Goal-Satisfaction Verification to Combination of Use Case Components

Saeko Matsuura[1], Shinpei Ogata[2] and Yoshitaka Aoki[3]
[1]*Graduate School of Engineering and Science, Shibaura Institute of Technology, Saitama, Japan*
[2]*Graduate School of Science and Technology, Shinshu University, Nagano, Japan*
[3]*Yoshitaka Aoki, Nihon Unisys Ltd. Tokyo, Japan*

Keywords: Requirements Analysis, Use Case, Non-functional Requirements, Model Checking.

Abstract: Functional requirements of a system can be specified by fundamental use cases that satisfy "the effective and useful scenarios in the system usage" so as to meet "the goal of the system". Ambiguous non-functional requirements against the system goal often cause uncertainty of use cases and scenarios at the early stage of software development. In this paper, from the viewpoint of non-functional requirements that are included in the goal, we discuss how to check satisfaction of the goal due to the combination of functional requirements during requirements analysis using an example.

1 INTRODUCTION

It is well known that requirement analysis is a key to success to develop high-quality system efficiently. Requirements specification has to be specified not only functional requirements but also non-functional requirements. Non-functional requirements include a system goal, external interfaces with the user, hardware, software, and communications. Moreover, user characteristics that are general characteristics of the intended users of the product including educational level, experience, and technical expertise affect not only the system usage but also the service quality.

On the other hand, functional requirements can be modelled as essential use cases by a semi-formal and widely used language such as UML (Unified Modelling Language) (OMG UML) in Model Driven Development (MDD). System usage scenarios can be defined by the combination of these use cases. However, the abovementioned architectural and external factors in non-functional requirements strongly affect the combination. Moreover, the presence or absence of the necessity of the use case may occur. Although such uncertainty of requirements is unavoidable at the early stage of system development, it is important to manage traceability of requirements connected with non-functional requirements in a system more formally.

To manage such traceability of requirements, this paper proposes the following modelling method and process at the early stage of system development.

- Assume that use cases are fundamental components of the requirements defined formally by UML models. In addition, a system usage scenario is constructed by the combination of the use case components under the conditions which may solve some uncertainty. Then, all scenarios are verified whether they satisfy the goal.
- An iterative cycle of analysis and verification in which the requirements specification of a system is defined incrementally.

The rest of the paper is organized as follows. Section 2 discusses the modelling method and process based on the problem of requirements specification verification. Section 3 explains our method and process using a case study of a reservation system of conference rooms. Section 4 discusses the related work.

2 REQUIREMENTS SPECIFICATION VERIFICATION PROBLEM

2.1 Requirements Specification in Model Driven Development

Model Driven Development (MDD) (S. J. Mellor,

Matsuura, S., Ogata, S. and Aoki, Y.
Goal-Satisfaction Verification to Combination of Use Case Components.
DOI: 10.5220/0006785003430350
In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2018)*, pages 343-350
ISBN: 978-989-758-300-1

OMG MDD) promoted by OMG is a promising approach to develop high-quality software products efficiently because it enables code generation and the translation rules give high traceability throughout the development. Changeability of platforms can be solved by separating concerns about platforms as Platform Independent Model (PIM) and Platform Specific Model (PSM) using UML. As mentioned before, aside from such platform view, there are various kinds of features that are different in level of abstraction within the requirements for a system and have a mutual relationship.

The initial model of a system depends on several features in non-functional requirements, because these features may restrict the service or expand the contents of the service. Therefore, the quality of the generated codes is affected by these source models that was separated various kinds of concerns within the requirements. Especially, initial requirements specification needs elaborating under consideration of these features step by step as discussed in the Twin Peaks Model (Nuseibeh).

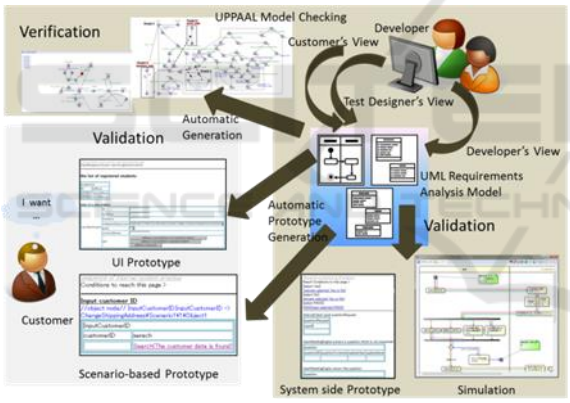

Figure 1: V&V of Requirements Analysis Model.

