Mutual Requirements Evolution by Combining Different Information Systems

Haruhiko Kaiya
Kanagawa University
Hiratsuka, Japan
kaiya@acm.org

Kaito Nagano
Kanagawa University
Hiratsuka, Japan

Ryoya Muto
Kanagawa University
Hiratsuka, Japan

Mizuki Yoshida
Kanagawa University
Hiratsuka, Japan

ABSTRACT
We propose a method of eliciting requirements for several different systems together. We focus on systems used by one user at the same time become such systems inherently give influences on with other. We expect such influences help a requirements analyst to be aware of unknown requirements of the user. Any modeling notations are used to explore the combination among systems causing such influences because the differences among the notations give diverse viewpoints to the analyst. To specify such mutual influences, we introduce semantic tags represented by stereo types. We also introduce other semantic tags so that the analyst can judge whether the combination brings advantages to the user. We apply our method to an example and we confirm the method works.

CCS CONCEPTS
- Software and its engineering → Requirements analysis;
Unified Modeling Language (UML).

KEYWORDS
requirements elicitation, creativity technique, systems combination

1 INTRODUCTION
When an information system is developed or revised, requirements are traditionally elicited from its stakeholders and the system is expected to satisfy the requirements. Although the requirements come from the stakeholders, the stakeholders do not know their own requirements completely in the first place. They usually become aware of missing requirements while the system is introduced into their activity. To overcome this problem, several techniques for discovering unknowns have been proposed. For example, interviews, observation, workshops and prototypes are typical examples of such techniques [15]. In addition, there already exist several creativity techniques [1, 12]. However, existing techniques are too abstract to stimulate new ideas and to find unaware requirements because information resources and concrete clues are insufficient. New technique is thus required so that the business and/or life activities supported by systems become better than ever.

We then focus on the fact that an actor such as human or a machine usually participates in several different activities at the same time. Information systems are usually used in each activity today. Therefore, an actor uses several different information systems together at the same time. We regard mutual effects on a system by another can become concrete clues to stimulate new ideas and to discover new requirements. On the basis of this idea, we have studied requirements elicitation techniques [7, 9]. However, the method was not well organized in our previous works.

In this paper, we propose and organize a method for eliciting requirements of several different systems together. We use models of each system in this method. Any kinds of modeling notations may be used. Following two notions are keys in this method: combination and evolution. When a new model element is discovered in or imported into a system model due to or from models of other systems, we regard the systems are combined. When such combination brings some advantages to stakeholders of each system, we regard the systems are evolved. Our method facilitates such combination, and helps analysts to examine whether the combination brings evolution. Meta-modeling and semantic tags represented by stereo types are used in our method as mentioned in the next section.

The rest of this paper is organized as follows. In the next section, we explain our method and related concepts. We then discuss the method according to its application in section 3. Section 4 briefly introduces related works, and finally we summarize our current results and show our future issues.

2 METHOD
2.1 Overview
We explain our mutual requirements evolution method by using a simple example mentioned below. In this method, requirements
We focus on two different information systems: EMAG and ESHOP.

2.2 Motivating Example

We focus on two different information systems: EMAG and ESHOP. First one called "EMAG" is a service of providing electronic magazines via the Internet on the basis of the subscription business model. A lot of topics such as clothes, cosmetics, shoes and travels are introduced in several magazines. Second one called "ESHOP" is a booking site like booking.com or expedia.com. Initially, these two systems are not related to each other.

2.3 Conceptual Meta-Model

Each modeling notation emphasizes specific characteristics of a system. For example, an activity diagram emphasizes the sequences of activities in the system. A class diagram emphasizes the static structure of the system. A use case diagram shows the list of functions of the system and their relationships. The first challenge of this method is to explore the joint points, where the entrance into the mutual evolution of systems. We assume comparing elements in different notations stimulates an analyst to discovery. For example, a magazine in EMAG introduces some beach resorts, hotels there and swimsuits. They are represented as the classes in a class diagram at the left side in Figure 1. On the other hand, ESHOP helps travelers to book a hotel in a beach resort and a seat of an aircraft to the resort. They are represented as the actions in an activity diagram at the right side in Figure 1. In this case, a concept "hotel" can be a joint point. EMAG may then introduce airlines as new topic, and ESHOP may recommend swimsuits as new action as shown in Figure 2. We may thus permit using several different notations together in the initial modeling activity.

