PAPER Special Section on Knowledge-Based Software Engineering

A Generation Method of Alternative Scenarios with a Normal Scenario

Atsushi OHNISHI[†a)], ***Member and*** **Koji KITAMOTO**[†*], ***Nonmember***

SUMMARY This paper proposes a method to generate alternative scenarios from a normal scenario written with a scenario language. This method includes (1) generation of alternative plans and (2) generation of alternative scenario by a user's selection of these plans. The proposed method enables users to decrease the omission of the possible alternative scenarios in the early stages of development. The method will be illustrated with some examples.

key words: *requirements elicitation, scenario analysis, scenario generation*

1. Introduction

Scenarios are important in software development, particularly in requirements engineering, by providing concrete system description [19]. In particular, scenarios are useful for system developers in defining system behaviors and validating the customers' requirements. In many cases, scenarios are foundation for system development. Incorrect scenarios will have a negative impact on the overall system development process. However scenarios are informal and it is difficult to verify the correctness of scenarios. The errors in incorrect scenarios may include (1) vague representations, (2) lack of necessary events, (3) extra events, and (4) wrong sequence among events.

The authors have developed a scenario language for describing scenarios. In this scenario language, simple action traces are embellished including typed frames based on a simple case grammar of actions and for describing the sequence among events [12], [20]. Since this language is a controlled language, the vagueness of the scenario written in this language can be reduced. Furthermore, the scenario with this language can be transformed into internal representation. In the transformation, both the lack of cases and the illegal usage of noun types can be detected, and concrete words will be assigned to pronouns and omitted indispensable cases [11], [20]. As a result, the scenario with this language can avoid the errors typed 1 previously mentioned.

Scenarios can be classified into (a) normal scenario, (b) alternative scenario, and (c) exceptional scenario. A normal scenario represents the normal and typical behavior of the target system, while an alternative scenario represents normal but untypical behavior of the system and an exceptional scenario represents abnormal behavior of the system. In order to grasp whole behaviors of the system, not only normal scenarios, but also alternative/exceptional scenarios should be specified in the requirements definition phase. However it is difficult to detect alternative scenarios and exceptional scenarios, whereas it is easy to think of normal ones.

Since we have already established a generation method of exceptional scenarios from a normal scenario [14], this paper focuses on how to generate alternative scenarios from a normal scenario. The similarity between a normal scenario and alternative scenarios is that they have the same purpose and achieve it finally. The difference between them is that their event sequences to achieve the purpose are different each other.

We adopt our scenario language prepared beforehand to write scenarios, because previously proposed scenario language is a control language and it is easy to analysis scenarios with that scenario language.

2. Scenario Language

2.1 Outline

The scenario language has already been introduced [12], [20]. In this paper, a brief description of this language will be given for convenience.

A scenario can be regarded as a sequence of events. Events are behaviors employed by users or the system for accomplishing their goals. It is assumed that each event has just one verb, and that each verb has its own case structure. The scenario language has been developed based on this concept. Verbs and their own case structures depend on problem domains, but the roles of cases are independent of problem domains. The roles include agent, object, recipient, instrument, source, etc. [6], [11].

Requirements frames were provided previously, in which their verbs and own case structures are specified. The requirements frame depends on problem domains. Each action has its case structure, and each event can be automatically transformed into internal representation based on the frame. In the transformation, concrete words will be assigned to pronouns and omitted indispensable cases. With Requirements Frame, users can detect both the lack of cases and the illegal usage of noun types [11].

It is assumed that four kinds of time sequences among events, that is, 1) sequential, 2) selective, 3) iterative, and

Manuscript received July 1, 2009.
Manuscript revised October 16, 2009.

[†]The authors are with the Faclty of Information Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.
[*]Presently, with the NTT Data Corporation.
a) E-mail: ohnishi@cs.ritsumei.ac.jp

DOI: 10.1587/transinf.E93.D.693

4) parallel. Actually most events are sequential events.

This scenario language defines the semantic of verbs with their case structure. For example, data flow verb has source, goal, agent, and instrument cases. Since our scenario language provides limited vocabulary and limited grammar, there exist two benefits, namely, (1) homonym problem can be solved, because limited vocabulary does not permit such problem and (2) the abstraction level of any scenarios becomes almost same, because case structures of actions depend on a certain abstraction level.

