A Feature Interaction Benchmark for IN and Beyond

E. Jane CAMERON, Nancy D. GRIFFETH, Yow-Jian LIN
Margaret E. NILSON, William K. SCHNURE
Bellcore, 445 South Street, Morristown, NJ 07962-1910, USA
Hugo VELTHUIJSEN
PTT Research, St. Paulusstraat 4, 2264 XZ Leidschendam, The Netherlands

Abstract. Rapid creation of new services for telecommunications systems is hindered by the *feature interaction* problem. This is an important issue for development of IN services, not only because of interactions among IN services themselves but because of interactions of IN services with switch-based services and potential interactions with services not yet developed. Furthermore, the problem is fundamental to service creation; it is not restricted to IN services. Any platform for telecommunication services requires a method for dealing with the feature interaction problem. A number of approaches for managing feature interactions have been proposed. However, lack of structured ways to categorize feature interactions makes it difficult to determine if a particular approach has addressed some, if not all, classes of interactions.
We describe and analyze a number of feature interactions by using two independent classification schemes. This paper is a step to achieving the goal of a coherent industry-wide collection of illustrative features and their interactions. The collection will help convey the scope of the feature interaction problem. It will also serve as a benchmark for determining the coverage of various approaches, and as a guideline for identifying potential interactions in software architectures and platforms.

1 Introduction

This paper outlines two methods for categorizing feature interactions. The goal is to improve understanding of the scope of the feature interaction problem in telecommunications systems and to analyze the coverage of approaches to solving the problem.

Features in a telecommunications system are packages of incrementally added functionality providing services to subscribers or the telephone administration [7]. For the purposes of discussion in this paper, the meaning of feature will be interpreted broadly. Some features, such as *Plain Old Telephone Service (POTS)*, offer basic means of setting up conversations and billing for services,[1] whereas others, such as *Call Forwarding*

[1]We consider POTS as a feature, so that we can address all interactions in a uniform way even when POTS is involved.

or *Call Waiting*, modify the basic telephone service that is offered to subscribers. Still others, such as those features and services offered by *ISDN* rely on sophisticated transport protocols to carry signals to and from subscribers, thus increasing the signaling capabilities and modifying the software protocols for controlling calls. In this paper, we will not make a distinction between the terms *feature* and *service*.

Feature interactions are understood to be all interactions that interfere with the desired operation of the feature and that occur between a feature and its environment, including other features or other instances of the same feature. Additionally, interference of one part of a feature with another part of that feature (e.g., in case of a distributed implementation of a feature) is considered to be a feature interaction.

In order to add a new feature to a telecommunications system, its interactions with the telecommunications system and other existing features must be managed in some fashion. A number of approaches have been proposed to address the feature interaction problem [2, 4, 6, 8, 9, 10]. While the increasing attention paid to this problem is encouraging, many approaches that claim to have addressed the problem fully in fact only solve a part of it. This is not surprising, since the problem itself is often vaguely defined through the use of instances of potential interactions or definition is avoided altogether. There has been no coherent collection of feature interactions available to illustrate the full range of the problem. This paper provides such a collection and proposes a taxonomy for categorizing it. This taxonomy is intended to serve as a benchmark for determining the coverage of various approaches.

The problem of feature interactions has created difficulties in the process of service deployment. Recent and foreseeable changes in the telecommunications industry complicate the problem further. Important changes include:

- the process of service creation is no longer largely governed by a single organization as it was prior to the divestiture of Bell System;
- the platform for an intelligent network promotes independent and rapid deployment of customized features by operating companies and their associated suppliers and independent information providers;
- the telecommunications network is becoming increasingly heterogeneous, with equipment and support software provided by several suppliers; and
- the service logic for controlling call processing is becoming increasingly distributed, with service logic programs distributed among various network components.

When a large number of features and equipment deployed by numerous suppliers must work together, the traditional approach of a manual feature-by-feature analysis is no longer feasible. Clearly, more powerful techniques for the management of feature interactions are needed.