We proposed a method of model-driven requirements analysis using UML. Figure 1 shows a Validation and Verification (V & V) requirements analysis process in which developers define and refine the requirements analysis model combining with the following verification process.

- Automatically generated three types of prototypes help customers and developers validate their requirements and the defined models by directly operating the HTML-based prototypes(Ogata).
- Simulating UML models by preparing test data from test designer's view helps developers find the inconsistency or some omissions or oversights for the defined requirements analysis model(Shikimi).
- Verifying some specification for UML requirements analysis models with the model checking tool UPPAAL helps developers detect defects of the defined UML models(Aoki). In this case, UML requirements analysis model and a model defined a feature that should be satisfied are translated into a UPPAAL model, and UPPAAL model checking tool automatically verifies whether the feature is satisfied the base model.

2.2 Use Case Components

UML requirements analysis model (RA Model) that is a target of validation and verification is defined as follows. RA Model is created based on a use case analysis, which is known to be an effective method for clarifying functional requirements.

We specify a use case from the following four viewpoints to answer the associated questions:

- Application requirements: what kinds of input data and conditions are required to execute a use case as expected?
- Application requirements observation: what kinds of conditions should be required when not executing the use case? Moreover, how should the system treat these exceptional cases?
- Use case conditions: what kinds of behaviours are required to execute the use case?
- Output: What kinds of data outputs result from these behaviours?

Both process flow and entity data, which are required to execute the target application requirements, are defined with UML activity diagrams and a class diagram based on the aforementioned four viewpoints and questions.

Figure 2 shows an outline of RA model. The relation between use cases in the specified use case diagram is expresses by an activity diagram includes some sub-activity nodes corresponding to use cases. Each use case is defined by an activity diagram. Activity diagrams specify not only normal and exceptional action flows, but also data flows that are related to these actions. Actions are defined by action nodes and data is defined by object nodes that are classified as members of a class that is defined in a class diagram. Accordingly, these two kinds of diagrams enable us to specify application process

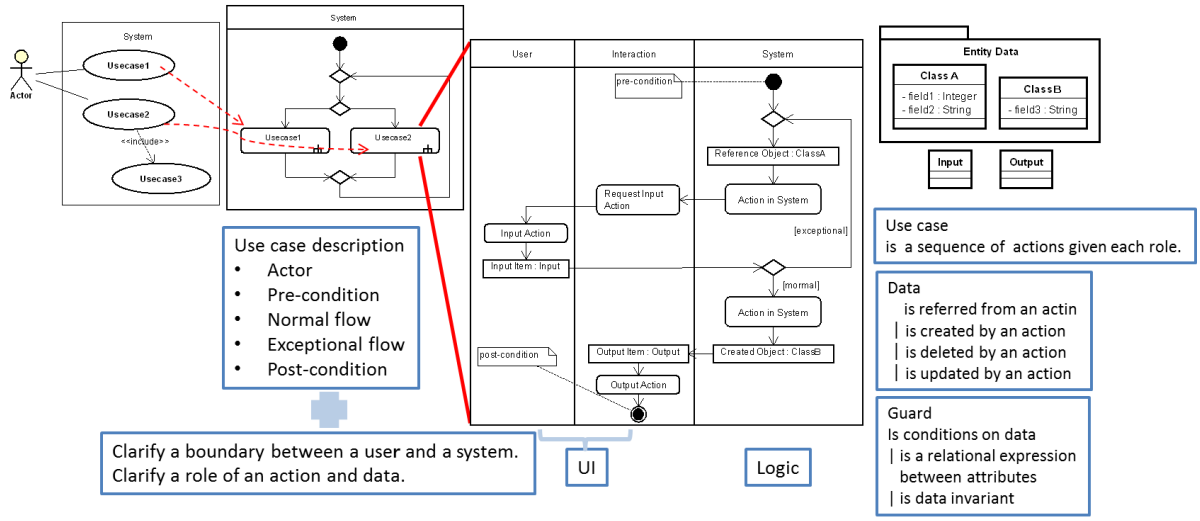

Figure 2: UML Requirements Analysis Model (RA Model).

flows in connection with the data. The interaction between a user and a system especially includes requisite flows and data on user input, conditions, and output to execute a use case correctly.

