A DETECTION METHOD OF FEATURE INTERACTIONS FOR TELECOMMUNICATION SERVICES USING NEW EXECUTION MODEL

Sachiko Kawada, Masayuki Shimokura and Tadashi Ohta
Information Systems Science, Soka University1-236, tangi-cho, hachioji-city, Tokyo, 192-8577, Japan

Keywords: Feature interaction, detecting interactions, seeming interaction, specification execution model.

Abstract: A service, which behaves normally, behaves differently when initiated with another service. This undesirable behavior is called a feature interaction. In investigating the international benchmark for detecting interactions in telecommunication services, it was found that many interactions that do not actually occur (called: "seeming interactions" in this paper) were mis-detected. The reason for mis-detection of seeming interactions is that interactions were detected using a state transition model which does not properly represent the process flow in a real system. Since seeming interactions cause an increase in time taken for solving interactions, avoiding mis-detection is an important issue. In this paper, a problem in implementing a detection system without mis-detecting seeming interactions is clarified and its solution is proposed. In addition, a new interaction detection method, which adopts the proposed solution and is based on a specification execution model which properly reflects the process flow in a real system, is proposed.

1 INTRODUCTION

A service, which behaves normally, behaves differently when initiated with another service. This undesirable behavior is called a feature interaction (hereafter abbreviated as an interaction) (Cameron, 1994). Many approaches, that formally and automatically detect feature interactions among given telecommunication services specifications, have been proposed (Amyot, 2003).

However, in investigating interactions that were described in the international benchmark for detecting interactions in telecommunication services (Griffeth, 2000), it was found that many interactions that do not actually occur (which are called "seeming interactions" in this paper) were mis-detected.

The authors have proposed a Trigger Point Model, (abbreviated as a "TP model") as a new specification execution model which properly reflects the process flow in a real system. They have also confirmed its effectiveness (Shimokura, 2004). In implementing a detection system based on the TP model without mis-detecting seeming interactions, a change of the meaning of an event causes a problem. To solve this problem, this paper proposes a method for identifying the meaning of an event and a new interaction detection algorithm based on the proposed method and the TP model, and confirmed that the proposed algorithm is effective.

In section 2, a concrete example of a seeming interaction caused by a change of the meaning of an event is explained. In section 3, the TP model that is a basis of this paper is briefly described. In section 4, a problem in implementing a detection system is described, and a method for identifying the meaning of an event is proposed as a solution. In section 5, a new detection algorithm for interactions based on the proposed model and the TP model is proposed. In section 6, the proposed algorithm is evaluated.

2 SEEMING INTERACTION

It is well known that telecommunication services specifications can be described as state transition diagrams. In this paper, hereafter, 'specification' means individual state transitions in the state transition diagram for a service. These state transitions are described formally so that a computer can understand them. So, a service specification means a set of all specifications for the service. 'Execution of a specification' means to execute a state transition described in the specification.

Kawada S., Shimokura M. and Ohta T. (2006).
A DETECTION METHOD OF FEATURE INTERACTIONS FOR TELECOMMUNICATION SERVICES USING NEW EXECUTION MODEL.
In *Proceedings of the First International Conference on Software and Data Technologies*, pages 190-195
DOI: 10.5220/0001310301900195

'Triggering a specification' means to initiate execution of the specification.

A concrete example of a seeming interaction between Call Forwarding service (CFV) and Calling Number Delivery service (CND), which is described in the international benchmark, is explained.

1) A specification of CFV

A typical specification of CFV is explained. Suppose that terminal A receives a dial tone (denoted by dialtone(A)), and terminal B has CFV activated and has registered terminal C as a forwarding terminal (denoted by cfv(B,C)). When terminal C is idle (denoted by idle(C)), if terminal A dials terminal B (denoted by dial(A,B)), the call from terminal A is forwarded to terminal C, then terminal A calls terminal C (denoted by calling(A,C)) (Figure 1).

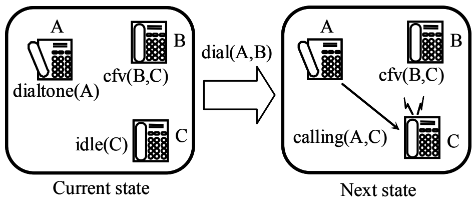

Figure 1: A specification of CFV.

