision making during the use case analysis process. Two real-life problems related to this process are described and their solutions through the suggested approach are given.

1 INTRODUCTION

Nowadays the volumes of information are continually increasing, the methods for storing and searching information become more complex and the software users - more demanding. One way to meet their expectations is to assure an effective engineering of the elucidated user-defined requirements. Requirements Engineering (RE) as a subfield of both Information Systems (IS) and Software Engineering (SE) concerns the real-life needs of the users and the constraints for the systems to be built.

Recognizing the importance of the RE in any software intensive system's life cycle and understanding the significant role of the requirements elicitation activity, we address a challenging issue – providing systematic approaches to the construction and constantly improvement of the requirements for information systems development.

The goal of this paper is to present our approach to IS requirements engineering, based on quality use cases. The applicability of a formal method for the proposed context-oriented use case analysis is investigated in Section 2. The most difficult activity – use case quality modelling in a specified context, is described in Section 3. Two examples are given, illustrating how the method can be successfully applied. In the Conclusions section some directions for future research and experimental work are shared.

2 BASIC DEFINITIONS AND RELATED APPROACHES

Requirements engineering is one of the initial and very important activities in software development. Lately its significance is realized and there are plenty of methods, frameworks, tools, etc., supporting RE (Pressman, 2009), (Sommerville, 2011), (Denny, 2005). From practitioner's point of view the great variety of approaches and available tools are more distractive than useful. This is especially true for information systems development, where the main requirements come from end users and usually are ambiguous, contradictory and incomplete (Pokorny, 2010). Due to such reasons the iterative approaches usually are more effective, because they support the requirements refinement in a more constructive way (Kaloyanova, 2012).

2.1 Use Case Modelling

A common form for describing functional requirements for a system is the ***use case modelling*** (Maciaszek, 2005)**.** It captures user requirements of a new system by detailing all scenarios that the users will perform. Each use case describes a specific way of using the designed system by actors – users with specific roles, trying to achieve their goals (Sinnig, 2010). The set of all use cases defined by actors describes the system functionality (Cockburn, 2000).

Kaloyanova K. and Maneva N.
An Approach to the Context-oriented Use Case Analysis.
DOI: 10.5220/0005425101350141
In *Proceedings of the Fourth International Symposium on Business Modeling and Software Design (BMSD 2014)*, pages 135-141
ISBN: 978-989-758-032-1

Use case modelling usually is recognized as a part of Requirement engineering discipline of the Unified Process (UP) that is focused primarily on the functional requirements of the system (Kruchten, 2004). The non-functional requirements are divided by the UP approach into several categories. The acronym FURPS+ presents the functional and several important non-functional requirements, which are quite relevant to IS development – usability, reliability, performance, and supportability of the system (Larman, 2004).

The use case model describes not only the system behaviour, but it also defines the boundary of the system and how this system communicates with different actors.

Besides, as use case modelling is a part of the UP, starting from the first phase of it - Inception, the use cases capture functionality of the system in an iterative process that is not only understandable for the users, but could also be detailed in the next phases of the UP – Elaboration and Construction.

Use cases are described in natural language. So they can be easily understood by non-technical people and can be further discussed by users, clients, analysts, designers, developers and their managers.

The use cases can be written with different level of details (Larman, 2004):

- Brief – one-paragraph summary, focused on the main success scenario;
- Casual – multiple paragraphs, covered various scenarios of user-system interaction;
- Fully dressed – all scenarios are written in detail, comprising some supporting sections, such as preconditions, post-conditions, special requirements, etc.

The different forms of use cases description can be created during successive iterations, following the UP. In this way, at every step some new information is added to the use case description. The analysis usually starts with a set of use cases in brief format and detail them through the next iterations. In order to do this systematically, the use cases have to be ranked, taking into account their significance, risk, complexity or other criterion of prioritization.