The feature of our method is an activity diagram that has three types of partitions: user, interaction, and system. These partitions enable ready identification of the following activities: user input, interaction between a user and system caused by the conditions for executing a use case, and the resulting output. Object nodes in the user, interaction, and system partitions represent input data, output data, and entity data, respectively.

The actions and the data in the user and interaction partition allow us to automatically generate a prototype system that shows the defined application process flow with the data.

Using the relation between the actions and the data that have own role specified by each partition enables us to systematically prepare test data for the application process flows.

Defining the state transition of data as conditions which must be met from data perspective against the application process flows allows us to verify whether the application process flow meets the condition. The conditions can be verified by combining with the application process flows using a model checking tool (Aoki.) This is one of the advantages of our method to use activity diagrams and class diagrams to specify some non-functional requirements such as security requirements.

Such use cases written in UML can have various verifiable features, so use cases that are fundamental components of the RA models are called *use case components*. The RA model is defined using a modelling tool astah*.

The point of this method is to define properties to be satisfied in the combination with use case components by a state machine diagram or a decision table. We have verified the following properties for the application process flow by a combination of some use case components.

- The related data with the use cases satisfy the universal rule concerning CRUD (Create, Read, Update, and Delete.)(Aoki)
- The security rules related to access control obtained by security requirement analysis (Aoki).

If we can define the behaviour of the combination of use case components so as to bring about the expected results, it can also contribute to the improvement of the quality of the code resulted by MDD. However, there are various non-functional requirements besides the functional requirements, it is not clear when and how to define and verify the requirement until extracting necessary and sufficient use case components.

2.3 Definition and Verification of Non-functional Requirements

Based on Chapter 4 “Considerations for producing a good SRS (Software Requirements Specifications)” and Chapter 5 “the parts on an SRS” in IEEEstd.8300-1998 (IEEE), we focus on the following parts about non-functional requirements and propose an iterative requirements analysis process as shown in Figure 3.

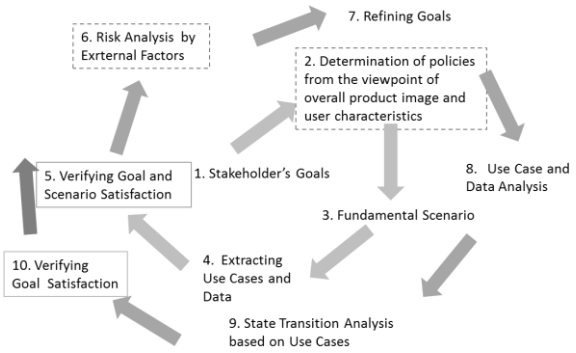

Figure 3: Iterative Requirements Analysis Process.

The process means that functional requirements of a system can be specified by fundamental use cases that satisfy "the effective and useful scenarios in the system usage" so as to meet "the goals of the system". We focus on the following internal and external factors of software.

1) Goals of the stakeholders
2) Overall product image based on the following two external factors.
   A) External interfaces with the user, hardware, software, and communications.
   B) User characteristics that are general characteristics of the intended users of the product including educational level, experience, and technical expertise which affect the system usage.
3) As software acts for a part of human activity in the real world, some entity objects in the software correspond to entity objects in the real world. The risk that such objects in the real world affect to the software should be analyzed and be appropriately coped with. We call such an object an external activity entity.

In this paper, we explain this process through a case study that analyzes a conference reservation system what RA models we define and verify these items as follows steps shown in Figure 3.

We begin at determining stakeholders of the target system and the goals of each stakeholder. Each original goal may unclear and there are sometimes conflicts between different goals. So we assume that these goals will be added and be detailed by checking goal satisfaction for fundamental use case components during the requirements analysis process. The above-mentioned factors may seriously affect determining use case components that have required functions.

We begin at specifying fundamental use cases that satisfy "the effective and useful scenarios in the system usage." After defining each use case component as mentioned Section B, we verify that they can meet "the goals of the system." As a result, the goals will be improved. A task enclosed by broken lines in Figure 2 shows the above-mentioned points.