In this paper, we present two different ways of categorizing feature interactions: by the *nature* of the interaction and by the *cause* of the interaction. The categorization by the *nature* of the interaction is defined by the following three dimensions: the *kind of features* involved; the *number of users* involved; and the *number of network components* involved. The categorization by the *cause* of the interaction includes: violation of assumptions about the system; limitations on network support; and problems intrinsic to large, heterogeneous distributed systems. An illustrative collection of feature interactions is presented, categorized by the two classification schemes, and cross-referenced.

However, interactions involving system security, mixed media, and administrative features are not discussed.[2]

We find that our taxonomy is illuminating when attempting to analyze various proposals that address the feature interaction problem. By identifying and classifying various issues regarding the feature interaction problem, we are now able to port ideas from different areas of computer science to address the appropriate issues (see also Sections 4 and 5). In addition, we believe that this benchmark can also serve as guidance for the development of more powerful service creation tools and environments. Note that the feature interaction problem is interpreted more broadly in this paper than is usually done elsewhere. Traditionally, the feature interaction problem only refers to situations where proper interworking of a group of features is not possible. Here we also include situations where the resulting joint behavior is not what the user would reasonably expect from understanding each individual feature alone. Such situations could arise because some feature descriptions are incomplete or ambiguous, or because the user does not envision all the implications of activating one feature, or because the user is not aware of some subtle differences in telecommunication terminology (e.g., that the directory number (DN) and the dialed number are not necessary the same).

In order to keep the examples simple and focused on the problem, we also elected to use features that are widely deployed (hence familiar to many telephone users) for illustrating feature interactions. We will briefly describe the features when they are used. The names and descriptions of these features are based on the collection of feature definitions in [1]. Some examples presented in this paper have actually been resolved in the current public network; therefore, they may not appear to be instances of the problem to some users. Nevertheless, it is worthwhile to recognize them as interactions, since the interactions have been handled case-by-case, without using or producing a global set of guidelines for handling interactions.

The paper is organized as follows. Section 2 presents a categorization of feature interactions by their *nature*. Also, all of our prototypical examples are introduced in this section. Section 3 provides a categorization of feature interactions by their *cause(s)*, referring back to the examples of Section 2. Section 4 summarizes our categorizations and indentifies potential approaches in managing feature interactions. The paper ends with some concluding remarks in Section 5.

2 Categorization of the Nature of Interactions

This categorization has three dimensions: the kind of features involved in the interaction, the number of users involved in the interaction, and the number of network components in the interaction.

The first dimension distinguishes interactions that involve only *customer features* from interactions that involve *system features*, instead of or in addition to customer features. Customer features include all of the call processing features visible to the general public, such as *Call Waiting*, *Call Forwarding*, *800 services*, features offered by *CENTREX*, and so on. System features include billing and other Operations, Administration, and Maintenance (OA&M) services. Interactions can occur among customer features; among customer features and system features; or among system features.

[2]These categories probably hold for these kinds of features as well as they do for the features presented here. Presenting a truly comprehensive set of features would take far too much space, without adding significantly to the substance of the benchmark.

The second dimension distinguishes *single-user* interactions from *multiple-user* interactions. Single-user interactions arise when different features, while simultaneously activated by a single user, interfere with each other, whereas multiple-user interactions arise when features activated by one user interfere with those activated by another user. The user of a feature is the one whose call processing logic includes the functionality of the feature; the user of a feature could be different from the subscriber to a feature, i.e., the one who pays for the feature. As an example, the municipal government that provides *911* emergency services to its community is the subscriber of *911*, but the user of *911* is the one who makes a *911* call: the logic of *911* is incorporated into the call processing logic of every line in the municipality. As another example, the user of the residential *Call Forwarding* feature can only be its subscriber: the logic of *Call Forwarding* is part of the call termination treatment associated with the subscriber only.

