PAPER Special Section on Knowledge-Based Software Engineering

Proposal for Requirement Validation Criteria and Method Based on Actor Interaction

Noboru HATTORI[†,††a)], **Shuichiro YAMAMOTO**[†††], **Tsuneo AJISAKA**[††], ***Members***, ***and*** **Tsuyoshi KITANI**[†], ***Nonmember***

SUMMARY We propose requirement validation criteria and a method based on the interaction between actors in an information system. We focus on the cyclical transitions of one actor's situation against another and clarify observable stimuli and responses based on these transitions. Both actors' situations can be listed in a state transition table, which describes the observable stimuli or responses they send or receive. Examination of the interaction between both actors in the state transition tables enables us to detect missing or defective observable stimuli or responses. Typically, this method can be applied to the examination of the interaction between a resource managed by the information system and its user. As a case study, we analyzed 332 requirement defect reports of an actual system development project in Japan. We found that there were a certain amount of defects regarding missing or defective stimuli and responses, which can be detected using our proposed method if this method is used in the requirement definition phase. This means that we can reach a more complete requirement definition with our proposed method.

key words: *requirements analysis, requirements validation, completion criteria, actor relationship analysis*

1. Introduction

The requirement definition phase largely affects the success or failure of software development projects. Mogyorodi introduced a study by James Martin, which states that the root cause of 56 % of all bugs identified in projects are defects introduced in the requirement definition phase, and roughly half of these bugs were due to requirements that were completely omitted [1]. The amount of effort it takes to fix bugs that arise from requirements is even higher at 82 %.

A unified modeling language (UML) is often used to describe software requirements. Use-cases are used for capturing domain-level requirements, and the domain model, i.e., the domain layer of software objects, is also used. However, these notations mainly describe software functions and data separately and are insufficient for analyzing problems in the real world. Detailed analyses conducted in the requirement definition phase are often omitted in documents to maintain a level of abstraction. Svetinovic et al. found that the domain model produced from use-case models by different modelers, for the same systems, can differ from one another. One reason for this lies in the difficulty of discovering concepts in a problem domain [2]. As a result, gaps between requirements and software functions are often clarified after the development phase is finished. Jackson also claims that focusing on the problem is important, and problem frames help us do this, instead of drifting into inventing solutions [3].

We propose requirement validation criteria and a method based on the interaction between actors, i.e. concerned parties that depend on each other in an organizational context. Information systems as well as humans can be regarded as actors. We focus on cycles of stimuli and responses from one actor to another. Each actor has situations described in a state transition table, which lists observable stimuli or responses the actors send or receive. This method is expected to clarify stimuli and responses and related situations of actors in the real world, which is difficult to exhaustively define using UML and to validate the completeness of stimuli and responses between actors. This completeness means that all responses of the concerned party to all stimuli in all realizable classes of situations are defined. This is one of the elements included in completeness defined in the recommended practice for software requirements specifications (SRS) in IEEE Std.830-1998 [4].

There have been empirical studies in which a well-defined scope, or decomposing, sizing, and managing features are significantly related to the success of the project [5], [6]. Each feature is generally described in a sequence of stimuli-response pairs [4]. Therefore, the clarification of stimuli and responses at an early phase is important for the success of a system development project.

In Sect. 2, we describe the components for describing requirements that are applicable to functional requirements. In Sect. 3, we describe our proposed requirement validation criteria and method based on actor interaction. In Sect. 4, we explain our proposed method with an example of a coin parking service in Japan. In Sect. 5, we show an industrial case study and evaluate the effectiveness of our method. We analyzed requirement defect reports of a system development project in a Japanese company to see if these defects could be detected if our proposed method is used for the requirements review at the requirements definition phase. In Sect. 6, discussion related to our industrial case study in Sect. 5 is described. In Sect. 7, we discuss related work, and finally, in Sect. 8, we make our conclusions and explain our

Manuscript received July 4, 2009.
Manuscript revised October 19, 2009.

[†]The authors are with R&D Headquarters, NTT DATA CORPORATION, Tokyo, 135–8671 Japan.
[††]The authors are with the Graduate School of Systems Engineering, Wakayama University, Wakayama-shi, 640–8510 Japan.
[†††]The author is with Headquarter of Information and Communication Services Information and Communication Planning Office, Nagoya University, Nagoya-shi, 464–8601 Japan.
a) E-mail: hattorinb@nttdata.co.jp

DOI: 10.1587/transinf.E93.D.679

future work.

2. Requirements Elicitation Based on Model of Actor's Situation

In this section, we propose components for describing software requirement specifications. The functional requirements for software can be thought of as follows. "An actor, in a certain situation, instructs a stimulus, which is input into a specified process and corresponding output. Then it is sent as a response to the actor, which is in a certain situation.".

Figure 1 shows this structure. The requirement components are defined as follows.

(1) Actor
A concerned party who explicitly or implicitly requests to start a software function and receives responses from that software.
(2) Actor's pre-situation
An actor's situation when he/she/it sends the corresponding stimulus to that function.
(3) Observable Stimulus
An event that triggers the function.
(4) Input
Signals, data, etc. that are related to the observable stimulus and received by the system.
(5) Processing
Operation of the function actually achieved by the information system.
(6) Output
An output of the operation result of the function created by an output screen, transmission message, etc.
(7) Observable response
The output to the actor for sending the result of the function.
(8) Actor's post-situation
A situation in which an actor receives the result of the function.

For the complete elicitation of functional requirements, the components shown in Fig. 1 must be completely defined. It is necessary to consider the completeness of the relationship between each component as well as the completeness of the component.

By analyzing functional requirements with these requirement component descriptions, we cannot only clarify these components but also analyze their relationship with one another. Therefore, we can set requirement validation criteria based on these components. This means that if we can completely define actors, actor's situations, observable stimuli and responses, input, processing, and output, we can completely define the requirements.

However, we believe it is not practical to define all combinations of these components. Therefore, we propose a method for dealing with this explosion of combinations.

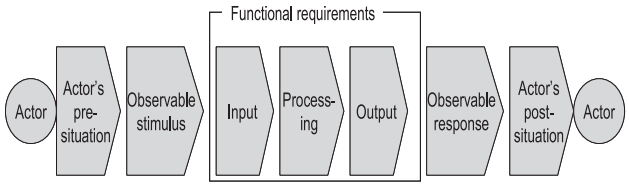

Fig. 1 Components for describing requirements.

3. Requirement Validation Criteria and Method Based on Actor Interaction

[Premise]
The number of situations of an actor in a system is not infinitely divergent, i.e., they are finite. An actor's situations are cyclical through interacting with the system.

This assumption is practical. For example, in resource management systems, a requestor reserves, cancels, utilizes, and makes a payment and releases a resource. In this case, the requestor and the resource can be regarded as actors. It is reasonable to assume cycles of an actor's situation. Under this premise, we propose a method for dealing with the explosion of combinations of requirement component descriptions, which was explained in Sect. 2.