3 CASE STUDY

3.1 Example Requirements

A case is a reservation system of conference rooms in our university. The users of the conference room are faculty members of the university and the total number of conference rooms is about 13. Currently, they submit a paper-based application form. In the administrative section of each school building, they are separately managed the reservation control register by Excel data. Recently, there are many cases where video conferences are held among three school buildings, and integrated management of reservations is desired.

3.2 System Goal and Goal Analysis

The goal of systematization is to abolish paper-based procedures and to improve the convenience of reserving procedures for conference rooms and to facilitate information management. According to the concern of each stakeholder of the system, the goal was analysed as shown in Table 1.

Table 1: The Goal of Each Stakeholder.

Stakeholder	Role	Goal		
Faculty staff	applicant	The user can make reservations easily.	The device can be used easily.	I
			If the user makes a reservation, he/she can get the right to use without other procedures.	II
			The user can operate the system easily.	III
		The reservation is guaranteed until the reserved date.		IV
Student affairs division staff	administrator	The location of responsibility is clear when a trouble occurs in the conference room/equipment due to the use of the conference room.		V
		There is no inappropriate use purpose or an occupied state so that the applicant can fairly use the conference room as a result.		VI
e-Lab staff	administrator of video teleconference system	For preparation and troubleshooting, he/she can know the schedule and location of the reservation that the video conference system is to be used.		VII

3.3 Fundamental Scenario Analysis

Next, based on the viewpoint 2) of the non-functional requirements described in Section II-C), we decide a policy for satisfying the goal as shown in Table 2. As shown by colouring, there are items related to multiple goals.

We consider the policy from the viewpoint of user characteristics and the system architecture. The goal I is the problem of a user environment. In this case, it is no problem to assume use from a smartphone or PC because the system will be used in the university. Reservation of the conference room may be done by accumulating reservation information and allowing it to be used by users of multiple roles. The system can be implemented as an ordinary Web application. However, under consideration of the goal III, hardware may be changed to make it easy to input some data. In addition, the assumed user has no trouble with the use of these devices.

Table 2: Policy for Satisfying the Goals.

	Policy
I	Assume the device to be used. In this case, no special consideration is required for the user characteristics and the usage situation.
II	Consider how to solve II under the constraint of VI.
III	Depending on the required input information, the hardware configuration may change.
IV	How can we guarantee the approved reservation? Refine the meaning of the guarantee clear.
V	Determine reservation management data that is required to satisfy this goal.
VI	How can we confirm that the constraints are satisfied?
VII	It is sufficient that the staff can only to browse the necessary reservation information to be known.

3.4 Impact Analysis of External Factors

The goal IV in Table 2 is an item to be considered after determining the reservation management data for the goal V. The goal VII is an item to consider when considering the V. To satisfy the goals II and VI, the rules to approve a reservation need to be decided according to them.

In order to achieve the goal V, the fundamental scenario for acquiring data that the reservation has been approved and the system should have is specified as follows.

> A faculty-staff reserves a conference room that he/she wants to use, and use it on the date. Regarding reservation, official approval by the student affairs division staff is not required, but we will check if there is inappropriate use purpose or an occupied state. If a user cannot make a reservation as he/she wishes, he/she can make another appointment. If a student affairs division staff judges it as inappropriate, he/she notifies it to the user by telephone etc. separately and encourages correction. Moreover, a student affairs division staff can make reservations for conference rooms necessary for university work as a priority.

Here, the underlined part is the rules to approve a reservation in accordance with the goals II and VI in Table 1. If there are many inappropriate circumstances, it may be necessary that the system checks them strictly by the way such as limiting use purpose and use time etc.

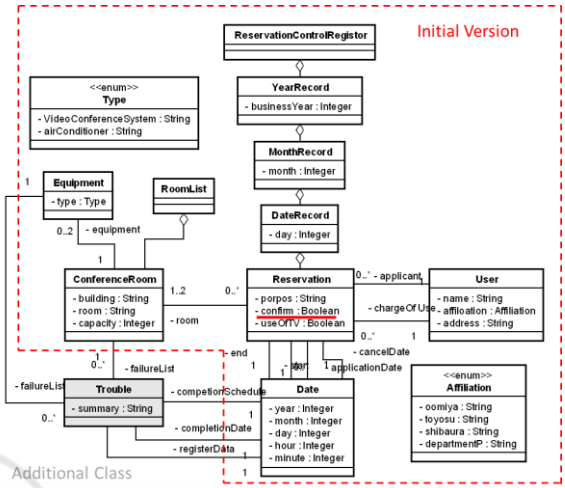

Figure 4: Final Class Diagram.

