ELSEVIER

Computer Networks 38 (2002) 591–602

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

A pragmatic approach to service interaction filtering between call control services

Mario Kolberg*, Evan H. Magill

Department of Computing Science and Mathematics, University of Stirling, Stirling, UK FK9 4LA

Received 15 November 2000; received in revised form 11 May 2001; accepted 12 October 2001

Responsible Editor: L.G. Bouma

Abstract

Triggered by the deregulation of the telecommunications market, it is expected that the number of services deployed in the telecommunications network will increase dramatically in the near future. It is expected that these services will be interworking. However, as widely reported, interworking services will be subject to the service interaction problem. It is essential to cope with the problem in a developing multi service provider environment as it may substantially delay service deployment and hence form a serious obstacle to rapid service provisioning.

In order to cope with the increasing number of services deployed in the network, previous work suggested the application of filtering approaches. This paper suggests such a technique. It is pragmatic in nature and does not require detailed service knowledge. Furthermore, it is shown how the approach can be incorporated into an overall comprehensive service interaction management process. Even though the approach is applied to call control aspects only, it could be extended to cover other areas. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Telecommunication services; Service interaction detection; Feature interaction; Interaction filtering

1. Introduction

1.1. Motivation

With the deregulation of the telecommunications market, the number of services deployed in the telecommunications networks will increase dramatically in the near future. Most of the services will aim at controlling the setup of connections, i.e. calls. Commonly, they are termed call control services. The increase in numbers is facilitated by initiatives such as JAIN [6] and Parlay [17]. For the first time, third party service providers, independent from the network operators, will be able to provide services. This allows for much more competition in the telecommunications industry. This competition will centre on which business can offer the most suitable service of the highest quality, most quickly. However, in sharp contrast to the new business opportunities, service creation is still a very costly and long process. One of the reasons for this is that extensive validation

*Corresponding author. Tel.: +44-1786-46-7440; fax: +44-1786-46-4551.

E-mail addresses: mko@cs.stir.ac.uk (M. Kolberg), ehm@cs.stir.ac.uk (E.H. Magill).

and testing activities are required. This has partly been addressed by developing new service architectures and APIs [6,17] as well as service creation approaches [14]. In spite of all these activities, one issue has not received sufficient attention—the service interaction problem which occurs when services interwork [3,12,13]. The same problem has also been referred to as feature interaction.

The service interaction problem arises when the behaviour of one service is affected by the behaviour of another service or another instance of the same service. Service interactions can give rise to a multitude of problems, such as user dissatisfaction.

Many service interaction scenarios have been identified in the past [3,7]. One well known example is between the services Originating Call Screening (OCS) and Call Forwarding Unconditional (CFU). OCS blocks outgoing calls to directory numbers which match entries in a database associated with that service. CFU forwards all incoming calls to the subscribers line to a predefined number. If user A has OCS with user C's number in the database and user B has CFU to user C, then if A calls B he is eventually connected to C. Clearly, this is a violation of the OCS service (also cf. Fig. 1).

While this well established example is used for illustration purposes it is pointed out that the goal of the research is to find new, *unknown* interactions. However, for validating new approaches, as required in this paper, well known services which produce well known interactions need to be used so that the performance of the approach can be determined.

Service interactions may involve any number of services, users and/or network components. They impact all phases of the software life cycle and can be categorised in a number of ways [3]. Most commonly, service interactions occur because new services violate original expectations or assumptions of the underlying network or other previously deployed services. Additionally, interactions are caused by the independent design and development of telecommunications services by competing businesses and their different and inconsistent assumptions. Clearly, the issue of service interactions is much more pressing in the deregulated market environment than ever before. This is due to the expected increased number of services deployed on the networks and hence the higher number of interworking services.

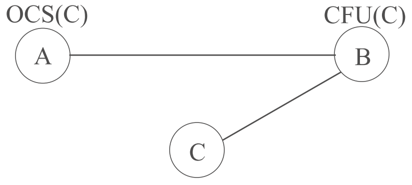

Fig. 1. A well known service interaction scenario.

The detection and resolution of service interactions is a complex problem because:

- the exponential growth of service combinations cannot be contained by manual inspection; automated analysis and resolution techniques are required,
- systems, their requirements, and the technology are evolving very rapidly,
- services are large and evolving software,
- services may be provided by a number of competing vendors,
- services may not be released with specifications.

