A SCENARIO GENERATION METHOD USING A DIFFERENTIAL SCENARIO

Masayuki Makino, Atsushi Ohnishi
Department of Computer Science, Ristumeikan University, Kusatsu, Shiga 525-8577, Japan

Keywords: Scenario analysis, Scenario generation, Requirements elicitation, Requirements definition.

Abstract: A generation method of scenarios using differential information between normal scenarios is presented. Behaviours of normal scenarios belonging to the same problem domain are quite similar. We derive the differential information between them and apply the information to generate new scenarios. Our method will be illustrated with an example.

1 INTRODUCTION

Scenarios are important in software development, particularly in requirements engineering, by providing concrete system description (Weidenhaupt et al., 1998). Especially, scenarios are useful in defining system behaviors by system developers and validating the requirements by customers. In scenario-based software development, incorrect scenarios will have a negative impact on the overall system development process. However scenarios are usually informal and it is difficult to verify the correctness of scenarios.

The authors have developed a scenario language for describing scenarios in which simple action traces are embellished to include typed frames based on a simple case grammar of actions and for describing the sequence among events (Zhang et al., 2004). Since this language is a controlled language, the vagueness of the scenario written with 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 (Ohnishi, 1996).

Scenarios can be classified into 1) normal scenario, 2) alternative scenario, and 3) 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. However it is difficult to hit upon most of alternative scenarios and exceptional scenarios, whereas it is easy to think of normal scenarios.

2 SCENARIO LANGUAGE

2.1 Outline

The scenario language named SLAF has already been introduced (Zhang, 2004, Toyama 2005). 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. We assume that each event has just one verb, and that each verb has its own case structure (Fillmore, 1968). 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.

We provide requirements frames (Ohnishi, 1996) in which verbs and their 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, we can detect both the lack of cases and the illegal usage of noun types.

We assume four kinds of time sequences among events: 1) sequential, 2) selective, 3) iterative, and 4) parallel. Actually most events are sequential events.

2.2 Scenario Example

We consider a scenario of train ticket reservation of a railway company. 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 with our scenario language based on videoized behaviors of both a user and a staff at a service center of a railway company (Railway Information System, 2001).

[Title: A customer purchases a train ticket of reservation seat]
[Viewpoints: Staff, customer]
1. A staff asks a customer about leaving station, destination and traveling date as customer's request.
2. The staff 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. The staff gets a permission to issue a ticket of the seat from the center.
11. The customer pays for the ticket by cash.
12. The staff gives the ticket to the customer.
13. **If** (changes exist) **then** the staff gives changes.

Figure 1: Scenario example.

A title of the scenario is given at the first line of the scenario in Fig.1. Viewpoints of the scenario are specified at the third line. In this paper, viewpoints mean active objects such as human and system appearing in the scenario. There exist two viewpoints, namely staff and customer. The order of the specified viewpoints means the priority.

In this scenario, almost all events are sequential, except for just two selective events (the 9th event and the 13th event). Selection can be expressed with if-then syntax like program languages. Actually, event number is for reader's convenience and not necessary.

2.3 Analysis of Events

Each of events is automatically transformed into internal representation. For example, the 2nd event "The staff 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 to say, 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 and "the staff" corresponds to the source case.

Table 1: Internal representation of the 2nd event.

Concept: Data Flow

source	goal	object	instrument
Staff	Reservation center	Customer's request	Private line

The internal representation is independent of surface representation of the event. Suppose other representations of 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 same as the 2nd event. These two events can be automatically transformed into the same internal representations.

3 DIFFERENTIAL SCENARIO

Systems belonging to the same domain similarly behave each other. In other words, normal scenarios belonging to the same domain resemble each other. Since our scenario language provides limited vocabulary and limited grammar, the abstraction level of any scenarios becomes almost same.

For one system, there exist several normal scenarios. In case of ticket reservation, reservation can be written as a normal scenario and cancellation can be written as another normal scenario. To make a differential scenario, we select two normal scenarios of two different systems. Each of the two scenarios represents almost same behavior, such as reservation of a ticket.

The differential scenario consists of 1) a list of corresponding words, 2) deleted events which appear in one scenario (say, scenario A) and do not appear in the other (say, scenario B), and 3) added events which do not appear in scenario A and appear in scenario B.

Fig. 2 shows a scenario of flight ticket reservation using credit card. By comparing two scenarios, we can get the differential scenario. The first four events of the scenario in Fig. 1 can be transformed as shown in Table 2. In fact, data flow concept has four cases, that is, source, goal, object, and instrument cases as shown in Table 1, but the instrument cases are omitted in Table 2 and 3 for the space limitation.

Since the sequence of the concepts of the first four events of the scenario in Fig. 1 is same as that of the scenario in Fig. 2, we can regard these events are corresponding each other. Then, the difference between cases of the corresponding events will be checked. In the case of the first event of the two scenarios, object cases of the events are different each other.

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

Figure 2: Normal scenario of flight reservation.

Table 2: The internal representation of the first four events of the scenario in Fig.1.

concept	**agent/ source**	**goal**	**objects**
query	staff	customer	leaving station, destination, traveling date
data flow	staff	reservation center	customer's request
retrieve	staff	available trains	request
data flow	staff	customer	list of available trains

Table 3: The internal representation of the first four events of the scenario in Fig.2.

concept	**agent/ source**	**goal**	**objects**
query	staff	customer	leaving airport, destination, departure date
data flow	staff	reservation center	customer's request
retrieve	staff	available flights	request
data flow	staff	customer	list of available flights

The difference between corresponding events will be stored as corresponding words in Table 4. The 12th and the 13th events of Fig. 2 are not-corresponding events and will be stored as added events, while the 12th event of Fig. 1 and the 14th event of Fig. 2 are corresponding events. The 13th event of Fig. 1 is a not-corresponding event and will be stored as a deleted event.

