Scenario Generation with Differential Scenario

ATSUSHI OHNISHI
Department of Computer Science
Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577
JAPAN
http://www.selab.cs.ritsumei.ac.jp/~ohnishi

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 alternative/exceptional scenarios. Our method will be illustrated with examples. This paper describes (1) a language for describing scenarios in which simple action traces are embellished to include typed frames based on a simple case grammar of actions, (2) introduction of the differential scenario, and (3) examples of scenario generation using the differential scenario.

Key-Words: - Scenario analysis, Scenario generation, Requirements elicitation, Requirements definition
540-238.pdf

1 Introduction

Scenarios are important in software development [4], particularly in requirements engineering, by providing concrete system description [15, 17]. 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 them. 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 author has developed a scenario language named SLAF 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 [18, 19]. Since this language is a controlled language, the vagueness of the scenario written with SLAF language can be reduced. Furthermore, a scenario with SLAF 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]. As a result, the scenario with SLAF can avoid the errors typed 1 previously mentioned.

Scenarios can be classified into (1) normal scenario, (2) alternative scenario, and (3) exceptional scenario. A normal one represents the normal and typical behavior of the target system, while an alternative one represents normal but untypical behavior of the system and an exceptional one 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 alternative scenarios and exceptional scenarios, whereas it is easy to think of normal scenarios.

This paper focuses on how to generate alternative/exceptional scenarios from normal scenarios of two different systems belonging to the same problem domain. We adopt SLAF language for writing scenarios, because SLAF is a control language and it is easy to analyze scenarios written with SLAF.

2 Scenario Language

2.1 Outline

The SLAF language has already been introduced [18, 19]. 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 [6]. 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].

We provide requirements frames 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 [11].

We assume four kinds of time sequences among events: 1) sequence, 2) selection, 3) iteration, and 4) parallelism. Actually most events are sequential events.

Our scenario language defines the semantic of verbs with their case structure. For example, data flow verb has source, goal, agent, and instrument cases.

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 [12].

A title of the scenario is given at the first line of the scenario in Figure1. Viewpoints of the scenario are specified at the third 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. In this example, the first prior object is staff, and the second is customer. In such a case, the prior object becomes the subject of an event.

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.

[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.

Table 1: Internal representation of the 2nd event.

Concept: Data Flow

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

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.

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 as shown in Table 1.

As related researches, we have developed scenario translation method between a scenario from a certain viewpoint and a scenario from different viewpoint [19], scenario integration method among scenarios from several viewpoints [18], and scenario verification method [16].

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.

Figure 2 shows a scenario of flight ticket reservation using credit card.

[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.

We compare the scenario of Figure 1 with the scenario of Figure 2 from top to bottom and regard events of same action concept are corresponding together. By comparing two scenarios, we can get the differential scenario.

The first four events of the scenario in Figure 1 can be transformed as shown in Table 2. In fact, data flow (dflow) 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 Figure 1 is same as that of the scenario in Figure 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.

The difference between corresponding events will be stored as corresponding words as shown in Table 4.

Table 2: The internal representation of the first four events of the scenario in Figure1.

Concept	**Agent/ source**	**goal**	**objects**
query	staff	customer	Leaving station, destination, traveling date
data flow	staff	Reservation center	Customer's request
retrieve	staff	Available flights	request
data flow	staff	customer	List of available trains

Table 3: The internal representation of the first four events of the scenario in Figure 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 12th and the 13th events of Figure 2 are not-corresponding events and will be stored as added events, while the 12th event of Figure 1 and the 14th event of Figure 2 are corresponding events. The 13th event of Figure 1 is a not-corresponding event and will be stored as a deleted event.

Table4: A list of corresponding words between scenarios of Figure 1 and 2.

station	airport
traveling	departure
train	flight
trains	flights
cash	credit card

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.

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

4 Scenario Generation with Differential Scenario

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.

4.1 Examples of generation

Figure 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, we can get a new exceptional scenario of flight ticket reservation as shown in Figure 4. In Figure 4, words of bold and italic font are exchanged according to Table 4. Since there are no events for payment with credit card, Table 5 and 6 are not used.

Figure 5 shows a scenario of cancellation of train ticket. This scenario is another normal scenario of train reservation system. After applying the differential scenario, we can get a new normal scenario of cancellation of flight ticket shown in Figure 6.

[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.

[Title: A customer cancels a train ticket of reservation seat.]
[Viewpoints: Staff, customer]
1. A staff is asked by a customer about cancellation of already reserved ticket.
2. The staff gets the customer's ticket and train information including seat id, traveling date, departure time and station, and arrival time and station.
3. The staff cancels the seat of the train.
4. The staff refunds for the ticket fee by cash while he charges the handling fee.

Figure 5: Scenario of cancellation of train ticket.

[Title: A customer cancels a flight ticket of reservation seat.]
[Viewpoints: Staff, customer]
1. A staff is asked by a customer about cancellation of already reserved ticket.
2. The staff gets the customer's ticket and ***flight*** information including seat id, ***departure*** date, departure time and ***airport***, and arrival time and ***airport***.
3. The staff cancels the seat of the ***flight***.
4. The staff refunds for the ticket fee by cash while he charges the handling fee.
5. ***The staff checks the credit card.***
6. ***The staff charges the ticket fee to the card.***

Figure 6: Generated scenario of cancellation of flight ticket.

