A Framework for Characterizing Usability Requirements Elicitation and Analysis Methodologies (UREAM)

Jos J.M. Trienekens
IE&IS
TUE
Eindhoven, The Netherlands
j.j.m.trienekens@tue.nl

Rob J. Kusters
Management Sciences
Open University
Heerlen, The Netherlands
rob.kusters@ou.nl

***Abstract*—Dedicated methodologies for the elicitation and analysis of usability requirements have been proposed in literature, usually developed by usability experts. The usability of these approaches by non-expert software engineers is not obvious. In this paper, the objective is to support developers and managers in a software development project in deciding on which methodology to select, taking into account local strengths and weaknesses. We define a framework based on a set of criteria that allow for the comparison of methodologies.**

Keywords-usability; usability requirements.

I. INTRODUCTION

In the development of interactive systems, usability is increasingly considered to be a crucial factor for the success of a software system [13]. However, identifying and specifying usability requirements are not trivial tasks. It is even further complicated by the existence of multiple, different definitions of usability. Multiple approaches have been proposed on how to elicit and analyze usability requirements. Therefore, a need arises to compare the available methodologies in order to make a well-founded decision about which can be used in a project, based on the specific characteristics of the project. In this paper, we present a structured comparison of usability elicitation and analysis approaches that is designed to help the stakeholders of a project, e.g., project coordinators, managers, and developers, decide on a methodology to use for usability requirements elicitation and analysis. We define a framework for extracting specific properties of a methodology so as to allow for a direct comparison of different approaches presented in literature. The selected methodologies represent a selection of what we believe are the most important approaches to usability requirements elicitation and analysis.

In Section 2, we give definitions of terms required to compare usability requirements elicitation and analysis approaches. Section 3 describes the aforementioned framework, and, in sections 4 to 7, this framework is applied to each methodology. Section 8 gives a comparison of the results obtained for each of the methodologies and section 9 concludes with an overview of the most relevant findings from this comparison.

II. DEFINITIONS

The following section gives definitions for the most relevant terms used throughout this paper:

- Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [7].
- UREAM: usability requirements elicitation and analysis methodology.
- Methodology: a coherent and structured set of procedures to carry out usability requirements elicitation and analysis in a step-wise and well-defined way.
- Method: a coherent set of steps in a methodology is defined as a method.
- Technique: a systematic way to carry out a particular procedure (for example: a survey and a questionnaire are techniques for a method that focuses on an analysis of user tasks.
- HCI: Human-Computer Interaction is a research area that studies of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings.

III. TOWARDS A FRAMEWORK FOR UREAM COMPARISON

As it is stated in the introduction, there are many different methodologies to elicit and analyze usability requirements. In order to support the developers or managers to compare the methodologies and to provide them with the criteria needed to select one to deploy, we propose the following framework to compare the different methodologies. The framework consists initially of three steps. First, each methodology will be decomposed into methods. Methods are coherent elements of a methodology. They describe a single function resulting in a sub-deliverable of the methodology. Examples are ‘pre-study’, ‘user profiling’, ‘task analysis’ and ‘usability specification’. In second step each method will be assessed using a set of criteria. Finally, the results of the assessment of each method will be combined to obtain the result for a methodology. This combination can be done in several ways according to the type of criterion. Some, such as required effort, can be added

 ISBN: 978-1-61208-230-1

across methods but if, e.g., a single method requires the availability of an HCI expert, this requirement translates to the methodology as a whole.

The reason we decompose the methodology into methods is to achieve a more accurate and concrete comparison. The methods employed are as will be shown easier to identify, describe and therefore easier to assess while the methodology as a whole will tend to be a fairly complex amalgamation of these constituent parts, which makes direct characterization of this combination much more difficult and dubious. Using the criteria to assess the methods first and then combining the result for each methodology will focus each discussion on a manageable level, thus helping developers to understand what the differences are and why there is a difference between the methodologies. Reasons may be, among others, that different methods or different techniques are used. Methodologies containing different methods will have different properties and therefore different results, but methodologies that include the same methods might also have different properties because the methods use different techniques.

The proposed framework consists of a set of criteria that can be used to assess the methodologies. This set of criteria is divided into four categories, namely the external factors (Section 3.1), the characteristics of the methodology (Section 3.2), the effort (Section 3.3) and the quality and effectiveness (Section 3.4). In the following, a short description of each criterion is given, and the arguments for selecting the criteria are described. Moreover, it is explained how the score is calculated and how the scores are combined for a particular methodology.

