Method to Define User Interfaces in the Requirements Analysis Phase

Junko Shirogane
Division of Psychology and Communication, Tokyo Woman's Christian University
junko@lab.twcu.ac.jp

Jose Ignacio Panach
Escola Tècnica Superior d'Enginyeria, Departament d'Informàtica, Universitat de València
joigpana@uv.es

Oscar Pastor
Centro de Investigación en Mètodos de Producción de Software, Universitat Politècnica de València
opastor@dsic.upv.es

ABSTRACT

Many requirements for quality in use are elicited in the late development phase. However, if requirements are elicited in the late development phase, the development may return to the previous phase or some requirements cannot be realized due to costs and schedules. To reduce these cases, we propose a method to elicit the requirements in the requirements analysis phase. First, software developers analyze the user characteristics (UCs) of the target users and specify important quality characteristics (QCs) for quality in use and UI design items based on the relationships among UC, QC, and UI design items. Because UI design items are considerations to develop UIs, the specified UI design items are elicited as UI requirements. Thus, when important QCs are specified, UI requirements can be easily elicited by tracing the relationships from QCs to UI design items.

KEYWORDS

User interface, Quality in use, Requirements analysis, Quality characteristics

ACM Reference Format:
Junko Shirogane, Jose Ignacio Panach, and Oscar Pastor. 2018. Method to Define User Interfaces in the Requirements Analysis Phase. In *EICS '18: ACM SIGCHI Symposium on Engineering Interactive Computing Systems, June 19–22, 2018, Paris, France.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3220134.3220137

EICS '18, June 19–22, 2018, Paris, France

ACM ISBN 978-1-4503-5897-2/18/06…$15.00
https://doi.org/10.1145/3220134.3220137

1 INTRODUCTION

User interfaces (UIs) are strongly related to quality in use. Many requirements for UIs are elicited in the late development phase [19] because users try to use the prototypes or actual software to identify issues with UIs in this phase. However, even if UI requirements are elicited in the late development phase, there are cases where the development must return to a previous phase or some requirements cannot be satisfied. To resolve these problems, it is necessary to elicit the UI requirements in the requirements analysis phase. In many cases, UI requirements are related to quality in use, which strongly depends on individual users. UI designs are based on the quality in use. How users feel about UI designs is assumed by the user characteristics (UCs). There are various types of UCs, whose values come from users. That is, the combinations of type-value sets of UCs represent individual users.

Quality in use consists of quality characteristics (QCs) [11][20]. QCs represent various aspects of the quality in use. The combinations of type-value sets of UCs affect whether QCs are satisfied. However, important QCs may differ from the combinations. In addition, strategies to improve quality in use determine UI designs. These strategies include which QCs are important and how to satisfy the QCs. UI designs are realized by combining various items. That is, UI design items are combined to satisfy quality in use.

Herein we propose a method to elicit the UI requirements in the requirements analysis phase. In our method, the relationships between UCs and QCs of quality in use and the relationships between the QCs and UI design items are clarified. Meanwhile, developers analyze the target users. Based on these relationships and UC analysis, important QCs for the target users are specified. This allows that UI design items could be identified. Then these UI design items are elicited as UI requirements. The scope of our method is to support elicitation of UI requirements using the relationships based on the UCs of target users, while the analysis of target users can be performed by existing methods.

2 RELATED WORKS

To develop usable UIs, various methods have been proposed. For example, Mejia-Figueroa et al. proposed a method to specify appropriate UI patterns [17]. They analyzed UCs in terms of users' senses (eyesight, hearing, and touch), cognitive functions (memory and attention), and motor functions (parts of body). Meanwhile, software was considered to be a set of functions and tasks, where tasks were considered to be a set of actions. Users' abilities to execute a task were analyzed. Then they specified UI patterns based on analyses of UCs and task execution abilities. Additionally, Oh et al. proposed a method to develop UIs based on users' cultural backgrounds [21]. They analyzed cultures using task models, specified appropriate colors and texts, and generated UIs in mobile environments. Although these methods analyze UCs to develop UIs, the target types of UCs are limited.

