INTEGRATIVE CREATION OF REQUIREMENTS DOCUMENTS FOCUSED ON USER EXPERIENCE

Junko Shirogane
junko@lab.twcu.ac.jp
Tokyo Woman’s Christian University, Japan

ABSTRACT
Typically user interface (UI) requirements are elicited in the late development phase because users try to use the software in this phase. UIs strongly reflect user experience (UX), which indicates the experience obtained using the software and includes usability, comfort, and satisfaction. Practical software must have a high UX. However, if the UI requirements are elicited in the late development phase, all requirements may not be realized, which may result in a low UX. To prevent this, UX requirements must be elicited in the early development phase. Herein a method is proposed to integrate requirements documents, such as requirements specifications, use case diagrams, and scenarios, and UX requirements. First, UX elements, which should be elicited as requirements and their description strategies, are defined based on the survey results of various papers and books. Second, use case diagrams and scenarios are generated from the described UX requirements. Because software is implemented based on use case diagrams and scenarios, gaps between the described requirements and use case diagrams and scenarios must not exist. Finally, developers customize the generated use case diagrams and scenarios. Hence, the proposed method reduces omissions in the UX requirements, realizing software with a high UX.

KEY WORDS
Requirements specification, User experience, Use case, Scenario

1 Introduction
User experience (UX) means experiences gained by using software [1]. Because UX emphasizes not only a sufficient number of functions and their usability but also comfort and satisfaction of users before/during/after using software, it is often employed to develop usable software. The diversity in both the users and the environment (i.e., conventional Web, computers, tablets, and smart phones) increases the importance of UX. To develop software with a high UX, user interfaces (UIs), which are software components that directly interact with users, are important because they allow users to interface with the software. Hence, UX must be related to UI designs.

To realize a high UX, users’ characteristics and use environments must be defined in detail. According to reference [2], UI requirements tend to be elicited in the late development phase because users try the actual software and provide feedback in this phase. Although the UI requirements typically appear when users deploy the software, the costs and development time that realize the requirements vary greatly. That is, if the requirements are elicited in the late phase, not all the requirements may be realized. Thus, UI requirements should be elicited in the requirements analysis phase of software development and be reflected in the requirements documents (e.g., requirements specification, use cases, and scenarios). To achieve this, user characteristics of the software must be specified as UX requirements.

Requirements are described by requirements specifications, use case diagrams, and scenarios. Requirements specifications include requirements for functions, users’ characteristics, and use environments [3]. Use case diagrams represent the interactions between users and functions, while scenarios represent the detailed interactions in the use cases. Functions are outlined in the requirements specifications, but their details are expressed in use case diagrams and scenarios. Although UX requirements are described in the requirements specifications, they are not described in the use case diagrams and scenarios. Because use case diagrams and scenarios form the basis of designing and implementing software, all the requirements described in the requirements specifications must be reflected in the use case diagrams and scenarios.

Thus, this paper proposes a method to not only describe UX requirements in the requirements specifications but also to reflect them in the use case diagrams and scenarios. The IEEE Std. 830-1998 (hereafter, IEEE-830) format [3] defines how to list the UX elements and describe them in the requirements specifications based on previous works. In the proposed method, this format is used to generate the use case diagrams and scenarios from the requirements specifications. Then software developers confirm and customize the generated use case diagrams and scenarios, enabling the UX requirements to be elicited in the requirements analysis phase of software development. Consequently, requirements specifications can be seamlessly connected to use case diagrams and scenarios.

The rest of paper organized as follows. Section 2 describes related works, while section 3 highlights the features of this method. Section 4 defines UX and the relationships between UX and UI design items. Section 5
shows the formats of the requirements specifications for the UX requirements. Section 6 provides strategies to generate use case diagrams and scenarios from requirements specifications. Section 7 evaluates this method, and section 8 concludes this paper.

2 Related works

Many studies for UX and requirements specifications have been performed. Ormeno et. al have proposed a method to capture usability requirements based on guidelines [4]. In this method, questions and selectable answers are created based on UI guidelines. For a selectable answer, detailed questions and answers are prepared. These questions and answers are organized as tree structures. Users select answers along with the tree structures, then UI designs are determined. Although users’ intents can be reflected to UI designs, UX requirements themselves are not be elicited.