Several questions arise here - how to implement the use case definition method in practice and how to evaluate the quality of use case description in different iterations. The analysts need a technique that supports a stable evolution of the specifications of every element of software functionality, described by the corresponding use case.

A number of works discuss different sides of the use case development – formalization (Sinning, 2009), content (Cockburn, 2000), inspection techniques (Anda, 2002), etc. Most of the research publications about the use case modelling are summarized in (Torner, 2006). Although the intensive research in the area it is hard to find a method that represents a holistic view on all aspects of the use case transformation.

In this paper we introduce a use case quality model for the description of the functionality, presented at different consecutive iterations of the same use case. Further this quality model can be built into a method allowing not only to assess the achieved use case quality, but to facilitate some decisions, made during the process of quality improvement. We base our approach on the formal Comparative Analysis method (Maneva, 2007). This method was chosen primarily because it shares the main objectives and methods of the broad theory of the Multiple Criteria Decision Making, trying to specify and apply them systematically.

2.2 Comparative Analysis

We will present the essence of the **Comparative Analysis (CA)** in order to explain how it can be used for the activity under consideration – use case analysis.

Comparative Analysis is a study of the quality content of a set of homogeneous objects and their mutual comparison in order to select the best, to rank them (establishing a preference order) or to classify each object to one of the predefined quality categories.

For CA use we distinguish two main roles: the ***CA Analyst***, responsible for all aspects of CA implementation, and a ***CA customer*** – a single person or a group of individuals, tasked with making a decision in a given situation. Depending on the identified problem to be solved at a given moment, a ***case*** should be opened to determine the context of the desired Comparative Analysis.

Each case is specified by the following elements:

case={View, Goal, Object, Competitors, Task, Level }

The **View** describes the CA customer's role and the perspective from which the CA will be performed.

The **Goal** expresses the main customer's intentions in CA accomplishment such as to describe, analyse, estimate, improve, predict or any other, formulated by the Customer, defining the case.

The **Object** represents the item under consideration. In the field of software engineering any studied object belongs to one of the following groups: products, processes or resources. For each

object for CA application, a quality model should be created – a set of characteristics, selected to represent the quality content in this context, and the relationships among them.

According to the goal, the set C of **Competitors, C = {C_1, C_2, ... , C_n}** – the instances of the objects to be compared – should be chosen.

The element **Task** of a case can be ***Selection*** (finding the best), ***Ranking*** (producing an ordered list), ***Classification*** (splitting the competitors to a few preliminary defined quality groups) or any combination of them.

The parameter **Level** defines the overall complexity of the CA and depends on the importance of the problem under consideration and on the resources needed for CA implementation.

Usually the object quality model is presented as a hierarchy. At the top of the hierarchy is the total object quality. The first level comprises some user-oriented attributes, called ***factors***. The next level describes a number of object-dependent attributes, providing quality. These ***criteria*** can be further decomposed to more simple and measurable characteristics. To each node at the hierarchy structure a weight (a coefficient of importance) is assigned, and for the leaves of the hierarchy some appropriate ***metrics*** are defined. Starting a bottom-up evaluation of characteristics at each level in the hierarchy and applying a modification of the MECCA (Multi-Element Component Comparison and Analysis) method, described in (Bowen, 1985), we can obtain the quantitative measures of all factors and fill the competitors-factors matrix **E(nxm)**, where **n** is the number of the competitors, defined in the set **C**, and **m** is the number of the quality factors. Each element **Ei,j** is the measure of the **i**-th competitor with respect to the **j**-th quality factor. The obtained matrix **E** is further used as input to the software tools, implementing the required selection, ranking or classification methods.

Additional details about the CA method can be found in a number of recent publications. In this paper we provide only the information, absolutely necessary for understanding the CA application as a method, supporting decision making.

3 A CONTEXT-ORIENTED USE CASE ANALYSIS

The CA method can be used in any decision making situation within the use case analysis, after specifying its context.