A. External factors

The first category concerns external factors. Information about the requirements of a methodology about the external environment in which it is to function is crucial for developers and managers to decide whether or not to apply this methodology in a particular context (also mentioned by Davis [8] as a first step of choosing a strategy for requirement elicitation). The external factors category consists of three criteria:

C1.1 Does a methodology / method need a human computer interaction (HCI) expert?

This criterion answers whether an HCI expert is needed to do this method or methodology properly. It is included in the framework because there are projects that do not allow for the involvement of an HCI-expert, e.g., due to budget reasons or a lack of qualified personnel. This criterion is mentioned in all four assessed methodologies [1], [2], [4], [3]. Each method and the methodology can be given either a plus or a minus for this criterion. A score of '+' indicates that the method/methodology needs an HCI-expert and a score of '-' indicates that it does not need one. If one of the methods needs an expert then the methodology needs an expert as well.

C1.2 Does a methodology / method need access to representative users?

This criterion indicates whether the methodology / method requires access to representative users. Involving users in the project increases the dependencies on external factors. Having access to the representative users and working with them is not a simple task. This property is also mentioned in all four assessed methodologies. Each method gets a plus or minus for this criterion to indicate whether or not it involves representative users. If the methodology does not involve users, a score of '-' is assigned. If there is some user involvement in a methodology, '+' is the result. A methodology that very strongly relies on user involvement gets a score of '++'.

C1.3 Does a methodology / method work with non-experienced users?

Some methods/methodologies require a certain level of knowledge or experience of the users to ensure an efficient communication and collaboration with them [8]. Inexperienced users might have difficulties with articulating their requirements [9], [10]. If this criterion is applicable for the method, then a score between 1 and 5 is assigned. If a method does not involve the user, this criterion is not applicable. For the methodology, a combined score on the same scale is calculated. However, this is not necessarily the arithmetic mean of the results for the methods because some methods may have greater influence on the overall score than others.

B. Characteristics

The second category focuses on the characteristics of the methodology. The characteristics provide the developers with insight whether a methodology is appropriate.

C2.1 Does a methodology / method give strict guidance to help the developers to carry it out?

The methods of the given methodology are assessed on how accurately they are described. Or in other words, whether a non-experienced developer can execute it well based on the description. A scale of '- -' to '++' is the range of the evaluation for this criterion. A combined score for the methodology (also'- -' to '++') is assigned.

C2.2 Does a methodology / method take the user feedback into account for further improvement?

It is very important to take the user feedback into account for further improvement with respect to usability of the system design [11]. A score of '+' or '-' is assigned to indicate whether feedback from users is taken into account. We consider user feedback as the input from the user that is based on a proposal made to the user or a prototype presented to them. If the methodology contains a certain number of methods which take the user feedback for further improvement, then it is argued that the methodology will also get a plus for this criterion.

C. Maintaining the Integrity of the Specifications

The third category is the effort, i.e., the time and the cost that is needed for a methodology. This helps the developer to make tradeoffs.

C3.1 Is a methodology / method time consuming?

This criterion indicates how time-consuming the methods are. A score between '- -' and '++' is the result of this criterion applied on the methods. A score of '++' indicates the method is very time consuming, while a '- -' indicates

 ISBN: 978-1-61208-230-1

that executing the method can be completed in a very short time. A cumulative score of each method is assigned to the methodology. If a project has a time constraint within which it needs to be finished, the cumulative score will help the developers to decide on a methodology.

C3.2 Is a methodology / method common in the software development process?

Time consumption is not an absolute value. It is also related to the degree of integration in common software engineering methods. Integration means less additional work and also will promote more experience with the approach among software engineers, impacting positively on the amount of effort required. The methods that are used in the elicitation and analysis process of the usability requirements might already be included or commonly used in the development process of the product for other reasons. Then the methods might be easily adapted such that it would not take any additional time. A list of commonly used functional requirements elicitation techniques indicate the answer to this criterion [12]. A value between '–' and '+' is assigned to each method to assess whether the method is common or not for software development processes. Of course, this provides only a guideline. Actual fit with a local process will still need to be determined when actually adopting an approach.

D. Quality and effectiveness

The last category is the quality and effectiveness of each method and the methodology. This will also help the developers to make the trade-offs. The objective of this set of criteria is to indicate the level of detail that is elicited.

C4.1 Does a methodology / method elicit enough information to help the developer specify the fit criterion?