To integrate requirements engineering (RE) and UX design (UXD), product life cycle management (PLM) [5][6] has been enhanced [7]. The authors listed analysis of misconceptions and myths for RE and UXD in industry. Based on the analysis, this method have proposed an integration framework based on the PLM. In this framework, RE and UXD professionals interact closely, artifacts are shared, and final documents are created. Due to this method, UX requirements can be appropriately elicited. However, support for descriptions of UI requirements without omissions to requirements specifications are not sufficient.

A method to describe requirements specifications based on the IEEE-830 format using ontology has been proposed [8]. In this method, first, requirements engineer capture requirements based on an ontology, and analyze them as user and system requirements, and the ontology is updated. Next, the requirements are formalized and clarified. Then, the requirements are verified, and requirements specifications based on the IEEE-830 format. Finally, the requirements are validated based on the ontology. Although requirements specifications that omissions of requirements are reduced can be described based on IEEE-830 format, support for eliciting UX requirements are not enough.

3 Features

3.1 Fewer omitted UX requirements

Requirements specifications tend to describe the requirements related to functions (e.g., concrete functions and external interfaces). Although the requirements related to functions are intensively elicited, the UX requirements are not. However, both are important. If UX requirements are insufficiently described in the requirements specifications, users feel that the software is unusable and are unsatisfied.

The proposed method prepares elements to describe UX requirements in the requirements specifications. These items can be used as a checklist to elicit the UX requirements. Hence, fewer UX requirements are omitted.

3.2 Requirements elicitation of UX in the early development phase

In many cases, UX requirements are realized as UIs. Often UI requirements are elicited in the late development phase [2] because this is the phase where users actually try the software. Unfortunately, time and costs may prevent UX requirements from being realized.

To circumvent this, UX requirements must be elicited in the early development phase. The proposed method allows the UX requirements to be determined in the early development phase.

3.3 Seamless connections between the requirements documents

Requirements specifications only outline functions. Software is implemented based on use case diagrams and scenarios, which provide details of functions. Thus, requirements specifications must be seamlessly connected to use case diagrams and scenarios.

In the proposed method, use case diagrams and scenarios are semi-automatically generated from the requirements specifications descriptions. Thus, requirements specifications can be reflected in the case diagrams and scenarios without gaps.

4 UX and UI design

4.1 Principles of UX

UX indicates experiences that users obtain from products (e.g., software, services, and tools) [1]. Software development commonly employs the terms of “usability” and “accessibility”, which focus on operability and potential use, respectively. UX is a wider concept, which also encompasses a positive attitude, satisfaction, and comfort before/during/after operating the software.

Software with a high UX can be realized based on UX requirements. First, many UX elements (e.g., age, disabilities, skills, and cultures) are defined after referring to various papers and books, such as reference [9]. Then these elements are classified into users’ characteristics, social environments, usage environments, etc.

In this paper, UX elements indicate the detailed types of UX. UX requirements, which are described in the requirements specifications, can be defined by eliciting the actual users’ situations that corresponds to the UX elements.
4.2 Relationships between UX elements and UI design items

UX is strongly reflected by UIs because users directly interact with UIs. To realize a high UX, UIs must be designed based on the UX requirements. However, the relationships between UX elements and UI design items must be specified because there are various UI design items.

UI design items are listed by surveying existing UI guidelines, such as “Guidelines” for Windows desktop applications [10] and OS X Human Interface Guidelines [11]. Examples of UI design elements include layout, size, widget types, colors, term usage, and input/output device. Based on a survey, Table 1 lists examples of the correspondences between UX elements and UI design items.

Table 1. Examples of relationships between UX elements and UI designs

<table>
<thead>
<tr>
<th>UX elements</th>
<th>UI design items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Size, colors, and terminology</td>
</tr>
<tr>
<td>Disability</td>
<td>Size, colors, and input/output devices</td>
</tr>
<tr>
<td>Skill</td>
<td>Interaction sequence and terminology</td>
</tr>
<tr>
<td>Culture</td>
<td>Term of usage, colors, and icons</td>
</tr>
</tbody>
</table>

5 Description of UX requirements

The proposed method assumes that requirements specifications are described in the IEEE Std. 830-1998 (hereafter, IEEE830) format [3], which defines considerations and characteristic to describe appropriate requirements specifications in the proper format. The format indicates a section structure (lists of sections and sub-sections) and defines the contents described in each section or sub-section, reducing the number of omitted requirements.