A method has been proposed to elicit requirements based on users' activity data [2]. First, the target end users were specified and data were collected by observing end users' activities. The collected data were classified into behaviors and environments. Next, users' desires related to the system were extracted. Then situation flows were generated as 3-tuple $\{d, A, E\}$ on time t. d indicated the predicted end user's desire, A indicated the end user's action set to achieve a goal corresponding to d, and E indicated the environmental context value. Finally a transition structure for the situations was generated and requirements were elicited. Li et al. developed a modeling language for non-functional requirements (NFRs) [16]. In their modeling language, quality was treated as NFR, together with universality, gradability and agreement. Universality was the degree of requirement subjects (such as websites) that should satisfy the target NFR, while gradability was the degree of NFR satisfaction that a requirement subject should satisfy. Agreement was the degree of subjective satisfaction. NFRs were described like a goal model [24]. However, these methods did not consider UCs.

3 BASIC CONCEPTS

The quality in use depends on feelings of the individual user due to different experiences and preferences. To develop UIs with a high quality in use, UCs must be analyzed and UIs must be implemented by considering the analysis results of UCs. That is, UI design items must be selected based on analysis. However, the abstraction levels of some UI design items are too low to be associated directly with UCs. Meanwhile, the quality in use is defined as several concrete QCs [11] [20]. QCs are satisfied by UI design items. Quality in use depends on the user. Consequently, important QCs must be specified based on the UCs of the target users.

When the UCs of the target users are analyzed and important QCs are specified, UI design items (UI requirements) that should be implemented are elicited. To realize this, it is necessary to clarify the relationships between UCs and QCs as well as the relationships between QCs and UI design items. Since QCs are not always independent and their scopes overlap, the relationships among QCs must be clarified. For example of expert users, QC "efficiency" is important. UI design item "Input support" can satisfy this QC. "Implement auto-complete functions of text" can satisfy "Input support".

4 RELATIONSHIPS OF QC, UC, AND UI DESIGN ITEMS

When the UCs of the target users are analyzed, our method specifies important QCs of quality in use based on the analysis and elicits UI design items that should be implemented as UI requirements. Thus, we previously clarified the relationships among the QCs of quality in use, between UCs and QCs, and between QCs and UI design items. Although we provided these relationships to elicit UI requirements, if QCs, UCs, and UI design items that are not targets are used, software developers must clarify their relationship. Our method can be applied based on the developers' relationships.

Relationships among QCs

First, we clarified the relationships among QCs of quality in use. We adopt QCs of ISO/IEC 25010:2011 (Systems and software engineering -Systems and software Quality Requirements and Evaluation, SQuaRE) [11]. SQuaRE is also a well-known international standard. The QCs of quality in use are defined by two quality models in SQuaRE: the quality in use model and the product quality model. The quality in use model consists of five QCs, where three of them have sub-QCs. Two other QCs, such as effectiveness and efficiency do not have sub-QCs. In the product quality model, QCs related to quality in use are defined as six sub-QCs in the QC "usability". In our method, important QCs for the target users are selected from the sub-QCs in all QCs in both models. Since effectiveness and efficiency do not have sub-QCs, they are also selected as important QCs.

To clarify the relationships among QCs, we construct the relationships among QCs [12][13]. Figures 1, 2, and Table 1 show parts of the results. In Fig. 1, a solid arrow indicates the QC and the sub-QC, while an dotted arrow indicates that the QC of the ending point is satisfied if the QC of the starting point is satisfied. In Fig. 2, if boxes of QCs overlap, their scopes overlap. In Table 1, ↙ indicates that the QC for the quality in use model is satisfied if the QC for the product quality model is satisfied.