2.2 Scenario Example

In this subsection, a scenario of train ticket reservation of a railway company is employed. Figure 1 shows a scenario of customer's purchasing a ticket of express train at a service center of a railway company. This scenario is written in our scenario language based on a video that records behaviors of both a user and a staff at a service center of a railway company [15].

A title of the scenario is given in the first line of the scenario in Fig. 1. Viewpoints of the scenario are specified in the second line. In this paper, viewpoints mean active objects such as human, system appearing in the scenario. There exist two viewpoints, namely staff, and customer. The order of the specified viewpoints means the priority of specified viewpoints. In this example, the object of the first priority is staff, and the second one is customer. In such a case, the first priory object becomes the subject of an event. Pronouns are available and pronouns will be assigned with concrete nouns in analysis. If there are several candidates for the assignment, scenario describer will select one of them.

In this scenario, almost all events are sequential, except for just two selective events (the 9th event and the 12th event). Selection can be expressed with if-then syntax like program languages. Actually, event number is only for reader's convenience and not necessary.

2.3 Analysis of Events

Each event is transformed into internal representation. For example, the 2nd event "He sends the customer's request to reservation center via private line" can be transformed into internal representation shown in Table 1.

In this event, the verb "send" corresponds to the concept "data flow." The data flow concept has its own case structure with four cases, namely source case, goal case, object case and instrument case. Sender corresponds to the source case and receiver corresponds to the goal case. Data transferred from source case to goal case corresponds to the object case. Device for sending data corresponds to the instrument case. In this event, "customer's request" corresponds to the object case. Since the pronoun "he" in the event should be "staff," concrete noun "staff" is assigned in the source case.

Let us consider other surface representations of the event, "Customer's request is sent from staff to reservation center via private line" and "reservation center receives customer's request from staff via private line." These events are syntactically different but semantically the same. Each of these two events can be transformed into the same internal representation shown in Table 1. In this sense, the internal representation is independent of surface representation of an event.

[Title: A customer purchases a train ticket of reservation seat]
[Viewpoints: Staff, customer]
1. A staff asks a customer about leaving station and destination as customer's request.
2. He sends the customer's request to reservation center via private line.
3. He retrieves available trains with the request.
4. He informs the customer of a list of available trains.
5. The customer selects a train that he/she will get.
6. The staff retrieves available seats of the train.
7. He shows a list of available seats of the train.
8. The customer selects a seat of the train.
9. **If** (there exists a seat selected by the customer) **then** the staff reserves the seat with the terminal.
10. He gets a permission to issue a ticket of the seat from the center.
11. The customer paid for the ticket by cash.
12. **If** (there exist change) **then** the staff gives change to the customer.
13. He gives the ticket to the customer.

Fig. 1 Scenario example (payment with cash).

Table 1 Internal representation of the 2nd event.

Concept: Dataflow

source	goal	object	instrument
staff	reservation center	customer's request	private line

3. Generation of Alternative Scenarios

When a customer buys a ticket, there exist several alternatives of payment, such as pay with cash, credit card, personal check, banking card, money order, and so on. When data is transmitted, there exist several alternatives, such as sending via e-mail, postal mail, FAX, FTP, and so on. These alternatives arise from the diversity of methods. In identification, there exist several ID, such as company ID card, driver's license card, and passport. In this case, alternatives arise from data variations. As for the first case, the diversity of payment method causes the alternatives. As for the second case, the diversity of sending method causes the alternatives. As for the third case, the diversity of data for identification causes the alternatives. These alternatives appear in a certain case of the case structure of a concept. For example, the diversity of sending method appears in the instrument case of the cases structure of data flow concept. In case of payment with cash, there exist alternatives, such as (1) credit card, (2) personal check, and (3) prepaid card.

We provide users with such alternatives using a database whose contents are (a) possible methods including an ordinary method and its alternative methods for a certain

case of a specified concept as plans and (b) each of event sequences for these methods as scenario templates. We call this database "alternative scenario DB." For example, possible payment methods are (1) credit card, (2) personal check, (3) prepaid card, and (4) cash in the instrument case of the concept, "payment." For each method, corresponding event sequence is stored. For example, an event sequence of the payment with credit card is shown in Fig. 2.

We assume that alternative scenario DB is already given, but describe how to construct it briefly. Alternative scenario DB consists of (1) plans and (2) scenario templates of the plans. In order to clarify plans including normal plan and alternative plans, first of all, we have to describe a normal scenario and corresponding alternative scenarios for a certain system.