3.5 Data and Behavioural Analysis of Use Case

Based on the above scenario, "reservation", "reservation condition" and "inappropriate use" are specified so as to satisfy the scenario, and the use cases for realizing the scenario is extracted.

A reservation is defined by a class diagram as shown a part enclosed by dotted lines in Figure 4, an applicant applies for "who (person in charge of use) uses which conference room from when (start) to what time (end)." Looking the contents of the reservation, each data can be selected from the data already included in the system except user required input, so there is no effect on the goal I.

In order for a student affairs division staff to check the inappropriateness, it is necessary for him/her to input and to keep the data on whether the reservation was confirmed or not. A use case for checking the inappropriateness named *Confirm a reservation* needs to be added to a use case diagram in addition to the essential use case extracted from the fundamental scenario considering functions for CRUD (Create/Read/Update/Delete) of a reservation data. Figure 5 except five colored use cases is a use case diagram in this stage. Moreover, to leave the data that a reservation is checked or not, an attribute confirm is added to *Reservation* class. The attribute underlined in red in Figure 4 is the added data.

For Update, we assume change of time and conference room from the attributes of *Reservation* class. For Read, to satisfy the goal VII in Table 1, we define a use case that checks the reservation schedule in which the video conference system will be used.

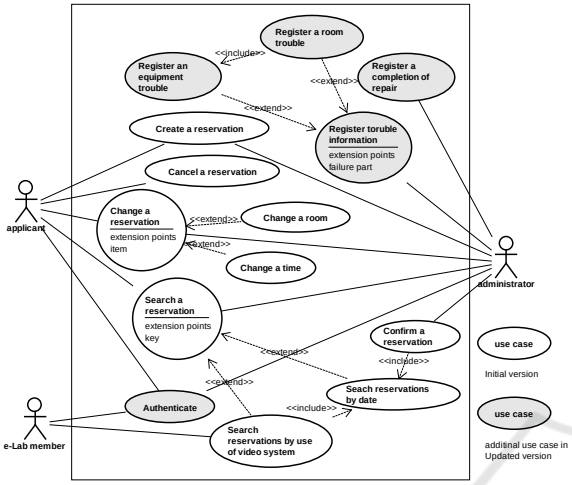

Figure 5: Final Use Case Diagram.

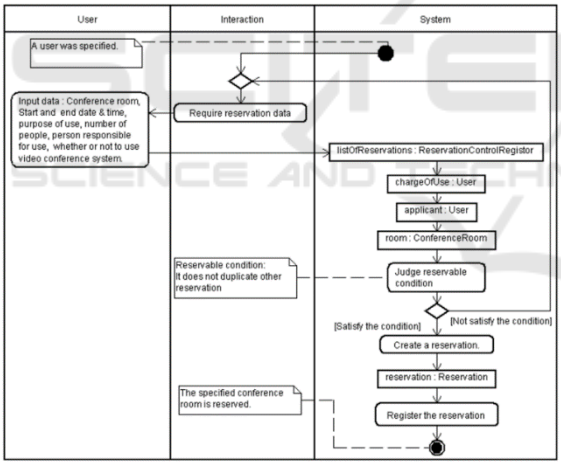

Figure 6: Use Case Description of Create a reservation.

Use case description of "Create a reservation" is defined by an activity diagram as shown in Figure 6. Pre-conditions, post-conditions and guards are defined by using a note node. From the viewpoint of data flow, we can confirm that each action can be realized with the data defined in the class diagram.

We will confirm that the combination of use case components meets the fundamental scenario described in Section III-D). We consider the goal's satisfaction with the goals of Table 1 using the use case components that have been decided so far. Table 3 shows the result.

