Deriving Interaction-Prone Scenarios in Feature Interaction Filtering with Use Case Maps

Masahide Nakamura
Informedia and Educational Devision
Cybermedia Center
Osaka University, JAPAN
masa-n@cmc.osaka-u.ac.jp

Pattara Leelaprute, Tohru Kikuno
Dept. of Informatics and Mathematical Science
Graduate School of Engineering Sciences
Osaka University, JAPAN
{pattara, kikuno}@ics.es.osaka-u.ac.jp

Abstract

Feature interactions (FIs, in short) occur when features of different communication services interfere with each other. The FI filtering is a pre-processing before the FI detection, which roughly identifies FI-prone service combinations based on simple indications of the FIs. We have previously proposed an FI filtering method at requirements stage using Use Case Maps (UCMs). This method identifies FI-prone service combinations by focusing on changes in user's scenarios before/after the service composition, but does not tell which scenarios in the compound services have a potential of FIs.

In this paper, as an extension of the previous method, we propose a new method to derive FI-prone scenarios from the FI-prone combinations obtained by the previous method. From many practical FIs, we first make the following two observations: (a) FI tends to occur in scenarios where both services are activated, and (b) FI tends to occur in scenarios where a service bypasses a feature of the other service. Then, based on the observations, we propose heuristics on the UCM scenario paths to derive FI-prone scenarios. An experimental evaluation demonstrates that the derived scenarios successfully cover all scenarios that lead to actual FIs.

1 Introduction

Recent advancement of networks, such as Advanced Intelligent Network (AIN), mobile and IP networks, enables functional enrichment of communication services. A variety of services from the conventional telecommunication services to multimedia services is being developed and deployed on the network.

The flood of services introduces a new problem, called Feature Interactions (FIs, in short) [9]. The FI is generally referred as a functional conflict between multiple services, which is never expected from services in isolation. This problem is known as a serious obstacle for rapid creation of new services, and should be detected and resolved. To tackle the problem, many formal methods to detect FIs have been proposed so far (See survey [3]). However, to detect all possible FIs exactly is generally an expensive task, due to combinational explosion in the numbers of service combinations, global states, scenarios, and so on.

In order to cope with the problem, a new notion of *FI filtering* is emerging recently [4]. The FI filtering is a pre-processing before the FI detection, which roughly identifies *FI-prone* service combinations based on simple indications of the FIs. Though the FI-prone combinations identified do not always cause actual FIs, the FI filtering narrows the service combinations to be investigated with a small cost. This enables a large cost reduction in the FI detection process.

We have previously proposed an FI filtering method at requirements stage [6], using Use Case Maps (UCMs) [1, 2], which is a graphical notation based on scenario paths (use cases). This method identifies FI-prone service combinations by focusing on changes in user's scenarios before/after the service composition, and outputs one of the following verdicts: (1) FI occurs, (2) FI does not occur, or (3) FI-prone. However, the method gives only the verdict to each service combination, but does not tell which scenarios in the compound services have a potential of FIs. Therefore, for each FI-prone combination, all possible scenarios have to be examined in the FI detection process.

In this paper, as an extension of the previous method, we propose a new method to derive FI-prone scenarios for the FI-prone combinations. From many practical FIs, we first make the following two observations: (a) FI tends to occur in scenarios where both services are activated, and (b) FI tends to occur in scenarios where a service bypasses a feature of the other service. Then, based on the observations, we propose heuristics on the UCM scenario paths to derive

FI-prone scenarios.

An experimental evaluation through the FI detection contest held in the year 2000 [5] demonstrates that the derived scenarios successfully cover all scenarios that lead to actual FIs. Also, it is shown that the proposed method can improve quality of FI filtering. As a result, the proposed method adds sophisticated information to the previous FI filtering method, which achieves more efficient FI detection.

2 Feature interactions (Practical examples)

Let us see practical examples of FIs, using the following three services. In the following, x, y, z and A, B, C denote variables and actual users, respectively.

Terminating Call Screening (TCS): This service allows a subscriber to screen incoming calls based on a screening list. Suppose that a user x registers y in the screening list. Then any call from y to x is screened.

Call Forwarding on Busy (CFB): A subscriber of this service can forward incoming calls to other predetermined number when the subscriber is busy. Suppose that a user y is setting the forwarding number to z, and that y is busy. If x dials y, then the call is forwarded to z and x is connected to z.

