

# Scenario Weaving for Security Requirements Elicitation

Hiroya Itoga and Atsushi Ohnishi

*Department of Computer Science, Ritsumeikan University, Japan.*

*{ itoga, ohnishi } @ cs.ritsumei.ac.jp*

## Abstract

*In this paper the authors propose security requirements elicitation method by scenario weaving. When the stakeholders write the behaviors of a system in scenarios or use cases, they may not have concern on software security.*

*We prepare scenario fragments based on security evaluation criteria and weave them into the scenarios. Then we elicit the advices of weaving. The advices express the condition of requirements sentences such as when they needed.*

*The authors explain the process of security requirements elicitation on proposed scenario language and show simple example with common criteria, most popular security evaluation criteria.*

Keywords: scenario analysis, scenario weaving, security requirements, security evaluation criteria

## 1. Introduction

Security problems, especially some techniques to implement these functions to software system, are often taken up as examples of Aspect-Oriented Software Development process. The security problems fit to be applied the process exactly, since stakeholders have concern in their business process and have no concern in software security.

Security requirements elicitation has not discussed in aspect-oriented point of view, though it has the same characteristics on security problem. We can define aspect-oriented security requirements elicitation process like some aspect-oriented implementation method of security functions.

The purpose of the authors in this paper is security requirements [1] elicitation but it is difficult process. Because stakeholders, particularly users, are not professional of the security problems of software, they do not have the knowledge of software security. It causes that they suggest to follow existing laws or security evaluation criteria as first step of the security require-

ments, and then they want to leave analysis for software requirements specification to developers. On the contrary, the developers know the details of problems and techniques of software security but they can not understand them in terms of the specified business process. They do not have precise methods to elicit security and usability requirements from the users who do not know the implementation of security functions.

The authors focus “scenario” for common-language between the users and developers. Scenarios are concrete descriptions of behaviors of users and systems. The developers can write some security functions and properties in concrete expression in scenarios. The users claim in usability because they could understand nothing but the behaviors.

In this paper, the authors propose a method to elicit security requirements using scenario weaving. We can weave scenario fragments from security evaluation criteria into normal scenario written by stakeholders. Then the stakeholders validate the behaviors of the woven scenario and find the requirements of weaving in order to make software requirements specification.

For example, during the sequence of order some products such as;

Mr. X watches the information about the product,

Mr. X orders the product,

Mr. X tells the number of his credit card,

we need the functions to identify Mr. X or to crypt the number of credit card to follow the laws and the security evaluation criteria.

When we introduce the functions, we must ask for the stakeholders; when we should identify and how many times, each time to be watched the information or only once to be ordered; which information we should encrypt and how hard.

So, we prepare the scenario fragments that express the typical behaviors of identification or encryption based on some security evaluation criteria. Then the users validate the behaviors on usability after weaving the scenario fragments into main successful scenario.

```

[ title of scenario ] [ viewpoints of actors ]
{
  ...
  Sequences of events.
  ...
  if ( condition ) then
    Sequence of events if the condition is satisfied
  fi
  ...
}

```

Figure 1: Format of scenario description language.

In this process, we can elicit the security requirements when the specified functions should be used.

The process is different from ordinary aspect-oriented techniques because it tries to get the precise advices from the result of scenario weaving.

It is also an effective example of traceability of early aspects. Since we include the conditions of using the functions into the software requirements specification, we can trace the reason of the requirement to the scenario fragment and its advice.

## 2. Scenario Description Language

Scenario [2][3] in this paper means the description of event sequence according to time which includes behaviors of the user, behaviors of the system, and interactions between the user and the system. Scenario is used on requirements elicitation because we can write concrete description in it from the user's point of view and we can validate the concrete behaviors. The use-case description, not a diagram, in UML use-case is an example of scenario [4].

The authors proposed a scenario language in order to analyze the scenario description machinery and implemented the system. The scenario language is based on requirements frame model [5], so we can analyze the scenario written in weak structured language not only on syntax but also semantics of scenario description.

The requirements frame model is proposed to describe Software Requirements Specification (SRS). It consists of noun frame, case frame, function frame, and some rules of descriptions [6].