3.1 Completion Criteria for Elicitation of Actor's Situation

By the premise described in the previous section regarding the cycle of an actor's situation, we can define a set of an actor's situations using a state transition table, where the vertical dimension indicates the current states, the horizontal dimension indicates the next states, and the cells contain the observable stimuli and responses that will lead to a particular state. Therefore, we can validate the completeness of an actor's situation if we can validate the completeness of this state transition table.

There are two types of actors, active and passive. An active actor sends a stimulus to another actor and receives the corresponding response. A passive actor receives a stimulus from the active actor and sends the corresponding response. There are observable stimuli and responses shared by information systems as well as actors in the real world. These stimuli and responses are implemented as information systems. There are observable stimuli and responses shared only by actors in the real world but not shared by information systems.

3.2 Completion Criteria for Elicitation of Observable Stimuli and Responses

With the same method described in Sect. 3.1, we can validate the completeness of an information system's status with observable stimuli and responses.

3.3 Completeness of Interaction between Actors

Each actor's situation changes when he/she/it interacts with another. This means that each actor's state transition table has to be complete, i.e., every stimulus and response as a

result of an actor interaction has to be included in the state transition table. The completeness of an actor's situation state transition table without considering interactions with other actors is meaningless.

3.4 Requirement Validation Method Based on Actor Interaction

The three types of completeness described above can be validated in the following steps.

Step 1. Focus on relationship between actors and classify them as active or passive.

Step 2. Elicit both actors' cyclical situations regarding interaction with another actor.

Step 3. Elicit courses of observable stimuli and responses in accordance with the cyclical transitions of both actors' situations elicited in Step 2.

Step 4. Construct both actors' state transition tables from the cyclical situations elicited in Step 2 and the stimuli and responses elicited in Step 3.

Step 5. Confirm that there are no missing statuses, stimuli, or responses by examining the interactions between the active and passive actors with their state transition tables.

Before the execution of these steps, organizational IT requirements engineering concerns, organizational actors and dependencies between them should be analyzed using goal-oriented approaches such as the i* framework [7] or the actor relationship matrix method [8]. After the execution of this method, the analysis results including potential requirement defects detected with this method should be validated by stakeholders such as customers of the system typically at requirement review meetings. This method is supposed to be used repeatedly and iteratively.

3.5 Guidelines for the Proposed Method

In Step 1, typically, managed resources and the information system can be passive actors, and their users can be active actors. Focusing on interactions between resources and their users is effective when we elicit actor's situations or courses of stimuli and responses.

In Step 2, a typical pattern of statuses of both active and passive actors can be described as follows.
{No interaction, Starting interaction, Being interactive, Finishing interaction}

If the resource is to be reused by another user, such as a rental commodity or space, the resource often returns to the initial status. If the resource is to be used only once, the resource finally leads to statuses such as "Discarded", or "Expired". These patterns can often be a starting point for eliciting situations of analyzed resources and their users. Actor's situations can often be expected to be derived from these typical patterns of statuses. If necessary, the statuses should be integrated or subdivided and more statuses should be added.

If we can elicit complete statuses in Step 2, it is not necessary to elicit all courses of observable stimuli and responses in Step 3. Defective or missing stimuli/responses can be detected by examining the state transition tables. However, it is often that there are some statuses that cannot be elicited in Step 2, and such statuses can be found by eliciting the courses of observable stimuli and responses. Examiners can use these courses when they trace the state transition table to confirm its correctness in Step 5. Therefore, it is preferable to elicit courses of observable stimuli and responses as much as possible in Step 3.

In Step 5, for the examination of the interaction between active and passive actors, we can make a direct product of the state transition table, i.e., the state transition table that contains the direct product of sets of both actors' statuses. The cells contain the observable stimuli and responses that will lead to a particular ordered pair of both actors' statuses.

We define X as the set of active actor's statuses and Y as the set of passive actor's statuses. Then, $X \times Y$, the direct product of X and Y, can be defined as follows.

$$X \times Y = \{(x, y) | x \in X, y \in Y\}$$

This is the set of all possible ordered pairs whose first component is a member of X and whose second component is a member of Y. If X has m-elements and Y has n-elements, the corresponding direct product $X \times Y$ has $m \times n$ elements.

In the same way, when the number of actors is p and the sets of each actor's statuses are defined as X_i, the direct product of these sets can be defined as follows.

$$\prod_{i=1}^{p} Xi = \{(x_1, \ldots, x_p) | x_1 \in X_1, \ldots., x_p \in X_p\}$$

This is the set of all possible ordered n-tuples. An ordered n-tuple is a set of n objects arranged in a specified order.

There are some rules that the direct product of the state transition table must satisfy. An examiner can detect defects when the direct product of the state transition tables does not satisfy the following rules.

(1) The direct product of state transition tables must be traced from the ordered n-tuple of all actors' initial statuses to all ordered n-tuples of all actors' statuses with stimulus/response pairs.
(2) The direct product of state transition tables must be traced from all ordered n-tuples with stimulus/response pair to the ordered n-tuple of all actors' final statuses.

Examiners can also detect defects when they find that some courses of observable stimuli and responses elicited in Step 3 cannot be traced in the direct product of the state transition tables if the courses are correct.

However, the state transition table cannot be validated based only on the examiner's knowledge. It is necessary for stakeholders, such as customers, to validate each cell in the direct product of the state transition tables. The examiners and the stakeholders should understand the meanings of current and next status and determine if a cell contains a observable stimulus/response pair.

It is often practical for examiners to omit an ordered n-tuple of statuses with no stimuli or responses because the number of elements of the direct product of each actor's statuses often becomes very large. When the examiners determine whether they can omit a certain ordered n-tuple of statuses or not, they should understand the meaning of the statuses of each actor and consider whether these statuses cannot occur simultaneously in the real world.

3.6 Potential Detectable Ability of Proposed Method

Requirements can be defined with the components described in Sect. 2. Therefore, if defects regarding an actor's situation or observable stimuli/responses are detected, defects of other related components may be detected. For example, if a missing stimulus is detected, defects of input, processing, or output may be detected.

Moreover, if we examine the cycles of an actor's situation against that of another actor, we may be able to find hidden relationship with still other actors or hidden new actors. They may be necessary to finish the cycles of the actor's situation when the actor cannot finish the cycles alone.

4. Example

In this section, we explain our proposed method by applying it to the requirement definition of a coin parking system in Japan. In coin parking, each parking space is equipped with a locking device, which activates automatically soon after a car parks in that space. The locking device deactivates when the driver pays the fee using an automatic payment system.

We can define two actors, the driver, as an active actor, and the parking space, as the passive actor. Figure 2 shows the interactions between the driver and the parking space.