“User characteristics” are mainly related to UX requirements, while “Functions” are used to generate use case diagrams and scenarios. “User characteristics” describe the users (e.g., age, skills, and knowledge), while “Functions” provide detailed behaviors of functions (e.g., input/output items, sequences of operations, and error handlings).

5.1 Format to describe UX requirements

Software often has several users, and UX requirements are different from users. UX requirements corresponding to UX elements include detailed types and characteristics/levels. Also, basic strategies of UI designs are determined by UX elements and the types. Thus, UX requirements are described using following format (hereafter called the UX format).

Element: Name of UX element (given in this method)

<table>
<thead>
<tr>
<th>Corresponding user:</th>
<th>Types of users that have this UX requirements as a user characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>State or values of users corresponding to this element</td>
</tr>
<tr>
<td>Characteristics/Level:</td>
<td>Level of state of the type (if necessary)</td>
</tr>
<tr>
<td>Related UI design item:</td>
<td>Considerations of UI designs for the UX element (given in this method)</td>
</tr>
<tr>
<td>Additional issue:</td>
<td>An additional UI design issue for the UX element except for “Related UI design item” (if necessary)</td>
</tr>
</tbody>
</table>

The “User characteristics” section of the requirements specifications lists the formats of all UX elements. “Element” is entered in advance with the name of the UX element. “UI design issue” is filled in based on Table 1. Then software developers enter the other items. Figure 1 shows an example of the user characteristics for users with weak eyesight.

![Figure 1. Example describing a user characteristics (UX requirements)](image-url)

The UX requirements can be elicited by various methods (e.g., interviews and questionnaires). I previously proposed a method to elicit requirements for disabled users by a questionnaire where the requirements were established by statistically analyzing concrete responses to a questionnaire derived from accessibility guidelines [12]. Like this, the proposed method assumes that the UX requirements are elicited by various methods to determine the UX requirements. Then the UX format can be used to create questions in the requirements elicitation.
5.2 Format to describe functions

Who use each function and what situation the function is used in are quite important for UX, because strategies of UI designs are different from the users and situations. Thus, descriptions of functions must clarify these. Also, it is necessary to generate use case diagrams and scenarios. To satisfy these, the proposed method defines the formats to describe the “Function” section of the requirements specifications as follows (hereafter referred to as the function format). This format includes contents defined by IEEE-830.

**Function name:** Name of the function

**Operator:** Permitted operators of the function

**Usage situation:** Situation to use the function (e.g., user’s mental condition and temporal condition)

**Meaning of input/output item:** Definition of each input/output item

**Operation and response sequence:** Sequence of user operations and responses, including input/output items

- Success/fail cases are separately described using “Case (Success)” /“Case (Fail)” keywords.
- Conditions that sequences become success/fail are described after “Case (Success)” /“Case (Fail)” keywords.

**Valid and invalid input:** Range or list of valid/invalid inputs

**Abnormal responses:** Error occurrence situations and their handling

**Formula:** Conversion formula of the input to the output

Figure 2 shows an example of user authentication, which describes a function using this function format.

6 Use case diagram and scenario generation

Because UIs reflect UX, UIs must be designed based on the UX requirements. There are various UI design items,
including window layouts, interaction sequences and colors. Thus, the use case diagrams and scenarios must reflect the UX requirements. In the proposed method, the correspondences between UI design items and UX elements are defined, the use case diagrams and scenarios are generated from requirements specifications, and then developers complete the process by customizing the generated use case diagrams and scenarios.

6.1 Classification of UI design items

As mentioned in section 4.2, UI design items are extracted from the UI guidelines. Some items should be described as requirements in the requirements analysis phase, while others should be determined in the design or implementation phase. Thus, UI design items are classified as either items for requirements analysis phase or items for other phases (Table 2).