Table 3: Consideration of Goal's Satisfaction.

role	Goal			Goal-satisfaction
applicant	The user can make reservations easily.	The device can be used easily.	I	○
		If the user makes a	II	○
		The user can operate the system easily.	III	○As required input and output data is defined, there is no hardware change. To satisfy the goal finally, it is necessary to elaborate operability at the implementation phase.
	The reservation is guaranteed until the reserved date.		IV	Any user can make a reservation if conditions are satisfied.
administrator	The location of responsibility is clear when a trouble occurs in the conference room/equipment due to the use of the conference room.		V	○Since the person in charge of the reservation and the contact address are recorded, it is possible to contact him/her if any trouble happens.
	There is no inappropriate use purpose or an occupied state so that the applicant can fairly use the conference room as a result.		VI	○The applicant who received the notification may cancel or update the reservation.
administrator of video teleconference system	For preparation and troubleshooting, he/she can know the schedule and location of the reservation that the video conference system is to be used.		VII	○For example, a user can search for reservations in which a video conference system is used in this week.

A reservation condition is "for all reservations in the reservation register, there exists no reservation that has attribute value being overlapped between the start and end times in the same conference room." This condition is satisfied because it can be judged by the data being accessed in the use case description.

At this stage, if this condition is satisfied, it is possible to make a reservation, and we can see that a reservation will be approved if there is no problem with the check by the student affairs division staff. In this sense, if the applicant does not violate the condition, the reservation is guaranteed until the reserved date. However, it is the final goal that the reservation is guaranteed even outside the applicant's responsibility.

3.6 Improvement Requirements Specification

For the goal IV, we will consider what is a probable factor outside the applicant's responsibility from the viewpoint 3) of the non-functional requirements described in Section 2.3).

From the class diagram in Figure 4, we analyse the risks of external activity entities that affect the services of the reservation system. Being able to respond to risks is a desirable result for users of the system, which leads to "the reservation is guaranteed even outside the applicant's responsibility" which is the goal of the applicant.

The affected external activity entity is "conference room" and its "equipment" and "user". The meaning of reservable is as follows.

- The applicant can use the room on the reserved date in actually.
- He/she can the required equipment on the reserved date in actually.

Although the goal V in Table 3 seems to be satisfied at first glance, it can occur that another

person different from the actual user is forced to have the responsibility when the applicant is impersonated by another user.

Therefore the goals should be redefined more precisely. Table 4 shows the added and detailed goals by risk analysis.

As a result, four new use cases are extracted and a class *Trouble* is added to the class diagram as shown in Figure 4 and 5. Since the responsible user and the applicant are necessary for the application, it is possible to correctly grasp the location of responsibility by certifying the applicant.

Table 4: Added and Detailed Stakeholder's Goals.

Stakeholder	Role	Goal	
Faculty staff	applicant	The reservation is guaranteed until the reserved date.	A user cannot actually reserve a conference room although it is unavailable.
			When there are some troubles, a user can know the situation promptly.
Student affairs division staff	administrator	When a trouble occurs in the conference room or the equipment, it can deal with so that inconvenience does not occur to the user who reserves it.	
		By using the conference room, when a trouble occurs in the conference room or the equipment, the location of responsibility can be grasped correctly.	

3.7 Verification of Goal Satisfaction

The reservation condition is determined by the attribute related to "reservation" of the class diagram. Due to the change of addition of *Trouble* class, the condition of "there are no troubles in the reserved room" is added to the condition. This change is dealt with by correcting the condition of the action "judge reservable condition" in Figure 6 with the attribute of the class.

The satisfaction level of the goal so far could be confirmed by the existence of elements of the model. In order to verify the satisfaction of the added and detailed goals in Table 4, we should consider what kind of results will be obtained if more than one use cases are executed during the service. Therefore, from the use case components extracted so far, using a state machine diagram, we analyse the influence of the trouble of the conference room on the state of the conference room of a certain reservation as shown in Figure 6.