An alternative scenario is a scenario that achieves the same goal of the normal scenario with different events, so the difference between an alternative scenario and the normal scenario will be (a) events whose one of the cases [6] are different each other between these two scenario, while actions are same and (b) deleted events from the perspective of the normal scenario and (c) added events from the perspective of the normal scenario.

As for (a) events whose one of the cases are different each other, say, when payment with cash in the instrument case is an event of normal scenario, payment with credit card and payment with check in the instrument cases may be alternative plans of the payment. In such a case, (1) the action concept, (2) its case name that distinguish the normal scenario from alternative scenarios, and (3) nouns assigned to the case as methods of the normal scenario and alternative scenarios are stored in the alternative scenario DB as plans. Figure 5 (b) shows the alternative plans of the payment when the payment with cash is a normal behavior.

As for (b) deleted events, they are events related to the payment with cash in the above example. As for (c) added events, they are three event sequences that are related to the payment with credit card, the payment with check, and payment with prepaid card, respectively. In these event sequences, some nouns are variable, but the others are not. For example, actor of the payment is variable, but credit card is not variable when he pays with credit card.

By DB constructor's identifying the variable nouns, these nouns will be replaced by the cases [6]. Figure 2 shows event sequences of the payment with credit card whose variable nouns are replaced by the cases. We call such event sequence scenario template. Scenario templates for both normal and alternative behaviors and corresponding methods are stored into the DB, too.

When scenario templates of payment with credit card, payment with cash, payment with check, and payment with prepaid card are stored in the DB, any possible alternative scenarios can be generated, if one of the methods is given as a normal plan. Then user can select one or more plans from the possible alternatives. Suppose a normal scenario of the payment with credit card and payment with check as an alternative plan. After the scenario template of the payment with credit card can be deleted from the normal scenario and the template of the payment with check can be added to, then concrete nouns will be assigned to the cases, we can get an alternative scenario of the payment with check.

As described above, users first specify a normal scenario, then possible alternatives are provided to the users. By users' selecting alternatives, alternative event sequence will be generated. By replacing the original event sequence with the alternative event sequence and by assigning concrete nouns to the cases, an alternative scenario will be automatically generated.

3.1 Generation Method of Alternative Scenarios

Our generation method of alternative scenarios is shown as follows. We assume that a normal scenario is written with our scenario language in advance as shown in step 0.

0: Scenario writer describes a normal scenario with our scenario language.
1: The normal scenario is transformed into internal representation. In this step each events is transformed into internal representation based on requirements frame. If there is an event whose concept and whose specified case is same ones stored as a plan in the alternative scenario DB. For example, there exists an event whose concept is payment and credit card is specified in the instrument case, we can detect that this event has alternatives of the payment methods.
2: Possible alternative methods are automatically generated and provided to the scenario writer. He/she selects appropriate alternatives. The describer can select one or more alternatives, or no alternatives.
3: Scenario templates can be derived from alternative scenario DB in accordance with the selected alternatives. There exist several lacks of cases in the scenario template, but the lacked cases are automatically compensated using the internal representation of the event. Details of compensation are in ([11]). Some events will be automatically deleted in the normal scenario, because these events depend on the normal behavior, but do not depend on the alternative behavior. Such events to be deleted are also stored in the alternative scenario DB.
4: Alternative scenarios are provided to the scenario writer. He/she can revise or customize them.

3.2 Example of Generating Alternative Scenario

The above 5 steps are illustrated with the example shown in Fig. 1. In the step1, two events are selected as alternative events. The 2nd event, " He sends the customer's request to reservation center via private line" can be transformed into internal representation shown in Table 1. Since the concept of the event is data flow, and since its instrument case is "private line," there exist several alternative events. The 11th event, "the customer paid for the ticket by cash." can

Table 2 Internal representation of the 11th event.

Concept: Payment

agent	object	instrument	goal
customer	ticket	cash	staff

[Title: Payment with a credit card]
[Viewpoints: (Agent), (Goal)]
1.(Agent) passes a credit card to (Goal).
2.(Goal) enters the credit card and amount of payment with a terminal.
3.(Goal) confirms that the card is authenticated.
4.(Goal) gets receipt and bill via terminal.
5.(Agent) gets the receipt and bill from (Goal).
6.(Agent) autographs the bill.
7.(Agent) passes the bill to (Goal).
8.(Goal) passes both the card and the receipt to (Agent).