Because it is hard to measure the non-functional requirements, eliciting information to specify the fit criterion of the usability requirement might be a crucial factor for selecting a certain methodology [12]. Juristo et al. argued that some proposed methodologies in the literature did not derive enough information to help the developers design and implement the elicited requirements [4]. The methods get a '-', a neutral or '+' for this criterion depending on whether they do not elicit enough information, it depends or it does (explicitly) elicit enough information, respectively. An average within the same range is given to the corresponding methodology.

C4.2 Does a methodology / method elicit the dependencies between the usability requirements and other functional and non-functional requirements?

Usability requirements are sometimes related to specific functional requirements [12]. Knowing the interdependencies between requirements is important for the system design and change management. Therefore, this criterion can be an important factor when selecting a methodology. A scale including '+', neutral, and '-' is used to indicate that dependencies are completely, partially, or not elicited, respectively. The proposed framework is applied to four selected methodologies of respectively Nielsen [1], Carlshamre et al. [2], Seffah et al. [3] and Juristo et al. [4].

IV. METHODOLOGY 1: THE USABILITY ENGINEERING LIFECYCLE (EUL)

The methodology, Usability Engineering Lifecycle (UEL) [1], was proposed in 1992 as one of the first approaches to usability engineering. It presents a practical usability engineering process that can be incorporated into the product development process. This methodology provides a very comprehensive set of methods that can be applied to elicit and analyze the usability requirements. Some of the other methodologies select a subset of the methods that are presented in this methodology. Therefore, this methodology is chosen to be assessed first and the framework is applied.

Ten methods are included in this methodology. Each method will be described shortly.

1. "Know the user". This is used to analyze the individual user's characteristics (e.g., work experience, knowledge level, work environment and social context), the user's current task (e.g., the overall goals, how they approach the task, the needed information, the way of dealing with exceptional circumstances or emergencies), to do functional analysis (e.g., the underlying functional reason for the task) and to have the evolution of the user (e.g., an educated guess about future users and uses).
2. "Doing competitive analysis". This analyzes the existing products heuristically according to established usability guide lines (e.g., usability goals and levels) and performs empirical user tests with these products.
3. "Setting usability goals". This is specified according to the five main usability characteristics (i.e., learnability, efficiency, ability of infrequent users to return to the system without having to learn it all over, frequency and seriousness of user errors and user satisfaction).
4. "Participatory design". This involves users in the design process through regular meetings to help the designer by asking questions and reacting to the designs that they do not like.
5. "Coordinated design of the total interface". This is to achieve consistency of the total interface. This can be approached by using interface standards and the product identity statement (a high-level description of what kind of product it is).
6. "Doing guidelines and heuristic analysis". A list of well-known principles of guidelines for user interface design should be followed. And a heuristic evaluation can be performed on the basis of the guidelines. Prototyping and empirical testing should be combined into iterative design to capture the design rationale, analyze the trade-offs, make the right decision and evolve the design. This combination will be considered as a method.
7. "Prototyping". This is commonly known and often deployed in software engineering.
8. "Empirical testing".
9. "Collect feedback from field use". This method is similar to empirical testing.

Each method is first analyzed separately. The result of applying the framework for all methods and methodologies can be found in Table 1.

 ISBN: 978-1-61208-230-1

TABLE I. RESULTS PER METHOD AND METHODOLOGY

Methods / Methodologies	External factors			Characteristics		Effort		Qual. and Effectiv.	
	C1.1 HCI expert	*C1.2 User access*	*C1.3 Non experts*	*C2.1 Strict guidance*	*C2.2 User feedback*	*C3.1 Effort*	*C3.2 Common in SRM*	*C4.1 Info for fit*	*C4.2 Depend-dencies*
Know the user	-	+	5	+	-	0	+	0	-
Competitive analysis	-	+	5	0	+	0	-	0	-
Setting usability goals	-	-	n/a	+	+	0	-	+	-
Participatory design	-	++	3	+	+	-	-	0	-
Coordinated design	-	-	n/a	-	-	+	0	+	-
Guidelines and heuristic analysis	+	-	n/a	0	-	+	-	-	0
Prototyping	-	-	n/a	-	0	0/++	+	-	+
Empirical testing	-	++	5	+	+	+/++	0	+	+
Collect feedback from field use	-	++	5	0	+	+	0	+	+
Total for UEL	+	+/++	5	0	+	++	-	+	+
Pre-study	-	-	n/a	-	-	-	+	-	-
User profiling	-	+	4	+	-	-	+	-	-
Task analysis	-	+	2	+	+	++	0	-	-
Usability specification	-	-	n/a	+	-	-	+	+	-
Prototype and usability testing	-	+	2	-	+	++	-	+	-
Total for Delta	-	+	3	+	+	-	+	+	-
System summary form	-	+	4	+	-	+	+	-	-
Compile system summary form	+	-	n/a	-	-	+	-	-	-
Context of use portfolio	+	-	n/a	--	-	+	-	-	-
Frs portfolio	-	-	n/a	--	-	+/-	+	-	+
Review and validate integrated picture	+	+	3	--	+	?	-	-	-
Total for ACUDUC	+	+	4	--	+	+	+	-	+
Apply the patterns	-	-	n/a	++	-	0	-	+	-
IFR table	-	+	2	++	+	+	-	+	++
Total for GEUF	-	+	2	++	0	0	-	+	+