The set of driver's statuses regarding the use of the parking space is {Driving, Grace period, Parking}. The grace period is the period given to the driver to exit before the car is re-locked into the space. The grace period is usually about 3 minutes in the real world. The set of parking space's statuses regarding being used by the drivers is {Vacant, Parked with locking device off, Parked with locking device on}.

The set of observable stimuli and responses found in the real world is {Entrance, Lock on, Payment, Lock off, Exit}. Courses of observable stimuli and responses are described as follows.

(1) The driver enters the parking space, makes a payment and exits the parking space.
(Driving) ⇒ Entrance to parking space ⇒ Lock on parking space ⇒ Payment ⇒ Lock off parking space ⇒ Exit from parking space ⇒ (Driving)

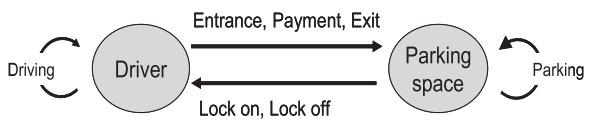

Fig. 2 Interactions between driver and parking space.

(2) The driver enters the parking space but exits within the grace period.
(Driving) ⇒ Entrance to parking space ⇒ Exit from parking space ⇒ (Driving)
(3) The driver completes payment, but he/she does not exit from the parking space within the grace period. This is regarded as the driver parking again.
(Driving) ⇒ Entrance to parking space ⇒ Lock on parking space ⇒ Payment ⇒ Lock off parking space ⇒ Lock on parking space ⇒ (Parking again)

Table 1 is the state transition table of a driver based on these cyclical transitions of his/her situations with the parking space. Table 2 is the state transition table of the parking space based on these cyclical transitions of its situations with the driver.

We then confirm that there are no missing stimuli and responses by examining interactions between the driver and the parking space with their state transition tables. Table 3 is the direct product of two state transition tables, i.e., Tables 1 and 2. The number of statuses in the direct product is the number of status products of the two transition tables. In Table 3, an ordered pair of statuses with no regular stimuli or responses is omitted. For example, the status of

Table 1 State transition table of driver.

(Next status) **Driver** (Current status)	Driving	Grace period	Parking
Driving	(Driving)	Entrance	-
Grace period	Exit	(Grace period)	End of grace period
Parking	-	Payment	(Parking)

Table 2 State transition table of parking space.

(Next status) **Parking Space** (Current status)	Vacant	Parked with locking device off	Parked with locking device on
Vacant	(Vacant)	Entrance acceptance	-
Parked with locking device off	Exit acceptance	(Parked with locking device off)	Lock on
Parked with locking device on	-	Lock off	(Parked with locking device on)

Table 3 Direct product of state transition tables of driver and parking space.

Driver (Current status)	(Next status) **Parking space** (Current status)	Driving Vacant	Grace period Parked with locking device off	Parking Parked with locking device on
Driving	Vacant	(Driving, Vacant)	Entrance	-
Grace period	Parked with locking device off	Exit	(Grace period, Parked with locking device off)	End of grace period / Lock on
Parking	Parked with locking device on	-	Payment/ Lock off	(Parking, Parked with locking device on)

Table 4 Direct product of state transition tables of manager, driver, and parking space. (when illegal abandonment of car occurs)

Manager		(Next status)	No need for intervention			Need for intervention	
	Driver	(Next status)	Driving	Grace period	Parking	Illegal abandonment	
		(Next status) **Parking**	Vacant	Parked with locking device off	Parked with locking device on	Parked with locking device off	Parked with locking device on
(Current status)	(Current status)	(Current status)					
No need for intervention	Driving	Vacant	(No need for intervention, Driving, Vacant)	Entrance	-	-	-
	Grace period	Parked with locking device off	Exit	(No need for intervention, Grace period, Parked with locking device off)	End of grace period / Lock on	-	-
	Parking	Parked with locking device on	-	Payment/ Lock off	(No need for intervention, Parking, Parked with locking device on)	-	Expiration of allotted time
Need for intervention	Illegal abandonment	Parked with locking device off	Forced exit	-	-	(Need for intervention, Illegal abandonment, Parked with locking device off)	-
		Parked with locking device on	-	-	-	Forced lock off	(Need for intervention, Illegal abandomnent, Parked with locking device on)

a parking space cannot be "Vacant" when the driver's status is "Parking" or "Grace period". Therefore, the ordered pairs of these statuses, i.e. (Parking, Vacant) or (Grace period, Vacant), can be omitted. From the direct product of the state transition tables, we can validate the completeness of regular stimuli and responses.

There are irregular stimuli in the real world such as forcibly exiting the parking space by destroying the locking device, illegal abandonment of a car, and incomplete payment due to the lack of small change. In these cases, it is necessary to include the manager of the parking space. We then examine the interactions between the manager and the parking space. We can also follow the steps mentioned above when we examine these interactions. Table 4 is the direct product of the state transition tables of the driver, parking space, and manager of the parking space when the parked car is regarded as illegally abandoned because the allotted time expired. In this case, the observable stimuli/responses such as "Forced lock off" and "Forced exit" by the manager are included in the table.

5. A Case Study

In this section, we verify the following hypothesis to validate the effectiveness of our proposed requirement validation criteria and method described in Sect. 3.

5.1 Hypothesis

We set the following verification hypothesis.

We can detect requirement defects or find missing requirements with the following steps.

Step 1. Elicit managed resources as passive actors and their users as active actors

Step 2. Examine users' cycles of situations regarding the use of managed resources

5.2 Experimentation

5.2.1 Outline of Analyzed System

We analyzed the development project for the core business system of a Japanese company. We describe our analysis and the effects we achieved using a virtual information system with basically the same business structure or workflow as the one we analyzed.

(1) The information system is of a Japanese retailer.
(2) The information system sells various items on the Internet.
(3) This system takes orders directly from customers through the Internet.
(4) There are operation managers for this system.
(5) Sometimes, operation managers suspend accepting orders from the customers.

For instance, operation managers suspend accepting orders when the number of orders is expected to be much higher than the expected number of arrival of goods. The operation managers also resume accepting orders.

5.2.2 Outlines of Requirement Defects Report

First, we analyze the requirement defect report from the system development project of the analyzed system to examine which requirement component description causes requirement defects.

We analyzed 332 requirement defect reports made after the requirement definition phase, i.e., in the design, implementation, or the test phase. Table 5 lists the results of this analysis. More than 75 % of the requirement defects came from the defects of input, processing and output, i.e., the machine world. Table 6 lists examples of requirement defects regarding an actor's situations, observable stimuli, and

Table 5 Analysis results of requirement defects from actual system development project.

Classification of requirement descriptions	Components for describing requirements	No. and % of defects found in each component		No. and % of defects found in each classification	
		No.	%	No.	%
Real world	Actors	6	1.8	37	11.1
	Actor's situations	7	2.1		
	Observable stimuli and responses	24	7.2		
Function	Input	98	29.5	250	75.3
	Processing	74	22.3		
	Output	78	23.5		
Others	Non functional requirements	19	5.7	19	5.7
	Documentation	26	7.8	26	7.8
Total		332	100.0	332	100.0

Table 6 Examples of requirement defects regarding actor's situations, observable stimuli, and responses.