2) A specification of CND

A specification of CND is explained. Suppose terminal C has CND activated (Figure 1 A specification of CFV denoted by cnd(C)). When terminal A receives a dial tone and terminal C is idle, if terminal A dials terminal C (denoted by dial(A,C)), terminal A calls terminal C, and a telephone number of terminal A is displayed on terminal C (denoted by display(C,A)) (Figure 2).

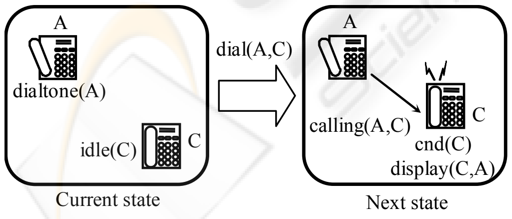

Figure 2: A specification of CND.

3) Occurrence of a seeming interaction

Since displaying a telephone number is executed after it is determined that terminal C is called, a specification of CND is triggered later than that of CFV. In the conventional detection methods, since it was supposed that when an event, dial(A,B) occurs, only a specification which has dial(A,B) as an event was triggered, only CFV is triggered. Therefore, a telephone number of terminal A is not displayed on terminal C despite terminal A calls terminal C. Since this is an abnormal state, an interaction is detected.

However, taking into consideration the process flow in a real system, after execution of CFV, the call from terminal A is forwarded to terminal C. Then, if terminal C is idle, terminal A calls terminal C. In effect, it can be said that a call from terminal A reaches terminal C. Thus, the meaning of the event, which is a trigger for executing specifications after execution of a specification of CFV, should be deemed to be 'a call from terminal A reaches terminal C', that is, in this case, dial(A,C). After a specification of CFV is executed, a specification of CND which has dial(A,C) as an event is triggered. As a result, a telephone number of terminal A is displayed on terminal C and an interaction does not occur.

Therefore, the interaction between CFV and CND shown as an example is a seeming interaction.

3 TP MODEL

The minimum explanation of the TP model, which is necessary for this paper, is given. For more details please refer to (Shimokura, 2004).

3.1 Overview

The TP model is designed, independently from individual services, based on state transition diagrams. To realize independency, each system state in state transition diagrams for supplementary services is represented as one of abstracted states in state transition diagrams for the basic service (POTS). Thus, each state transition for supplementary services can be represented as one of state transitions between common states, Sc and Sn, abstracted from states of POTS (Figure 3).

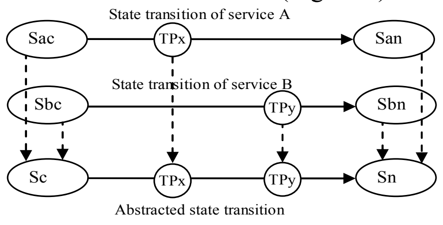
State transition of service A
Sac
TPx
San
State transition of service B
Sbc
TPy
Sbn
Sc
TPx
TPy
Sn
Abstracted state transition

Figure 3: Represented state transition.

On a common abstracted state transition diagram obtained in the way mentioned above, TPs are set as timing points for specifications. Since timing points for individual specifications are determined according to TPs, the order of triggering specifications is clear.

State transitions for CFV and CND, described in section 2, are explained using Figure 3. Current states for CFV and CND, Sac and Sbc, can be represented as Sc. Next states for CFV and CND, San and Sbn, can be represented as Sn. TPs for a specification of CFV and a specification of CND are described as TPx, and TPy, respectively. Thus, it is clear that a specification of CFV is triggered before a specification of CND.

3.2 Triggering a Specification

(1) Conditions for triggering a Specification

Suppose that in a specification, 'current state' which is a service state before the state transition, 'event' which is a trigger for the state transition, 'next state' which is a service state after the state transition, the name of a TP where the specification is initiated, and a name of a TP or a state which the process flow after execution of the specification reaches, are described. If the process flow after execution of the specification does not reach one of TPs or states, the last term, a TP or a state, is not described. In the TP model, when an event Ei occurs and the process flow reaches one of TPs, TPi, specifications, which have the same event as Ei and the same trigger point as TPi, are initiated. Thus, for one event, more than one specification can be triggered.