According to the definition already given in Section 2.2, the concrete situation for a particular decision making can be stated by a case, comprising six elements:

case={View, Goal, Object, Competitors, Task, Level}

The ***View*** describes the CA customer's role and focuses on the perspective of the Comparative Analysis. Taking into account the responsibilities and typical tasks of the main participants in the use case analysis, the following Customer's roles have been identified till now: ***Business Analyst, User, IS developer.*** Thus a lot of situations can be further specified, reflecting the specific participant's point of view to the analysed context.

As the ***Goal*** expresses the main Customer's intentions in CA accomplishment and bearing in mind the goal-oriented use case definition and its further improvement, the Analyst should decompose the goal stated for use case analysis to a number of CA-related goals, e.g. to describe, assess, estimate, improve, predict, etc.

The element **Object**, as it was mentioned above, represents the item under consideration. Particularly for the use case analysis, the following objects, classified in three groups, can be studied:

Products – use case, collection of use cases, use cases source (discussion memos, documentation, specifications, user's and developer's stories), prototypes for checking selected use case characteristics, like completeness, usefulness, etc.

Processes, related to use case analysis: elicitation, quality assessment, prioritization, documenting, maintenance, tracking, etc.

Resources, related to use case analysis: technological (e.g. method, technique, tool), or project-oriented (people, team, performance).

For each studied object a quality model should be created – a set of characteristics, selected to represent the quality content, and the relationships among them.

According to the goal, the set **C** of **Competitors** – the instances of the objects to be mutually studied and compared – should be chosen. If the Goal is only to create the quality model of the object under consideration, the set C is empty.

There are no special considerations, when we define the element ***Task*** of a situation for use case analysis. As usual, the task can be *Selection*, *Ranking*, *Classification* or any combination of them.

It is not difficult to define the degree of complexity (simple, medium or high) presented by the element **Level.**

One of the challenges in the implementation of the CA is the construction of a model, adequate to the quality content of the object, pointed in a given

case. According to the prescriptions, the Analyst is responsible for accomplishment of this task, which is with very high cognitive complexity and usually requires unified efforts of the Analyst and the Customer, ordered the CA.

3.1 A Quality Model for Object "Use Case"

As an example, we will describe the basic hierarchical model, constructed for the quality of object "use case". In this model we propose three user-oriented characteristics (factors) – Applicability, Validity and Utility.

The factor ***Applicability*** concerns the relation between the use case and the real system functionality and addresses the question of the use case legality. Next follows a brief definition of the quality characteristics selected for this factor and the corresponding metrics for the leaves of the tree structure.

The hierarchy obtained after the decomposition of the ***Applicability*** factor is shown in Table 1.

The first level of the hierarchy comprises three criteria affecting the factor **Applicability** - ***Accuracy, Trackability,*** and ***Adequacy.***

Table 1: Decomposition of the factor Applicability.

Factor Applicability	
Criteria	Item-Oriented Characteristics
Accuracy	
Trackability	
Adequacy	Robustness Completeness Correctness

Accuracy defines how the use case corresponds to the reality. It shows whether the goal of this use case could be reached by the presented use case description. Accuracy can be measured by the ratio of correct features to the total number of features, described in the use case.

Trackability reflects whether the changes in reality can be accompanied by changes in the use case.

The ***Adequacy*** criterion is more complex**.** It should be evaluated by answering several important questions like:

- Is the use case robust?
- Is the use case complete? Are all possible scenarios defined?
- Is the use case correct? Is the right sequence of steps described in the scenarios?

In the frame of ***Adequacy*** the characteristic **Correctness** can be measured by the ratio of correct steps to the total number of steps, described in the use case. More sophisticated evaluation of the correctness can be done by splitting this characteristic into two more – *Compliance* and *Homogeneity,* defined as follows:

– *Compliance*: Does the use case description follow the proper separation between the positive scenario and other alternative scenarios?

– *Homogeneity:* Are the descriptions of all scenarios with the same level of details?