Combining the results from the individual methods allows us to judge the methodology as a whole. In order to deploy this methodology, the following criteria for the external factors have to be fulfilled: The developers need to have access to an HCI expert to do the guidelines and heuristic analysis properly (C1.1: +). The methodology needs frequent and reliable access to the representative users in order to perform some of the methods (C1.2: +/++), but it does not require the users to be experienced (C1.3: 5). The methodology does not give very strict guidance to help the developers (C2.1: 0). It suggests a set of techniques to do some of the methods. And the methodology includes methods such as participatory design and empirical testing to elicit the user feedback and take it into account to improve the usability (or the specification of requirements) (C2.2: +). The effort that needs to be put into the methodology is high. Because of the comprehensive set of methods, the methodology is very time consuming. And only a small part of methods are a part of the regular software engineering process (C3.2: -). The rest needs to be added (C3.1: ++). But, the quality of the methodology is fair. It gives enough information about the quantities of usability requirement to specify the fit criterion and it gives an indication about the dependencies between the requirements (C4.1, C4.2: +).

V. METHODOLOGY 2: THE DELTA METHOD

The Delta method [2] is a task-based and usability-oriented approach to requirement engineering. This method was applied in a project to improve the overall usability of the systems delivered. The results prove that the delta method rendered usable systems and helped in eliciting functional requirements in a natural way. This methodology derives its method from the usability definitions in ISO 25062 [7]. Each method corresponds to the users, goals and context of use in the usability definition. This methodology consists of five methods.

1. "Pre-study". Here the scope of the prospective system, the customer categories, and the fundamental services of the system are identified.
2. "User profiling". This provides an overview of the prospective users by means of questionnaire.
3. "Task Analysis". This captures the work tasks of the users through interviews.
4. "Usability specification". This defines an agreed level of usability that the system should supply.
5. "Prototype and usability testing". This tests the results of the last method.

The results for all methods can be found in Table 1. Combining the results from the individual methods delivers the following results. This methodology does not involve usability experts in any of its methods (C1.1: -). Representative users are accessed in most of the methods and this methodology works well with users of moderate level of experience (C1.2: +; C1.3: 3). Guidance is provided by means of questionnaire, activity graph and usability levels (C2.1: +), but most of the methods do not give quantitative

 ISBN: 978-1-61208-230-1

results and users feedback is considered only in prototype testing method (C4.1: +). This methodology proves not to be very time consuming (C3.1: -).

VI. Methodology 3: Approach Centered on Usability and Driven by Use Cases (ACUDUC)

Seffah, Djouab and Antunes [3] present a method that combines usability-centered requirements engineering processes with those based on use cases. They compare similar approaches in both processes and define their own method, called ACUDUC, Approach Centered on Usability and Driven by Use Cases. This is based on the Unified Software development Process [5] as a representative of use-case-driven software engineering methodologies and the RESPECT framework (Requirements Specification in Telematics) [6], a user-centered requirements process. They use the definitions of usability given by ISO 9241 and ISO 9126. The methodology involves five methods.

1. "System Summary form". Here, the stakeholders fill in a form about general characteristics of the system.
2. "Compile System Summary forms". A usability expert combines the stakeholder's forms into a summary form.
3. "Creating the context of use portfolio". This results in a document, which describes all the aspects that have an important impact on the system usability [3].
4. "Creating the functional requirements portfolio". This document consists of use cases and system functionalities as well as characteristics and constraints of the system and UI-prototypes.
5. "Review". Here, all artifacts are being reviewed to ensure integrity and consistency among them.