Voice Mail (VM): This service allows a subscriber to redirect incoming calls to a voice mail center while the subscriber is busy. Suppose that a user y subscribes to the VM, and that y is busy. If x dials y, x can record a voice message after the announcement, instead that x receives busytone [1] .

Then, the following FIs occur. Note that these FIs do not occur if the services are used in isolation.

FI-(a) - Interaction CFB & VM: Suppose that B subscribes to both CFB and VM, and that B is busy. At this time, if A dials B, then a non-deterministic behavior occurs: should the call be forwarded by CFB or be redirected to voice mail center by VM?

FI-(b) - Interaction TCS & CFB: Suppose that C subscribes to TCS and puts A in the screening list, and that B subscribes to CFB and sets the forwarding address to C. Now, if A dials B when B is busy, then CFB forwards the call to C. As a result, A can call C, which is a violation of TCS's feature.

[1] In general, VM works also when the subscriber does not answer the phone. However, for simplicity, we suppose in this paper that VM is activated only when the subscriber is busy.

3 FI filtering with UCMs

This section presents a brief review of the previous FI filtering method [6].

3.1 Use Case Maps

Use Case Maps (UCMs, in short) is a requirements notation method based on a scenario path structure [1, 2]. UCMs visually describe causal relationships between responsibilities of one or more use cases.

Figure 1 shows an example of UCMs. In UCMs, *scenario paths* are depicted by wiggly lines. A path starts at a *starting point* (depicted by a circle) and ends at an *end point* (shown as a bar). *Responsibilities*, depicted by crosses with labels, are abstract activities that can be refined in terms of events, tasks, functions, etc. Tracing a path from the start to the end explains a scenario (use case) by a causal sequence of events. *Fork* and *join* are used to make scenarios branch off and merge together in one diagram. A fork may be associated with a *guard*, which represents a condition for the path selection. A guard is described by a text with brackets.

Next, a *dynamic stub*, depicted by a dotted diamond, specifies a place in scenarios, where the details of the scenarios are explained by other UCMs, called *submaps*. The dynamic stub allows UCMs to have hierarchical structure and dynamic behavior. A UCM in the top hierarchy is called *root map*. A submap can be *plugged into* a stub by binding entries and exits of the stub with starting and end points of the submap, respectively. It is also possible to plug multiple submaps into a dynamic stub. The selection of the submaps are done at run time by a *selection policy*, which is usually specified in *pre-conditions* of the submaps.

The primary purpose of the UCM notation is to bridge the gap between requirements and design specification. Since details of system are usually not determined at requirements stage yet, UCMs do not have such strict formalism as seen in *formal description techniques (FDTs)*. UCMs only specify path structure and causal relationships between responsibilities. Other things such as labels, guards and conditions do not have formal meanings. They are just used to help human understanding.

3.2 Describing services with UCMs

3.2.1 Basic call

Figure 1 shows scenarios for simplified basic call model (default service). The root map in the figure describes core scenarios commonly used by all services. Two submaps def_1 and def_2, shown in the lower part, are respectively plugged into the stubs 1 and 2 in the root map, which complete the scenarios of the basic call. The submaps are called

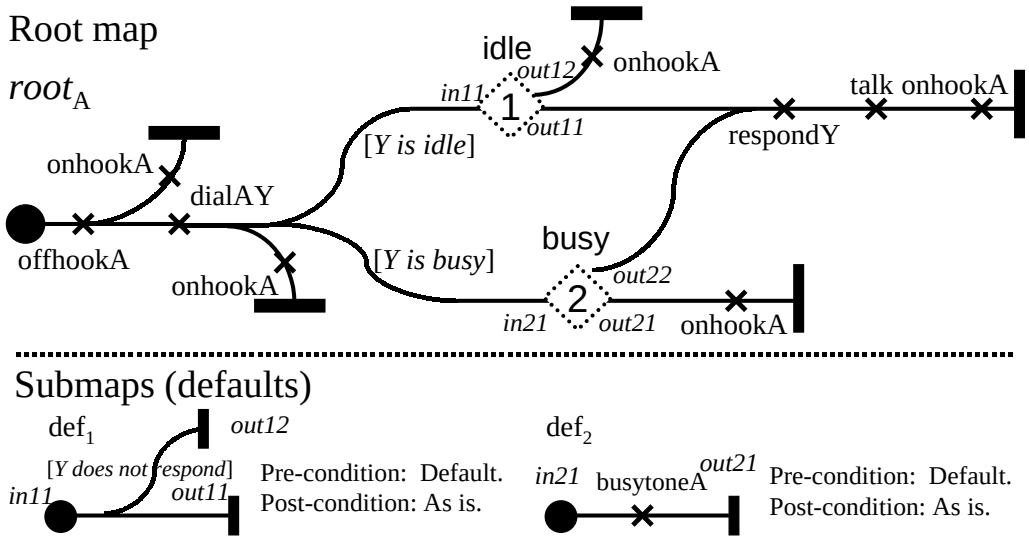

Figure 1. Use Case Maps for simplified basic call

default submaps, in the sense that they are for the default service.