The second suggested factor ***Validity*** concerns the inner capacity of the use case to produce results corresponding to the expected ones.

Table 2: Decomposition of the factor Validity.

Factor Validity	
Criteria	Item-Oriented Characteristics
Reliability	
Modifiability	Complexity Modularity
Understandability	Self-descriptive Concise Structured
Measurability	

Table 2 illustrates the proposed hierarchy for this factor based on the selected quality criteria: ***Reliability, Modifiability, Understandability and Measurability***.

The quality criterion ***Reliability*** is evaluated through the answers of two questions:

- Is the use case adequate?
- Is the use case robust?

The next criterion ***Modifiability*** shows how easily the use case can be modified. It is decomposed further in two measurable characteristics:

- *Complexity* – can be considered as textual or algorithmic, measured by the appropriately constructed metrics;

- *Modularity* – usually measured by the number of all alternative scenarios.

Nevertheless the quality criterion ***Understandability*** is well known concept, for the use case analysis it concerns variety of details, which can be taken into account:

- Is the use case self-descriptive? Are all steps clear to follow?
- Is the use case concise? Is the description of the steps brief and clear?
- Is the use case well structured? Do all scenarios fit each other? Is the main positive scenario properly defined and all alternative scenarios are adequate to it?

The used metrics here reflect the structure – *linear* (usually for the brief form of the use case) or *hierarchical* (for the description of the alternative scenarios and their connections to the steps of the main positive scenario).

When we consider the criterion ***Measurability***, it is obligatory to select some concrete metrics for use case assessment like:

- The number of steps of the successfull scenario;
- The number of steps, performed by each actor, involved into the success scenario;
- The number of system steps, described by the success scenario;
- The number of all alternative scenarios;
- The number of steps of alternative scenarios;
- The number of steps of the negative alternative scenarios;
- The number of all steps, described in the studied use case;
- The number of preconditions in the studied use case.

The third factor ***Utility*** deals with the relation between use case and users and addresses the question of the possibility to derive conclusions from the created use case. The proposed hierarchy for this factor is shown in Table 3, presenting a number of quality characteristics.

Table 3: Decomposition of the factor Utility.

Factor Utility	
Criteria	Item-Oriented Characteristics
Content	
Fidelity	
Constructiveness	
Stability	
Usability	Effectiveness Efficiency User satisfaction

The ***Content*** criterion focuses on the text description. Several questions arise here:

What does the use case represent?

Is use case detailed enough? Are all steps clear and described with appropriate level of detail?

The criterion ***Fidelity*** answers the question if different users will get similar results using the same use case.

The next criterion - ***Constructiveness*** explains how the use case facilitates some future IS-related activities like design, coding, testing, etc.

The criterion ***Stability*** explains if the use case can be manipulated to obtain false results or if the new version of the use case corrupts any of the previous versions.

The last considered criterion ***Usability*** can be evaluated from three different perspectives:

- *Effectiveness:* Does the use case match the purpose? Are all steps of the main positive scenarios leading to the goal?
- *Efficiency*: Does the use case fulfil the stated goal with minimal resources? Is the sequence of the steps is the shortest way to the goal?
- *User satisfaction:* Can be measured by the percentage of unsatisfied users and the average rating, given by users.

3.2 A Quality Model for Object "A Set of Use Cases"

This object can appear in a situation involving a set **U** of interconnected use cases, which can be studied as a whole due to some customer's considerations. For this object a simple linear model can be created, comprising four quality characteristics - ***Completeness***, ***Consistency,*** ***Relevance,*** and ***Correctness.***

For the purposes of the use case analysis we define their meaning as follows:

Completeness – the set **U** comprises all developed use cases, necessary to described the functionality of a system or a system's compound element – component, subsystem, etc.;

Consistency – all involved in **U** use cases are in reasonable and logical harmony, without any contradictions in their content;

Relevance – all involved in **U** use cases should possess a direct and clearly identified connection to the studied system's functionality. For this factor two measures have been applied – ***Recall*** and ***Precision***:

Recall measures how exhaustive a description of the scenarios is. It can be calculated by the following formula:

***Recall* = x / (x+y) * 100%,** where

- x is the number of described in **U** relevant scenarios,

- y is the number of the relevant, but not described yet in **U** scenarios.