The third dimension distinguishes between *single-component* interactions, which arise when only one network component (switching system, adjunct, SCP, services node, CPE) is involved in the processing, and *multiple-component* interactions, which arise when features supported on one network component interfere with the operation of features supported on another network component. Multiple-component interactions are becoming more common as features are increasingly being supported by a distributed architecture. These interactions are especially difficult to resolve, since they may require distributed control or inter-supplier collaboration to get features to interoperate correctly.

In this section, the examples of feature interactions are presented in the following order:

- SUSC (Single-User-Single-Component) interactions among customer features;
- SUMC (Single-User-Multiple-Component) interactions among customer features;
- MUSC (Multiple-User-Single-Component) interactions among customer features;
- MUMC (Multiple-User-Multiple-Component) interactions among customer features;
- CUSY (CUstomer-SYstem) interactions among customer features and system features.

Since we found many more examples of interactions among customer features, we focus in this paper mainly on interactions that interfere with the expected functioning of customer features. The full range of choices among the number of users and number of components is addressed only for interactions among customer features. Some interactions among customer features and systems features (which are mostly multi-component interactions) are also discussed. Interactions that involve only system features are not addressed at all in this paper.

Some multiple-user interactions can be either MUSC type or MUMC type, depending upon whether all users involved are associated with a single network component. In our categorization we put all such multiple-user interactions in the MUMC category; only those multiple-user interactions that arise specifically in a single network component are categorized as MUSC interactions. By the same token, some single-user interactions can be either SUSC type or SUMC type, depending upon whether all features involved are deployed in a single network component or distributed among several. In this case, we put all such single-user interactions in the SUSC category; only those single-user interactions that explicitly involve multiple network components are put in the SUMC category. Of course, with a distributed architecture such as that of AIN, many SUSC interactions could also have been defined as SUMC interactions.

In most cases, we will give several examples for the same category of feature interactions (according to the present categorization). In Section 3, these examples will be used to illustrate a further categorization of feature interactions by their causes.

2.1 SUSC Interactions

SUSC (Single-User-Single-Component) interactions occur because incompatible features are simultaneously in use by a single user in a single network component such as a switching element or a service control point (SCP). Some of these interactions arise because of *functional ambiguities* between the features (that is, two different features are designed to deal with the same call processing situation, but differ as to how it will be handled), or because of *interferences* (that is, one feature will preclude the proper execution of another feature). Other of these interactions arise because of resource limitations or signaling limitations.

There are many combinations of features that produce functional ambiguities. Features such as *Call Waiting*, *Call Forwarding*, *Answer Call* (or *Voice Mail*), and *Automatic Recall* provide alternatives that enable the calling party to contact the called party on calls that would otherwise be unsuccessful. A person who subscribes to more than one such feature could encounter interactions when features compete for call control.

Example 1 *Call Waiting* and *Answer Call.*
When a call attempts to reach a busy line, *Call Waiting* generates a call-waiting tone to alert the called party, whereas *Answer Call* connects the calling party to an answering service. Suppose that **A** is a subscriber of both features. If **A** is already on the line when the second call comes in, should **A** receive a call-waiting tone or should the second call be directed to the answering service?
□

Of course, these interactions can be resolved simply by establishing precedence relations among conflicting features. The resolution can be either determined for all subscribers during service creation or personalized during provisioning. Note that this is not to be confused with "signal ambiguities" (cf. Example 2) which can be resolved by a richer set of functional signals.