The 5th event and the 6th event in Figure 6 are added, since these two events are listed in added events in Table 5. By contrast, deleted event in Table 6 is not applied, since this event does not exist in the scenario in Figure 5. Actually, added events of Figure 6 are not correct in this case. Added events and deleted events are valid if customer pays the ticket fee with not cash but credit card.

By investigating the automatically generated scenario, this scenario will be revised as shown in Figure7 by hand.

[Title: A customer cancels a flight ticket of reservation seat.]
[Viewpoints: Staff, customer]
1. A staff is asked by a customer about cancellation of already reserved ticket.
2. The staff gets the customer's ticket and flight information including seat id, departure date, departure time and airport, and arrival time and airport.
3. The staff cancels the seat of the flight.
4. ***The staff gets the credit card number.***
5. ***The staff refunds for the ticket fee by credit card while he charges the handling fee.***

Figure 7: Revised scenario of cancellation of flight ticket.

4.2 Revising the differential scenario

By comparing between two scenarios in Figure 3 and Figure 4, we found that there exist no problems in the differential scenario. However, by comparing between two scenarios in Figure 5 and Figure 7, we should revise the differential scenario.

Since there exist no contradictions in corresponding words shown in Table 2, the list of corresponding words will not be modified. By contrast, both added events in Table 5 and deleted events in Table 6 should be modified. We introduce conditions in the differential scenario. The revised tables are shown in Table 7 and Table 8.

Table 7: The revised added events.

event	condition
The staff checks the credit card.	Payment with credit card
The staff charges the ticket fee to the credit card	Payment with credit card
The staff gets the credit card number.	Refund to credit card
The staff refunds for the ticket fee by the credit card while he charges the handling fee	Refund to credit card

Table 8: The revised deleted event.

event	condition
If (changes exist) then the staff gives changes	Payment with credit card

By revising the differential scenario, the quality of automatically generated scenarios with our method will be improved.

5 Related Works

Ben Achour proposed guidance for correcting scenarios, based on a set of rules [1, 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. 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 [5]. However, his model strongly depends on a specific domain. 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 generation method of alternative paths for each normal sequence from exception types for events and generic requirements with abnormal patterns [15]. Our approach for generating scenarios with a differential scenario is independent of problem domains.

6 Conclusion

We have developed a frame base scenario language and a generation method of scenarios using differential scenario. With our method, we can get new scenarios of a certain problem domain using scenarios belonging to the same domain and differential scenario between two systems.

Through our examples, we found the scenario generation method will improve the quality of scenario. Especially the productivity of scenarios and the correctness of generated scenarios will be improved.

We have to validate the ideas more thoroughly by applying to several different problem domains, such as web-based sales ordering system. 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

[1] Achour, C. B.: Linguistic Instruments for the Integration of Scenarios in Requirements Engineering, Proc. of the Third Requirements Engineering: Foundation for Software Quality (REFSQ'97), 1997, pp. 93-106.

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

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

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

[5] Cramp, D.G., Carson, E.R.: Assessing Health Policy Strategies: A Model-Based Approach to Decision Support, Proc. International Conference on System, Man and Cybernetics, Vol.3, 1995, pp.69-7.

[6] Fillmore, C.J.: The Case for Case, in Universals in Linguistic Theory, Bach and Harms, Chicago, Eds Holt, Rinehart and Winston, 1968.

[7] Jackson, M.: Problems and requirements, Proc. 2nd IEEE International Symposium on Requirements Engineering (RE), IEEE Computer Soc., 1995, pp.2-8.

[8] Leite, J.C.S.P., et al.: Enhancing a Requirements Baseline with Scenarios, Proc. of the 3rd RE, 1997, pp.44-53.

[9] Maiden, N.A.M., Manning, M.K., Ryan M.: CREWS-SAVRE: Systematic Scenarios Generation and Use, Proc. IEEE 3rd International Conference on Requirements Engineering (ICRE), 1998, pp.148-155.

[10] Maiden, N.A.M., Hare, M.: Problem Domain Categories in Requirements Engineering, International Journal of Human-Computer Studies, 49, 1998, pp.281-304.

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

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

[13] Ridao, M., Doorn, J., Leite, J.C.S.P.: Domain Independent Regularities in Scenarios, Proc. of the Fifth RE, 2001, pp.120-127.

[14] Sutcliffe, A.G., Ryan, M.: Experience with SCRAM, a SCenario Requirements Analysis Method, Proc. of the 3rd ICRE, 1998, pp.164-171.

[15] Sutcliffe, A. G., et al.: Supporting Scenario-Based Requirements Engineering, IEEE Trans. Software Engineering, Vol.24, No.12, pp.1072-108, 1998.

[16] Toyama, T., Ohnishi, A.: Rule-based Verification of Scenarios with Pre-conditions and Post-conditions, Proc. of the 13th RE, 2005, pp.319-328.

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

[18] Zhang, H., Ohnishi, A.: Integration and Evolution Method of Scenarios from Different Viewpoints, Proc. International Workshop on Principles of Software Evolution (IWPSE 2003), 2003, pp.183-188.

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