Precision measures the amount of noise in scenarios description, based on the formula:

***Precision* = x / (x+z)** * 100%, where

- x is the number of described in **U** relevant scenarios;

- z is the number of described in **U**, but non-relevant scenarios.

The last quality characteristic – ***Correctness,*** is related to some scenarios errors in **U**, which have

been found and fixed during the use case analysis. It can be measured by the ratio of corrected scenarios to the number of all scenarios described in the set **U.**

4 EXAMPLES

The usefulness and feasibility of our idea to apply the CA method to use case analysis have been examined for solving two real-life problems, described below.

Case Study 1: How the CA can Support the Iterative Use Case Development. The iterative use case development is based on defining the initial version of the use case and then - a repetitive process of use case quality assessment and further refinement. According to the results of assessment, next iteration of the use case should be created through some changes that improve the recent quality content and functionality of the use case.

The CA with properly defined context can be used as a form of research for communicating and evolving a use case, as consecutive versions of the use case are implemented within the following step-wise procedure:

Step 1. ***Pre-analysis*** – description of a number of real-life problems encountered during the activity under consideration, for which the CA method seems to be useful. During the iterative use case development some additional information about the current state should be gathered to decide how to continue. From CA perspective this can be done as follows:

- Creating a derivative quality model of the use case, including only those quality characteristics from the generic model, which are relevant to the considered context;
- Evaluating the current version of a use case from the point of view of different actors;
- Comparing two consecutive iterations of the same use case to observe the effect of the performed changes on the quality;
- Evaluating the last iteration of a use case to decide how to proceed. In this situation the use case estimates are compared with those of a virtual "perfect" use case, whose current state is described by the same quality characteristics, but with preliminary assigned threshold values.

Step 2. ***Preparation*** – defining the CA context and planning the CA implementation. The elements (View, Goal, Object, Competitors, Task and Level) of each case must be specified. The relevant sources of information, needed for the CA performance, are identified and made available. A CA plan is created, describing some parameters of the work – duration, cost, personnel involved, tasks and responsibilities allocation and schedule.

For the above mentioned activities the possible elements of a case can be:

View – that of any already identified participant in use case analysis - user, IS developer or business analyst;

Goal – to assess, to compare or any other, defined by the CA Customer;

Object – a use case;

Competitors – an instance of a use case (and a "perfect" use case for the last action for comparison of the current use case with the "perfect" one);

Task – usually it is ranking;

Level – simple, middle or high.

Step 3. Construction – building a quality model, corresponding to use cases quality content for a defined case.

Step 4. ***Execution*** – evaluating the quality factors and accomplishment of the CA Task.

As an example, let us describe this step in CA for the case, designed for comparison of the last use case version and the "perfect" use case. The final results of the bottom up evaluation are presented in an objects-factors matrix. In it the first row comprises the threshold values, assigned to the perfect use case, and the second row comprises the results of use case evaluation (see Table 4).

Table 4: A filled objects-factors matrix.

Use case/ Factors	F1 Applicability weight - 0.3	F2 Validity weight -02	F3 Utility weight -05
Perfect use case	1	1	1
Investigated use case	0.8	0.7	1

Step 5. ***Completion*** - analysis, interpretation and drawing conclusions from the results.

In the discussed case the deviation of the estimated measures from the desired values for each factor are calculated and described, in order to make decision what to do further. The first possibility is to continue with new iterations, trying to improve the values for the second and the third factor. In this case we have to define the scope and contents of some additional use case refining activities. The second possibility is to accept the achieved quality level as appropriate and to stop analysing this use case.