Fig. 2 Scenario template of the payment with credit card.

be transformed into an internal representation shown as Table 2. There is no noun for the goal case in this event, but analyzer compensates with a noun, "staff" for the goal case object.

Since the concept of the event is payment, there exist several alternatives for this event. In the step 2, alternatives are shown with alternative scenario DB. In case of sending data via private line, there exist alternatives, such as (1) public line, (2) FAX, (3) e-mail, (4) postal mail, and (5) FTP. A describer can select one or more alternatives. If he/she cannot find any appropriate alternatives, he/she may not select any alternatives. Here, we assume that no alternatives are selected.

In case of payment with cash, there exist alternatives, such as (1) credit card, (2) personal check, (3) banking card, and (4) money order. Here, we assume credit card is selected as alternative payment.

In the step 3, a scenario template for the payment with credit card is derived from the alternative scenario DB. This template is shown in Fig. 2. In this step, the 11th event and the 12th event in the normal scenario is deleted, because these two events are registered as a scenario template of the payment with cash.

In this template, the goal case and the agent case are not specified. Since the goal case of the 11th event in the normal scenario and the agent case of the event are "staff" and "customer" respectively, both the goal case and the agent case in the template will be "staff" and "customer" respectively. By compensating with these two nouns, the scenario becomes as follows.

In the step 4, alternative scenario shown in Fig. 4 is provided to the scenario writer. The 11th event of normal scenario in Fig. 1 is expanded with the compensated scenario template of Fig. 3.

Last, the scenario writer checks the alternative scenario and revises it if necessary.

[Title: Payment with a credit card]
[Viewpoints: customer, staff]
1.Customer passes a credit card to staff.
2.Staff enters the credit card and amount of payment with a terminal.
3.Staff confirms that the card is authenticated.
4.Staff gets the receipt and bill via terminal.
5.Customer gets the receipt and bill from staff.
6.Customer autographs the bill.
7.Customer passes the bill to staff.
8.Staff passes both the card and the receipt to customer.

Fig. 3 Compensated scenario template of the payment with credit card.

[Title: A customer purchases a train ticket of reservation seat]
[Viewpoints: Staff, customer]
1. A staff asks a customer about leaving station and destination as customer's request.
2. He sends the customer's request to reservation center via private line.
3. He retrieves available trains with the request.
4. He informs the customer of a list of available trains.
5. The customer selects a train that he/she will get.
6. The staff retrieves available seats of the train.
7. He shows a list of available seats of the train.
8. The customer selects a seat of the train.
9. If (there exists a seat selected by the customer) then the staff reserves the seat with the terminal.
10.He gets a permission to issue a ticket of the seat from the center.
11.Customer passes a credit card to staff.
12.Staff enters the credit card and amount of payment with a terminal.
13.Staff confirms that the card is authenticated.
14.Staff gets the receipt and bill via terminal.
15.Customer gets the receipt and bill from staff.
16.Customer autographs the bill.
17.Customer passes the bill to staff.
18.Staff passes both the card and the receipt to customer.
19.He gives the ticket to the customer.