To unify model elements in different notations, we define a conceptual meta-model. We also define a super class of each model element in a concrete modeling notation. For example, the conceptual meta-model contains "Actor", which is a shared super class of "Actor" in a use case diagram, "Partition" in an activity diagram and "Entity" in a data flow diagram. The meta-model also contains "Action", "Object", "Goal" and so on so that it can deal with most modeling languages.

2.4 Guidelines for finding Joint Points

The same or similar concepts in different models become joint points. Basically, model elements that are the sub-type of an element in a conceptual model become the candidates of joint points. For example, a use case in a use case diagram and a process in a data flow diagram can be joint points.

Because the joint points are just the entrance to combine several models, they do not have to be the same or similar exactly. Practically, an analyst may decide several model elements are joint points by referring the label of each element. For example in Figure 2, we just refer the keyword "hotel", and decide two elements are joint points. As shown in the figure, stereo types such as JOIN0, JOIN1 or JOIN2 are attached to joint points.

We may use a sub-structure of a model as a joint point. For example, a has-a sub-structure in a class diagram, a sub-sequence of actions in an activity diagram or state transitions in a state machine diagram can be a joint point.

2.5 Guidelines for Combination

When new model elements are imported or discovered in each system, we regard the systems are combined. Such combination...
sometimes makes unnecessary or duplicated elements in each system to be removed. A joint point itself and its neighboring elements are candidates for combined elements. For example in Figure 1, “hotels” in EMAG and “book a hotel” are joint points. A sibling of “hotels” in EMAG, i.e. “swimsuit”, is then focused, and it triggers off the discovery of “choose a swimsuit” as shown in Figure 2. The next action of “book a hotel” in ESHOP, i.e. “book a seat”, is also focused, and it triggers off the discovery of “airlines” in EMAG as shown in Figure 2.

To specify combination, we introduce the following four stereo types. Each of them is attached to a combined element so that the role of the element is clarified. Each stereo type also has number postfix in the same way as JOIN.

- TRI (trigger): When TRI is attached to an element of a system, the element triggers off the discovery of new model elements in other systems. For example in Figure 2, a class “swimsuit” in EMAG triggers off the discovery of an action “choose a swimsuit” in ESHOP. An action “book a seat” in ESHOP also triggers off the discovery of a class “airlines” in EMAG.

- DIS (discovered): When DIS is attached to an element of a system, the element is discovered due to the trigger by another element in another system. We can find its examples in Figure 2. The same postfix number is given to corresponding pairs of TRI and DIS as shown in the figure.

- EXP (export): When EXP is attached to an element of a system, the element is exported to other systems and utilized for their evolution.

- IMP (import): When IMP is attached to an element of a system, the element is imported from another system. In the same way of TRI and DIS pair, the same postfix number is given to corresponding pairs of EXP and IMP.

2.6 Advantage Analysis to Examine Evolution

To analyze the advantage of combination, we also introduce two additional stereo types below.

- TBD (to be developed): Before combining the systems, each system has its own additional requirements that have not been implemented yet. TBD is attached to elements in models corresponding to such requirements. We expect each TBD element does not have to be implemented due to the import of some elements from others.

- REM (removed): When some model elements are imported or discovered, existing model elements are sometimes replaced to such new elements. The existing elements also become unnecessary. REM is attached to such replaced or unnecessary elements so that an analyst clarifies that they may be removed. We expect the number of such REM elements increases as much as possible. The stereo type REM of such an element has the same postfix number as one of the IMP or DIS element causing the removal.

We regard the combination of the systems has advantages to the systems in the following cases.

- Human actors can perform actions less, more easily or more comfortably than ever.

- Own additional requirements corresponding to TBD elements do not have to be implemented as much as possible.