Noun frame express types of the nouns. The types are *human* type, *function* type, *data* type, *instrument* type, and so on. For instance, an actor which means a person in scenarios is *human* type, and the information the person deal with is *data* type. In actual scenario, the phrase Mr. X corresponds to *human* type.

Case frame defines the concept of the action of a verb and the structure of nouns related to the verb. For

```

[ title of measure ][ viewpoints of actors ]
event [ event sentence causes exception ]
cond [ description of the exception ]
{
  ...
  Sequences of event against to exception.
  ...
}

```

Figure 2: Format of scenario fragment.

example, in the sentence *an actor sends a request to a system by the form*, the verb *send* means data flow; we write the concept as DFLOW. Still *receive*, *input*, *display*, etc. have same concept of DFLOW. The concept DFLOW must have source case, goal case, and objective case to express the precise concept. Instrument case can be added to the sentence to express how to send the object. In the actual scenario, the source and the goal case are *an actor* and *a system*.

Function frame means the structure between the requirements sentences under some conditions.

These verbs and nouns are prepared in specified dictionaries which include actual words and their types. By preparing them in advance, we can reduce the problem of granularity in event description, i.e. we can prevent the ambiguity from writing the same concept in different words or writing the actions in some words, which is included in another action.

The scenario language has control structure like some programming languages as *if* and *while* structures. And it also has time sequence structure as *AND-fork*, *OR-fork*, and *XOR-fork* on sequence diagrams or activity diagrams in UML [7]. These expressions allow us to check the behaviors in alternative paths during writing the scenario different from extends expression on use-case diagram.

Figure 1 shows the format of the scenario language. It has title of the scenario, viewpoints of the scenario, and event sequences with a control structure.

## 3. Scenario Weaving for Security Requirements Analysis

In this section, the authors explain the concept of scenario weaving and their relations to Aspect-Oriented concepts.

### 3.1. Scenario Fragments

Scenario fragment in this paper correspond to an aspect.

Figure 2 shows the format of a scenario fragment. It is almost same as scenario description but it has event sentence which may cause exceptional behaviors, the description of exception, and behaviors of countermea-

sure, for example, check an ID, failure of checking ID, and retype the ID; instead of actor's objective to archive in scenario, for example, order products or make reports.

The scenario fragments are imported from a research about discovering exceptions and its measures on scenarios. Exception means anything but we assumed in scenarios. In that research we aim to get precise requirements from analyzing more complete behaviors between actors and systems. Security requirements are same characteristics.

Generally the scenario fragments for specific scenarios, which means behaviors related to security, can be discovered by brain storming or using domain knowledge base. But it causes lacks of behaviors because of quality of knowledge base or experiences of attendants.

In this paper, we translate the security evaluation criteria into scenario fragments.

Security evaluation criteria are described in general requirements sentences in order to apply them to various domains. These generic descriptions are difficult to understand for users. But they can understand the behavior if the descriptions are enough concrete such as scenarios. Furthermore, the security experts can write the countermeasures in scenario fragments because they do not need concern the limitation with specified business rules.

### 3.2. Matching Event Sentences

Pointcut in aspect-oriented programming is considered as the particular behavior such as creation of instance, calling the method, and so on. But on scenarios, all event sentences may cause the exception, so the authors consider all the events are pointcuts.

When we weave the scenario fragments into normal scenarios, we must judge the equivalence of the sentences. The scenario description language, which the authors proposed, introduces the case grammar. Therefore, we can find the sentences which have same meaning with event sentence in a scenario fragment though transcribes of words or orders of the phrases are different from the event sentence. In other words, we can find the sentence *an actor sends to a system* and *a system receives from an actor* are same.

When the case is lacked in the event sentence or the phrase is an optional case, the lacked phrase is matched to anything. If the lacked case is necessary case, the lacked ones in the measure behavior of the scenario fragment is complemented the same phrase from the normal scenario. The complement allows us to write the scenario fragment more abstract or without limitation of expression with specified domain or system. For instance, *when can't receive an order sheet*

do not need to distinction of sender; Mr. X can receive it in normal scenario, it becomes *when Mr. X can't receive an order sheet*, and Ms. W can, *when Ms. W can't receive*.