The results can be found in Table 1. Combining the results from the individual methods delivers the following results. As some of the methods involve usability experts and need representative users, the overall methodology does so as well (C1.1: + and C1.2: +). However, most of the methods do not directly involve users. Those methods that do involve users, can deal with inexperienced users and therefore the overall methodology can be considered to work with inexperienced users rather well (C1.3: 4). The description of the methodology is rather vague in parts and therefore it does not provide the stakeholders of the software engineering project with strict guidance (C2.1: '- -'). There is only one method that takes the user feedback into account. However, we consider this sufficient to conclude that the methodology takes the user feedback into account (C2.2: +). Many of the methods can be assumed to be rather time consuming and so is the complete methodology (C3.1: +). However, it combines, as stated earlier, the elicitation and analysis of functional and usability requirements in a single methodology. Therefore, the methodology overall can be considered to be common for requirements elicitation and analysis because no work is done exclusively for elicitation and analysis of usability requirements (C3.2: +). From the methods as they are described by the methodology's authors, it can be doubted that the non-functional requirements are quantified very precisely as there is no method that does so. Therefore, the whole methodology is judged as not eliciting enough information about quantity (C4.1: -). Because of its integrative (functional and nonfunctional) approach, the methodology can elicit the relation between functional and non-functional requirements rather well (C4.2: +).

VII. Methodology 4: Guidelines for Eliciting Usability Functionalities (GEUF)

The methodology Guidelines for Eliciting Usability Functionalities (GEUF) was proposed in 2007 [4]. It refines the method guidelines and heuristic analysis of the first methodology (UEL). The methodology addresses usability requirements as functional requirements during the requirements engineering stage. Based on the guidelines that are provided in the usability literature, the authors have listed a list of functional usability features as a starting point for identifying usability features with an impact on software system functionality. Based on the HCI literature about each feature (if enough is found), the subtypes are listed for each of the features (called usability mechanisms). For each mechanism, the elicitation and specification guides are defined from a development perspective. A set of issues is derived from the elicitation process and needs to be discussed with stakeholders. An initial common vision of system functionality is built before the developers and the users can discuss whether and how specific usability mechanisms affect the software. Two methods are used.

1. "Apply the patterns". Here a template derived from the research is applied to the specific situation.
2. "Applying the Issue/Functionality/Requirement (IFR) table to the issues".

The results for all methods can be found in Table 1. Combining the results delivers the following results.

The result of the methodology is combined as follows. If the developers select this methodology, there is no requirement for having access to an HCI expert (C1.1: -). The methodology does require the involvement of users to discuss the issues (C1.2: +). Therefore, the users should have a high level of knowledge and/or experience to help the developers find correct answers to the issues (C1.3: 2). It was already indicated that the methodology does take the user feedback into account. But this happens only once, there is no iterative design and continuous involvement of the users, therefore it only gets a neutral (C2.2: 0). The methodology is well explained and makes it easy to systematically apply the templates and the table. Hence, it does give a very strict guidance (C2.1: ++). It is not time consuming as it is a one-time task and only considered the proposed mechanisms (C3.1: 0). It does not elicit other functional requirements nor analyze other aspects (e.g., task analysis). But it does take some effort to learn it because it is a new methodology and needs patience to apply it. The template helps the developers to specify the fit criterion using standardized sentences and using the results of IFR table to fill in the specification (C4.1: +). It also explicitly captures the dependencies between requirements in the table (C4.2: +).

VIII. Discussion

For the external factors, there are only few differences. Both methodologies Delta Method and GEUF can be used

 ISBN: 978-1-61208-230-1

without access to an HCI expert or with only little involvement of such an expert. Unsurprisingly, all methodologies rely on having access to representative users. This can be attributed to the fact that usability requirements are very individual and therefore cannot be properly identified and analyzed without contacting the prospective users. The biggest difference within this category can be identified for criterion C1.3. While the UEL methodology works especially well with inexperienced users, the GEUF methodology should not be used with inexperienced users because it is likely that their needs would not be captured correctly within this methodology. Within the category characteristics, the most noticeable difference lies in C2.1. While Delta method and GEUF give good guidance and UEL earns a neutral score, the ACUDUC methodology scores poorly. The only methodology that does not sufficiently take user feedback into account is GEUF. The third category, time and effort, shows notable differences in results for both of the criteria. The results for criterion C3.1 have great variance. While we consider the Delta method methodology to be least time consuming, the UEL methodology is considered most time consuming. This can be attributed to its comprehensive set of methods. Overall, ACUDUC and Delta method are considered to mainly consist of methods that are common in software development. For ACUDUC, this can at least partially outweigh the relatively high effort in time needed for this methodology. For Delta method, it points to a comparatively small overall effort. Within the category of effectiveness, only the Delta method cannot elicit dependencies between functional requirements and usability requirements. All methodologies except ACUDUC have the potential to analyze the requirements in enough detail to be able to specify a fit criterion.