Finally, we can get the differential scenario between train ticket reservation and flight ticket reservation shown in Table 4, 5, and 6.

Table 4: A list of corresponding words between scenarios of Figure 1 and 2.

Fig.1	**Fig.2**	**Fig.1**	**Fig.2**
station	airport	trains	flights
traveling	departure	cash	credit card
train	flight		

Table 5: Added events.

The staff checks the credit card.
The staff charges the ticket fee to the card

Table 6: Deleted events.

If (changes exist) then the staff gives changes.

4 SCENARIO GENERATION

Once differential scenario between system A and B given, we can apply it to another scenario of system A and get a new scenario of system B by changing corresponding words and by deleting or adding not-corresponding events.

Fig. 3 shows an exceptional scenario of ticket reservation. In this scenario, the customer cannot get any available trains with respect to the first request. So, the customer changes the traveling date and then gets available trains.

By applying the differential scenario in Table 4, 5, and 6, we can get a new exceptional scenario of flight ticket reservation as shown in Fig. 4.

[Title: A customer purchases a train ticket of reservation seat, but cannot find available train, so he gives the second choice.]
[Viewpoints: Staff, customer]
1. A staff asks a customer about leaving station, destination and traveling date as customer's request.
2. The staff sends the customer's request to reservation center via private line.
3. He cannot find available trains with the request.
4. He informs the customer of no available trains and asks the customer about another traveling date.
5. The customer gives another traveling date.
6. The staff sends the customer's request to reservation center via private line.
7. He retrieves available trains with the new request.
8. He informs the customer of a list of available trains.
9. The customer selects a train that he/she will get.
10. The staff retrieves available seats of the train.
11. He shows a list of available seats of the train.
12. The customer selects a seat of the train.
13. ...

Figure 3: An exceptional scenario.

[Title: A customer purchases a flight ticket of reservation seat, but cannot find available flight, so he gives the second choice.]
[Viewpoints: Staff, customer]
1. A staff asks a customer about leaving airport, destination and departure date as customer's request.
2. The staff sends the customer's request to reservation center via private line.
3. He cannot find available flights with the request.
4. He informs the customer of no available flights and asks the customer about another departure date.
5. The customer gives another departure date.
6. The staff sends the customer's request to reservation center via private line.
7. He retrieves available flights with the new request.
8. He informs the customer of a list of available flights.
9. The customer selects a flight that he/she will get.
10. The staff retrieves available seats of the flight.
11. He shows a list of available seats of the flight.
12. The customer selects a seat of the flight.
13. ...

Figure 4: Generated a new exceptional scenario.

5 RELATED WORKS

Ben Achour proposed guidance for correcting scenarios, based on a set of rules (Achour, 1998). 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. We propose generation methods of exceptional scenarios and alternative scenarios from a normal scenario.

Derek Cramp claimed the importance of alternative scenarios. He proposed a model to create alternative scenarios (Cramp et al., 1995). However, his model strongly depends on a specific domain.

Ian Alexander proposed a scenario-driven search method to find more exceptions (Alexander, 2000). 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 (Maiden et al, 1998). These classes are generic exceptions, permutations exceptions, permutation options, and problem exceptions. With these classes, alternative courses are generated. They proposed a generation method of alternative paths for each normal sequence from exception types for events and generic requirements with abnormal patterns (Sutcliff et al., 1998).

Our approach for generating scenarios with a differential scenario is independent of problem domains.

6 CONCLUSION

We have developed a generation method of scenarios using a differential scenario. Because of the space limitation, we showed just one example, but we confirmed that alternative scenarios and different normal scenarios can be generated with our method.

We have to validate the ideas more thoroughly by applying to several different problem domains. We have been developing a prototype system based on the method. The evaluation of our method through the use of the prototype system is another future work.

REFERENCES

Achour, C. B., 1998: Guiding Scenario Authoring, Proc. of the Eight European-Japanese Conference on Information Modeling and Knowledge Bases, pp.181-200.

Alexander, I., 2000: Scenario-Driven Search Finds More Exceptions, Proc. 11th International Workshop on Database and Expert Systems Applications, pp.991-994.

Cramp, D.G., Carson E.R., 1995: Assessing Health Policy Strategies: A Model-Based Approach to Decision Support, Proc. International Conference on System, Man and Cybernetics, Vol.3, pp.69-73.

Fillmore, C.J., 1968: The Case for Case, in Universals in Linguistic Theory, Holt, Rinehart and Winston.

Maiden, N.A.M., Manning' M.K., Ryan M., 1998: CREWS-SAVRE: Systematic Scenarios Generation and Use, Proc. IEEE 3rd ICRE'98, pp.148-155.

Ohnishi, A., 1996: Software Requirements Specification Database on Requirements Frame Model, Proc. IEEE 2nd ICRE'96, pp.221-228.

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

Sutcliffe, A. G., Maiden, N. A. M., Minocha S., Manuel D., 1998: Supporting Scenario-Based Requirements Engineering, IEEE Trans. Software Engineering, Vol.24, No.12, pp.1072-1088.

Toyama, T., Ohnishi, A., 2005: Rule-based Verification of Scenarios with Pre-conditions and Post-conditions, Proc. 13th IEEE RE2005, pp. 319-328.

Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P., 1998: Scenarios in System Development: Current Practice, IEEE Software, March, pp.34-45.

Zhang H., Ohnishi, A., 2004: Transformation between Scenarios from Different Viewpoints, IEICE Trans. Information and Systems, Vol.E87-D, No.4, pp.801-810.