The authors are now investigating the formalize scenario fragments that can be used commonly in a domain. In this case, we prepare the domain specific dictionary and system specific dictionary, and translate the words from one to another.

### 3.3. Scenario Weaving

In scenario weaving process, we weave scenario fragments into normal, main successful scenario using *if* or *while* control structures that are prepared in the scenario language. It is because that the normal scenario is given from the stakeholders and we assume it must be right on behavior, and exceptional behaviors and their measures should be alternative paths of the normal one.

The following are the process of scenario weaving.

- 1) Search for the sentences which have same meaning of event sentence in the scenario fragment. The event sentence is expressed more abstract, so we use it as key to search.
- 2) Complement lacked case phrase if needed. If a phrase is complemented, we should do as same way on countermeasure event sentences. For example, *anyone inputs password to a system* matches *Mr. X...*, we prepare the scenario fragment whose *anyones* are replaced into *Mr. X*.
- 3) Insert the scenario fragment into the normal scenario with *if* or *while* block. When the condition is expressed in *when*, *if* block is used; when the condition is expressed in *during*, *while* block is used.

If the scenario fragment does not include the same sentence with event sentence, *if* or *while* block is inserted just after the matched sentence, since we often judge the result of the matched sentence in scenario fragment. In the case of password checking, we should check it after an actor input it.

In another case, when the scenario fragments include the same sentence with event sentence, *if* or *while* block is inserted around the matched sentence as the same order of events with the scenario fragment. It is used for repeat the behavior until a condition is satisfied. It could succeed by insertion just after the event, but it reduces the readability of scenario; an actor input the password and failed the authentication, we must insert the events of input and checking once again. This can omit the detail description of advice, such as before, after, around on aspect-oriented programming.

<p>7 FIA Identification and authentication</p> <p>7.1 Authentication failures ( FIA_AFL )</p> <p>FIA_AFL.1.1 The system shall detect when (configurable number of) unsuccessful authentication attempts occur related to (authentication events).</p> <p>FIA_AFL.1.2 When the defined number of unsuccessful authentication attempts has been met or surpassed, the system shall (list of functions).</p>
---

Figure 3: An example entry of common criteria

### 3.4. Requirements Elicitation from Woven Scenario

The most important task in this process is requirements elicitation from scenarios in which scenario fragments woven. Scenario is an example of interaction between actors and systems, so we must analyze and elicit more abstract requirement sentences.

We must analyze whether the pointcuts and advices are valid to specified system. On validation, we compare usability of the systems and risk of the security problems. If the stakeholders are not the professional of the security problem, they can understand the concrete behavior of the actors and the systems. Then they judge need or not of the weaving of particular scenarios, and define the advice precisely.

The scenario fragments, or aspect descriptions of scenario, are extracted from security evaluation criteria. The sentences of requirements can be found out on the criteria by tracing the original ones.

It should be validated by the stakeholders that the parameters of sentences for specified system such as when a function is available, which condition is satisfied, and how much harden with security.

### 3.5. Conditions of Actors

If the event sentence of a scenario fragment matches the sentence in normal scenario, the fragment is always woven. But we need to control weave or not the scenario fragment by context of the normal scenario. It reduces the cost to check meaningless weaving.

The conditions of actors are focused to get the context of scenario because scenarios are described from the point of users' view.

We must change the behavior or select the scenario fragment to weave from;

- 1) An actor has specified role such as manager.

<pre>[ Authentication failures ] [ user, system ] event[ System authenticates user by password ] cond[ while authentication failure ] {   User is requested the password from the system.   User inputs the password to the system.   User is authenticated by the system via password.   if ( number of attempts more than 3 times ) then     System switches to <b>invalidate ID</b>.   if }</pre>
--

Figure 4: An example entry of scenario fragment

- 2) An actor has specified access permission for system, device, or files.
- 3) An actor has been authorized by system for secret information.

The authors proposed the method to add the modal verbs in requirements frame model and to analyze the scenario using flow analysis of programming languages.