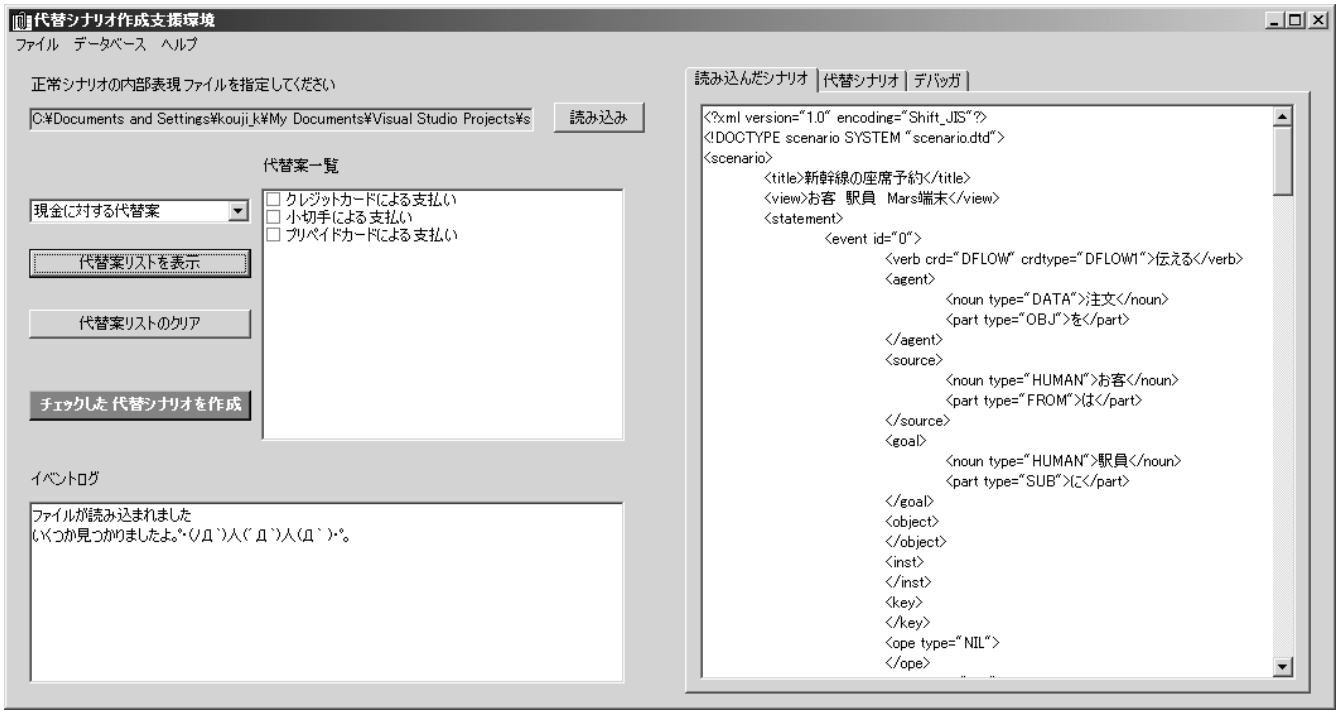
Fig. 4 Alternative scenario for the normal scenario in Fig. 1.

3.3 Supporting Tool for Making Alternative Scenario

We have developed a supporting tool based on our method with VisulaBasic.NET 2003. Figure 5 (a) shows display image of the tool. The left side of Fig. 5 (a) shows alternatives of payment methods. Figure 5 (b) shows the list of alternatives of payment in English. The right side of Fig. 5 (a) shows a normal scenario. Figure 5 (c) shows a normal scenario in English. We use a transformer from scenario with our scenario language to scenario with XML format. Our system accepts a scenario with XML format.

In the left side of Fig. 6 (a), user selected the first payment method. This method is payment with credit card. The right side of Fig. 6 (a) shows an alternative event sequence generated by compensating scenario template with the payment method using credit card.

Figure 6 (b) shows the alternative event sequence in English. Since Fig. 3 shows alternative events for payment with credit card, Fig. 3 and Fig. 6 (b) are mostly same. The differ-



(a)

☐ Payment with credit card
☐ Payment with check
☐ Payment with prepaid card

(b)

1.A customer sends his order to a staff.
2.The staff sends the order to system.
3.The system displays available seats of the train.
4.The staff selects a seat and enters the seat id to the system.
5.A ticket of the seat is issued.
6.The staff notifies the customer of the total amount fee.
7.The customer pays the fee with cash.
8.The staff hands out the ticket of the reserved seat.

(c)

Fig. 5 (a) Original scenario and list of alternatives. (b) List of alternative methods of the payment. (c) Original normal scenario.

ence between them is system's viewpoint is included in the scenario or not. In Fig. 6 (b), the 3rd and the 4th events that state system's behavior are included. By replacing the 7th event of Fig. 5 (c) with the events in Fig. 6 (b), an alternative scenario can be automatically generated.

4. Evaluation

In order to evaluate our method, we adopt a normal scenario and its alternative scenarios of a real project [4] and discuss whether these alternative scenarios can be generated from the normal scenario with our method. A normal scenario of purchasing goods has 8 exceptional scenarios and 5 alternative scenarios [4]. We focus on the 5 alternative scenarios. Both the five normal plans and corresponding alternative plans are shown in Table 3.

As for the first alternative plan, alternative scenario can be generated with our method by providing a verb "describe" and its object case in the scenario language and by regarding the diversity of the object case of this verb as alternatives.

Actually, "Clerk describes all of the items" in normal scenario can be transformed into a representation shown in Table 4. Table 5 shows alternative plans for the diversity of the object case of the concept, "write."

Figure 7 (a), (b) and (c) show scenario templates for the corresponding plans. If the second plan in Table 5 is selected, the above event will be replaced by the scenario template shown in Fig. 7 (b) and then clerk which is the agent case in Table 4 will be assigned in the (Agent) case of Fig. 7 (b). As a result, alternative event sequence can be generated as shown in Fig. 7 (d).