The UCMs explain scenarios that a user A calls other user Y (here, we assume that Y is variable, i.e., the actual callee is determined at run time). Let us explain some scenarios. First A offhooks and dials Y. If the callee Y is busy, A receives busytone and onhooks to disconnect the line. If the callee Y is idle and responds to the call, A and Y can talk. Finally A onhooks to disconnect.

3.2.2 Supplementary services

Supplementary services (e.g., TCS, CFB, VM in Section 2) extends the basic call (default service) by adding special features. In this sense, we call the supplementary services simply *features* in the following. Adding features to the basic call is achieved easily in terms of the dynamic stubs. Instead of the default submaps, *feature submaps*, which describe scenarios specific to the feature, are plugged in the dynamic stubs in the root map. The prug-in operation is performed dynamically, according to the subscription conditions specified in pre-conditions of the feature submaps.

Figure 2 shows the feature submaps. Let us add CFB to the basic call. Suppose that B subscribes to the CFB and forwards the call to user C. For convenience, we introduce a notation F_u to denote a feature subscription where user u subscribes to feature F. In the root map in Figure 1, consider A's call scenarios when B is busy. First, A offhooks and dials B (now the variable Y is regarded as B). Since B is busy, the scenario proceeds to the stub 2. The feature submap $cfbBC_2$ in Figure 2 has a pre-condition $Y = B$. Since the pre-condition matches the scenario that the callee Y is B, $cfbBC_2$ is plugged into the stub 2 instead of def_2. In $cfbBC_2$, the call to B is forwarded to C by *forwardBC*. Next, if C is idle and responds to the call, the scenario proceeds to *out22*. The callee Y is regarded as C in the following scenario, as specified in the post-condition for *out22*. The scenario goes back to the root map and A and C talk. For the stub 1 in the root map, def_1 is reused.

Similarly, $tcsC_1$ is plugged into stub 1, if C subscribes to TCS and A calls C. Also, vmB_2 is plugged into stub 2, if B subscribes to VM and A calls B. Note in the example that if the callee Y does not subscribe to any feature, def_1 and def_2 are used. Using dynamic stubs and feature submaps, a feature (with a subscriber) can be roughly characterized by a *stub configuration*, i.e., allocation of submaps to stubs in the root map.

3.3 FI filtering (Previous method)

Researchers agree on an informal proposition of FI: *FI occurs iff the combined use of multiple features changes the requirements properties of each feature in isolation.* Though many researches have been conducted so far, there is no universal strict definition of FIs. In general, different frameworks adopt different definitions. In the FI filtering method we proposed in [6], we briefly characterize FIs by the following conditions.

Condition C1 (Necessary condition for FIs): If an FI occurs, then composition of features changes some user's call scenarios in an individual feature

Condition C2 (Sufficient condition for special FIs): If a composition of features enables different call scenar-