Table 1: Relationships between QCs in the quality in use model and the product quality model

			QC "usability" of product quality					
			Appropriateness recognizability	Learnability	Operability	User error protection	User interface aesthetics	Accessibility
Quality in use	Effectiveness				↙	↙		↙
	Efficiency				↙	↙	↙	↙
	Satisfaction	Usefulness			↙	↙	↙	↙
		Trust	↙			↙		
		Pleasure					↙	
		Comfort						

Table 2: Selected relationships between UCs and QCs

UC			Considerations to UI design	Important QC
Type	Value	Condition		
Disability	Blind	Cannot see texts and images	Support devices (e.g., braille display), support tools (e.g., screen reader), operation by only keyboard	• Accessibility • Operability • User error protection
Skill	Novice	Low experience of using the target system	Understandability and undo function are important, dialog type interface is suitable	• Operability • User error protection
	Expert	High experience using the target system	Efficiency is important (minimum messages of warnings and confirmations, and only user-selected help)	• Efficiency • Operability • User error protection

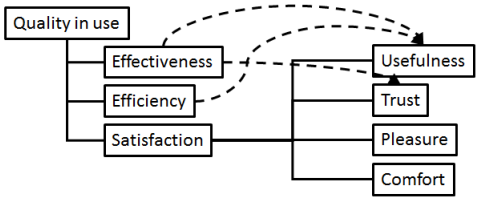

Figure 1: Relationships among QCs in the quality in use model

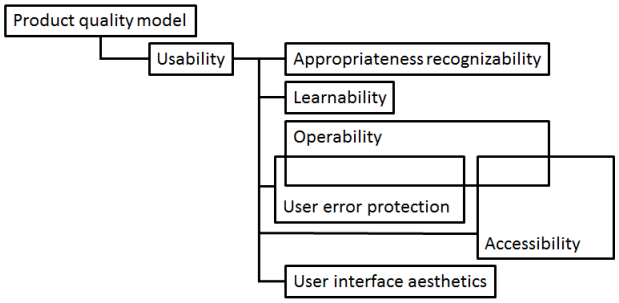

Figure 2: Relationships among QCs in the product quality model

Relationships between UCs and QCs

Important QCs differ from the UCs of target users. UCs consist of various types such as age, sex, computer proficiency, and work [22] [5]. After concrete values are assigned to the UCs by analyzing the target users, important QCs can be specified. Thus, we clarify the relationships between sets of UC types and values (UC type-value sets) and QCs.

Many references describe the types, available values, and conditions of users on the values of UCs and the considerations to UI designs for UC type-value sets [10][8][8][22][23][15]. Herein we analyze these descriptions, QC meanings, and the relationships among QCs in Section 4 and we associate UC type-value sets with QCs. In addition, the relationship levels of QC for UC type-value sets depend on the conditions. Thus, the relationship levels (five levels) are assigned to conditions and QCs. Table 2 provide illustration of the results.

Relationships between QCs and UI design items

The QCs of quality in use are satisfied by designing UIs. In other words, UI design items are combined to satisfy the quality in use. Various UI design items at different abstraction levels have been proposed. Examples of UI design items with high abstraction levels are described in references [8][14][4], while UI design guidelines [18][9][6] are examples of low

Table 3: Selected relationships between QC and UI design items

QC	UI design item (In (): Contribution level) High ← *Abstractionlevel* → Low		
Effifiency	Efficiency of operation	Limit amount of user operations (2)	Make auto-save available (4)
		Input support (2)	Implement auto-complete functions of text (4)
			Do not divide input fields (3)
	UI by skill levels	Use wizard (Note: for skill - novice users) (4)	
		Short cut key (Note: for skill - expert users) (5)	Assign short-cut keys to menu Use well-known short-cut keys (5)
Operability	Operation by users' intents	Make some operation methods available for a task (4)	
		Make UIs customizable (3)	Make bars customizable (4)
			Permit users to change data color and theme (3)