As for the second alternative plan, alternative scenario

(a)

[Title: Payment with credit card]
[Viewpoints: customer, staff, system]
1.The customer passes his credit card to the staff.
2.The staff enters the credit card and amount of payment with a terminal.
3.The system authenticates the card.
4.The system issues a bill and receipt.
5.The staff gets the receipt and bill via terminal.
6.The customer gets the receipt and bill from the staff.
7.The customer autographs the bill.
8.The customer passes the bill to staff.
9.The staff passes both the card and the receipt to customer.

(b)

Fig. 6 (a) Compensated scenario template of the selected payment method. (b) Alternative events for payment with credit card.

Table 3 Normal plans and thier alternative plans in a real project.

normal plan	alternative plan
Clerk fills all of the items	Clerk fills a part of items. Purchasing department fills unfilled items
Manager approves the request	Superior person except for manager approves the request
Purchasing department checks the stock and orders goods	Purchasing department checks the stock, delivers goods in stock and orders the remains of the goods
Purchasing department orders goods to a vendor	Purchasing department orders goods to multiple vendors
Receiver checks the goods. He records the delivery of the goods	Receiver checks a part of the goods. He records the delivery of a part of the goods

can be generated by providing a verb "approve" and its agent case in the scenario language and by regarding the diversity of the agent case as alternatives.

Table 4 Internal representation of the 1st normal plan.

Concept: write, **Verbs:** write, describe, fill

agent	object	instrument
clerk	all of items	-

Table 5 Alternative plans.

Concept: write, **Case:** object

plan1	plan2	plan3
all of items	a part of items	none

In case of the third plan, we cannot generate this alternative scenario. However we can write this normal event sequence and its alternative plan as an integrated normal event sequence as shown in Fig. 8.

As for the fourth plan, alternative scenario can be generated by providing a verb "order" and its goal case in the scenario language and regarding the diversity of the goal

[Title: Fill items by Agent]
[Viewpoints: (Agent)]
1.(Agent) describes all of the items.

(a)

[Title: Fill items partly]
[Viewpoints: (Agent)]
1.(Agent) describes a part of items.
2. Purchase department describes unfilled items.

(b)

[Title: Fill items by purchase department]
[Viewpoints: (Agent)]
1.(Agent) describes none.
2. Purchase department describes all of the items.

(c)

[Title: Fill items partly]
[Viewpoints: clerk]
1. Clerk describes a part of items.
2. Purchase department describes unfilled items.

(d)

Fig. 7 (a) Scenario template of the 1st plan. (b) Scenario template of the 2nd plan. (c) Scenario template of the 3rd plan. (d) Generated alternative events of the 2nd plan.

[Title: Deliver goods in stock and order the goods]
[Viewpoints: Purchasing department]
1.Purchasing department checks the stock.
2.**If** (there exist the goods in stock) **then**
he delivers goods in stock. he orders the remains of the goods.
else he orders the goods.

Fig. 8 Integrated plans into a normal event sequence.

case as alternatives.

As for the last plan, alternative scenario can be generated by providing a verb "check" and its object case in the scenario language and regarding the diversity of the object case as alternatives.

In the above discussion, we can generate 4 alternative scenarios from a normal scenario of a real project. Just one scenario cannot be generated, but the alternative plan of the not generated scenario can be described as a normal scenario. We are sure that our method is useful to generate alternative scenarios in real scenario-based software developments.

As another evaluation, the following experiment was performed. We adopted a scenario based software project of developing a bill management system of an insurance company. In this project, analysts wrote not only a normal scenario, but also other scenarios, that is, alternative scenarios and exceptional scenarios. We applied our method to the normal scenarios and got alternative scenarios. Then we compared alternative scenarios that developed at the projects with automatically generated scenarios. Since original normal scenarios are written with natural language, we rewrote the normal scenarios with our scenario language prior to the experiments.

Table 6 Result of alternative scenarios of the project.

	Total	Same	New	Not generated
Original	4	3	-	1
Method	5	3	2	-

In this project, one normal scenario, 4 alternative scenarios, and 5 exceptional scenarios are specified. By applying our method, we could get 5 alternative scenarios. By comparing original alternative scenarios with generated scenarios, we found that 3 scenarios are same respectively, 2 scenarios are newly generated and effective, and 1 scenario is not generated. Table 6 shows the above result.