No.	Component	Examples
1	Actor's situations	There was defect in conditions when operation managers suspended accepting orders due to shortage of expected number of goods.
2		There was defect in conditions when operation managers resumed accepting orders.
3	Observable stimuli and responses	It was not documented that it was necessary to accept order cancellation in suspension of accepting orders.
4		It was not documented that some managed information such as number of orders or amount of stock can change when order cancellation was accepted in suspension of accepting orders.
5		It was not documented that information system must cancel processing orders when an item cannot be reserved for certain time after terminating accepting orders.
6		It was not documented that when system received same requesting stimulus from operation managers, system has to reject latter one.
7		It was not documented that operation managers can send requesting stimulus for resumption of accepting orders to system only in the status of "Suspending accepting orders".

responses.

5.3 Analysis of Actor Interactions

In this section, we describe the actor interactions regarding this case study. Table 7 lists the combination of managed resources and their users.

5.3.1 Interaction between Customer and Commodity Space

Figure 3 shows the interaction between a customer and a commodity space. A commodity space consists of the commodity itself and the information system that manages the commodity.

The set of customer's statuses regarding the use of the commodity space is {Non-possession, Ordering, Possession}, and the set of commodity space's statuses regarding being used by the customer is {Out-of-stock and non-reserved, Out-of-stock and reserved, In-stock and non-reserved, In-stock and reserved, Possessed, Discarded}. In this case study, the commodity space is required to receive orders that is out-of-stock. Therefore, two statuses should be defined when the commodity space is "non-reserved".

Table 7 Combination of managed resource and user.

No.	Managed resource	User
1	Commodity	Customer
2	Commodity	Operation manager

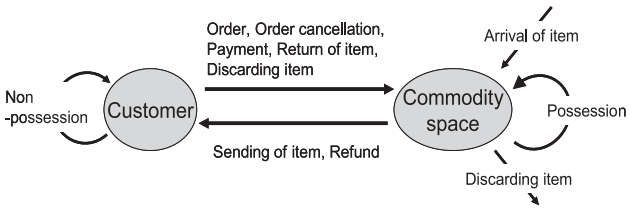

Fig. 3 Interaction between customer and commodity space.

The set of observable stimuli and responses shared by the two actors are {Order, Order cancellation, Payment, Sending of item, Return of item, Refund, Discarding item} "Discarding item" is the stimulus that is not shared by the information system, i.e., the stimulus is assumed to be sent directly from the customer to the commodity.

Courses of observable stimuli and responses based on the cycles of the customer's situation regarding the commodity space are described as follows. Nos. 4 to 7 are the courses of observable stimuli and responses including the "arrival of item" event.

(1) The customer orders an item in stock, makes a payment for it, receives it, and finally discards it.
(Non-possession) ⇒ Order ⇒ Payment/Sending of item ⇒ Discarding item ⇒ (Non-possession)
(2) The customer orders an item in stock, makes a payment for it, receives it, but returns it and is refunded.
(Non-possession) ⇒ Order ⇒ Payment/Sending of item ⇒ Return of item/Refund ⇒ (Non-possession)
(3) The customer orders an item in stock but cancels the order.
(Non-possession) ⇒ Order ⇒ Order cancellation ⇒ (Non-possession)
(4) The customer orders an item that is out of stock. The item arrives later, and the customer makes a payment for it, receives it, and finally discards it.
(Non-possession) ⇒ Order ⇒ Arrival of item ⇒ Payment/Sending of item ⇒ Discarding item ⇒ (Non-possession)
(5) The customer orders an item that is out of stock. The item arrives later, and the customer makes a payment for it, receives it, but returns it and is refunded.
(Non-possession) ⇒ Order ⇒ Arrival of item ⇒ Payment/Sending of item ⇒ Return of item/Refund ⇒ (Non-possession)
(6) The customer orders an item that is out of stock. The item arrives later, but the customer cancels the order.
(Non-possession) ⇒ Order ⇒ Arrival of item ⇒ Order can-

cellation ⇒ (Non-possession)
(7) The customer orders an item that is out of stock, but the customer cancels the order before the arrival of the item.
(Non-possession) ⇒ Order ⇒ Order cancellation ⇒ (Non-possession)

Table 8 is the state transition table of the customer based on the cycle of the customer's situations. Table 9 is the state transition table of the commodity space.

Table 8 State transition table of customer.

(Next status) **Customer** (Current status)	Non-possession	Ordering	Possession
Non-possession	(Non-possession)	Order request	-
Ordering	Order cancellation	(Ordering)	Payment /Receipt of item
Possesion	・Discarding item ・Return of item/Refund	-	(Possession)

Table 10 is the direct product of the two state transition tables, i.e., the direct product of Tables 8 and 9. In Table 10, an ordered pair of two actor's statuses with no regular stimuli or responses is omitted.

5.3.2 Interaction between Operation Manager and Commodity Space

Figure 4 shows the interaction between an operation manager and a commodity space.

The set of statuses of the operation manager regarding

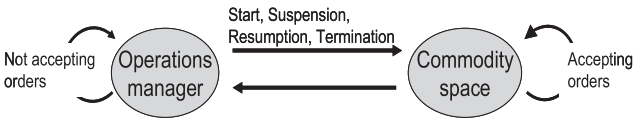

Fig. 4 Interactions between operation manager and commodity space.

Table 9 State transition table of commodity space.

(Next status) **Commodity space** (Current status)	Out-of-stock and non-reserved	Out-of-stock and reserved	In-stock and non-reserved	In-stock and reserved	Possessed	Discarded
Out-of-stock and non-reserved	(Out-of-stock and non-reserved)	Order acceptance	Arrival of item	-	-	-
Out-of-stock and reserved	Order cancellation acceptance	(Out-of-stock and reserved)	-	Arrival of item	-	-
In-stock and non-reserved	-	-	(In-stock and non-reserved)	Order acceptance	-	-
In-stock and reserved	-	-	Order cancellation acceptance	(In-stock and reserved)	Payment acceptance /Sending of	-
Possessed	-	-	Return of item acceptance /Refund request	-	(Possessed)	Discarding item
Discarded	-	-	-	-	-	(Discarded)

Table 10 Direct product of state transition tables of customer and commodity space.