As outlined above, being able to provide new, enhanced services of high quality quickly and for the best price will distinguish service providers and hence will be an essential business factor in a deregulated telecommunication market [11]. However, service interactions affect the quality of the services perceived by the user. Thus the service interaction problem needs to be addressed. To date, many powerful approaches have been devised [2,7,10]. However, only very few of them have been put into practise because business related issues have often been overlooked [18]. More specifically, many of the approaches require detailed knowledge or even extremely complex formal models of the services involved. Clearly, these approaches are of limited use in the new business environment. However, on the other hand, the ad hoc methods commonly applied in industry today are also insufficient in the future business environment. In other words, many approaches developed so far solve the problem only partially and are not capable of coping with the expected increased number of services. Hence an overall approach embedded in business processes and which combines various techniques to tackle the service interaction problem needs to be applied [9,11]. The European project EURESCOM P509 (Handling Service Interactions in the Service Life Cycle) developed such an overall comprehensive approach to service interaction handling [9]. P509 ran between 1995 and 1997 and partners included major telecommunication companies, such as Deutsche Telekom (Germany), France Telekom (France) and Telia (Sweden).

1.2. The P509 approach

The P509 approach is incorporated into business processes and addresses the factors of a competitive market. P509 defined two business roles called *service integrator* and *service mediator*, to host the processes for the intra- and inter-domain service portfolio, respectively. One of the main novel ideas in the approach is that a separate Service Interaction Handling Process (SIHP) was defined. In other words, Service interaction handling is not part of the service creation process, but rather a part of service integration. The SIHP is divided into four main subprocesses:

1. *Preparation* updates the service descriptions and decides which services are to be analysed in the current run of the SIHP.
2. *Filtering* performs a rough and quick analysis of the service combinations to select the interaction-prone ones.
3. *Detection* carries out a much more detailed analysis of the service combinations to identify service combinations with interactions.
4. *Solving* is concerned with finding solutions to the detected problems.

All the subprocesses are executed sequentially, although, there may be some overlap between filtering and detection as well as detection and solving.

While the SIHP provides a very useful overall framework for interaction handling, the specific algorithms used inside the subprocesses are paramount for the overall success of the approach. Although P509 suggested a number of techniques for each subprocess, it was not a goal of the project to elaborate on them. This is especially true for the filtering algorithms. Moreover, two of their filtering techniques are based on natural language and informal notations. While this information sounds very straightforward to achieve, it tends to be error prone.

Other approaches on filtering include the work by Keck [8], Yoneda and Ohta [19], Heisel et al. [5], and Nakamura et al. [16]. However, especially the latter requires very detailed information. It is based on use case maps and is rather complex. Furthermore, the chosen level of abstraction is very low for a filtering approach. In other words, detailed service knowledge is required. Heisel proposed a heuristic approach to interaction detection. Again, the disadvantage with that work is that formalising the requirements requires detailed knowledge. Keck's technique is aimed at the IN architecture. As with the other two approaches, detailed service knowledge including detection points and service specific data is required. While Yoneda and Ohta's approach does not require a lot of detailed service knowledge they chose a restricted set of services to demonstrate the applicability of approach. They target their approach at active networks and demonstrate the functionality of the approach only by using Single User Single Component (cf. classification in Ref. [3]) interactions. Furthermore, there are open questions with regards to modelling services. Also it appears that new signals are introduced by some services (e.g. Call Waiting). However, a discussion on how these relate to existing ones is unclear. How is a Call Waiting ringing different from a normal POTS ringing?

As explained above, often the highlevel of detail of knowledge required by these approaches cannot be met. In this paper an alternative technique for filtering of call control services is suggested. The main objective was to keep the notations and algorithms as simple as possible to allow for a truly quick analysis of the features. While the technique has been constrained to call control aspects, it is believed to be extendible to other aspects, such as billing. The next section describes the notation and algorithm of the technique. Section 3 presents results obtained in a case study.

2. Analysing call control services

2.1. Outline of the approach

Many approaches developed so far attempt to tackle the whole issue of the interaction problem—and often fail due to the complexity of the task [1,4]. One reason for the complexity is the number of possible scenarios of service combinations which need to be investigated. Thus, the prime motivation behind the approach presented in this paper was to cut down the number of scenarios which need to be investigated in detail. This is achieved by filtering all scenarios which are not interaction prone, that is, scenarios which cannot occur in practise or scenarios where the services do not affect each other. Afterwards, during the detection stage, the potentially problematic service combinations can be investigated more fully; for instance with respect to whether the changes in behaviour are desirable or not.

The approach was not targeted at a low technical level but rather at the user level. In other words, it should detect that certain combinations of services change the behaviour as seen by a user. More specifically, if a different connection to that originally intended is set up, a potential problem is reported. The highlevel view was chosen to form a more pragmatic approach and to allow for a simple notation and algorithm. Abstracting to the user level is believed to be legitimate as practically all low level problems have also a manifestation at a higher one. Some examples of those include triggering two services simultaneously, endless loop with redirection services, or preventing an action which is the goal of the other service. Not using detailed technical knowledge has a positive impact on the applicability of the approach: firstly, it allows for using the approach very early in the development cycle when no detailed descriptions exists, and secondly, it can be applied in a competitive business environment where no detailed technical information will be available. A third important advantage is that the applicability of the approach is not limited to services provided on any specific network architecture. The approach works regardless whether services are deployed in an intelligent network or on top of Parlay or in a Voice over IP environment.