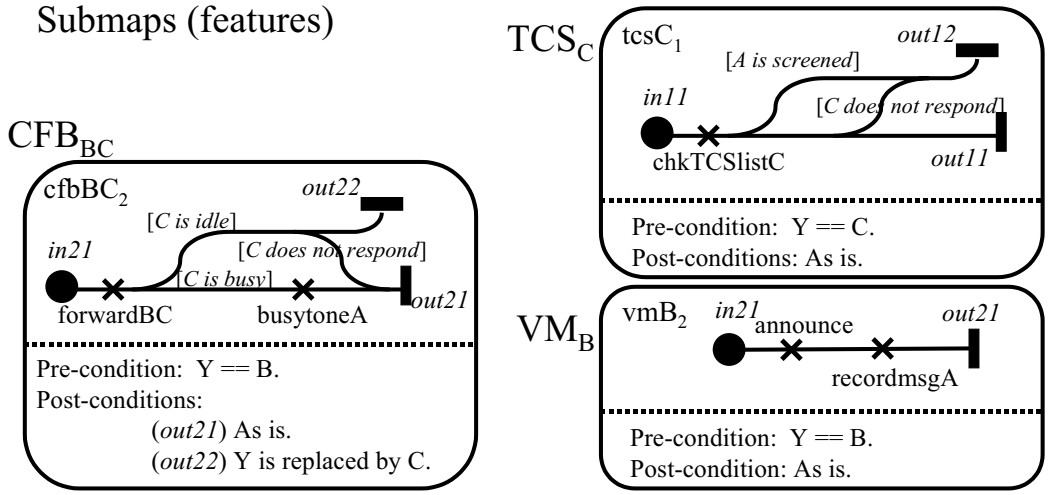

Figure 2. Submaps for supplementary features

ios to be performed under the same conditions, then a non-deterministic FI [8] occurs.

Although these seem to be a bit loose conditions, the filtering method does not need such strict definitions as seen in FI detection. This is by the nature of FI filtering. Based on these conditions, we explain the previous filtering method.

We first duplicate the root map for all possible users assumed. For example, consider again the root map in Figure 1. Here, if we assume three users A, B and C in FI filtering, then three copies are created as shown in Figure 3. Let us consider the stub configuration CFB_B shown in a rectangle in Figure 3(a). When B subscribes to CFB, and A or C calls B, CFB may be activated. Thus, B's subscription to CFB affects scenarios of A and C. Hence, the submap $cfbBC_2$ (in Figure 2) may be plugged into the stubs 2 in the root maps of A and C (i.e., $root_A$ and $root_C$). Similarly, other features respectively form their own stub configurations under the root maps of all assumed users.

Once a feature is characterized by a stub configuration, the method combines stub configurations of two features[2]. Then, according to the following simple checking, the method outputs one of the following verdicts: (1) FI occurs, (2) FI does not occur, or (3) FI-prone.

FI Filtering Procedure:

Step 1: By the composition, if feature submaps of different features conflict in the same stub, then we conclude (1) FI occurs (non-determinism).

Step 2: For each user's root map, if the stub configuration does not change before/after the feature composition, then we conclude (2) FI does not occur.

[2]Strictly speaking, the composition $\oplus$ is defined by a matrix operation [6].

Step 3: Otherwise, we conclude (3) FI-prone.

Let us see how the filtering method works using Figure 3. Figure 3(a) shows a case with the verdict *FI occurs*. Here, B subscribes to both CFB and VM. Therefore, as explained in Section 3.2.2, CFB uses $cfbBC_2$ in stub 2, while VM puts vmB_2 in the stub2 also. Composition of two features causes a conflict of the two feature submaps at stub 2. So, by Step 1, this combination is concluded to be "FI occur", which reflects the example **FI-(a)** in Section 2. This verdict is justified by Condition C2.

Next, Figure 3(b) shows a case with the verdict *FI-prone*. Here, C subscribes to TCS and B subscribes to CFB. By the composition, there is no conflict of feature submaps. However, a new stub configuration appears in $root_A$ after the composition. So, Step 3 concludes that this combination is "FI-prone". The new configuration in $root_A$ indicates that some scenarios of user A may be changed. Hence, according to Condition C1, FI *may* occur. This feature combination actually causes an FI explained in **FI-(b)** in Section 2. However, the filtering method cannot tell the strict answer at this time, just concluding FI-prone. The further analysis is left to the subsequent FI detection process.

Thus, by using simple checking, the previous filtering method conducts a rough and quick evaluation for the tendency of FIs. However, the method only gives a verdict for a combination, but does not provide concrete scenarios in which a potential FI may occur. Especially, as for FI-prone combinations, it is essential to know which scenario has a high possibility of FIs. If we have the FI-prone scenario, it greatly helps the subsequent FI detection process, since the scenario narrows the scope of FI detection.