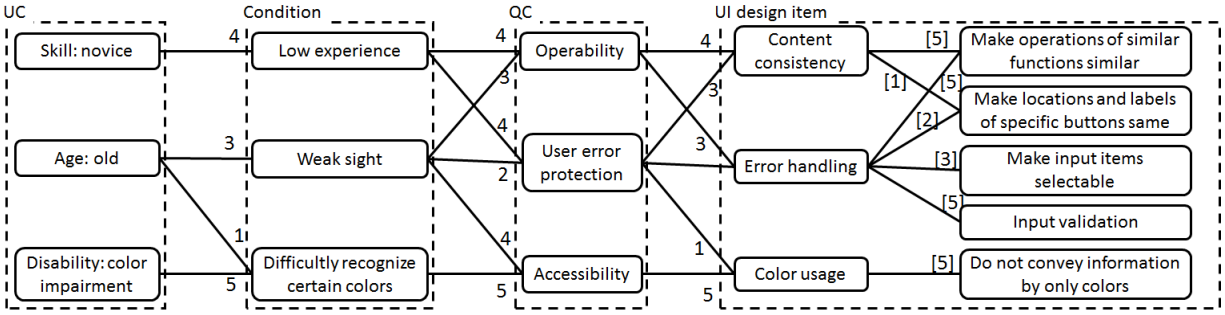

Figure 3: Image of relationships amang UC, QC, and UI design items

abstraction levels. Although it is difficult to directly associate UI design items with low abstraction levels and QCs, UI design items with high abstraction levels can be considered as solutions satisfying QCs, while UI design items with low abstraction levels can be considered as solutions realizing UI design items with high abstraction levels. That is, UI design items with high abstraction levels are associated with low abstraction levels like goal models, and QCs are associated with the UI design items with high abstraction levels.

Thus, we collected UI design items from various references and guidelines, analyzed their meanings, and associated them. Since the abstraction levels of these UI design items are for more than two levels, UI design items are associated as hierarchies. The contribution levels of UI design items with low abstraction levels are assigned. The contribution level means how much a UI item with a low abstraction level helps realize a UI design item with a high abstraction level. Additionally, the target UC type-value sets are specified for some UI design items. Thus, these target UC type-value sets are added to the UI design items as notes. Table 3 provide illustration of these results.

5 STRATEGY TO ELICIT UI REQUIREMENTS

To elicit the UI requirements, software developers must initially analyze target users and specify the UC type-value sets and user conditions corresponding to the sets based on the relationships between the UCs and conditions in Section 4. Then the target UC type-value sets and conditions must be extracted from Table 2. Although the relationships represent available conditions for the UC type-value set, the condition levels depend on the actual users. Thus, developers also specify the relationship levels of the actual users to the related UC type-value sets. We assume that there are five relationship levels.

Figure 3 shows an image of the relationships between UC type-value sets and the conditions. This figure is created by representing the UC type-value sets and the conditions of the extracted relationships as nodes, which are connected using edges. The numbers assigned on the edges between

the UC type-value sets and conditions show the relationship levels.

Second, software developers extract the corresponding relationships of QCs and UI design items for the above relationships of UCs and QCs from Table 3. Similar to the relationships between UC and QC, QC and UI design items are represented as nodes that are connected by edges. Figure 3 shows an image of the relationships among conditions, QCs, and UI design items.

Third, the relationship levels between QC and UI design items are determined. Although relationships between QCs and UI design items are clarified in Section 4, the relationship levels of UI design items to satisfy QCs depend on the relationships between the conditions and QCs, especially for UI design items where the UC type-value sets are added as notes. Thus, the relationship levels are assigned to the UI design items based on the relationships between conditions of the UC type-value sets and QCs. The numbers on the edges between QCs and UI design items and among the UI design items are the relationship levels in Fig. 3. The numbers without [] are the relationship levels, while the numbers in [] are the contribution levels described in Section 4.