The not generated scenario is regarded as an alternative scenario at the project, but it should be categorized into a normal scenario, because this scenario specifies normal behavior of the bill management system and has a different purpose from that of the corresponding normal scenario. Actually, an event "A clerk registers a damage in the system." is in the normal scenario, while an event "A clerk changes registered damage into different one in the system" is alternative. In this case, the normal scenario treats newly registration, but the alternative one treats modification of registered damage. Since the purposes of these two scenarios are quite different, these scenarios should be written as two normal scenarios.

5. Related Works

Ben Achour proposed guidance for correcting scenarios, based on a set of rules [2]. These rules aim at the clarification, completion and conceptualization of scenarios, and help the scenario author to improve the scenarios until an acceptable level in terms of the scenario models. Ben Achour's rules can only check whether the scenarios are well written according to the scenario models, while we propose a generation method of alternative scenarios from a normal scenario.

Derek Cramp claimed the importance of alternative scenarios. He proposed a model to create alternative scenarios [5]. However, his model strongly depends on a specific domain. Our method for generating alternative scenarios can be applied to several different domains by switching alternative scenario DB.

Ian Alexander proposed a scenario-driven search method to find more exceptions [3]. In his approach, a model answer was prepared with knowledge of all exception cases identified by stakeholders. For each event, related exceptions are listed as a model answer. His model answer, however, strongly depends on a specific domain.

Neil Maiden et al. proposed classes of exceptions for use cases [9]. These classes are generic exceptions, permutations exceptions, permutation options, and problem exceptions. With these classes, alternative courses are generated. For communication actions, 5 problem exceptions are prepared, that is, human agents, machine agents, human-machine interactions, human-human communication, and machine-machine communication. They proposed a gener-

ation method of alternative paths for each normal sequence from exception types for events and generic requirements with abnormal patterns [17]. We focus on generation of alternative scenarios by providing more precise model based on both case structure of actions and actor types.

In the author's previous work [11], we proposed to build software requirements from textual requirements in Japanese, based on a typology of concepts very similar to the semantic roles of the case grammar [6]. Another related work is Ben Achour's use of case grammar in scenario analysis [1], [2]. Ben Achour focuses on how textual scenarios could be integrated into different existing methods, and proposes guidance for writing scenarios. He provides style and content guidelines referring to conceptual and linguistic model of scenarios, based on the case grammar. These works demonstrate that the case grammar is suitable to the semantic characterization of any design models as well as the semantic characterization of any natural language sentence.

6. Conclusion

The authors have proposed a generating method of alternative scenario. We provide plans of normal and alternative behaviors and their templates with an alternative scenario DB. By selecting plans, a template corresponding to the selected plan is provided, by deleting events of normal behavior , by adding events of alternative behavior and by compensating the templates, we can automatically get alternative scenarios. Our method contributes to lessen developers' work of making several scenarios and to improve the quality of scenarios.

The proposed method was demonstrated by the example and was evaluated. The evaluation results show that our method is valid in software development.

The quality of generated alternative scenario depends on alternative scenario DB. An alternative scenario DB fully depends on a certain problem domain and the quality of the DB depends on ability and knowledge of domain experts that construct the DB. So, we have a plan to automatically derive alternative scenario DB from software documents of high quality in order to improve the quality of the DB. We will evaluate and improve our method and system by applying them to several scenario-based software system developments. These are left as future works.

Acknowledgement

We thank to Mr. Taishi Yamamoto (currently at NTT Data co.), Mr. Hiroki Shudo (currently at TIS co.), Dr. Hiroya Itoga at Ritsumeikan University, and other members of Software Engineering laboratory, Department of Computer Science, Ritsumeikan University, Japan.

References

[1] C.B. Achour, "Linguistic instruments for the integration of scenarios in requirements engineering," Proc. Third International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'97), pp.93–106, Barcelona, Spain, 1997.

[2] C.B. Achour, "Guiding scenario authoring," Proc. Eight European-Japanese Conference on Information Modeling and Knowledge Bases, pp.181–200, Vamala, Finland, May 1998.

[3] I. Alexander, "Scenario-driven search finds more exceptions," Proc. 11th International Workshop on Database and Expert Systems Applications, pp.991–994, London, U.K., Sept. 2000.

[4] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, USA., 2001.