Table 2. Examples of UI design classification

<table>
<thead>
<tr>
<th>Requirements analysis phase</th>
<th>Other phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction sequence</td>
<td>Window layout</td>
</tr>
<tr>
<td>Terminology</td>
<td>Icons</td>
</tr>
<tr>
<td>Error handling</td>
<td>Operation method</td>
</tr>
<tr>
<td>Feedback</td>
<td>Widget Size</td>
</tr>
<tr>
<td>Display timing</td>
<td>Widget type</td>
</tr>
</tbody>
</table>

UI design items that should be considered in the requirements analysis phase are reflected to use case diagrams and scenarios. In contrast, other UI design items are reflected in UI prototyping, which is a future extension of this method.

6.2 Generation

6.2.1 Use case diagram generation

Use case diagrams consist of actors and use cases. Actors are users and external systems that interact with the target software. Section 5.2 describes operators specified in the “Operator” item of the function formats, and the operators are identified as actors in the proposed method. For example in Fig. 2, because “Teacher” and “Student” are identified as actors, the UX requirements in the “User characteristics” section of the requirements specifications are added to the actors as additional information.

Because use cases are considered as software functions, each function described in the function format is identified as a use case. “Function name” of the function format is applied to the name of the use case. After generating the actors and use cases, the interactions between actors and use cases are represented. The “Operator” items (that is, actors) connect the interactions with use cases.

In addition, some functions are executed only in specific situations, such as users’ mental conditions, temporal conditions, and execution mode, as described in the “Usage situation” item of the function formats. The description is set as pre-condition of the use case.

Figure 3 shows an example of a generated use case diagram from Figs. 1 and 2.
6.2.2 Scenario generation

Scenarios represent concrete interaction sequences between actors and software in a use case. A use case has several scenarios, which can be categorized in three types: main scenarios, alternative scenarios, and exceptional scenarios [13]. Main scenarios represent typical interaction sequences, while alternative scenarios represent optional interaction sequences. Both of these successfully execute the use case. On the other hand, exceptional scenarios represent error sequences; that is, exceptional scenarios fail to execute the use case.

Scenarios are generated from the “Operation and response” items of the function formats in section 5.2. In the proposed method, three types of scenarios are generated. Additionally, success/fail sequences are represented by “Case (Success)”/“Case (Fail)” keywords. Thus, sequences that are excluded in the description of “Case (Success)”/“Case (Fail)” keywords are applied as parts of the interaction sequences of all scenarios for the use case. Sequences described using “Case (Success)”/“Case (Fail)” keywords are applied as parts of the interaction sequences of individual scenarios. Sequences of “Case (Success)” become part of the interaction sequences of the main or alternative scenarios, while those of “Case (Fail)” become part of the interaction sequences of exceptional scenarios. The first success sequence in the function format is indicated as a main scenario, while the other success sequences are identified as alternative scenarios. In addition, each interaction is represented by an itemized sentence with one subject and one predicate. Thus, descriptions that include sentences connected by “and” are divided into individual sentences. Figure 4 shows examples of generating scenarios from “Operation and response sequence” item in Fig. 2. After generating scenarios, developers customize and complete them.

7 Evaluation

To evaluate the effectiveness of this method, followings were confirmed.

- Whether contents of user characteristics in existing requirements specifications could be described using the proposed method’s UX format as UX requirements.
- Whether use case diagrams and scenarios could be generated from descriptions of functions using this method’s function format.

Requirements specifications for these evaluations were searched in internet, and documents that satisfied following conditions were selected as the target requirements specifications.

- The type of document files were PDF.
- The document names included the term “requirements specification”.
- The documents included descriptions of software functions.

7.1 Descriptions using UX format

To confirm whether contents of UX requirements in existing requirements specifications could be described using UX format, descriptions about target users were analyzed. Table 3 shows the results.

In this table, “A”–“D” indicate requirements specifications. “Type” indicates the system types described in requirements specifications. “All” indicates the number of UX elements that should be described in the requirements specifications. “Possible” indicates the number of UX elements that could be described using UX format, while “Difficult” indicates the number of UX elements that the descriptions in the requirements specifications were difficult to describe using the UX format. “Not enough” indicates
Table 3. Results of UX requirements descriptions

<table>
<thead>
<tr>
<th>Type</th>
<th>All</th>
<th>Possible</th>
<th>Difficult</th>
<th>Not enough</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Library</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>B Electrical</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C Digital archive</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D Reservation</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

the number of UX elements that should be described in requirements specifications, but were not described enough. “None” indicates the number of UX elements that should be described, but were not.