Finally, a model of the relationships among UC type-value sets, their conditions, QC, and UI design items like in Fig. 3 is complete, and the UI design items are elicited as UI requirements. For a UI design item, the sum of the numbers of relationships and the contribution levels on all edges that trace from a UC type-value set to a UI design item are calculated. Each UI design item has a sum value. A large sum value indicates a strong connection of the UI design item to QC and UI type-value sets.

6 EVALUATION

To confirm the effectiveness of our method, we developed an application and performed evaluations by subjects.

Procedure

We analyzed the UCs of the subjects as the target users. The subjects were students in Universitat de Valencia (Spain) studying information science. Based on the analysis of their UCs, we developed a calendar application on Android smartphones and evaluated the quality in use of the application. We selected a calendar application because the subjects had smartphones and sufficient domain knowledge.

First, we implemented a questionnaire, which 41 subjects completed, to analyze their UCs. Then we created a persona [3]. Analysis of the persona revealed five important QCs based on Section 4: effectiveness, efficiency, usefulness, operability, and user error protection. Hereafter, "important QCs" and "other QCs" indicate these five important and all other QCs, respectively. Next we elicited UI requirements based on Section 4.

Finally, we evaluated the quality in use of the application and analyzed the results statistically. The evaluation involved 28 subjects, all of whom were involved with the persona creation step. Subjects operated the target application on their own smartphones along with the given tasks, and then answered a questionnaire. The tasks were to customize the colors and event/to-do types, create new events, and view existing events. The questionnaire involved two parts: (1) important QCs when choosing a calendar application, and (2) satisfaction of the QCs for the target application. For each part, the questions were answered on seven-level scale of "not important" - "important", "strongly disagree" - "strongly agree", and "not satisfied" - "satisfied", respectively. Additionally, there were free comments about the target application. For Parts (2), subjects could answer "not applicable" to questions inapplicable to the target application.

To analyze this evaluation, we classified QCs into two groups: important QC (important group) and other QC (other group). The confidence interval was 95% for all statistical analysis.

Evaluation results of important QCs

Figure 4 shows the averages of the answers to important QCs. We applied the T-test [7] to the students' answers of the important group and the other group. By the T-test, averages of students' answers of important group ant the other group were compared. These two groups differed significantly ($t = 3.860$, $df = 316$, $p < .05$). That is, subjects answered that the QCs of the important group were more important than the QCs of the other group.

In addition, multiple comparisons analyzed individual important and other QCs [1]. Significant differences between important QCs and other QCs include: effectiveness - environmental risk mitigation, efficiency - environmental risk mitigation, usefulness - environmental risk mitigation, operability - economic risk mitigation, health and safety mitigation, and environmental risk mitigation, and user error protection - environmental risk mitigation.

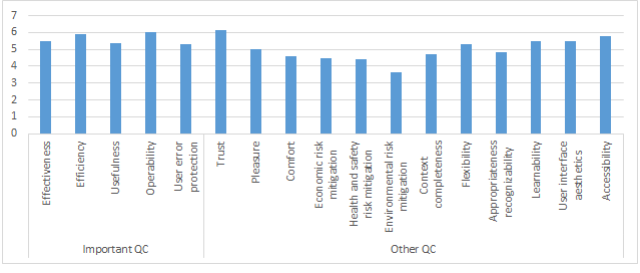

Figure 4: Results of important QCs (Average)

Evaluation results of QC satisfaction

Figure 5 shows the averages of the answers to the satisfaction of QCs for the target application (Part (2)). The T-test indicated that important and other groups significantly differed ($t = 2.476$, $df = 438$, $p < .05$). That is, subjects considered that the important QCs were satisfied more than the other QCs. Individual QCs did not significantly differ in the multiple comparisons.