Because of the highlevel nature of the approach, it cannot only be used by programmers and service architects but also by product managers or even by customer facing business consultants. Being able to include the latter group as users of the approach is very important as they are the only people who understand the needs of customers. Hence potential problematic interaction cases can be investigated further in light of what the customer needs.

As discussed in Section 1.2, the filtering process should feed directly into the detection process. By using the filtering approach presented in this paper this can be achieved. As will be described in the following sections, the output of the filtering approach are precise interaction prone call configurations. These include the involved services, the triggering party (TP) of the services, and the connections attempted to establish. This information will narrow the detailed analysis to specific call situations and thus reduce the time required in the detection process.

The next section shows how the services are modelled and describes the syntax and semantics of the used notation. Section 2.3 discusses the algorithm which is applied to those service specifications to find interaction prone service configurations.

2.2. Modelling

The technique assumes call control services to be extensions of the basic call model. That leads to the definition of service interactions as problems between different extensions (services) to the basic call model. As a consequence, only the pure service functionality is modelled, i.e. without the basic call.

The approach aims at detecting interactions in all possible call configurations. To achieve this, three different users are represented in the model: User A, User B and User C. In addition, there is the concept of Treatment which parties can be connected to instead. Treatments are announce-

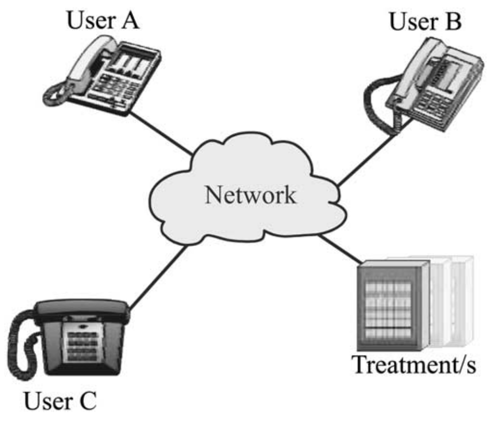

Fig. 2. The network model.

ments or tones triggered by the network to handle certain conditions during a call. For example when a call is screened. Unlike the users which can only have a single instance, treatments can have multiple instances in one call scenario. For instance, two services involved in the same call scenario might intend to connect a party to treatments. However, these two treatments are unlikely to be the same. The network model employed in the approach is depicted in Fig. 2. The shadows behind the icon representing the treatment symbolise possible multiple instances.

Modelling the services consists of two elements: firstly the TP and a connection equation. The latter consists of two parts: the original connection to be set up before the service is activated and the connection set up after the service has been triggered.

An example should make this more illustrative. For instance, the service Call Forwarding Unconditional (CFU) which redirects all incoming calls to a predefined third user, can be described by the connection equation shown in Fig. 3.

The call starts with A attempting to connect to B. However, because of CFU, A is connected to C instead. The TP for CFU is B, because CFU is activated at the terminating side of the original connection. As can be seen, the specification includes only the incremental service behaviour, not the behaviour of the Basic Call Model.

The approach does not consider time. While in the above CFU example the resulting connection is set up immediately after triggering the service, this might not be the case with some other services. For instance the Automatic Ringback (AR) service which is triggered by an incoming call to the subscribers line while he is engaged in another call, only returns the call to the caller after the subscriber is idle again. The approach presented here is to be used for filtering and thus aimed to be as straightforward as possible. However, including time in the models would make them considerably more complex. Hence time is not modelled. Indeed, our results in the case study (Section 3) show, that omitting time in the model is not an issue. Fig. 4 contains the specification for the AR service.

Data such as screening lists or data associated with forwarding services is also not considered in this approach. Such information is regarded to be too detailed for a filtering approach and should only be considered during the detection stage. For the filtering approach, the behaviour of such services is modelled in the way the services behave if the data would lead to triggering the service. For example a screening feature will always screen and block the call. Fig. 5 shows the specification of the service Originating Call Screening (OCS). With OCS all the directory numbers of outgoing calls are screened against a screening list. If a number matches an entry in the list, the call is blocked and the caller is played an announcement.

Interaction prone scenarios are found by applying an algorithm to a pair of service equations. In order to find all possible interactions in all possible call configurations, the three users (A, B, and C) in the model of one service and treatments need to assume the role of each other in the

TP.: B; Conn: (A, B) $\rightarrow$ (A, C)

Fig. 3. Specification of Call Forwarding Unconditional.

TP.: B; Conn: (A, B) $\rightarrow$ (B, A)

Fig. 4. Specification of Automatic Ringback.

TP.: A; Conn: (A, B) $\rightarrow$ (A, Treat)

Fig. 5. Specification of Outgoing Call Screening.

