# A Scenario Description Language Based on Action Frame

Zhang Hong Hui and Atsushi Ohnishi

Department of Computer Science, Ritsumeikan University, Shiga 525-8577, Japan,
{zhh,ohnishi}@selab.cs.ritsumei.ac.jp

**Abstract.** Scenarios that describe concrete behaviors of system play an important role in system development and in particular requirements engineering. Scenarios are informal, and are difficult to be processed automatically. This paper describes a language for describing scenarios in which simple action traces are embellished to include typed frames based on a simple case grammar of actions.

## 1 Introduction

Scenarios are important in system development and in particular requirements engineering for providing concrete system description, especially when used to define system behavior by system developers and to validate the requirements by customers. However, scenarios are informal, and are difficult to be processed automatically.

Ben Achour proposed guidance for writing scenarios [1]. He provides style and content guidelines referring to conceptual and linguistic model of scenarios, based on the case grammar. This work demonstrates 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. However, in Ben Achour's guidance, styles and contents are restricted, so it may be difficult to describe a scenario.

In the authors' previous work [2], we proposed to build software requirements in VRDL (Visual Requirements Definition Language) from textual requirements in Japanese, based on a typology of concepts very similar to the semantic roles of the case grammar. Scenarios can be regarded as a sequence of events. Each event corresponds to a certain action. We have developed a scenario description language based on this notion. We outline a frame-based approach for structuring the actions in a scenario, and use ideas from the previous work and Jackson's problem frames for structuring the content of scenarios and action descriptions in specifications. Scenarios described with this language have clear syntax and semantics, and can be transformed into internal representations automatically.

The authors have introduced the scenario description language in [3]. In this paper, a brief description of this language will be given, and then the evaluation of this language will be presented in detail.

## 2 Scenario Description Language

We assume that scenario is composed of description of event and description of sequence among events.

Events are behaviors employed by users or system for accomplishing their goals. We assume that each event has just one verb, and that each verb has its own case structure. Verbs and their own case structures depend on problem domains, but roles of cases are independent of problem domains. There exist several roles, such as agent, object, recipient, instrument, source, and so on [3]. We provide action frames in which verbs and their own case structures are specified [3]. The action frame depends on problem domains.

Just like Ohnishi's Case Frame [2], each action has its case structure. For example, action "move" has four cases, such as "agent," "source," "goal," and "instrument." A sentence "Mr. X moves from Tokyo to Vienna by airplane" can be transformed into an internal representation as shown in Table 1.

**Table 1.** Internal representation example

| **Action** | **agent** | **source** | **goal** | **instrument** |
|---|---|---|---|---|
| move | Mr. X | Tokyo | Vienna | airplane |

We assume that a scenario represents a sequence of events, and each event can be transformed into internal representation based on the action frame. In the transformation, concrete words will be assigned to pronouns and omitted indispensable cases. Some cases are not indispensable, but optional. In case of optional cases, such as the instrument case of action "notify," a concrete word may not be assigned. Just like Requirements Frame [2], we can detect both lack of cases and illegal usages of noun type.

We also assume several kinds of time sequences among events, such as sequential events, selective events, iterative events, and parallel events (AND/OR/XOR) [3]. A scenario example is given in [3].

Scenario description language has been applied to various problem domains, such as (a) program chair's jobs, (b) seat reservation, (c) goods purchase in company, (d) claiming payment of motoring accidents, (e) use case description of report sales, (f) elevator control. This language depends on problem domains. By changing the action and case frame according to domain, (1) in case of (a) a scenario of 172 lines is described, (2) in cases of (c), (d), and (e), scenarios that contain same information with scenarios appeared in materials open to the public, can be described, and (3) in cases of (b) and (f), information of existing systems can be described as scenarios.

## 3 Evaluation

### 3.1 Ease of writing

Scenario description language is a semi-formally controlled language. In order to evaluate whether it is easy to describe scenario with this language, following experiments have been performed.

**exp.1** ease of writing: comparison between scenario description language (SL) and natural language (NL)

**exp.2** ease of writing: comparison between scenario description language (SL) and activity diagram (AD)

Two testers alternated their description order of different languages on two problems, so proficiency should not be reflected in experiment results. Time for describing scenario and a percentage of correct events and time sequence (cor.) were measured. Experiment results of exp.1 are shown in Table 2.