Conditions for triggering a specification are given as follows:

Condition 1: The process flow for the event reaches a TP described in the specification.

Condition 2: An event described in the specification occurs.

Condition 3: A state described in the current state of the specification is the same as a service state when the process flow reaches a TP described in the specification.

(2) Triggering two specifications

In the TP model, when two specifications are given, each specification is initiated as follows:

(i) In case two specifications have the same TP.

When a process flow reaches a TP described in both specifications, firstly, a specification, which satisfies Condition 2 and Condition 3, described above in this section, is triggered. Where both specifications satisfy the conditions, either specification is triggered first. After execution of the first specification, if the process flow reaches a TP described in the second specification and the second specification satisfies the conditions, the second specification is also triggered.

(ii) Two specifications have different TPs which are set on the same state transition in the TP model.

When a process flow reaches a TP described in the specification triggered first, if the specification satisfies Condition 2 and Condition 3 described above in this section, it is triggered. After execution, if the process flow reaches a TP described in the other specification and the specification satisfies the conditions, the specification is also triggered.

(iii) Two specifications have different TPs which are set on different state transitions, respectively, in the TP model.

Since the temporal order of TPs cannot be determined, both specifications are triggered in the same way as case (i).

4 MEETING CONDITIONS FOR TRIGGERING: A PROBLEM AND ITS SOLUTION

4.1 Problem

To detect interactions, it is judged whether two given specifications can be triggered or not.

In the TP model, when an event, Ei, occurs and the process flow reaches a TP, TPi, a specification which is triggered at the TPi has Ei as an event. But, as mentioned in section 2, there is a case where the meaning of an event is changed by execution of a specification. Therefore, in this case, after execution of the specification, a specification that has an event other than Ei may satisfy Condition 2 described in section 3.2. In this case, another specification that has Ei as an event does not satisfy the condition.

Therefore, to judge whether a specification which is executed after execution of the first specification, satisfies Condition 2 described in section 3.2 or not, it is necessarily to identify what an event means after execution of the first specification.

4.2 Solution

(1) Identification Method

The meaning of an event used in POTS is well known. But, the meaning of a new event used in supplementary services cannot be known beforehand. The meaning of an event commonly used in two supplementary services causes the problem in judging if Condition 2 described in section 3.2 is satisfied. But, most of those events are used in POTS. Therefore, in this paper, targeted events are restricted to those used in POTS. The meaning of those events is classified, and a method for identifying the meaning of events is discussed. Because of space limitation, a method for identifying the meaning of dial(A,B) is discussed here.

For POTS, there is a specification that represents a state transition: when a service state is {dialtone(A), idle(B)}, if dial(A,B) occurs, the service state transits to calling(A,B). This specification can be taken as, when terminal B is idle, if a call from terminal A reaches terminal B, terminal A calls terminal B. Besides, only this specification represents a state transition to calling(A,B). That is, 'when terminal B is idle, terminal A calls terminal B' is a necessary and sufficient condition for the meaning of an event, which is a trigger for initiating this specification, to be that a call from terminal A reaches terminal B. Thus, if calling(A,B) is described in the next state of the specification, after execution of the specification, arguments X and Y of dial(X,Y) which means 'a call from terminal X reaches terminal Y', are A and B, respectively.

Thus, if calling(A,B) is described in the next state of a given specification s (in this case, s is called as s1) which has dial(X,Y) as an event, after execution of s, dial(X,Y) should be considered to be changed to dial(A,B).

An identification method in the case where calling(A,B) is not described in the next state of s is discussed in (2).