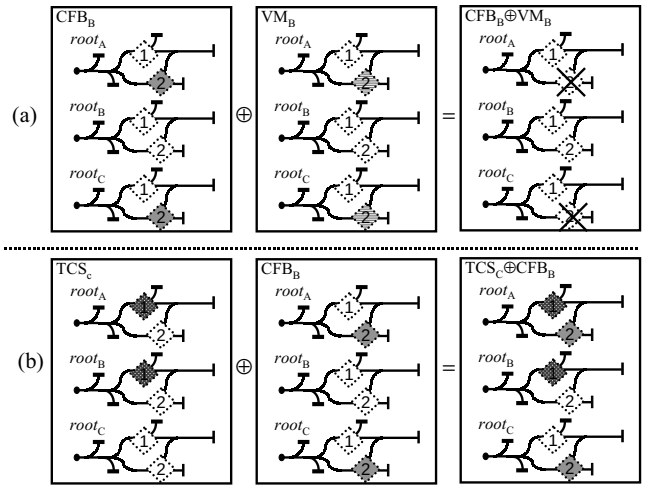

Figure 3. FI filtering (a) FI occurs (b) FI-prone

4 Deriving FI-prone scenarios

As an extension of the previous filtering method, we propose a new method to derive FI-prone scenarios for feature combinations with the verdict FI-prone.

4.1 Heuristics to identify FI-prone scenarios

In the previous method, the verdict FI-prone is derived by Condition C1, i.e., some user's scenarios in the root maps have been changed by the feature composition. What we have to consider next is how the scenarios change and which scenarios have a potential of FIs. In many practical examples, we have had the following observations of FIs.

Observation 1: FI tends to occur in the scenarios where *two different features are activated*.

Observation 2: FI tends to occur in the scenarios where *execution of the one feature bypasses execution of the other feature*.

For example, consider again the examples FI-(a) and FI-(b) in Section 2. FI-(a) occurs in a scenario, where B subscribes both CFB and VM and, A (or C) calls B while B is busy. In this scenario, both two services CFB and VM are activated (though this combination is concluded to be (1) FI occurs by the previous method). So, Observation 1 is justified. Next, FI-(b) occurs in a scenario, where B subscribes to CFB, C subscribes to TCS, and A calls B when B is busy. In the scenario, a feature of TCS, which checks C's screening list, is bypassed by CFB. From this, we can see that Observation 2 is also reasonable.

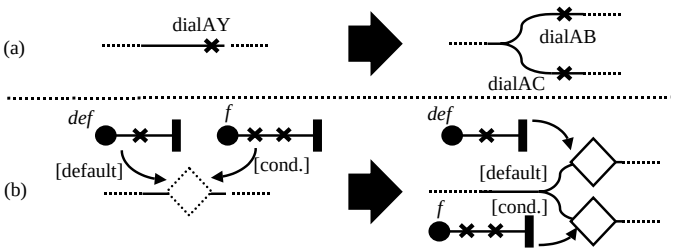

Figure 4. Expanding dynamic elements

According to Condition C1, the FI-prone scenarios must be contained in a root map in which the stub configuration is modified by feature composition. For instance, let us see Figure 3. For the composition (a), $root_A$ and $root_C$ include FI-prone scenarios. For (b), if an FI occurs then the FI must be in a scenario in $root_A$. Consequently, in order to derive FI-prone scenario, we have to do the following two steps:

Step 1: Identify root maps that may contain FIs.

Step 2: Examine scenarios of the root maps in which the FIs tends to occur.

Step 1 is easily done by looking at stub configuration of the root maps. To achieve Step 2, we propose heuristics on the UCMs based on the Observations 1 and 2, as presented in the following subsections.

4.2 Expansion of dynamic elements in scenarios

In order to derive scenarios satisfying Observations 1 and 2, we eliminate *dynamic elements* in the root map suspected in Step 1. The dynamic elements in our root maps are responsibilities with variables and dynamic stub. The dynamic elements help the UCMs to describe dynamic scenarios, i.e., the detailed conditions of the scenarios are determined and interpreted at run time. For the efficiency of the scenario derivation, we expand the dynamic scenarios into static ones. This is done by replacing the dynamic elements by branches (fork, join) of all the possible scenarios.

For responsibilities with variables, we use a fork to describe a possible branch with respect to range of the variable. For example, Figure 4(a) shows the case of a responsibility "dialAY", where callee Y is a variable. Assume that the range of Y includes B and C. Then two subsequent scenarios are possible, where A calls B or A calls C. So, "dialAY" is expanded into two (static) responsibilities "dialAB" and "dialAC".