## 4. Example of Scenario Weaving

### 4.1. Common Criteria

ISO/IEC 15408 common criteria [8] is most popular security evaluation criteria. It includes the description of security functional requirements that are classified by objectives and functions of software system. It also includes the simple behaviors for inspection.

The authors examined by creating practical scenario fragments for the following functional families in the common criteria; identification and authentication/authentication failures, target of evaluation access/session locking, and user data protection/user data integrity transfer protection. In this paper, the authors show the example of authentication failures.

Figure 3 shows the practical common criteria of authentication failures. It shows only the part of behavior, but they also have the ways to evaluate or detail descriptions, and so on.

Now we are investigating other security requirements in the common criteria because the requirements which change the behavior in according to authorities of the actors need to extend the scenario language for modal verbs. Some requirements to get the information in order to evaluate the system can not validate by scenario which have users' viewpoint.

```

[ Order the products ] [ Mr. X ]
{
  Mr. X inputs order to system.
  Mr. X is requested ID from the system.
  Mr. X inputs the ID to the system.
  Mr. X is requested password from the system.
  Mr. X inputs the password to the system.
  Mr. X is authenticated by the password.
  Mr. X is inquired the order from the system.
  Mr. X accepts the order to the system.
}

```

## 4.2. Scenario Fragments from Common Criteria

Scenario fragments are described by experts of security technologies from the behavior description of common criteria, in this process. The behaviors are easy to translate into behaviors in the scenario language and enough to be generic from business rules that the experts may not know. Another reason is that any vulnerability must not be injected.

When they translate the common criteria to scenario fragments, they should express the parameters or options are changeable by the orders of stakeholders. And they should also describe the fragments abstract enough at the actors or functions.

Figure 4 shows the example of a scenario fragment for authentication failures. “3” and “invalidate ID” are changeable and we express it with bold font.

## 4.3. Woven Scenario

Figure 6 shows the woven scenario which is generated by weaving the scenario fragments shown in Figure 4 into the normal, main successful scenario shown in Figure 5. The scenario assumes a product order system on web.

We assume that the result of scenario weaving is customized by validating the behavior. The number of failures was 3 in the fragment, but it is too small in web system. So we change it 3 to 5 because misses in password input may occur more often in web system and invalidation of ID cause more complications. The reason is only from the viewpoint of usability, but it also reduces the cost to support the consumers.

## 5. Discussion

In this section, we discuss the merits and problems of the method that weaving the scenario fragment to normal scenario; the fragments are generated from the security evaluation criteria.

In this method, we may validate the single scenario which includes complete behavior of a system. The

```

[ Order the products ] [ Mr. X, system ]
{
  Mr. X inputs order to system.
  Mr. X is requested ID from the system.
  Mr. X inputs the ID to the system.
  Mr. X is requested password from the system.
  Mr. X inputs the password to the system.

  While ( authentication failure )
  do
    Mr. X is requested the password from the system
    Mr. X inputs the password to the system.
    Mr. X is authenticated by the system via password.
    if ( number of attempts more than 5 times ) then
      System switches to Re-registration ID.
    fi
  done

  Mr. X is inquired the order from the system.
  Mr. X accepts the order to the system.
}

```

Figure 6: An example of woven scenario

original normal scenario includes only the business process. We use aspect-oriented way because there are different concerns between business process and security requirements, but the method may requires to validate all different concerns at once. And the scenario weaving increases the complexity of the scenario since many *if* and *while* blocks are inserted and they may be nested. The original normal scenario has only a path but the woven scenario have many paths. It causes difficulty of validation.

The proposed scenario language has the concept of viewpoint. It can hide the event sentences which have no relationship with the user’s concern. Or the authors plan to make a scenario viewer which hide the *if* and *while* blocks in the scenario according to the condition.

When we find the oppositions of requirements in scenario validation, we do not have effective method to mitigate them. The oppositions may be caused between usability and security, but they have little compromise way generally. We think goal-oriented requirements analysis answers it.

Formalize method to translate abstract scenario fragments into specialized scenario fragments along normal scenario is needed. We complement the phrases which lacked in the fragments only, but we need to define some domain specific translation rules.