Case Study 2: How the CA Can Support the Use Case Prioritization. One of the most difficult problems in use case analysis is to create an ordered list of all identified as important use cases. Such prioritization will make possible to perform the use case analysis in a more systematic and efficient way, facilitating the proper distribution of the planned and usually insufficient resources.

A Real-life Problem: After the use case quality content assessment is done, it is necessary to assign a priority to each use case in a given set of already developed use cases.

Some cases for this problem can be defined as a combination of the following elements:

View – can be that of the User, Business Analyst, Project manager or IS Developer;

Goal – to compare the use cases, selected for prioritization;

Object **–** use case;

Competitors – all developed use cases, chosen to be prioritized;

Task – ranking to produce an ordered list of the compared use cases;

Level – simple, middle or high.

5 CONCLUSIONS

The main purpose of this paper is to propose a systematic approach to the creation and continuous improvement of use cases, based on the Comparative Analysis method. It supports the decision making in some significant situations for the use case analysis.

The basic advantage of the CA is the possibility for adjustment to the context, specified by the elements of the investigated case. The most difficult activity – object quality modelling – has been described and illustrated with models for two basic objects – use case and a set of use cases. As example, three CA successful implementations in the field of use case analysis have been given.

Some possible directions for future work can be:

- To identify and describe entirely some other CA situations within the use case analysis, so as to enrich the collection of the re-used items – situations, models, metrics, etc.;
- To examine the possibility to apply the CA to other use case related activities as validating, tracking, change management, design and implementation of already defined use cases.

ACKNOWLEDGEMENTS

This work is partly supported by the National Scientific Research Fund under the Contract ДТК 02-69/2009 and partially supported by Sofia University "St. Kl. Ohridski" SRF/2014 under the Contract "Successful Practices for Information Systems Analysis and Design".

REFERENCES

Anda, B., D. Sjøberg, 2002. Towards an Inspection Technique for Use Case Models, In *Proceedings of the 14th Int. Conf. on Software Engineering and Knowledge Engineering*, pp. 127-134.

Bowen, T., G.B. Wigle, J. Tsai, 1985. Specification of software quality attributes, *Software quality evaluation guidebook,* RADC-TR-85-37, vol.III.

Cockburn, A., 2000. *Writing Effective Use Cases*, Addison-Wesley.

Denny, R., 2005. *Succeeding with Use Cases: Working Smart to Deliver Quality*, Addison Wesley.

Kaloyanova, K., 2012. Design from Data: How To Use Requirements for Better IS Analysis and Design, *Proceedings of the Int. Conference Informatics in Scientific Knowledge*, pp. 189-197.

Kruchten, P., 2004. *The Rational Unified Process: An Introduction*, Pearson Education.

Larman, G., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented analysis and Design and Iterative Development*, 3rd Edition, Prentice Hall.

Maciaszek, L., 2005. *Requirements Analysis and System Design*, Addison-Wesley Longman, Inc.

Maneva, N., 2007. Comparative Analysis: A Feasible Software Engineering Method, *Serdica J. of Computing, 1(1), pp. 1-12*.

Pokorny, J. at all, 2010. *Information Systems Development: Business Systems and Services: Modelling and Development*, Springer.

Pressman, R., 2009. *Software Engineering: A Practitioner's Approach*, 7th Edition, McGraw-Hill.

Sinnig, D., P. Chalin., F Khendek, 2009, LTS Semantics for Use Case Models, In *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 365-370.

Sinnig, D., H. Javahery, 2010. Mastering Use Cases: Capturing Functional Requirements for Interactive Applications, In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 373-374.

Sommerville, I., 2011. *Software Engineering*, 9th Edition, Addison Wesley.

Tornew, F., M. Ivarsson, F. Pettersson, P. Ohman, 2006. Defect in automotive use cases, In *Proc. of 2006 ACM/IEEE international symposium on Empirical software engineering*, pp.115-123.