**Table 2.** Ease of writing: comparison between SL & NL

| problem | tester | description order | language | time | cor. |
|---|---|---|---|---|---|
| (i) | A | 1 | NL | 10 min. | 0.8 |
| | | 2 | SL | 10 min. | 1 |
| | B | 1 | SL | 13 min. | 0.8 |
| | | 2 | NL | 7 min. | 0.4 |
| (ii) | A | 1 | SL | 18 min. | 1 |
| | | 2 | NL | 15 min. | 1 |
| | B | 1 | NL | 15 min. | 0.6 |
| | | 2 | SL | 10 min. | 0.6 |

As shown in Table 2, (1) regardless of language, a tester describes a problem firstly takes more time than that same tester describes same problem secondly, and (2) regardless of description order, scenario described with SL has higher cor. than scenario described with NL. As a result, SL and NL have the same degree of ease of writing. However, if using SL the quality of scenario can be increased. Furthermore, SL is superior to NL in clarity, completeness, rigorousness, and verifiability.

Because description time of AD depends on used tool, in exp.2 handwritten time was measured for both AD and SL. Experiment results of exp.2 are shown in Table 3.

**Table 3.** Ease of writing: comparison between SL & AD

| problem | tester | description order | language | time | cor. |
|---|---|---|---|---|---|
| (iii) | A | 1 | SL | 30 min. | 1 |
| | | 2 | AD | 55 min. | 1 |
| | B | 1 | AD | 12 min. | 0.4 |
| | | 2 | SL | 15 min. | 0.6 |
| (iv) | A | 1 | AD | 23 min. | 1 |
| | | 2 | SL | 17 min. | 1 |
| | B | 1 | SL | 15 min. | 0.9 |
| | | 2 | AD | 13 min. | 0.6 |

In case of tester A, cor. is high, and description time of SL is shorter than description time of AD. In case of tester B, cor. is low, therefore description time does not make much sense, but SL is superior to AD in cor. As a result, SL is easier to write than AD.

### 3.2 Understandability

We assume that SL has same understandability with NL. Following experiment has been performed to compare the understandability of SL and AD.

**exp.3** understandability: comparison between SL and AD

The scenario described with SL and the scenario described with AD, which represent the same problem and so contain same information, were provided to two testers. Testers read these scenarios, then answered prepared questions. Four questions (a,b,c,d) that have same degree of difficulty were prepared for each problem.

Time for answering questions, until answers became correct, was measured. Experiment results are shown in Table 4. As shown in Table 4, SL and AD have the same degree in understandability.

**Table 4.** Understandability: comparison between SL and AD

| | | first | | | second | | |
|---|---|---|---|---|---|---|---|
| **problem** | **tester** | **language** | **question** | **time** | **language** | **question** | **time** |
| (iii) | C | SL | a,b | 5min. | AD | c,d | 6min. |
| | D | AD | a,b | 6min. | SL | c,d | 10min. |
| (iv) | C | AD | a,b | 10min. | SL | c,d | 10min. |
| | D | SL | a,b | 8min. | AD | c,d | 12min. |

### 3.3 Total evaluation

In exp. 1 and 2, not only the ease of writing but also the cor. was compared between languages, so we can get the evaluation of accuracy. The total evaluation of languages is shown in Table 5 according to results of exp.1, 2, and 3.

**Table 5.** Total evaluation of languages

| **language** | **ease of writing** | **understandability** | **accuracy** | **total** |
|---|---|---|---|---|
| SL | A | A | A | A |
| AD | B | A | B | B- |
| NL | A | A | B | B |

A: excellent, B: medium, C: poor

## 4 Conclusion

Since our scenario description language enables us to define both the syntax and the semantics of scenarios, it is easy to analyze and validate scenarios written with this language. Using this scenario description language, we can distinctly describe scenario, and improve the correctness of scenario description. In addition, scenarios can be transformed into internal representation automatically based on the action frame.

## References


1. Ben Achour, C.: Guiding Scenario Authoring, Proc. of the Eight European-Japanese Conference on Information Modeling and Knowledge Bases, Vamala, Finland, May 25-29, (1998)
2. Ohnishi, A.: Software Requirements Specification Database Based on Requirements Frame Model, Proc. of the IEEE second International Conference on Requirements Engineering (ICRE'96), (1996) 221–228
3. Ohnishi, A., Zhang, H.H., Fujimoto, H.: Transformation and Integration Method of Scenarios, Proc. of 26th International Computer Software and Applications Conference (COMPSAC), Oxford, U.K., Aug. (2002) 224–229