Customer (Current status)	(Next status) (Next status) **Commodity space** (Current status)	Non-possession: Out-of-stock and non-reserved	Non-possession: In-stock and non-reserved	Non-possession: Discarded	Ordering: Out-of-stock and reserved	Ordering: In-stock and reserved	Possession: Possessed
Non-possession	Out-of-stock and non-reserved	(Non-possession, Out-of-stock and non-reserved)	Arrival of item	-	Order	-	-
	In-stock and non-reserved	-	(Non-possession, In-stock and non-reserved)	-	-	Order	-
	Discarded	-	-	(Non-possession, Discarded)	-	-	-
Ordering	Out-of-stock and reserved	Order cancellation	-	-	(Ordering, Out-of-stock and reserved)	Arrival of item	-
	In-stock and reserved	-	Order cancellation	-	-	(Ordering, In-stock and reserved)	Payment / Sending of item
Possession	Possessed	-	Return of item /Refund	Discarding item	-	-	(Possession, Possessed)

the interaction with the commodity space is {Still not accepting orders, Accepting orders, Suspending accepting orders, No longer accepting orders}. The set of statuses is the same as that of the operation manager and commodity space.

Courses of observable stimuli and responses based on the cycles of the operation manager's situation regarding the commodity space are described as follows.

(1) An operations manager makes a request to the commodity space to start accepting orders, and then he/she makes a request to terminate accepting orders.
(Still not accepting orders) ⇒ Start of accepting orders ⇒ Termination of accepting orders ⇒ (No longer accepting orders)
(2) An operations manager makes a request to the commodity space to start, suspend, and resume accepting orders. Finally he/she makes a request to terminate accepting orders.
(Still not accepting orders) ⇒ Start of accepting orders ⇒ Suspension of accepting orders ⇒ Resumption of accepting orders ⇒ Termination of accepting orders ⇒ (No longer accepting orders)
(3) An operations manager makes a request to the commodity space to start, suspend accepting orders, and finally he/she makes a request to terminate accepting orders. (After suspension, there was no resumption of accepting orders)
(Still not accepting orders) ⇒ Start of accepting orders ⇒ Suspension of accepting orders ⇒ Termination of accepting orders ⇒ (No longer accepting orders).

Table 11 is the state transition of the operation manager. Table 12 is the state transition table of the commodity space. Table 13 is the direct product of the two state transition tables, i.e., the direct product of Tables 11 and 12. This state transition table is the same as that of the operation manager and commodity space.

Table 11 State transition table of operation manager.

(Next status) **Operation manager** (Current status)	Still not accepting orders	Accepting orders	Suspending accepting orders	No longer accepting orders
Still not accepting orders	(Still not accepting orders)	Start request	-	-
Accepting orders	-	(Accepting orders)	Suspension request	Termination request
Suspending accepting orders	-	Resumption request	(Suspending accepting orders)	Termination request
No longer accepting orders	-	-	-	(No longer accepting orders)

Table 12 State transition table of commodity space.

(Next status) **Commodity space** (Current status)	Still not accepting orders	Accepting orders	Suspending accepting orders	No longer accepting orders
Still not accepting orders	(Still not accepting orders)	Start acceptance	-	-
Accepting orders	-	(Accepting orders)	Suspension acceptance	Termination acceptance
Suspending accepting orders	-	Resumption acceptance	(Suspending accepting orders)	Termination acceptance
No longer accepting orders	-	-	-	(No longer accepting orders)

5.3.3 Interactions among Customer, Operation Manager, and Commodity Space

Both the customer and the operations manager interact with the commodity space. Therefore, it is necessary to examine the cycles of the commodity space interacting with both users. Table 14 is the direct product of the state transition tables of these three actors. Due to lack of space, Table 14 shows only when the operation manager's statuses are "Accepting orders" and "Suspending accepting orders". The cells with only parentheses are described in the diagonal elements of the table which give statuses of multiple actors.

In Table 14, we made the direct product of all the statuses of the three actors, even when the consumer status is "Non-possession" and the status of the commodity space is "Discarded". When the consumer status is "Non-possession" and the status of the commodity space is "Discarded", the ordered triple of these two actors' statuses and that of an operation manager is not effective, i.e., does not need to be considered. The direct product of the state transition tables, such as Table 14, is thought used by human reviewers mainly in the requirement definition phase. Human reviewers are expected to judge the effectiveness of each ordered n-tuple of different actors' statuses by understanding the meaning of observable stimulus or responses in the cell.

5.4 Detectable Requirement Defects

We can examine observable stimuli and responses and their related state transitions exhaustively using the direct products of the state transition tables of actors. If the concerned stimuli or responses can be detected explicitly in the cell of the direct product of the state transition tables, we can judge that the requirement defects can be detected with our proposed method.

We describe three types of detectable defects, which can be detected with our proposed method with the examples of the requirement defects in Table 6.

Table 13 Direct product of state transition tables of operation manager and commodity space.

Operation manager (Current status)	(Next status) **Commodity space** (Current status)	Still not accepting orders	Accepting orders	Suspending accepting orders	No longer accepting orders
Still not accepting orders	Still not accepting orders	(Still not accepting orders)	Start	-	-
Accepting orders	Accepting orders	-	(Accepting orders)	Suspension	Termination
Suspending accepting orders	Suspending accepting orders	-	Resumption	(Suspending accepting orders)	Termination
No longer accepting orders	No longer accepting orders	-	-	-	(No longer accepting orders)

Table 14 Part of direct product of state transition tables of customer, operation manager, and commodity space.

Operation manager	(Next status)		Accepting orders						Suspending accepting orders					
	Customer	(Next status)	Non-possession			Ordering		Possesion	Non-possesion			Ordering		Possession
		(Next status) Commodity space	Out-of-stock and non-reserved	In-stock and non-reserved	Discarded	Out-of-stock and reserved	In-stock and reserved	Possessed	Out-of-stock and non-reserved	In-stock and non-reserved	Discarded	Out-of-stock and reserved	In-stock and reserved	Possessed
(Current status)	(Current status)	(Current status)												
Accepting orders	Non-possession	Out-of-stock and non-reserved	( )	Arrival of item	-	Order (Note 1)	-	-	Suspension	-	-	-	-	-
		In-stock and non-reserved	-	( )	-	-	Order (Note 1)	-	-	Suspension	-	-	-	-
		Discarded	-	-	( )	-	-	-	-	-	-	-	-	-
	Ordering	Out-of-stock and reserved	Order cancellation (Note 1)	-	-	( )	Arrival of item	-	-	-	-	Suspension	-	-
		In-stock and reserved	-	Order cancellation (Note 1)	-	-	( )	Payment request/ Sending of item	-	-	-	-	Suspension	-
	Possession	Possessed	-	Return of item/ Refund	Discarding item	-	-	( )	-	-	-	-	-	Suspension
Suspending accepting orders	Non-possession	Out-of-stock and non-reserved	Resumption	-	-	-	-	-	( )	Arrival of item	-	- (Note 2)	-	-
		In-stock and non-reserved	-	Resumption	-	-	-	-	-	( )	-	-	- (Note 2)	-
		Discarded	-	-	-	-	-	-	-	-	( )	-	-	-
	Ordering	Out-of-stock and reserved	-	-	-	Resumption	-	-	Order cancellation (Note 2)	-	-	( )	Arrival of item	-
		In-stock and reserved	-	-	-	-	Resumption	-	-	Order cancellation (Note 2)	-	-	( )	Payment / Sending of item
	Possession	Possessed	-	-	-	-	-	Resumption	-	Return of item/ Refund	Discarding item	-	-	( )

Note 1. In the status of "Accepting orders", both Order and Order cancellation should be accepted.
Note 2. In the status of "Suspending accepting orders," Order should NOT be accepted but Order cancellation SHOULD be accepted.