(2) calling(A,B) is not described in specification s

A method for identifying the meaning of an event in the case where calling(A,B) is not described in the next state of s, s2. A concrete example where calling(A,B) is not described in s2 is explained.

s2, which defines a state transition when terminal C registered as a forwarding terminal in CFV is not idle, is shown in Figure 4. Figure 4 is explained. Suppose terminal B has CFV activated and has registered terminal C as a forwarding terminal. When terminal C is not idle (denoted by not[idle(C)]), if terminal A dials terminal B, terminal A receives a busy tone (denoted by busy(A)). Thus, when s2 is executed, a call from terminal A reaches terminal C. Thus, the meaning of the event after execution of s2 should be regarded as not dial(A,B) but dial(A,C).

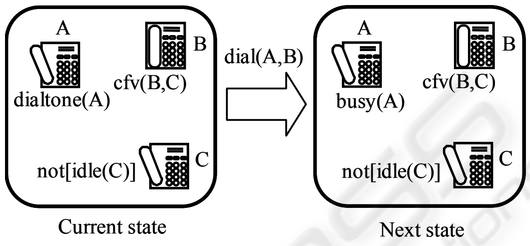

Figure 4: A specification of CFV (in case terminal C is not idle).

However, since calling(A,C) is not described in the next state of s2 shown in Figure 4, the identification method proposed in (1) above cannot identify the meaning of dial(X,Y).

But, in general, there are two cases for terminating terminal, idle and not idle, the service specification should have both specifications.

Thus, when calling(A,C) is not described in the next state of s2, by finding out another specification, s3, which has calling(A,C) in the next state, the meaning of the event can be identified as dial(A,C).

(3) An event identification method

For a method for identifying the arguments X and Y in dial(X,Y), after execution of specification s that has dial(A,B) as an event, discussion (1) and (2) mentioned above are summarized. When calling(P,Q) is described in the next state of s (s1), dial(X,Y) after execution of s1 is regarded as dial(P,Q). Here P and Q represent arbitrary terminals. When calling(P,Q) is not described in the next state of s (s2), identify another specification s3, that has dial(X,Y) as an event and calling(P,Q) is described in its next state in the service specification to which s2 belongs. If s3 is found, dial(X,Y) after execution of s2 is regarded as dial(P,Q). If s3 is not found, since arguments in dial(X,Y) cannot be identified, the arguments in dial(X,Y) after execution of s2 are regarded as unchanged. Consequently, there is a possibility that real interactions are not detected and/or seeming interactions are mis-detected. This possibility is evaluated in section 6.

Based on the discussion above, a method for identifying the arguments in an event after execution of specification s which has dial(A,B) as an event, is proposed as follows:

Step 1) If calling(P,Q) is described in the next state of specification s, go to Step 3.

Step 2) Search for another specification si, that has dial(X,Y) as an event and calling(P,Q) is described in its next state, in the service specification to which specification s belongs. If specification si is not found, go to Step 4.

Step 3) Identify the meaning of dial(X,Y) after execution of specification s as dial(P,Q), and end identification.

Step 4) Identify the meaning of dial(X,Y) after execution of specification s as dial(X,Y), and end identification.

5 DETECTION METHOD FOR INTERACTIONS

A new interaction detection algorithm, which is based on the TP model and adopts solutions described in section 4.2, is proposed. In interaction detection, for given two specifications depicted from two service specifications, respectively, non-determinacy interactions and semantic interactions, which means abnormality of a system state or a state transition after execution of two specifications (Ohta, 1994)(Ohta, 1998) are detected, as conventional detection methods.

5.1 Detection Scenario for Non-determinacy Interactions

A non-determinacy interaction occurs when the order of triggering two specifications cannot be determined. In the TP model, the order of triggering two specifications cannot be determined in the following cases: Both TPs and events described in each specification are the same, or events described in each specification are the same and the TPs of each specification are set on different state transitions in the TP model. So, in either case, a non-determinacy interaction is detected.

5.2 Detection Scenario for Semantic Interactions

(1) Checking conditions for a specification to be executed

Since a specification is described as a state transition, when two specifications that belong to different services are given, a specification, which satisfies all of the following conditions, should be executed.

(a) The process flow reaches a TP described in the specification.

(b) An event described in the specification is the same as one that actually occurs.

(c) The state described in the current state of the specification exists in the current service state of a compound

The conventional detection methods (Ohta, 1994) (Yoneda, 2003) can be used for judging conditions (c). Judging conditions (b) can be made by using the identification method proposed in section 4.2. Therefore, the method for judging condition (a) is discussed.

