Hidden-use Case for Eliciting Quality in Use

Natsuko Noda
Shibaura Institute of Technology
Tokyo, Japan
nnoda@shibaura-it.ac.jp

Tomoji Kishi
Waseda University
Tokyo, Japan
kishi@waseda.jp

Shin'ichi Fukuzumi
RIKEN
Tokyo, Japan
shin-ichi.fukuzumi@riken.jp

***Abstract*— Finding use cases and developing a use case diagram is one of the most powerful techniques for eliciting and specifies requirements. However, the notion of use case cannot handle the needs of indirect users correctly, because the use case only describes the direct interaction between the system and the actors. In this short position paper, we describe our ideas about extension of the concept of use case and use case description, to describe and analyze the needs of the indirect users.**

Keywords*— *quality in use, indirect user, use case, use case diagram, quality model

I. Introduction

Eliciting requirements is important and at the same time very difficult. If there is any requirement that is hidden and not described explicitly, appropriate software cannot be developed. However, digging up all requirements and expressing them explicitly is hard.

For this task, finding use cases [1][2] and developing a use case diagram [2] is one of the most powerful technique. A use case is a description of an interaction sequence between actors and a target system. It expresses a behavior of the system. Therefore, it exposes necessary functions of the system. It does not describe any quality requirement directly, but it makes it easy to consider necessary quality attributes along with the interactions indicated by it; e.g., the system must respond in 1 msec at this step in this use case.

Quality requirements are ideally considered and organized in reference to the quality models, which are standardized in ISO/IEC 25010:2011 [3], one of the SQuaRE series. The models are the product quality model and the quality in use model. Quality in use is the quality that a stakeholder recognize when he/she utilizes a system. The stakeholder may be of a various type. In the SQuaRE, stakeholders include an indirect user, who receives output, but does not interact with the system.

To consider the needs of the indirect users, can we utilize the concept of use cases and use case diagram? Maybe no; a use case describe an "interaction" sequence between actors and a target system, and indirect users do not have interaction with the system. However, these indirect users may relate to an interaction occurred in a specific use case, because the indirect users receive output, which is produced only when the system is used. Considering this situation, how about enlarging the concept of the use case and extending the use case diagram?

In this short position paper, we describe our ideas about extension of the concept of use case and use case description.

II. Use Case

The notion of use cases was originally introduced by Jacobson [1]. According to his definition, a use case is a special sequence of transactions in a dialogue with the system, that will be performed by a user. This kind of user is called as an actor. Jacobson also introduced a model for requirement specifications; that is a use case model, consists of actors and use cases.

The concept and the notation for the model are adopted in the UML. The definition of the use case is almost same as Jacobson's. According to the specification of the UML [2], each use case specifies some behaviors that a subject (target system) can perform in collaboration with one or more actors.

In use case models, actors may be various stakeholders; not only (narrowly defined) users, but also maintainers, system operators, outside systems, and so on. Fig.1. shows a sample of use case diagram. In this diagram, "Customer", "Administrator", and "Bank" are actors. "Customer" is a type of ordinary user, "Administrator" and "Bank" are more general users in broader meaning.

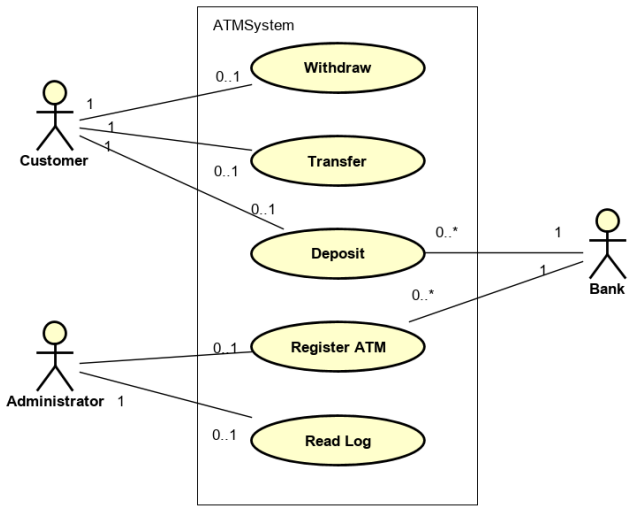

Fig. 1. Sample use case diagram (from [2])

As the definitions and this sample shows, actors may be various types of stakeholders. In this sense, the use case and use case diagram of UML handle various types of stakeholders as actors. However, these actors must interact with the system. A stakeholder who does not directly interact with the system cannot be an actor.

Each use case is a grouped functions that the system (the subject in use case diagrams) provides. Therefore, use cases and use case diagrams are suitable to analyze and specify functional requirements. On the other hand, non-functional requirements are not described directly by use cases. We need other techniques to describe those requirements. However, use cases help us to analyze non-functional requirements, especially quality requirements along with the use case scenarios.

III. Stakeholders in Quality in Use Model

In SQuaRE, various types of stakeholders are taken into account; users, developers, regulators, and society. In these

stakeholders, users are mostly related to the quality in use model. In ISO/IEC 25010:2011, the following types of users are considered [3].

- Primary user: person who interacts with the system to achieve the primary goals.
- Secondary user: person who provides support, for example
  a) content provider, system manager/administrator, security manager;
  b) maintainer, analyzer, porter, installer.
- Indirect user: person who receives output, but does not interact with the system.