5.4.1 Detection of Necessary Alterations of Observable Stimuli or Responses in State Transition Tables

Detectable defects can be found in Nos. 3 and 4 in Table 6. Table 14 is the direct product of the state transition tables of the three actors when the status set by an operation manager is "Accepting orders" and "Suspending accepting orders". The gray-colored cells in Table 14 reflect these defects.

When the status set by an operation manager is "Suspending accepting orders", it is not possible to accept an "Order request" from the customer. We can consider whether we should accept an "Order cancellation request" or not with the direct product of the state transition tables. Usually, an "Order cancellation request" should be accepted even when the status set by an operation manager is "Suspending accepting orders". We can detect the necessary alterations regarding how to deal with observable stimuli or responses from a customer by using the direct product of the state transition tables including that of operation managers.

5.4.2 Detection of Missing Stimuli or Responses Necessary to Finish the Cycles of User's Situation against Commodity Space

Detectable defects can be found in No. 5 in Table 6. Table 16 is the direct product of the state transition tables of the three actors when the status of the operation manager is "No longer accepting orders". No. 5 in Table 6 occurs in the gray-colored cell in Table 15, i.e., when the status of the customer is "Ordering" and the status of the commodity is "Out-of-stock and reserved". If all items arrive, even after accepting orders has terminated, the cycles of the customer's situation can be completed. However, it is necessary to cancel existing orders because the items do not always arrive in the commodity space. In this case, it is normal to cancel orders from the commodity space, not from the customer. By "Order cancellation", the status of the customer transits to "Non-possession" and the status of the commodity space transits to "Out-of-stock and non-reserved". If the project members were conscious that the cycles of the customer's situation must be completed, the missing observable stimuli or responses detected in No. 5 would have been detected.

Table 15 Direct product of state transition tables when status set by operation manager is "No longer accepting orders".

Operation manager (Current status)	**Customer** (Current status)	(Next status) **Commodity space** (Current status)	No longer accepting orders					
		(Next status)	Non-possession			Ordering		Possession
			Out-of-stock and non-reserved	In-stock and non-reserved	Discarded	Out-of-stock and reserved	In-stock and reserved	Possessed
No longer accepting orders	Non-possession	Out-of-stock and non-reserved	(No longer accepting orders, Non-possesion, Out-of-stock and non-reserved)	Arrival of item	-	-	-	-
		In-stock and non-reserved	-	(No longer accepting orders, Non-possesion, In-stock and non-reserved)	-	-	-	-
		Discarded	-	-	(No longer accepting orders, Non-possesion, Discarded)	-	-	-
	Ordering	Out-of-stock and reserved	Order cancellation (Note 1)	-	-	(No longer accepting orders, Ordering, Out-of-stock and reserved)	Arrival of item	-
		In-stock and reserved	-	Order cancellation	-	-	(No longer accepting orders, Ordering, In-stock and reserved)	Payment/ Sending of item
	Possession	Possessed	-	Return of item / Refund	Discarding item	-	-	(No longer accepting orders, Possession, Possessed)

Note 1. Normally it is the commodity space that should trigger "Order cancellation".

Table 16 State transition table of operation managers.

(Current status) **Commodity space** (Stimulus)	Still not accepting orders	Accepting orders	Suspending accepting orders	No longer accepting orders
Start	Accepting orders	- (Note 1)		
Suspension		Suspending accepting orders	- (Note 1)	
Resumption	- (Note 2)	- (Note 1),(Note 2)	Accepting orders	- (Note 2)
Termination		No longer accepting orders	No longer accepting orders	- (Note 1)

Note 1. Stimuli from operation managers are unacceptable after the commodity space transits to the status after accepting the stimuli.
Note 2. Resumption is acceptable only when the status of the commodity space is "Suspending accepting orders".

5.4.3 Clarification of Observable Stimuli That Should be Rejected in Certain Status

Detectable defects can be found in Nos. 6 and 7 in Table 6.

Observable stimuli that should be rejected can be clarified by examining another type of state transition table. Table 16 is the state transition table of the commodity space where the vertical dimension indicates the observable stimuli, the horizontal dimension indicates the current states, and the cells contain the next status. We judged that these defects can be regarded as detectable as a result of clarification of actor's statuses and stimuli/responses of our proposed method.

For example, once the commodity space receives the "Start Request" from the operation manager, the commodity space sends "Start Acceptance" to the operation manager and leads to the status "Accepting orders". We can consider how the commodity space, with the state "Accepting orders", should do if it receives "Start Request" again in one of the gray-colored cells in Table 16. We can consider what the commodity space should do if it receives a request from the operation manager, when the commodity space's status already transited to the status after the request has been received. We can clarify what the commodity space should do when it receives "Resumption Request" in each status by considering the other gray-colored cells in Table 16.

5.5 Undetectable Requirements Defects

Undetectable defects can be found in Nos. 1 and 2 in Table 6, which are the defect reports regarding the actor's situation when the operation manager is the actor.

In No. 1, it is necessary to clarify what kind of operation manager's situation should produce the stimulus "Suspension request". For example, if the expected amount of orders is more than N times larger than the expected amount of arrivals, it may be necessary to suspend accepting orders. Also, in No. 2, it is necessary to clarify what kind of operation manager's situation should produce the stimulus "Resumption request". These situations should be elicited from the stakeholders and are difficult to detect by the examiners.

It may be possible to clarify some cycles of the operation manager's situation based on the amount of orders and arrivals. However, there still remain situations that are difficult to clarify. For example, the suspension of receiving orders can occur due to heuristic decisions of operation managers in real system operation. There are various unpredictable situations and stimuli in the real world, and it is often the operation managers who must first respond to such stimuli.

In this case study, all defects regarding an actor's situation were those of operation managers. However, there are

Table 17 Relationship between examined interactions and number of detected defects.

No.	Examined interactions	Types of detected defects		Total
		Detection of missing or defective stimuli/ responses	Clarifications of stimuli/responses to be rejected	
1	Only customer and commodity space	0	0	0
2	Only operation manager and commodity space	0	9	9
3	Customer, operation manager, and commodity space	11	4	15
Total		11	13	24

other actors' situations that cannot be detected with the proposed method unless they are elicited from the stakeholders.