IX. CONCLUSIONS AND FUTURE WORK

By comparing the four different methodologies for usability requirement elicitation and analysis on the basis of our UREAM framework we could reach the following conclusions. In terms of external factors (like the need of usability experts, access to representative users and their experience) the Delta method and GEUF are probably most cost-efficient as they can be executed without the help of an HCI expert. All methodologies need access to representative users. However, the Delta method and GEUF can be applied well even with non-experienced users. In terms of characteristics of the methodology, the internal factors like taking user feedback into account is considered in all methodologies except GEUF. Both the Delta method and GEUF provide a strict guidance to the developer for executing the methods. So the Delta method obtains a better score in characteristics compared to the other methodologies. In terms of effort, the Delta method is probably more effective as it can handle a tight project schedule and most of the methods are in common to functional elicitation and analysis, so that the effort to capture them is minimized. In terms of quality and effectiveness, GEUF scores well as the developers can do a quantitative analysis with respect to most of the methods, and the methods support the developers to understand the dependencies between other functional requirements. We suggest that regarding UREAM selection for a concrete project, first the individual characteristics of the project have to be considered, and subsequently the framework-based tables can be used. We are convinced that the (initial) UREAM framework has been validated in this research project. However, further research is needed to elaborate the UREAM framework further so that it (i.e., dimensions and criteria) can also offer a structured basis for the development of new and advanced usability requirements elicitation and analysis methodologies.

ACKNOWLEDGMENT

We would like to acknowledge the support of X. Lu, P. Meenakshy, and T. Milde in preparing this paper.

REFERENCES

[1] J. Nielsen, “The usability engineering life cycle”’ IEEE Computer vol. 25, nr. 3 (March) 1992, pp. 12-22.

[2] P. Carlshamre and J. Karlsson, “Usability-oriented approach to requirements engineering”, Proc. ICRE96, IEEE Computer Society Press, Los Alamitos, CA, pp. 145-152., 1996.

[3] A. Seffah, R. Djouab, and H. Antunes, “Comparing and Reconciling Usability-Centered and Use Case-Driven Requirements Engineering Processes”, Australian Computer Science Communications, Vol. 23, nr. 5, 2001, pp. 132 – 139.

[4] N. Juristo, A.M. Moreno, and M. Sanchez-Segura, “Guidelines for eliciting usability functionalities”, IEEE Trans Softw Eng 2007; Vol. 33, nr. 11, pp. 744-758.

[5] Jacobson I., Booch G., Rumbaugh J. The Unified Software Development Process. Object Technology Series. Addison Wesley, 1999.

[6] M.C. Maguire, User-centred requirements handbook. EC Telematics Applications Programme, Project TE 2010 RESPECT (Requirements Engineering and Specification in Telematics), WP4 Deliverable D4.2, version 3.3, May 1998.

[7] ISO IEC 25062 Software engineering — Software productQuality Requirements and Evaluation(SQuaRE) — Common Industry Format(CIF) for usability test reports, Geneva, International Organization for Standardization

[8] G.B. Davis, “Strategies for information requirements determination”, IBM Systems Journal, Vol. 21, nr. 1, pp. 4-30, 1982.

[9] B. Nuseibeh, and S. Easterbrook, “Requirements Engineering: A Roadmap”, The Future of Software Engineering, Special Issue 22nd International Conference on Software Engineering, ACMIEEE, pp. 37-46, 2000.

[10] S. Adikari, C. McDonald, and N. Lynch, “Usability in Requirements Engineering”, Proc. ACIS 2006, Adelaide, Paper 76.

[11] K. Van de Berg, J.M. Conejero, and R. Chitchyan, AOSD Ontology 1.0 - Public Ontology of Aspect-Orientation, Report UTwente, 2005.

[12] H. van Vliet, Software engineering: principles and practice. Wiley, 2008.

[13] A.E. Bayraktaroglu, F. Calisir, and C.A. Gumussoy, “Usability and functionality: A comparison of project managers' and potential users' evaluations”, Procs. IEEE IEEM, 8-11 Dec. 2009, Hongkong, pp. 2019 – 2023.

 ISBN: 978-1-61208-230-1