These users' needs have to be considered in requirements definitions and specifications. For these processes, use cases and use case diagrams can be utilized, as described in the previous section.

Primary users can be obviously actors of use cases. Their needs can be analyzed along with the use cases related to corresponding actors. Even though functions for secondary users sometimes tend to be left out of essential services the system provides that are easily considered as use cases, secondary users can also be actors, since they interact directly with the system. The notion of use cases does not exclude services for these secondary users, and these services should be considered using the notion of use cases. We don't have to enlarge the notion; we utilize this for the secondary users. However, indirect users cannot be actors, because they do not interact with the system. When we consider quality in use related to indirect users, the concept of use case is difficult to be used.

IV. HIDDEN-USE CASE

To tackle the issue described in the previous section, we are trying to enlarge the concept of the use case and extending the use case diagram.

We introduce a new concept of "hidden-use case." A hidden-use case describes an effect that the subject (the system) provides to indirect users of the subject. The hidden-use cases may include behaviors of the system and send results of the behaviors unilaterally to the indirect users. They do not interact with any users; if they provide any effect to indirect users, the indirect users do not respond to the subject.

Hidden-use cases cannot exist without use cases and actors. They appear in consequence of the execution of ordinary use cases. For example, when the direct user of an AI assistant service execute the use case "ask schedule" of the service, a person around the direct user may hear sound from the service. Here, "hear sound" is a hidden-use case of this AI assistant service. The person around does not interact with the service, but the person may hear sound. And this "hear sound" hidden-use case never exists without the use case "ask schedule" trigged by the direct user.

Hidden-use cases may be categorized in two types;

- Type 1: its effect appears only from the system. In "hear sound" hidden-use case, this effect may be produced directly in the consequence of the use case "ask schedule." The actor "Owner" triggers the use case, but the actor never works over the produced effect of the use case, before this effect reaches to the "person around."
- Type 2: its effect appears from the behavior of the actor who received the output of the corresponding use case. For example, if the use case "ask schedule" is executed and the actor can meet the schedule because of the exact information produced by the use case "ask schedule, " an indirect user "Supervisor" of the actor may meet a team schedule. However, this result cannot be obtained only with the result of the original use case; if the actor does not meet its schedule even with the exact information produced by the use case, the supervisor may not meet a team schedule. The effect depends on the response of the actors to the original use case.

This categorization is only from the one viewpoint. There may be other various categorization. For example;

- A hidden-use case that is related to the original purpose of the original use case. "meet schedule" may be one of this type.
- A hidden-use case that is as a side effect of the original use case. "hear sound" may be one of this type.
- A hidden-use case that is produced as the result of the wrong usage of the original use case.
- A hidden-use case that may be a misuse case [4].

The above categorization is the only example. We have to consider more explicit and accurate viewpoint to categorize hidden-use cases.

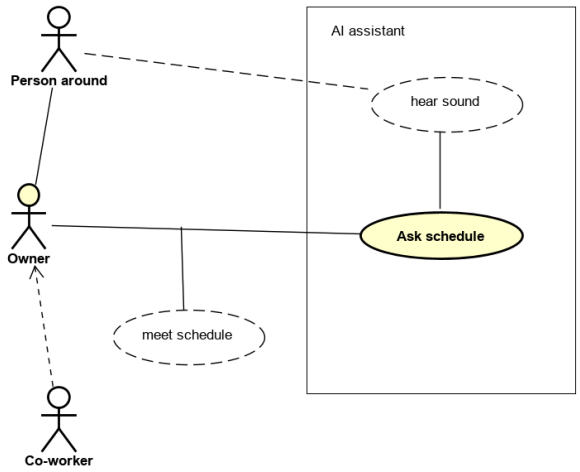

Fig. 2. Enhanced use case diagram with the notion of "hidden-use case"

Now we are trying to define the notation of the hidden-use cases and to enhance the use case diagram. Fig. 2 shows an example of the enhanced use case diagram by our first draft notation. A dashed ellipse is a hidden-use case. Indirect users are shown using the same symbol of actors. A hidden-use case is connected to indirect users with a dashed line. A type 1 hidden-use case is connected to the original use case with a line and is placed in the subject. A type 2 hidden-use case is connected to an association between the original use case and an actor. The indirect user of this hidden-use case depends on the actor of the original use case; therefore, the indirect user is connected to the actor with the dependency association.

This notion of the hidden-use case and the enhancement of the use case diagram with the hidden-use case may make the

usage of the system clear. It is expected that we can analyze and specify qualities in use, especially those related to indirect users.

Also, we expect that this hidden-use case concept will help to re-consider various qualities in use and to re-organize the model of qualities in use.

V. Conclusion

In this short position paper, we introduce a new unique notion of hidden-use case and an enhancement of use case diagram with this notion. This is just a first draft; we have to consider more deeply and refine the idea.

In the workshop, we would like to discuss the issues related to quality in use based on our idea described in this paper.

References

[1] I. Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley Professional, 1992

[2] Object Management Group, OMG Unified Modeling Language (OMG UML) Version 2.5.1, 2017

[3] International Organization for Standardization, ISO/IEC 25010:2011 SQuaRE -- System and Software Quality Models, 2011

[4] G. Sindre and A. L. Opdahl, "Eliciting Secutiry Requirements by Misuse Cases", Proc. TOOLS Pacific 2000, pp 120-131, 20-23 Nov 2000.