5.6 Results from Hypothesis Verification

Table 17 shows the relationship between the examined interactions of the actors and the number of detected defects. In this case study, all detectable defects are regarding observable stimuli and responses. The ratio of defects that can be detected with the proposed method to all defects is 7.2 %, i.e., the ratio of all defects regarding observable stimuli and responses in Table 5.

Therefore the hypothesis, which was set in Sect. 5.1, was affirmed by this case study.

The reasons above are that, in requirement defects reports of actual system development projects, there were a number of defects that could be detected using the proposed method. We could detect some defects by validating the necessary alterations of observable stimuli or responses in the direct product of the state transition tables of multiple actors. We also could detect some missing observable stimuli or responses by examining whether the cycles of a user's situation are complete.

6. Discussion

6.1 Relationship between Examined Interactions and Number of Detected Defects

Table 17 shows the relationship between the examined interactions of the actors and the number of detected defects regarding observable stimuli and responses. In Table 17, no defects could be detected in the examination of the interactions between the customer and the commodity space. Also, in the examination of the interactions between the operation manager and the commodity space, only the defects regarding the observable stimuli, which should be rejected in certain statuses, were detected. All defects regarding missing or defective stimuli and responses were detected in the examinations of the three actors, i.e., the customer, operation manager, and commodity space. It is obvious that this is because the examination of interactions becomes more difficult and tends to be omitted as the number of actors increases.

Table 18 Work item and Man-hours of our analysis.

Work item	Prepa-ration	Step 1	Step 2	Step 3	Step 4	Step 5	Total
Man-hours	56.5	6	13	7	21	46	149.5

Table 19 Defect detection rate. (No. of defects/man-hour)

Our case study	Case study by Berling et al. [9]				
	Project A	Project B	Project C	Project D	Average
0.161	0.173	0.174	0.081	0.083	0.127

6.2 Ratio of Detectable Defect in All Requirement Defect Reports

In Table 5, the ratio of defects that can be detected with our proposed method to all defects is 7.2 %, i.e., the ratio of defects regarding observable stimuli and responses. This number may not seem high, but we believe that in the requirement definition phase, defects regarding observable stimuli and responses are more severe than those in input, processing, and output because they are produced as a result of real-world analysis. We first heard this opinion from the project manager of the analyzed system development project. We believe that the requirement validation criteria and method based on actor interaction can contribute to the improvement in the quality of requirement definitions.

6.3 Efficiency of Requirement Defect Detection

Efficiency of a defect detection method is usually evaluated by defect detection rate, i.e., the number of detected defects per man-hour. In our case study, we did not first analyze the requirement specifications but the requirement defect reports after finishing the requirement definition phase. We calculated the provisional defect detection rate of our proposed method from the time required in our analysis.

Table 18 shows the work item and the time required of our case study. In preparation phase, we analyzed 332 requirement defect reports, found defects regarding actor's situation, observable stimuli, and responses and then analyzed the requirement specifications. Then we performed our method from Steps 1 to 5 in Sect. 3.4.

We consumed 149.5 man-hours and found 24 defects regarding observable stimuli and responses are detectable. The defect detection rate was 0.161 defects per man-hour. Berlin et al. conducted an industrial case study of the verification and validation activities including inspection for requirement specifications [9]. Table 19 lists their results. We judged that their case study can be compared with ours because they calculated the defect detection rate divided by the time spent for preparation as well as the review meetings, they gathered data only from formal inspections, and their requirement specifications are relatively mature. The

Table 20 Effect of cost reduction to fix requirement defects at each development phase.

Phase	Assumed value		No. of detected defects of stimuli/responses		No. of frozen requirements (Xi)		Cost to fix requirement defects (Ci × Xi)		Effect of cost reduction	
	Relative cost to fix a requirement defect (Ci)	percentage of requirements frozen	Before	After	Before	After	Before (P)	After (Q)	Q-P	(Q-P)/P(%)
Requirements definition	1	70	a	a+24	775	799	775	799	24	3.10
Design	5	10	8	0	111	103	555	515	-40	-7.21
Coding	10	10	8	0	111	103	1110	1030	-80	-7.21
Unit/Integration Testing	30	9	7	0	100	93	3000	2790	-210	-7.00
System/Acceptance Testing	50	1	1	0	10	9	500	450	-50	-10.00
Total	-	100	a+24	a+24	1107	1107	5940	5584	-356	-5.99

efficiency of our requirements defect detection method is almost the same as theirs.

6.4 Effect of Cost Reduction of Proposed Method

We calculated the assumed effect of cost reduction if our proposed method was applied and all defects regarding observable stimuli and responses were detected in the requirement definition phase. We assume that the cost to fix a requirement defect is unchanged if it is detected in the same development phase. We define E, the total cost to fix requirement defects, as follows.

$$E = \sum Ci \times Xi,$$

where C_i is the cost to fix a requirement defect phase and X_i is the number of requirements frozen at each development phase.

Table 20 shows the assumed numbers of requirements frozen, the cost to fix requirement defects and change of required cost at each development phase. In Table 20, we referred to Mogyorodi [1] for the assumed value of relative cost to fix a requirement defect. We referred to Sanker et al. [10] for the assumed percentage of requirements frozen because we could only obtain the number of requirement defects after finishing the requirement definition phase.

In Table 20, the total cost to fix requirements defects decreases by 356, although the cost increases by 24 at the requirements definition phase. Our proposed method can be used to reduce 5.99 % of the total cost for fixing requirement defects. This means that our method can contribute to the success of a software development project because it can decrease the total cost of the development project.

7. Related Work

In goal-oriented approaches, the i* framework is popular for analyzing dependencies between actors [7]. However, validating the completeness of the model is reported to be difficult, especially when this method is applied to an industrial case with a large number of actors [11]. Yamamoto et al. proposes an actor relationship matrix method (ARM), which clarifies the dependencies between actors in a cell of the table [8]. ARM is a rather easy and practical method, which enables requirement engineers to better ensure completeness of requirements between actors than with the i* framework. Our proposed method is intended for more precise and complete analysis of requirements, especially for an actor's situation and related stimuli and responses.

A problem frame is a method for analyzing problems in the real world rather than inventing solutions [3]. A problem diagram is made up of requirements, real-world domains, a machine, and the links of phenomena they share. Various methods are proposed for describing further details of each element. Classen et al. use Event Calculus [12], Seater et al. use the Alloy Analyzer and its language [13], Salifu et al. uses a state chart to describe events in real-world domains and related behavior when they analyze requirements for context awareness applications [14], and in our method, state transition tables are used for precise description of events, i.e., stimuli and responses in the real-world domain.

Leveson describes the following seven types of completion criteria for requirement analysis [15].