TP.: B; Conn: (A, B) → (A, C)
TP.: C; Conn: (A, C) → (A, B)
TP.: A; Conn: (B, A) → (B, C)
TP.: C; Conn: (B, C) → (B, A)
TP.: A; Conn: (C, A) → (C, B)
TP.: B; Conn: (C, B) → (C, A)

Fig. 6. Variations on the CFU service.

connection equation of the other service. The number of the variations depends on the number of parties in a service pair and can be calculated with the formula $n!$, n being the number of variables, i.e. users. For instance, for a service pair which involves all three users and treatment, $4! = 24$ combinations need to be created. If, however, only two parties are involved, only $2! = 2$ combinations are derived.

Applying this approach to the CFU example, and assuming that the second service does not employ any treatment, six variations as depicted in Fig. 6 can be derived.

If each of these variations are paired with the original equation of another service, potential interactions can be detected by applying a basic algorithm detailed in Section 2.3. It is noted that only the original equation of the second service is required. This is sufficient as the algorithm is commutative.

2.3. The algorithm

With the algorithm the specifications of two services are compared. That is, the approach is targeted at finding interaction prone service *pairs*. As outlined in the previous section, the specification of one service is compared with all the variations of the specification of another service. This means that the parties in the first service assume the role of all the parties in the second service. This way calls with both services in all possible call configurations are covered. Hence, following the equation above the algorithm has to be applied $N!$ times, N being the number of parties involved, to decide if a service pair is not interaction prone.

In the following six rules are presented. If a service pair fulfils any of the six rules the pair is interaction prone. In other words, a service pair is interaction prone if the following equation is true:

$$\text{Interaction Prone} = \text{Rule 1} \vee \text{Rule 2} \vee \text{Rule 3} \vee \text{Rule 4} \vee \text{Rule 5} \vee \text{Rule 6}$$

Rule 1—single user–dual service control: If the original connection of both service specifications is identical and the TP is the same, the pair is interaction prone. Even if the services aim at setting up the same connection, the services might clash as they might be triggered by the same event. An example is given in Fig. 7.

Rule 2—connection looping: The second type of an interaction prone service pair can be recognised by the following behaviour. The original connection of one service is identical with the connection after activating the second service and the original connection of the second service is identical with the connection set up after triggering the first service. As both services are trying to divert the call, a possible endless loop which results in a call deadlock is caused. Fig. 8 gives an example.

Rule 3—diversion and treatment: Two services are also interaction prone if one service redirects the call (not to a treatment) and the resulting connection is the original one of the other service which redirects the call to a treatment. The sources of the original connections need to be the same and the sinks be different. The Triggering Parties of the services have to be different.

This scenario is a potential problem as the redirection service establishes a connection which has a treatment assigned. In other words, the connection which one service is trying to establish may be prevented by the other. Fig. 9 provides an example.

CFB: TP.: B; (A, B) → (A, C)
CFU: TP.: B; (A, B) → (A, C)

Fig. 7. Call Forwarding on Busy and Call Forwarding Unconditional.

CFB: TP.: B; (A, B) → (A, C)
CFU: TP.: C; (A, C) → (A, B)

Fig. 8. Call Forwarding on Busy and Call Forwarding Unconditional revisited.

CFB: TP.: C; (A, C) → (A, B)
OCS: TP.: A; (A, B) → (A, Treat)

Fig. 9. Call Forwarding on Busy and Originating Call Screening.

Rule 4—reversing and treatment: This rule is very similar to the previous one. One service reverses the call and the resulting connection is equal to the original connection of the other service which connects to a treatment. In contrast to the previous rule, the same users need to be involved in both original connections, although their roles (source, destination) may be swapped. Fig. 10 illustrates this case.

Rule 5—diversion and reversing: The scenario here is caused by one service forwarding a call which is subsequently reversed to the originator by the other service. More precisely, one service forwards the call (not to a treatment) and the resulting connection is the original one of the other service which reverses the call. A common manifestation of the problem is the originator of the call is phoned back by a person he never contacted. An example of the configuration is given in Fig. 11.

Rule 6—reversing and diversion: This scenario is closely related to rule 5. However, the forwarding happens after the reversing of the call. Consequently, one service reverses the call and the resulting connection is the original one of the other service which forwards the call to a third party. Fig. 12 illustrates this rule.

Finally, it is important to note that the used models are on a per service base, not per service pair. It is only the algorithm which combines two service specifications to pairs. This greatly reduces the complexity of the specification activity, because when new services are introduced no existing models need to be changed.

AR: TP.: B; (A, B) → (B, A)
OCS: TP.: B; (B, A) → (B, Treat)

Fig. 10. Automatic Ringback and Originating Call Screening.

CFB: TP.: C; (A, C) → (A, B)
AR: TP.: B; (A, B) → (B, A)

Fig. 11. Call Forwarding on Busy and Automatic Ringback.