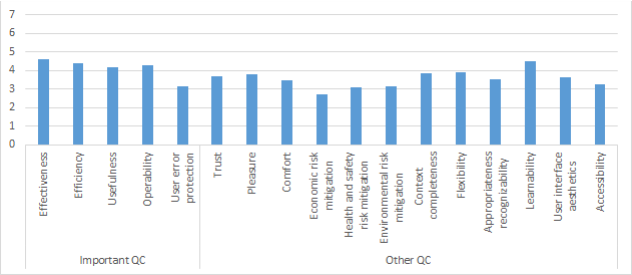

Figure 5: Results of QC satisfaction for the target application (Average)

Discussion

For the important QC analysis results of Part (1), the average of the subjects' answers to all QCs of the important group was 5.62. The QCs of the other group with individual averages greater than 5.62 were "trust" and "accessibility". The QC of "trust" had the highest average, indicating that the subjects verified that the calendar application was suitable for their intent. There are two reasons to explain why "trust" was important for most subjects. First, many people frequently use a calendar application as part of daily activities. Second, obstacles may occur if the application does not work as intended. Meanwhile, the QC of "accessibility" tends to focus on users with disabilities and elderly users. Because the QC of "accessibility" also targets other users, the subjects might consider that it is important. To analyze whether the QC is an important QC, the characteristics of the target application must be included in the analysis of important QC, and UCs must be analyzed in more detail. However, considering these findings and the T-test holistically, important QCs were more important for the subjects as a whole, confirming that our method can specify important QCs.

For QC satisfaction of Part (2), the average of the subjects' answers to all QCs in the important group was 4.05, while only one QC ("learnability") in the other QC group had an average of more than 4.05. As described above, since subjects were familiar with calendar applications, they might be able to easily understand the operations of the target application. In addition, individually important QCs did not significantly differ from other QCs in the multiple comparisons. However, subjects' answers to the important group significantly differed from the other group according to the T-test. That is, the target application realized important QCs more than other QCs by UI design items, confirming the appropriateness of relationships between QCs and UI design items.

On the whole, our method elicited important QCs and UI requirements, allowing the target application to be developed appropriately. Thus, the effectiveness of our method is confirmed.

7 THREATS TO VALIDITY

In this paper, the target users of our methods were university students using a calendar application. However, UCs depend on the users. Thus, different target users may elicit different important UC and UI requirements. Similarly, the target application also affects which QCs are important. Different target users or a different target application may change the evaluation results in Section 6.

We used 95% as the confidence interval to analyze the evaluation results statistically. Although this value is common in statistical analysis, whether significant differences between important and other groups/QCs are judged may change if another value is used as the confidence interval.

8 CONCLUSION

To reduce returning to a previous development phase or not considering UI requirements, we propose a method to elicit UI requirements in the requirements analysis phase. Our method provides the relationships among UCs, QCs, and UI requirements in advance. Then software developers analyze the UCs of target users, specify important QCs, and elicit UI design items to be considered as UI requirements based on the relationships. If necessary, the relationships that developers create can be used in the elicitation. We applied our method to a calendar application on Android smartphones, and evaluated its effectiveness. The evaluation consisted of a questionnaire that asked the subjects about important QCs, evaluations of quality in use for the target application in terms of QCs, and a questionnaire for QC satisfactions of the target application. Although individual important QCs did not significantly differ from other QCs, as a group, important QCs significantly differed from the other QCs, confirming that our method can effectively elicit UI requirements.

Future works include:

- Applying and evaluating our method by subjects
- Specifying important QCs by analyzing the application.
- Proposing methods to generate UI prototypes

ACKNOWLEDGEMENT

We would like to thank the students in Universitat de Valencia (Spain) for answering questionnaire and evaluating our application. This paper has the support of Generalitat Valenciana through project IDEO (PROMETEOII/2014/039) and Spanish Ministry of Science and Innovation through project DataME (ref: TIN2016-80811-P).

REFERENCES

[1] Yosef Hochberg and? Ajit C. Tamhane. 2009. *Multiple Comparison Procedures.* Wiley.