(1) Human-Computer Interface Criteria
(2) State Completeness
(3) Input and Output Variable Completeness
(4) Trigger Event Completeness
(5) Output Specification Completeness
(6) Output to Trigger Event Relationships
(7) Specification of Transitions Between States

It seems difficult to validate completeness of all these types due to the limitation of cost or scheduling. However, if these completion criteria are not considered, the project may be too optimistic in estimation of effort, cost, or scheduling. Our proposed method can be used to validate Leveson's completion criteria effectively. By regarding a human operator as an active actor and a computer as a passive actor, we can validate the completeness of a trigger or stimulus and output an event or response. We can also validate the completion of a status by examining an actor's situation.

State machine-based formalism, such as labeled transition systems (LTS), are expected to describe more complete requirement specifications. Uchitel et al. propose an algorithm for translating scenarios into behavioral specification with LTS [16]. They also propose using partial LTS to clarify the remaining definition of system behavior [17]. Our

proposed method can also be useful for translating requirement specifications into LTS.

8. Conclusions and Future Work

We proposed requirement validation criteria and a method based on the interaction between actors. We focused on the relationship between active and passive actors and stimuli and responses between them. An active actor has cycles of his/her/its situations against a passive actor, and both actors' situations can be described in a state transition table, which describes observable stimuli or responses the actors send or receive. Examination with the direct product of the state transition tables of the concerned actors enables us to detect missing or defective requirements, i.e., missing or defective observable stimuli or responses. As a case study, we analyzed requirement defect reports of an actual development project of the core business system in a Japanese company. We found that there are a certain amount of defects regarding stimuli and responses, which can be detected with our proposed method if this method is used in the requirement definition phase.

For future work, we are planning to apply our proposed method to an actual system development project from the beginning of the requirement definition phase.

References

[1] G. Mogyorodi, "Requirements-based testing: An overview," Proc. 39th International Conference and Exhibition on Technology of Object-Oriented Language and Systems (TOOLS'01), pp.286–295, 2001.
[2] D. Svetinovic, D.M. Berry, and M. Godfrey, "Concept identification in object-oriented domain analysis: Why some students just don't get it," Proc. 13th IEEE International Conference on Requirements Engineering (RE'05), pp.189–198, 2005.
[3] M. Jackson, Problem Frames: Analysing & Structuring Software Development Problems, Addison-Wisley Professional (ACM Press), 2000.
[4] IEEE Computer Society, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std., 830–1998, 1998.
[5] J. Verner, K. Cox, S. Bleistein, and N. Cerpa, "Requirements engineering and software project success: An industrial survey in Australia and the U.S.," Australian J. Information Systems (AJIS), vol.13, no.1, pp.225–238, 2005.
[6] D. Damian and J. Chisan, "An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management," IEEE Trans. Softw. Eng., vol.32, no.7, pp.433–453, 2006.
[7] i*: http://www.cs.toronto.edu/km/istar
[8] S. Yamamoto, K. Ibe, J. Verne, K. Cox, and S. Bleistein, "Actor relationship analysis for the i* framework," Proc. 11th International Conference on Enterprise Information Systems, (ICEIS 2009), LNBIP, vol.24, pp.491–500, Springer-Verlag, Heidelberg, 2009.
[9] T. Berling and T. Thelin, "An industrial case study of the verification and validation activities," Proc. 9th International Software Metrics Symposium (METRICS'03), pp.226–238, 2003.
[10] K. Sankar and R. Venkat, "Total requirements control at every stage of product development," Proc. 15th IEEE International Requirements Engineering Conference (RE'07), pp.337–342, 2007.
[11] H. Estrada, A. Martínez, O. Pastor, and J. Mylopoulos, "An experimental evaluation of the i* framework in a model-based software generation environment," CAiSE 2006, E. Dubois, K. Pohl, eds. LNCS, vol.4001, pp.513–527, 2006.
[12] A. Classen, P. Heymans, and P.Y. Schobbens, "What's in a feature: A requirements engineering perspective," Proc. 11th International Conference on Fundamental Approaches to Software Engineering (FASE'08), LNCS, vol.4961, pp.16–30, 2008.
[13] R. Seater and D. Jackson, "Requirement progression in problem frames applied to a proton therapy system," Proc. 14th IEEE International Conference on Requirements Engineering (RE'06), pp.166–175, 2006.
[14] M. Salifu, Y. Yu, and B. Nuseibeh, "Specifying monitoring problems in context," Proc. 15th IEEE International Conference on Requirements Engineering (RE'07), pp.211–220, 2007.
[15] N. Leveson, SAFEWARE: System Safety and Computers, Addison-Wesley, 1995.
[16] S. Uchitel, J. Kramer, and J. Magee, "Synthesis of behavioral models from scenarios," IEEE Trans. Softw. Eng., vol.29, no.2, pp.99–105, 2003.
[17] S. Uchitel, J. Kramer, and J. Magee, "Behavior model elaboration using partial labelled transition systems," Proc. 11th ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE'03), pp.19–27, 2003.

Noboru Hattori is an associate researcher, at the Center for Applied Software Engineering, Research and Development Headquarters of NTT DATA Corp. Currently he is also a Ph.D. student at the Graduate School of Systems Engineering at Wakayama University. He received his B.Ec from Keio University and joined NTT DATA Corp. in 1989. He has worked in research and development of communication protocols, empirical software engineering, and requirements engineering.

Shuichiro Yamamoto is a Professor in the Headquarter of Information and Communication Services Information and Communication Planning Office at Nagoya University, Japan. Previously, he was engaged in the development of programming languages, CASE tools, network-based smart card environments, and distributed application development platforms. His research interests include distributed information systems, requirements engineering, ubiquitous computing, Knowledge creation and Knowledge management. He joined NTT in 1979. He received his B.S. in information engineering from Nagoya Institute of Technology in 1977, and his M.S. and Ph.D. in information engineering from Nagoya University in 1979 and 2000. He joined NTT DATA Corp. in 2002 and had been Deputy Senior Executive Manager (2002–2007). He had also?been Research Fellow and Director of Research at the Institute of System Science (2007–2009), Research and Development Headquarters of NTT DATA Corp.

Tsuneo Ajisaka is a Professor in the Faculty of Systems Engineering at Wakayama University, Japan. He has been a researcher and educator of software engineering with Kyoto University (1985–1997, assistant/associate professor) and Wakayama University (1997–). His research interests include software analysis/design methods, requirements engineering, and program comprehension. He received his B.S. in physics and his M.E. and Ph.D. in informatics from Kyoto University.

Tsuyoshi Kitani is Deputy Senior Executive Manager, Research and Development Headquarters, NTT DATA Corp. He started his research career at a NTT Research Laboratory after receiving his B.S. in Electrical Engineering from Keio University in 1983. At the laboratory, he did research in the area of natural language processing, particularly word segmentation from Japanese text. After he joined NTT DATA Corp. in 1988, he joined the Center for Machine Translation, Carnegie Mellon University as a visiting researcher where he stayed from 1991 to 1993. His research included information extraction, information retrieval and text categorization. He received his Ph.D. in electrical engineering from Keio University in 1995. He now sees software engineering R&D at NTT DATA Corp.