Next, the dynamic stub can have multiple submaps to be plugged into. The selection of the submaps is determined at run time by a selection policy that is usually specified in pre-conditions of the submaps. The dynamic stub can be

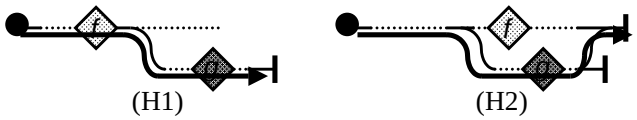

Figure 5. Heuristics H1 and H2

also expanded into a branch of the static stubs (denoted by solid diamonds). Let us see Figure 4(b). In the figure, two submaps def and f can be plugged into the dynamic stub, thus, there are two possible scenarios. So, we expand the scenario into two using a branch with guards (taken from the pre-conditions of the submaps). Then for each scenario, we put a static stub in which a submap f or def is plugged. Also, for the submaps that have post-conditions modifying conditions in a scenario (e.g., $cfbBC_2$), it might be necessary to merge some expanded scenarios. For this situation, we use a join.

4.3 Deriving FI-prone scenarios

After eliminating the dynamic elements on the root map, here we derive FI-prone scenarios in the suspected root map. Note that each feature is characterized a stub configuration and a set of feature submaps. Among all scenario paths in the expanded root map, a feature is activated in a path passing through a static stub with the feature submap. So, scenario paths corresponding to the Observations 1 and 2 in Section 4.1 can be respectively derived by the following heuristics on the expanded root map.

Let f and g be feature submaps of features F and G, respectively.

Heuristic H1: Derive a scenario path that passes through both f and g.

Heuristic H2: Derive a scenario path in which f is bypassed by g, and vice versa.

Figure 5 illustrates Heuristics H1 and H2. As shown in the figure, Heuristic H1 (and Heuristic H2) derives a scenario path corresponding to Observation 1 (and Observation 2, respectively). Based on them, we derive FI-prone scenario paths from the expanded root map.

As an example, let us derive an FI-prone scenario for the combination TCS_C and CFB_B in Section 3.3. The previous method concludes that this combination is FI-prone (see Figure 3). Since $root_A$ has a new stub configuration, we suspect that FI-prone scenarios should be in $root_A$. Next, the dynamic elements of $root_A$ are expanded as shown in Figure 6. Finally, according to Heuristic H2, an FI-prone scenario, depicted by a dotted line, is derived. The derived scenario is interpreted as follows: "B subscribes to CFB forwarded to C, and C subscribes to TCS. If A dials B when B

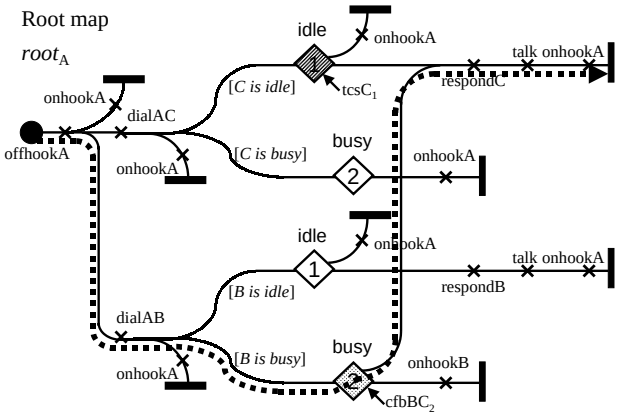

Figure 6. Scenario derivation for TCS_C and CFB_B

is busy, the call is forwarded to C. Finally A can call C without checking procedure of TCS screening list". The derived scenario explains the situation of FI-(b) in Section 2.

Here we should note that the derived scenarios *do not* guarantee the existence of actual FIs. That is, Heuristics H1 and H2 derive just *FI-prone* scenarios. Hence, not all derived scenarios contain FIs, and some of them might be FI-free. The exact FIs will be detected in the FI detection process, which is the next step of FI filtering. The goal of the proposed method is to provide the FI-prone scenarios as an essential information for efficient FI detection.

5 Evaluation

5.1 Preliminary

We have applied the proposed method to specifications of eight features taken from the second FI detection contest [5]. The features include: (1) Call Forwarding on Busy (CFB), (2) Teen Line (TL), (3) Terminating Call Screening (TCS), (4) Reverse Charge (RC), (5) Call Number Delivery Blocking (CNDB), (6) Ring Back when Free (RBF), (7) Voice Mail (VM), (8) Split Billing (SB).

Since the contest specifications are given by Communicating Finite State Machines (CFSMs), we first construct UCMs (root map and default/feature submaps), so that the causal relationships among events are preserved. The UCMs consists of a root map with eight stubs, eight default submaps, and nine feature submaps. For each pair among the eight features, we combined them in the following two ways: (a) both features are allocated to the same user, and (b) different users subscribes to different features. Then, for each combination, we applied the previous filtering method.