- A system contributes to others as much as possible by exporting its elements or triggering off discovery in others. Especially, each system mutually contributes to each other.

- The contribution mentioned above makes the existing system save time and hardware in the systems.

- The combination brings new values to someone including its original user.

If the combination of systems shows tendencies above, we regard the systems are evolved.

3 DISCUSSION

We applied our method to combining a music player like Spotify and a running support application like Runkeeper. Through our experiences of applying our method, we confirmed our method worked. However, it largely depends on the insight of analysts. It is not the problem of our requirements evolution method because systems are never evolved by themselves even with a state of the art AI techniques. Although such techniques will enumerate the possibilities of combination, human always has to judge whether the combination can bring evolution or not.

Patterns of systems’ evolution also help us to find the combination possibilities. Examples of such patterns are as follows.

- Neighboring elements of joint points can be candidates of new elements added to other systems as shown in Figure 2.

- Neighboring elements of joint points can be candidates of new elements instead of existing ones in other systems.

- A TBD (to-be-developed) element in a system can be developed in another system, and the element is also used in other cases.

Collecting such patterns encourages the usage of our method.

4 RELATED WORK

As cited and discussed in our previous paper [7], there are few approaches for eliciting requirements for several different systems together. A little bit similar concept is co-evolution. However, most papers [4, 17] focus on co-evolution of several different aspects such as requirements, architecture, design and/or organization related in a system. In a book chapter [19], the co-evolution of several open source systems is mentioned. However, they do not focus on the structural aspect of the systems. They rather focus on their organizational aspects.

Although researches of Systems of systems (SoS) [11] seemed to be related to our research in this paper, they are a little bit different from ours. SoS researches do not focus on discovery of new or unknown purposes of systems. Our research mainly focuses on such discovery. In this sense, our research is more related to creativity techniques [1, 12] and techniques for finding missing requirements [18, 21].

In the context of researches of digital transformation, interaction or combination of systems or businesses are focused. In a paper [3], a platform business model is focused as a part of digital transformation. In the model, business is expected to create value by facilitating direct interactions between two or more distinct types of customers. Our idea also focuses on facilitating different types of elements. However, we focus not only on customers but also any elements in systems and business activities.
5 CONCLUSION

In this paper, we proposed and exemplified a method of eliciting requirements of several systems together. Because an actor such as human or devices usually participates several different activities at the same time, analyzing mutual interaction among systems is natural. In addition, such analysis helps analysts to discover unaware requirements for each system, and such discovery will improve each system and its supported activities. In our method, analysts may use any kinds of modeling languages so that they can focus on different aspects of systems. To unify models written in different modeling languages, we proposed a conceptual meta-model. When a new model element is discovered or imported into a system due to or from other systems, we regard the systems are combined. When such combination brings advantages to a system, we regard the system is evolved. To facilitate combination of systems, we introduced several semantic tags represented in stereotype types such as JOIN (joint point), TRI (trigger), DIS (discovered), EXP (export) and IMP (import) each of which is attached to a model element. To clarify the evolution, we also introduced stereotype types: TBD (to be developed) and REM (removed). Our method including the conceptual meta-model and stereotype types helps analysts to perform such combination and evolution.

We finally summarize our future issues. As mention in section 3, the insight of analysts is required to perform our method. Patterns of system combination and evolution will mitigate the load of the analysts. We thus gather and organize such patterns so that the analysts can use them. Some of modeling activities using our meta-model and stereotype types are tedious but important. For example, post fix numbers in the stereotype types should be carefully matched. We thus have to develop a CASE tool for this method. Our experiences of developing UML related tool [8] help us to develop such tool. Currently, main criteria of evaluating advantages are very classic: decreasing the cost of human and systems, i.e. efficiency. However, we have to focus on other criteria. The candidates are sustainability and expandability [6]. We expect activities supported by systems may continue as long as possible. We also expect people joyfully participate in the activities more than ever. However, too much efficiency sometimes becomes an obstacle to such sustainability and expandability. We want to take such criteria into account.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number 18K11249.

REFERENCES