On ordinary method, the behaviors related to the security functions are described with the same granularity in scenarios. But it is difficult to trace the reason why the requirements are needed, and who require them, when they are required. It is just traceability

problem with separation of concern in requirements elicitation.

The proposed method deal with the security functional requirements or non-functional requirements as scenario fragments apart from the requirements from the users. The security requirements which come from the laws or security evaluation criteria are separated from the security requirements which users want. Furthermore, we can find out the reason of customized parameter or options.

The requirements frame model used in the scenario language is defined for software requirements specification originally, so we can easy to define the specification from scenario and we can trace the scenario from a requirement sentence by matching the meaning of it.

The backward traceability of software requirements specification, scenario, and security evaluation criteria is just realized by aspect-oriented method.

## 6. Related Works

Another application of aspect-oriented method for use case is proposed [9].

AspectU [10] is an aspect-oriented use case description language which has 1) pointcuts on use case, extension, step of use case description 2) advice before, around, after on pointcuts.

The scenario weaving method, we do not need to learn the new language or concepts. Pointcut is one event sentence included in the original language and advice is condition to find after weaving. Furthermore, we do not need to understand the concepts of aspect-oriented methods.

Misuse case [11] and abuse case [12] is proposed for security requirements elicitation. They show the threats and make the hint for discussion; the countermeasures are found out in brainstorming or something.

The security evaluation criteria guarantee the rightness of countermeasure behavior in the scenario weaving method.

## 7. Conclusion

In this paper, the authors propose and discuss the method of scenario weaving on the proposed scenario language. We prepare the scenario fragments based on the security evaluation criteria and weave them into normal, main successful scenario, and then we validate the woven scenario. We can find the opposition between security and usability requirements.

If there are proper scenario fragments for security, we can get the behaviors which satisfy the criteria automatically.

Ordinal aspect-oriented methods, we deal with the aspect and advice was known, but in the proposed method we check woven scenario and find out precise advices. The conditions for requirements sentences have important meaning in software requirements specification, so the advices are also important.

In the future, the authors will implement the scenario weaving tool and evaluate the method in actual software development process. And they will also establish the method to find the precise advice concerned with cost and security risk.

## Acknowledgements

This research is partly supported by Grant-in-Aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan (No. 17700038).

## References

- [1] IEEE Std. 830-1998, IEEE Recommended Practice for Software Requirements Specifications, 1998.
- [2] John M. Carroll, *Making Use: Scenario-based Design of Human Computer Interactions*, MIT Press, 2000.
- [3] Ian F. Alexander and Neil Maiden, *Scenarios, Stories, Use Cases – Through the Systems Development Life-Cycle*, John Wiley & Sons, 2004.
- [4] Alistair Cockburn, *Writing Effective Use Cases*, Addison Wesley, USA, 2001.
- [5] Atsushi Ohnishi, "Software requirements specification database based on requirements frame model," *Proceedings of the Second IEEE International Conference on Requirements Engineering*, pp. 221-228, 1996.
- [6] C. J. Fillmore, *The Case for Case, Universals in Linguistic Theory*, ed. Bach & Harms, Holt, Rinehart and Winston Publishing, Chicago, 1968.
- [7] HongHui Zhang and Atsushi Ohnishi, "Transformation between Scenarios from Different Viewpoints," *IEICE transactions on Information and Systems*, Vol. E87-D, No. 4, pp. 801-810, 2004.
- [8] ISO/IEC 15408 common criteria.
- [9] Ian Jacobson and P.W. Ng, *Aspect-Oriented Software Development with Use Cases*, Addison Wesley, USA, 2004.
- [10] Jonathan Sillito, Christopher Dutchyn, Andrew David Eisenberg, and Kris De Volder, "Use Case Level Pointcuts," *Proceedings of 18th European Conference on Object-Oriented Programming (ECOOP-2004)*, Oslo, Norway, pp. 244-266, June 2004.
- [11] Guttorm Sindre and Andreas L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, Vol. 10, pp. 34-44, 2005.
- [12] John McDermott, Chris Fox, "Using Abuse Case Models for Security Requirements Analysis," *Proceedings of the 15th IEEE Annual Computer Security Applications Conference (ACSAC'99)*, pp. 55-65, 1999.