Finally, for combinations with (3) FI-prone, we applied the proposed method to derive FI-prone scenarios.

The evaluation is conducted from the following viewpoints:

Filtering quality: how many feature combinations are filtered at FI filtering process.

Scenario coverage: whether the derived scenarios cover actual FIs or not.

As a reference of the actual FIs among the eight services, we used FI detection results submitted by the team of Ottawa University [7].

5.2 Filtering quality

Table 1 shows the filtering result by the previous method only. Each entry of the table represents one of the verdicts: (1) FI occurs, (2) FI does not occur or (3) FI-prone. The *same* (or *diff*) means that two services are allocated to the same users (or different users, respectively), as mentioned in Section 5.1. The shaded entries represent that the combinations cause actual FIs detected in [7].

It can be seen in Table 1 that all combinations causing FIs are covered by the verdicts (1) or (3). However, most combinations have a verdict (3), and no concrete scenario is available at this time. The number of combinations that have strict answer, i.e., (1) or (2), is 14, and 50 FI-prone combinations still have to be examined the subsequent FI detection process. So, it can be said that the filtering quality of the previous method only is 22.9% (=14/64).

On the other hand, Table 2 shows the filtering result, obtained by using the proposed scenario derivation together with the previous method. For the combinations with (3), concrete FI-prone scenarios are derived by Heuristics H1 and/or H2. $H1$ (and $H2$) in the table means that the combination has a scenario derived by Heuristic H1 (and H2, respectively) [3]. $Other$ means that no scenario has been derived by H1 nor H2.

It can be seen in the table that scenarios in Other do not cause actual FIs. Therefore, if we conclude the combinations with Other to be FI-free, then more combinations can be filtered at the filtering process. Since the number of combinations with (1), (2) or Other is 22, and the filtering quality is improved to be 34.4% ($= 22/64$). Thus, by using the proposed method, we do not only get concrete FI-prone scenarios, but also we can expect the improvement the filtering quality.

5.3 Scenario coverage

Next, we conduct a scenario-wise investigation of the result obtained by the previous method with the proposed scenario derivation. Figure 7 represents a set of all scenarios in FI-prone combinations. Among total 74 scenarios investigated, 48 scenarios met Heuristic H1 (depicted by a set H1 in the figure), while 6 scenarios were derived by Heuristic H2 (shown by a set H2). 20 scenarios matched neither H1 nor H2 (described by a set Other). A set FI in Figure 7 represents a set of scenarios containing actual FIs that were identified in [7]. Out of 25 scenarios in the set FI, 20 scenarios were contained in the H1, and 5 scenarios were in the H2.

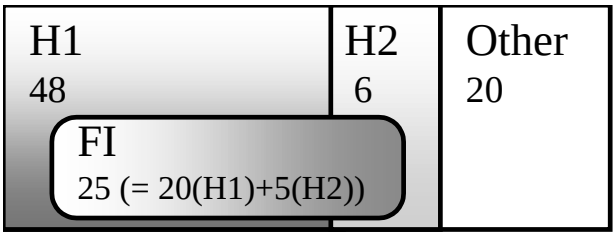

Figure 7. Result of scenario coverage

From the result, it can be seen that all FI scenarios are contained in the FI-prone scenarios derived by Heuristics H1 and H2. None of FI scenarios belong to the set Other. That is, it can be said that Heuristics 1 and 2 sufficiently cover actual FI scenarios in the experiment.

6 Conclusion

This paper proposed a new method to derive FI-prone scenarios, as an extension of the previous FI filtering method with UCMs. Based on the two heuristics, the proposed method derives FI-prone scenarios from root maps of FI-prone service combinations. The experimental evaluation through the FI detection contest showed that the derived scenarios successfully covered all scenarios of actual FIs. It was also observed that, if no FI-prone scenario is derived by the heuristics from a FI-prone combination, then the combination did not cause actual FIs. This fact implies that the heuristics are quite reasonable and that the proposed method can improve filtering quality. After all, the proposed method adds essential information to the previous method, which allows more efficient FI detection.

Our future works are summarized as follows: It will be necessary to establish an efficient framework to use the derived FI-prone scenarios in FI detection process (e.g., test case generation, etc). Also, we plan to apply the proposed method to more practical services, which may reveal more effective heuristics for FI filtering.