[2] Nimanthi L. Atukorala, Carl K. Chang, and Katsunori Oyama. 2016. Situation-Oriented Requirements Elicitation. In *Procs. of IEEE 40th Annual Computer Software and Applications Conference.*

[3] Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel. 2014. *About Face: The Essentials of Interaction Design.* Wiley.

[4] Scott Faranello. 2016. *Practical UX Design.* Packt Publishing.

[5] Elizabeth Goodman, Mike Kuniavsky, and Andrea Moed. 2012. *Observing the User Experience: A Practitioner's Guide to User Research.* Morgan Kaufmann.

[6] Google. 2015. User Interface Guidelines. Retrieved 11 Jan., 2017 from `https://developer.android.com/guide/practices/ui_guidelines/index.html`. (2015).

[7] Scott Hartshorn. 2017. *Hypothesis Testing: A Visual Introduction To Statistical Significance.* Independently published.

[8] Ichiro Hirata and Toshiki Yamaoka. 2013. A logical design method for user interface using GUI design patterns. In *Proceedings of the 15th international conference on Human-Computer Interaction: human-centred design approaches, methods, tools, and environments*, Vol. Part I.

[9] Apple Inc. 2017. Human Interface Guidelines. Retrieved 11 Jan., 2017 from `https://developer.apple.com/ios/human-interface-guidelines/overview/ themes/`. (2017).

[10] ISO. 2006. *ISO 20282-1:2006, Ease of operation of everyday products – Part 1: Design requirements for context of use and user characteristics.*

[11] ISO/IEC. 2011. *ISO/IEC 25010:2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.*

[12] ISO/IEC. 2016a. *ISO/IEC 25022:2016, Systems and software engineering – Systems and software quality requirements and evaluation (SQuaRE) – Measurement of quality in use.*

[13] ISO/IEC. 2016b. *ISO/IEC 25023:2016, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality.*

[14] Jeff Johnson. 2014. *Designing with the Mind in Mind, Second Edition: Simple Guide to Understanding User Interface Design Guidelines.* Morgan Kaufmann.

[15] Michal Levin. 2014. *Designing Multi-Device Experiences: An Ecosystem Approach to User Experiences across Devices.* O'Reilly Media.

[16] Feng-Lin Li, John Mylopoulos, Renata S. S. Guizzardi, Giancarlo Guizzardi, Alexander Borgida, and Lin Liu. 2014. Non-functional Requirements as Qualities, with a Spice of Ontology. In *Procs. of 22nd IEEE International Conference on Requirements Engineering.*

[17] Andres Mejia-Figueroa, Maria de los Angeles Quezada Cisnero, and J. Reyes Juarez-Ramirez. 2016. Developing Usable Software Applications for Users with Autism: User Analisys, User Interface Design Patterns and Interface Components. In *Procs. of 4th International Conference in Software Engineering Research and Innovation.*

[18] Microsoft. 2015. *Windows 10 User experience guidelines for Universal Windows Platform (UWP) apps.*

[19] Takako Nakatani, Shouzo Hori, Michio Tsuda, Mari Inoki, Keiichi Katamine, and Masaaki Hashimoto. 2009. Towards a Strategic Requirements Elicitation - A proposal of the PRINCE Model. In *Procs. of the 4th International Conference on Software and Data Technologies (ICSOFT2009).*

[20] Jacob Nielsen. 1994. *Usability Engineering.* Morgan Kaufmann.

[21] Jung-Min Oh and NamMee Moon. 2012. Towards a cultural user interface generation principles. *Multimedia Tools and Applications* 63, 1 (2012).

[22] Michael Richter and Markus Flueckiger. 2016. *User-Centred Engineering: Creating Products for Humans.* Springer.

[23] Ben Shneiderman, Catherine Plaisant, Maxine Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. 2017. *Designing the User Interface: Strategies for Effective Human-Computer Interaction, Global Edition.* Pearson Education Limited.

[24] Axel van Lamsweerde. 2004. Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In *Procs. of 12th IEEE International Conference on Requirements Engineering.*