AR: TP.: B; (A, B) → (B, A)
CFB: TP.: A; (B, A) → (B, C)

Fig. 12. Automatic Ringback and Call Forwarding on Busy.

3. Case study

3.1. Selected services

In order to validate the presented approach a case study was carried out. Twelve widely used services were selected for this purpose. A very important criteria for selecting the services was that they represent a wide variety and a good selection of the type of call control functionality available today. Additionally two services are included which were quoted in a recent study of Feature Interactions in Internet Telephony [15]: Forking and Camp-On. These services were included to demonstrate the applicability of the approach to services deployed on new architectures.

Generally, while call control services will be more advanced in their functionality in the near future, it is expected that they will be based on the services known today. It is expected that future services will be much more flexible and offer a much higher degree of configurability than existing ones. With the convergence of telephony and IP-based networks services will be extended to include added capabilities. For example, call forwarding will be able to forward a call to a webpage or to email. However, the 'base functionality' of call control services, such as forwarding or screening, will remain the same. In other words, future services will exhibit many characteristics of today's services. Hence an approach which can cope with the various types of services today is very likely to achieve a good performance in the future also. The following list provides a short description of the selected services.

Call Forwarding Unconditional redirects all incoming calls to a predefined third Party.

Call Waiting notifies the subscriber about incoming calls while being busy in another call. The subscriber may accept the new call by putting the original one on hold. Subsequently, the subscriber may toggle between the two calls.

Call Forwarding on Busy redirects all incoming calls to a predefined third Party when the subscriber is busy.

Originating Call Screening screens all directory numbers of outgoing calls against a database. If a number matches an entry in the database the call is blocked and the subscriber is redirected to an announcement.

Terminating Call Screening screens all directory numbers of the originating party of incoming calls against a database. If a number matches an entry in the database the call is blocked and the originator is redirected to an announcement.

Voice Mail System redirects the caller to a voice mail treatment when the subscriber is busy.

Automatic Ringback is activated when the subscriber is busy. It returns the call to the caller after the subscriber is idle again.

Automated Callback allows a caller to automatically recall a busy callee, after the callee is gone onhook.

Do Not Disturb redirects all incoming calls to the subscriber's line to an announcement.

Group Ringing allows for an incoming call to ring on an additional phone(s). The phone which goes offhook first is getting connected to the originator. The remaining phone will be idle again. In a SIP environment this functionality is known as *Forking*.

Alarm Phone enables the subscriber to phone a special number and to set the alarm time. At the alarm time the service calls the subscriber.

Camp-On allows the subscriber to periodically retry to connect to a busy destination until the terminating party becomes free.

Following the modelling approach discussed in Section 2.2 the connection equation for each of the services was developed. Table 1 contains the specification of the services.

As can be seen from the table, some services which are actually quite different are modelled in a rather similar way. For instance, the services OCS and TCS differ only in their Triggering Party and the services AR and ACB only in the direction the final connection is set up. Finally, the specifications for the services ACB and CW are identical. This is due to the fact that time is not included in the models and the final connections are the same in both services.

Table 1
Specifications of the case study services

Feature	Triggering party	Equation
CFU	B	$(A, B) \rightarrow (A, C)$
CW	B	$(A, B) \rightarrow (A, B)$
CFB	B	$(A, B) \rightarrow (A, C)$
OCS	A	$(A, B) \rightarrow (A, \text{Treat})$
TCS	B	$(A, B) \rightarrow (A, \text{Treat})$
VMS	B	$(A, B) \rightarrow (A, \text{Treat})$
AR	B	$(A, B) \rightarrow (B, A)$
ACB	B	$(A, B) \rightarrow (A, B)$
DND	B	$(A, B) \rightarrow (A, \text{Treat})$
GR	B	$(A, B) \rightarrow (A, C)$
ALM	Treat	$(A, \text{Treat}) \rightarrow (\text{Treat}, A)$
CON	B	$(A, B) \rightarrow (A, B)$

In the following, the interaction prone scenarios involving the services above are presented. Some interesting cases are discussed in more detail at the end.

3.2. Results

Between the eleven selected services 95 interaction prone scenarios have been found. Clearly, a number of service pairs exhibit interaction prone behaviour in different call configurations. Table 2 provides details on the interaction prone scenarios. The numbers in the cells refer to the rules discussed in Section 2.3. In other words, they show which rules detect a particular interaction prone pair. Multiple numbers in one field refer to multiple interaction prone call configurations involving the same services.