[3] Sometimes, multiple scenarios are derived from a combination.

Table 1. Filtering result (previous filtering only)

	CFB		TL		TCS		RC		CNDB		RBF		VM		SB	
	same	diff	same	diff	same	diff	same	diff	same	diff	same	diff	same	diff	same	diff
CFB	-	(3)	(2)	(2)	(3)	(3)	(3)	(3)	(2)	(3)	(1)	(3)	(3)	(3)	(3)	(3)
TL	-	-	-	(2)	(2)	(3)	(2)	(3)	(3)	(2)	(3)	(3)	(3)	(3)	(2)	(3)
TCS	-	-	-	-	-	(3)	(3)	(3)	(2)	(3)	(3)	(3)	(3)	(3)	(3)	(3)
RC	-	-	-	-	-	-	-	(3)	(2)	(3)	(3)	(3)	(3)	(3)	(1)	(3)
CNDB	-	-	-	-	-	-	-	-	-	(2)	(3)	(3)	(3)	(3)	(2)	(3)
RBF	-	-	-	-	-	-	-	-	-	-	-	(3)	(3)	(3)	(3)	(3)
VM	-	-	-	-	-	-	-	-	-	-	-	-	-	(3)	(3)	(3)
SB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	(3)

Table 2. Filtering result (previous plus scenario derivation)

	CFB		TL		TCS		RC		CNDB		RBF		VM		SB	
	same	diff	same	diff	same	diff	same	diff	same	diff	same	diff	same	diff	same	diff
CFB	-	H1	(2)	(2)	Other	H1	H1	H2	(2)	H1	(1)	H1	Other	H1	H1	H2
TL	-	-	-	(2)	(2)	H1	(2)	H1	H1	(2)	H1	H1	H2	H1	(2)	H1
TCS	-	-	-	-	-	Other	H1	H1	(2)	H1	Other	H1	H2	H1	H1	Other
RC	-	-	-	-	-	-	-	Other	(2)	H1	H1	H1,H2	H1	H1	(1)	Other
CNDB	-	-	-	-	-	-	-	-	-	(2)	H1	H1	H1	H1	(2)	H1
RBF	-	-	-	-	-	-	-	-	-	-	-	H1	H1	H1	H1	H1,H2
VM	-	-	-	-	-	-	-	-	-	-	-	-	-	H1	H1	H1
SB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	Other

Acknowledgments

This work is partly supported by Grant-in-Aid for Encouragement of Young Scientists (No.13780234), from Japan Society for the Promotion of Science.

References

[1] Amyot, D., Logrippo L., Buhr, R.J.A. and Gray, T., "Use Case Maps for the capture and validation of distributed systems requirements," *Proc. of Fourth International Symposium on Requirements Engineering (RE'99)*, pp. 44-53, June, 1999.

[2] Buhr, R.J.A., "Use Case Maps as architectural entities for complex systems," *IEEE Transactions on Software Engineering*, Vol.24, No.12, pp. 1131-1155, 1998.

[3] Keck, D.O. and Kuehn, P.J., "The feature interaction problem in telecommunications systems: A survey," *IEEE Trans. on Software Engineering*, Vol.24, No.10, pp.779-796, 1998.

[4] Keck, D.O., "A tool for the identification of interaction-prone call scenarios", *Proc. of Fifth Workshop on Feature Interactions and Software Systems (FIW'98)*, pp.276-290, IOS Press 1998.

[5] Kolberg, M., Magill, E.H., Maples D., and Reiff S., "Second Feature Interaction Contest", Proc. of Sixth Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems (FIW'00), pp.293-310, May 2000.

[6] Nakamura, M., Kikuno, T., Hassine, J., and Logrippo L., "Feature interaction filtering with Use Case Maps at requirements stage", Proc. of Sixth Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems (FIW'00), pp.163-178, May. 2000.

[7] Nakamura M., Ding, T., Sincennes, J., Lu, X. and Logrippo L., "Second Feature Interaction Contest - Contest Report", `http://lotos.csi.uottawa.ca/FI/FIW00/Results/`.

[8] Yoneda, T. and Ohta, T., "A formal approach for definition and detection of feature interactions", *Proc. of Fifth Workshop on Feature Interactions and Software Systems (FIW'98)*, pp.165-171, IOS Press 1998.

[9] Feature Interaction in Telecommunications, Vol. I-VI, IOS Press (1992-2000)