If two specifications have the same TP, or different TPs which are set on different state transitions in the TP model, the both specifications are judged to satisfy condition (a).

In case that two specifications, sa and sb, have different TPs (TPa and TPb) that are set on the same state transition in the TP model, if TPa is set ahead TPb in the state transition in the TP model, specification sa is judged to satisfy condition (a). For sb, only if the process flow in the TP model reaches TPb after execution of specification sa, specification sb is judged to satisfy condition (a).

These judgments can be made by comparing a TP, described in specification sb, and a destination reached by a process flow after execution of specification sa, described in the specification sa.

(2) A detection scenario

A detection scenario of semantic interactions is proposed. When the given two specifications have different events, firstly, an event described in either of two specifications is supposed to occur, and detecting interactions is done. Then, suppose that the other event occurs, and detecting interactions takes place. The detection scenario is as follows:

Step 1) According to the triggering methods described in section 3.2, execute a specification that is triggered first.

Step 2) After execution of the specification, the event is changed if needed, according to the identification methods of events described in section 4.2 (3).

Step 3) Execute another specification according to the triggering methods described in section 3.2

and obtain a state of a compound service after a state transition.

Step 4) Judge whether each specification should be executed or not according to the method described in (1) above. If both specifications are judged to be executed, go to Step 6.

Step 5) If a state described in the current state of the specification, which is judged as not to be executed, does not exist in a state of a compound service after execution of the other specification, an interaction is detected. Go to Step 7.

Step 6) If each state described in the next states of each specification does not exist in a state of a compound service after execution of both specifications, an interaction is detected.

Step 7) If a state of a compound service after execution of specification/specifications violates either service constraint (Ohta, 1998), an interaction is detected.

6 EVALUATION

The event identification method proposed in section 4.2 and the new detection method for interactions proposed in section 5 were applied to specifications for 12 services, which are described in the international benchmark (Griffeth, 2000). In the international benchmark, 98 interactions are reported (Griffeth, 2000). But, among them, there are 22 interactions that do not actually occur because system states just before executing the specifications cannot actually exist. According to our investigation beforehand, it was confirmed that 39 interactions out of 76 interactions are seeming interactions.

For the identification method: in all cases for all pairs of 12 services, all events are correctly identified. Thus, the proposed identification method was confirmed to be reasonable.

For the detection method: in 39 seeming interactions (8 non-determinacy interactions and 31 semantic interactions) were avoided to be mis-detected, and 37 actual interactions were detected. Thus, the proposed detection method was confirmed to be effective.

7 CONCLUSION AND FUTURE WORK

To implement a detection system without mis-detecting seeming interactions, a method for identifying the meaning of an event was proposed. In addition, a new method for detecting interactions was proposed. The proposed method was applied to specifications of 12 services described in the international benchmark for interaction detection, and it was confirmed that the proposed methods were reasonable and effective.

For future work, an automatic detection system based on the proposed methods will be implemented and evaluated in more detail.

REFERENCES

Amyot, D., 2003. Feature Interactions in Telecommunications and Software Systems VII. *Proc. of FIW'03*. IOS Press.

Cameron, J., 1994. A Feature Interaction Bench mark for IN and Beyond. *Proc. of FIW'94*. IOS Press.

Griffeth, N., 2000. A feature interaction benchmark for the first feature interaction detection contest. *The International Journal of Computer Networks, Vol.32*.

Ohta, T., 1994. Classification, Detection and Resolution of Service Interactions in Telecommunication Services. *Feature Interactions in Telecommunications Systems*. IOS Press.

Ohta, T., 1998. Formal Detections of Feature Interactions in Telecommunication Software. *IEICE, Trans. on Fundamental, Vol.E81-A, No.4*.

Shimokura, M., 2004. Service Specification Description Model for Avoiding Redundancy in Detecting Feature Interactions. *Proc. ATNAC2004*.

Yoneda, T., 2003. Formal Approaches for Detecting Feature Interactions, Their Experimental Results, and Application to VoIP. *Proc. of FIW03*. IOS Press.