3.3. Discussion of the results

While detecting 95 interaction prone call configurations between twelve services might suggest that not that many configurations have actually been filtered out, this is only true at a superficial

Table 2
Results of the case study

	CFU	CW	CFB	OCS	TCS	VMS	AR	ACB	DND	GR	ALM	CON
CFU		1	1,2	3	1,1,3	1,3	1,5,6	1	1,1,3	1,2	5,5,6,6	1
CW			1		1	1	1	1	1	1		1
CFB				3	1,1,3	1,1,3	1,5,6	1	1,1,3	1,2	5,5,6,6	1
OCS							4			3		
TCS						1	1,4	1	1	1,1,3	6	1
VMS							1,3	1	1	1,1,3	6	1
AR								1	1,4	1,5,6	6	1
ACB									1	1		1
DND										1,1,3	6	1
GR											4,4,6,6	1
ALM												1
CON												

glance. From the twelve services 66 service pairs can be created. This figure excludes pairs consisting of the same services (cf. Table 2). Further, as explained in Section 2.2 up to 24 call configurations can be derived for every service pair. This depends on the number of users involved in the specific case. For the case study 1026 call configurations were derived from the 66 service pairs. Table 3 shows how many configurations could be derived against the number of service pairs. It is noted that domain knowledge was used when automatically creating the call configurations; for instance a service cannot be triggered by a treatment unless the service is specifically designed for it (e.g. Alarm Phone). These special cases are handled in the implementation of the approach—no manual selection is required. This straightforward optimisation is the reason that the number of call configurations per service pair do not necessarily correspond to the equation $n!$

Considering the fact that 1026 call configurations were investigated and 95 interaction prone scenarios were found, it can be argued that the approach considerably cuts down the number of scenarios requiring a more detailed study. In other words, only less than 10% of the derived call configurations need further investigation.

Clearly, the six rules contribute differently to the overall success of the technique. Rule one leads to the detection of 54 interaction prone pairs, the second rule to three, the third to 13, the fourth to five, the fifth to seven, and the sixth to 13. Obviously, rules one, three, and six are the most effective ones. However, as is discussed in the following, these figures are heavily dependent on the nature of the investigated services.

For instance, most of the services are operating on the terminating end of the call. That is, rule one applies to many combinations of services. Further, five services are applying some sort of a treatment to the calling party. Six services are trying to redirect the call to some other location. Combinations of the last two categories of services are filtered out by rule three. Rule six appears very successful as the service ALM matches with other redirecting services and services employing treatments. Rules four and five should apply to combinations including a service which returns a call to the caller. Of this kind of service there are only two in the selection. Rule two in turn requires two forwarding services.

It can be argued that rules four, five, and six are specifically aimed at the services automatic ringback and alarm phone. Indeed, in this case study only service combinations involving these services

Table 3
Details of call configurations for case study

No. of service pairs	Configurations per pair	Sum
15 out of 66 @	24 (3 users + Treatment)	360
30 out of 66 @	18 (3 users + Treatment −optimisations)	540
21 out of 66 @	6 (3 users)	126
		1026

respond to these rules. However, as the goal of the approach is to be applicable to a wide variety of call control capabilities, these rules are necessary. In the future, advanced services, for instance security services, might return calls to the calling party. In other words, the functionality performed by Automatic Ringback will reappear in many services.

A more detailed study of the results of the case study shows that no scenarios which have been filtered out contain service interactions. In other words, no *false negatives* have been found in the case study. This point is fundamental for a filtering approach if its results are to be trusted and hence be useful. On the other hand, all scenarios found to be interaction prone actually exhibit some sort of problematic behaviour. However, often this is subjective to individual users and their expectations. Some examples are discussed in the following.

Fig. 13 shows a scenario where a forwarded call is redirected to a voice mail treatment. Some users might see the call forwarding service as a service to increase their reachability, for instance when they are at a different location. While the voice mail service does not restrict or affect the behaviour of the call forwarding directly, some subscribers to CFB might not expect a voice mail to answer their calls. Also the caller might be surprised to get a voice mail announcement from a party they did not try to reach.

A different scenario is shown in Fig. 14. Here a call which is returned to the originator by the AR service is blocked by Terminating Call Screening. Arguably, the TCS service prevents the AR service from working correctly. However, some users might expect that the call returned by AR is established since they originated the first call.

CFB: TP.: C; (A, C) → (A, B)
VMS: TP.: B; (A, B) → (A, Treat)

Fig. 13. Call Forwarding on Busy and Voice Mail.

TCS: TP.: A; (B, A) → (B, Treat)
AR: TP.: B; (A, B) → (B, A)

Fig. 14. Terminating Call Screening and Automatic Ringback.

CFU: TP.: B; (A, B) → (A, C)
AR: TP.: B; (A, B) → (B, A)

Fig. 15. Call Forwarding Unconditional and Automatic Ringback.

The last example depicted in Fig. 15 is showing another interaction prone scenario which depends on user expectations. Here a call forwarded by CFU is returned to the originator by Automatic Ringback. One option is that the originator might be surprised to receive a call from someone he did not call or does not even know. However, the expectations are completely different if the party the original call was forwarded to is a secretary or even the mobile phone of the originally called person.