The most of “Difficult” were the descriptions that UX requirements were not needed. For example, “Target users do not have disabilities” were described in requirements specification B. Because UX format of this method assumes that characteristics that users have are described, UX requirements that were not needed cannot be described enough by the UX format. Thus, it is necessary to improve UX format in order to describe them.

Also, there were many “Not enough” and “None” of UX requirements. The examples of “Not enough” were ages and skills, while the examples of “None” were disabilities and skills. Because these UX requirements strongly affect UI designs, it is necessary to describe them in requirements specifications.

According to these results, there were many UX requirements that should be described but were not described in requirements specifications. This means that omissions of UX requirements can expect to be reduced by preparing UX formats. According to these results, although improvements of UX formats are required, they can contribute requirements elicitations and requirements descriptions.

7.2 Generation of use cases and scenarios

In this evaluation, whether use case diagrams and scenarios could be generated from existing requirements specifications were analyzed. Table 4 shows the results.

Table 4. Results of use case diagram and scenario generation

<table>
<thead>
<tr>
<th></th>
<th>Use cases</th>
<th>With actors</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>24</td>
<td>3</td>
<td>133</td>
</tr>
<tr>
<td>B</td>
<td>16</td>
<td>16</td>
<td>31</td>
</tr>
<tr>
<td>C</td>
<td>45</td>
<td>45</td>
<td>71</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>16</td>
<td>72</td>
</tr>
</tbody>
</table>

In this table, “A”-“D” indicate requirements specifications, same as Table 3. “Use cases” indicates the number of generated use cases, while “With actors” indicates the number of use cases that interacted actors could be clearly recognized. “Scenarios” indicates the number of generated scenarios.

Use cases and scenarios were generated almost appropriately from the descriptions in requirements specifications. However, target users of functions could not be clearly recognized in the requirements specification A. Target users were described in only some requirements as the subjects. In other requirements specifications, the target users were clearly described by functions. When target users were not described, interactions between users and use cases could not generated. Target users of functions must be described in requirements specifications. It is important to describe target users for functions to realize UX, because UI designs are different from user characteristics.

Also, varieties of sequences were not described clearly in requirements specifications A, C, and D. In case of varieties of sequences, alternative and exceptional scenarios could not be generated. Varieties of sequences are also important contents of requirements specifications.

According to these results, usage of function formats in this method can resolve these problems. Thus, effectiveness of function formats in this method could be confirmed.

8 Conclusion

This paper proposes a method to describe the requirements specifications with UX requirements and generate use case diagrams and scenarios from the descriptions. After defining the formats to describe the requirements related to UX, existing requirements specifications with an emphasis on functions related to UX are determined. Because UX elements are listed in the requirements specification in advance, the proposed method clarifies the requirements, decreasing UX requirements omissions. Additionally, UX requirements can be elicited in the early development phase, reducing the development time and costs. Finally, use case diagrams and scenarios can be generated from the described requirements specifications, enabling a seamless connection between use case diagrams and scenarios. As evaluations, descriptions of UX requirements and functions using UX and function formats were confirmed. Reductions of omissions of UX requirements and function descriptions were expected by the proposed method. Appropriate generation of use case diagrams and scenarios were also confirmed.

Future research includes:
Developing support tools to describe requirements specifications
There are many types of UX requirements and elements. To fill out UX and function formats without omission, it is necessary to prepare its support tools for describing UX requirements.

Improving the strategies to generate use case diagrams and scenarios
Currently, use case diagrams and scenarios are generated from descriptions of functions and UX requirements. However, various other descriptions, such as external interfaces and constraints, are included in requirements specifications. Use of these other descriptions and generation of more appropriate use case diagrams and scenarios must be considered.

Developing prototype generation system that apply UX requirements from the generated and customized use case diagrams and scenarios
Although it is important for users to confirm the actual UI design of the target software in terms of whether their requirements are appropriately reflected to requirements documents, confirmation using only use case diagrams and scenarios is difficult. Prototypes of UIs are effective for the confirmation. Thus, development of UI prototype generation system is one of the future works. UI prototypes are generated from the generated and customized use case diagrams and scenarios in requirements analysis phase.

References