[5] D.G. Cramp and E.R. Carson, "Assessing health policy strategies: A model-based approach to decision support," Proc. International Conference on System, Man and Cybernetics, vol.3, pp.69–73, 1995.

[6] C.J. Fillmore, "The Case for case," in Universals in Linguistic Theory, ed. E. Bach and R. Harms, Holt, Rinehart and Winston, Chicago, 1968.

[7] M. Jackson, "Problems and requirements," Proc. 2nd International Symposium on Requirements Engineering (RE'95), pp.2–8, York, England, March 1995.

[8] J.C.S.P. Leite, G. Rossi, F. Balaguer, V. Maiorana, G. Kaplan, G. Hadad, and A. Oloveros, "Enhancing a requirements baseline with scenarios," Proc. 3rd IEEE International Symposium on Requirements Engineering (RE'97), pp.44–53, Annapolis, U.S.A., Jan. 1997.

[9] N.A.M. Maiden, M.K. Manning, and M. Ryan, "CREWS-SAVRE: Systematic scenarios generation and use," Proc. 3rd International Conference on Requirements Engineering (ICRE'98), pp.148–155, Colorado Springs, U.S.A., April 1998.

[10] N.A.M. Maiden and M. Hare, "Problem domain categories in requirements engineering," Int. J. Human-Computer Studies, vol.49, pp.281–304, 1998.

[11] A. Ohnishi, "Software requirements specification database based on requirements frame model," Proc. IEEE second International Conference on Requirements Engineering (ICRE'96), pp.221–228, Colorado Springs, U.S.A., April 1996.

[12] A. Ohnishi and C. Potts, "Grounding scenarios in frame-based action semantics," Proc. 7th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'01), pp.177–182, Interlaken, Switzerland, June 2001.

[13] A. Ohnishi, H. Zhang, and H. Fujimoto, "Transformation and integration method of scenarios," Proc. 26th Annual International Computer Software & Applications Conference (COMPSAC02), pp.224–229, Oxford, England, 2002.

[14] A. Ohnishi, "A generation method of exceptional scenarios from a normal scenario," IEICE Trans. Inf. & Syst., vol.E91-D, no.4, pp.881–887, April 2008.

[15] Railway Information System Co., Ltd., JR System, http://www.jrs.co.jp/keiki/en/index_main.html, 2001.

[16] M. Ridao, J. Doorn, and J.C.S.P. Leite, "Domain independent regularities in scenarios," Proc. Fifth IEEE International Symposium on Requirements Engineering (RE'01), pp.120–127, Toronto, Canada, Aug. 2001.

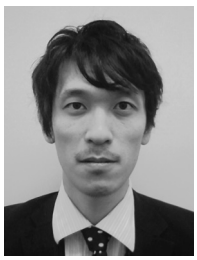
[17] A.G. Sutcliffe and M. Ryan, "Experience with SCRAM, a scenario requirements analysis method," Proc. 3rd International Conference on Requirements Engineering (ICRE'98), pp.164–171, Colorado Springs, U.S.A., April 1998.

[18] A.G. Sutcliffe, N.A.M. Maiden, S. Minocha, and D. Manuel, "Supporting scenario-based requirements engineering," IEEE Trans. Softw. Eng., vol.24, no.12, pp.1072–1088, 1998.

[19] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer, "Scenarios in system development: Current practice," IEEE Software, pp.34–45, March 1998.

[20] H. Zhang and A. Ohnishi, "Transformation method of scenarios from different viewpoints," Proc. 11th Asia Pacific Software Engineering Conference (APSEC2004), pp.492–501, Busan, Korea, Nov.-Dec., 2004.

Atsushi Ohnishi received B. of Engineering, M. of Engineering, and Dr. of Engineering degrees from Kyoto University in 1979, 1981, and 1988, respectively. He was a Research Associate and an Associate Professor of Kyoto University from 1983 to 1994. Since 1994 he has been a Professor at Dept. Computer Science, Ritsumeikan University. He was a visiting scientist at UC Santa Barbara from 1990 to 1991 and also a visiting scientist at Georgia Institute of Technology in 2000. His current research interests include requirements engineering, object oriented analysis, and software design techniques. Dr. Ohnishi is a member of IEEE Computer Society, ACM, Information Processing Society (IPS) Japan, and Japan Society for Software Science and Technology (JSSST).



Koji Kitamoto received B. of Engineering, M. of enginering from Ritsumeikan University in 2003 and 2005, respectivly. He is now at NTT Data Co.