Many similar cases can be imagined. For instance between CFU and DND; one could argue that these two services are in direct contradiction to each other because CFU is supposed to allow for an increased reachability of the subscriber while DND restricts reachability. However, in some situations the interworking between the two services might be desirable to both subscribers, e.g. during short private personal meetings.

It is noted that these scenarios are not detected by any specific rule. The scenarios presented above were detected by rules three, four and one, for instance. It is a more general problem, that user's expectations cannot easily be captured in a simple notation. All these scenarios are regarded as interaction prone by the presented approach. In the worst case, they would be *false positives*, scenarios falsely highlighted. For the nature of the presented approach this is perfectly acceptable, as more detailed studies should be carried out during the detection and resolution stages as defined by P509.

The case study also shows that the rules are not overlapping. In fact, all scenarios have only been detected by one rule (double numbers in Table 2 refer to two call configurations involving the same services). Finally, the case study was performed using a C++ implementation of the approach. Results could be obtained immediately after running the program which, for the case study service set, returns instantaneously from execution.

4. Conclusions

The service interaction problem appears at many levels and has many dimensions. They affect the user's impression of the quality of the services and hence cause dissatisfaction with users and providers alike. To make things worse, the number of services deployed on the telecommunication networks will increase dramatically in the near future. To tackle the problem at a higher level and to include it within business processes, the project P509 developed an overall approach to service interaction handling. In their approach they use interaction filtering to cut down the problem space. However, the goal of P509 was to define the overall processes. Consequently, they did not elaborate on individual techniques, and indeed the suggested filtering techniques have some disadvantages.

In this paper, a filtering technique has been proposed which can be used. The approach concentrates on call control aspects. However, it is believed to be possible to extend it to other areas. Central to the approach is modelling the service functionality as connection equations. More specifically, the originally intended connection and the finally established connection after triggering the service are modelled. Applying a straightforward algorithm to a pair of service specifications discards all service pairs which do not exhibit interaction prone behaviour. The remaining scenarios bearing interaction prone characteristics can be investigated further to fully establish whether a service interaction is likely to occur.

Clearly, service interaction handling is about detecting *unknown* interactions. However, as an approach can only verified by using known service interactions, the case study presented in the previous section, concentrated on known services. However, it is believed that the approach can be applied to new services to find interaction prone service configurations. This belief is based on the careful selection of the case study services, to cover a wide spectrum of possible types of services. In fact, the chosen services operate on both sides of the call (originating and terminating end) and span across all major call control functions, such as call diversion, screening, or busy treatment.

The two main contributions in this paper are firstly, the simple and highlevel way to describe telecommunications services, and secondly, the rules to find interaction prone call scenarios.

The highlevel and thus straightforward way of modelling the services allows for the approach to be used by people who are not necessarily familiar with implementation details of the services. This includes most importantly customer facing business consultants. After the filtering stage they can analyse the result with regards to what the customer needs.

The output of the approach feeds directly into the detection process by providing information on the services involved in the particular scenario, the Triggering Parties of the services and the call connections tried to set up. These are important parameters which can directly be used in more detailed approaches.

As far as the case study can show, the approach exhibits a very good performance. As the most important criteria for a filtering approach, no *false negatives* have been found. These are scenarios which contain service interactions but were not marked as interaction prone by the approach. Moreover, all scenarios marked as interaction prone exhibit behaviour which contains some problematic aspects. Some users might expect such behaviour, some other might find it surprising or disturbing. Establishing solutions for those scenarios is part of the detection and solution stages as defined by P509.

While the work presented in this paper is focussing on pairs of services and also on pairwise connections only, it is recognised that removing these restrictions is important. Thus future work will include applying the approach to services which establish multiple connections, such as three-way-calling or conference calls. Further, the adaption of the approach to triple service interactions will be investigated.

References

[1] M. Calder, What use are formal design and analysis methods to telecommunications services, in: K. Kimbler, L.G. Bouma (Eds.), Feature Interactions in

Telecommunications and Software Systems V, IOS Press, Amsterdam, 1998, pp. 23–31.

[2] M. Calder, E.H. Magill (Eds.), Feature Interactions in Telecommunications Software Systems VI, IOS Press, Amsterdam, 2000.

[3] E.J. Cameron, N.D. Griffeth, Y.-J. Lin, M.E. Nilson, W.K. Schnure, H. Velthuijsen, A feature interaction benchmark for IN and beyond, in: W. Bouma, H. Velthuijsen (Eds.), Feature Interactions in Telecommunications Systems, IOS Press, Amsterdam, 1994, pp. 1–23.

[4] C. Capellmann, P. Combes, J. Pettersson, B. Renard, J.L. Ruiz, Consistent interaction detection—a comprehensive approach integrated with service creation, in: P. Dini, R. Boutaba, L. Logrippo (Eds.), Feature Interactions in Telecommunications and Distributed Systems IV, IOS Press, Amsterdam, 1997, pp. 183–197.