In Figure 7, every transition event is an activation of a use case in the use case diagram. The subject of an event is the applicant or the administrator as shown in Figure 7. Therefore, the subject of the use case recognizes the state occurring as a result of his/her action but cannot recognize it if it is not the executor. Even in the reserved state, the information of availability of equipment is sometimes important, so it is always necessary to

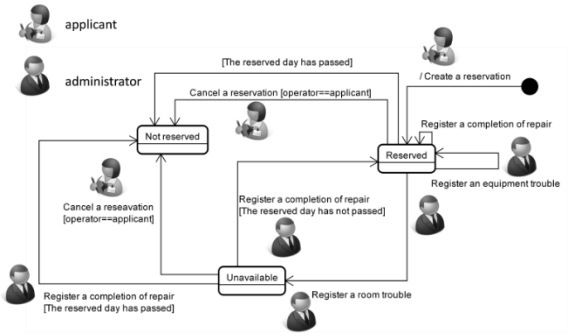

Figure 7: State of a conference room on the other day.

recognize the trouble by notice of conference room and equipment trouble.

When notifying the trouble from the system, add a notification action to the action flow in the corresponding use case description of "Register an equipment trouble" and "Register a room trouble" use cases regarding the breakdown of the conference room. Due to this refinement, the refined goal of the applicant in Table 4 is satisfied anyway, but it is not sufficient for the administrator's added goal. Namely, if it is not possible to use the room in the end, or if the date of use is near, it is unsatisfactory to the applicant, and the applicant himself/herself needs to reserve again.

4 DISCUSSION AND CONCLUSIONS

Garlan (David Garlan) says, "It is important to treat uncertainty as a primary concern of software development." During acquiring requirements for a desirable system, it is a fact that there are many such uncertain factors caused by non-functional requirements, especially items mentioned in Section II-C).

There are some studies on requirements acquisition such as Goal-oriented analysis (Chung, L.). However, it is difficult to treat uncertainty caused by non-functional requirements about external activity entity such as hardware, users and physical things as a primary concern.

In this paper, separating concerns for users and physical things such as a room or an equipment from the fundamental scenario lead systematically to finding new use cases that reduce the risks caused by them.

In this case, user characteristics give no influence to a system. However, they often affect the system architecture and kinds of use cases. Therefore, the iterative analysis is necessary while

goal satisfaction of combination of use case components is verified during the early stage of requirements analysis. Modelling use case components semi-formally related with goal satisfaction make it possible to module uncertain requirements with traceability to functions.

REFERENCES

Moore, R., Lopes, J., 1999. Paper templates. In *TEMPLATE'06, 1st International Conference on Template Production*. *SCITEPRESS*.

Smith, J., 1998. *The book, The publishing company. London, 2nd edition.*

OMG, "Unified Modeling Language", http://www.uml.org/

S. J. Mellor, K. Scott, A. Uhl, and D. Weise, MDA Distilled Principles of Model-Driven Architecture. *Addison-Wesley, 2004.*

OMG, "MDA Guide Version 1.0.1." Object Management Group, Tech. Rep., 2003. http://www.omg.org/mda/

Nuseibeh, B., Weaving the Software Development Process Between Requirements and Architectures, *IEEE Computer, 34(3), pp. 115-117, 2001*.

Shinpei Ogata, Saeko Matsuura,Evaluation of a Use-Case-Driven Requirements Analysis Tool Employing Web UI Prototype Generation, *WSEAS Transactions on Information Science and Applications, Issue 2, Volume 7, pp.273-282, February 2010.*

R. Shikim, S. Ogata and S. Matsuura, Test Case Generation by Simulating Requirements Analysis Model, *Proc. of 2012 IEEE 36th International Conference on Computer Software and Applications, pp 356-357, 2012.*

Y. Aoki, S. Ogata, H. Okuda and S. Matsuura, Data Lifecycle Verification Method for Requirements Specifications Using a Model Checking Technique, *Proc. of The Eighth International Conference on Software Engineering Advances (ICSEA2013), pp.194-200, 2013.*

Yoshitaka Aoki and Saeko Matsuura, Verifying Security Requirements using Model Checking Technique for UML-Based Requirements Specification, *Proc. of 1st International Workshop on Requirements Engineering and Testing, pp.18-25, 2014.*

IEEE Computer Society, *IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998 (1998).*

David Garlan. Software Engineering in an Uncertain World. In Proceedings of the FSE/SDP workshop on *Future of Software Engineering Research, pp. 125-128, ACM, New York, NY, USA, 2010*. ISBN 978-1-4503-0427-6.

Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J., Non-Functional Requirements in *Software Engineering, Academic Publishers, 1999.*