[5] M. Heisel, J. Souquieres, A heuristic approach to detect feature interactions in requirements, in: K. Kimbler, L.G. Bouma (Eds.), Feature Interactions in Telecommunications and Software Systems V, IOS Press, Amsterdam, 1998, pp. 165–171.

[6] JAIN, Java APIs for integrated networks, for more information on JAIN see: http://java.sun.com/products/jain.

[7] D.O. Keck, P.J. Kühn, The feature and service interaction problem in telecommunications systems: a survey, IEEE Transactions on Software Engineering 24 (10) (1998) 779–796.

[8] D.O. Keck, A tool for the identification of interaction-prone call scenarios, in: K. Kimbler, L.G. Bouma (Eds.), Feature Interactions in Telecommunications and Software Systems V, IOS Press, Amsterdam, 1998, pp. 276–290.

[9] K. Kimbler, C. Cappelmann, H. Velthuijsen, Comprehensive approach to service interaction handling, Computer Networks and ISDN Systems 30 (15) (1998) 1363–1387.

[10] K. Kimbler, L.G. Bouma (Eds.), Feature Interactions in Telecommunications and Software Systems V, IOS Press, Amsterdam, 1998.

[11] M. Kolberg, K. Kimbler, Service interaction management for distributed services in a deregulated market environment, in: M. Calder, E.H. Magill (Eds.), Feature Interactions in Telecommunications and Software Systems VI, IOS Press, Amsterdam, 2000.

[12] M. Kolberg, E.H. Magill, Service and feature interactions in TINA, in: K. Kimbler, L.G. Bouma (Eds.), Feature Interactions in Telecommunications and Software Systems V, IOS Press, Amsterdam, 1998, pp. 78–84.

[13] M. Kolberg, R.O. Sinnott, E.H. Magill, Engineering of interworking TINA-based telecommunication services, Proceedings of Telecommunications Information Networking Architecture Conference, Oahu, Hawaii, 1999.

[14] M. Kolberg, R.O. Sinnott, E.H. Magill, Experiences modelling and using formal object-oriented telecommunication service frameworks, Computer Networks 31 (23–24) (1999) 2577–2592.

[15] J. Lennox, H. Schulzrinne, Feature interactions in Internet telephony, in: M. Calder, E.H. Magill (Eds.), Feature Interactions in Telecommunications and Software Systems VI, IOS Press, Amsterdam, 2000, pp. 38–50.

[16] M. Nakamura, T. Kikuno, J. Hassine, L. Logrippo, Feature interaction filtering with use case maps at requirements stage, in: M. Calder, E.H. Magill (Eds.), Feature Interactions in Telecommunications and Software Systems VI, IOS Press, Amsterdam, 2000, pp. 163–178.

[17] Parlay Working Group, Parlay Specification 1.2, available from http://www.parlay.org/products.html.

[18] S. Tsang, E.H. Magill, The network operator's perspective: detecting and resolving feature interaction problems, Computer Networks and ISDN Systems 30 (15) (1998) 1421–1441.

[19] T. Yoneda, T. Ohta, The declarative language STR (state transition rule), in: S. Gilmore, M. Ryan (Eds.), Language Constructs for Describing Features, Springer, Berlin, 2001, pp. 197–212.

Mario Kolberg studied Computer Science at the HTWS Zittau/Goerlitz in Germany. After spending one year in the Computer Science Department at Humboldt University in Berlin, Mario joined the Communications Division in the Department of Electronic and Electrical Engineering at the University of Strathclyde in June 1997. While at Strathclyde, he mainly worked in the European Union funded ACTS TOSCA project on issues related to the rapid development of distributed telecommunications services based on the TINA architecture. In September 2000 Mario moved to the Department of Computing Science and Mathematics at Stirling University where he became a member of academic staff.

Recently, his work concentrated on the EPSRC SEBPC (Systems Engineering for Business Process Change) project. His research interests include service creation, feature interaction management, and networked appliances. Mario is a member of the Telecommunications Service Engineering Research Group. In parallel to his occupation as a Lecturer, Mario is finalising his Ph.D. in the area of feature interaction management.

Evan Magill, following almost a decade in industry both in Canada and the UK, returned to University to gain a Ph.D. in Electronic and Electrical Engineering in 1991 from the University of Strathclyde, Glasgow. His work focussed on the automatic analysis of SDL specifications. Afterwards he became a Lecturer and later Senior Lecturer in the same Department. In 2000 he joined the Department of Computing Science at the University of Stirling as Professor.

His main research interests include telecommunications service creation, and in particular feature interaction. He has maintained links with a number of companies through research projects covering telecommunications resource allocation, SDL, feature interaction, and service creation. He is a member of the Telecommunications Service Engineering Research Group in the Department.