Example 2 *Call Waiting* and *Three-Way Calling.*
The signaling capability of customer premise equipment (CPE) is limited. As a result, the same signal can mean different things depending on which feature is anticipated. For example, a flash-hook signal (generated by hanging up briefly or depressing a 'tap' button) issued by a busy party could mean to start adding a third party to an established call (*Three-Way Calling*) or to accept a connection attempt from a new caller while putting the current conversation on hold (*Call Waiting*). Suppose that during a phone conversation between **A** and **B**, an incoming call from **C** has arrived at the switching element for **A**'s line and triggered the *Call Waiting* feature that **A** subscribes to. However, before being alerted by the call-waiting tone, **A** has flashed the hook, intending to initiate a three-way call. Should the flash-hook be considered the response for *Call Waiting*, or an initiation signal for *Three-Way Calling*?
□

A similar situation occurs when lifting a handset is interpreted as accepting the incoming call, even though the user's intention in doing so is to initiate a call—remember the cases when one picks up the phone in the absence of ringing and somebody is already on the other end of the line. The call processing is behaving just as it was designed to, but some users may be momentarily puzzled.

Call Waiting and *Three-Way Calling* can also interact due to resource limitations: since both features need a three-way bridge, according to [1] one feature is automatically disabled once the bridge is used by the other. Nevertheless, as a later example (cf. Example cwtwc2) will show, even when the simultaneous use of both features are possible, more confusion could be introduced.

Now we present some interference examples where one feature will not function properly in the presence of another feature.

Example 3 *911* and *Three-Way Calling.*
A *Three-Way Calling* subscriber must put the second party on hold before bringing a third party into the conversation. However, the *911* feature prevents anyone from putting a 911 operator on hold. Suppose that **A** wishes to aid a distressed friend **B** by connecting **B** to a *911* operator using the *Three-Way Calling* service. If **A** calls **B** first and then calls 911, **A** can establish the three-way call, since **A** still has the control of putting **B** on hold before calling 911. However, if **A** calls 911 first, then **A** cannot put the *911* operator on hold to call **B**; therefore **A** cannot make the three-way call.

Assume that **A** has made the three-way call according to the former scenario. Whether the *911* operator has the control over the entire call remains ambiguous: it appears that the *911* operator does not have control over **B**, even though the operator has control over **A**. Therefore, the operator cannot prevent **B** from dropping out of the call.
□

The main reason for preventing a caller from dropping out of a 911 call is to facilitate tracing the origin of the call. Since the ID of a caller can be delivered in an advanced network, the "no interruption" policy of *Emergency 911* could be replaced by a mandatory *Calling Number Delivery.* Consequently, the above-mentioned interaction regarding setting up a three-way call with a 911 operator can be resolved. On the other hand, which ID gets delivered becomes a new issue. For example, can the *911* operator see the ID of **B**, in addition to that of **A** who sets up the three-way call?

Example 4 *Terminating Call Screening* and *Automatic ReCall.*
Terminating Call Screening assumes that every incoming call will be screened against the incoming call screening list. However, *Automatic ReCall* (i.e., automatically returning the last incoming call), if processed before *Terminating Call Screening* when the line is busy, could just register the incoming call number without having the number screened. Suppose that **A** has both features, and **B**'s number is among those that **A** refuses to accept via the *Terminating Call Screening* feature. Now, if **B** calls **A** while **A**'s line is busy, then when **A**'s line becomes idle the *Automatic ReCall* feature will initiate a connection attempt back to **B**, which nullifies the purpose of the *Terminating Call Screening* feature.
□

The AIN framework [4] requires local switching elements to trigger service logic programs stored in SCPs, adjuncts, or other network components. Restrictions imposed by suppliers on the triggering capabilities of some switching elements can cause interactions, as we will show in the next example.

Example 5 *Originating Call Screening* and *Area Number Calling.*
Originating Call Screening aborts attempts to connect the subscriber to directory numbers in the screening list, whereas *Area Number Calling* decides the actual terminating number based upon the originating number and the dialed number. Each of them needs to launch a query for call origination treatment during call set up. Suppose that **A**, who is a subscriber of the *AIN Originating Call Screening* feature, calls Domino's area number to order a pizza. If the switching element imposes a restriction on the number of queries per call, e.g., one query per call, then after launching a query for the *Originating Call Screening* feature that **A** subscribes to, the component will not be able to launch another query for the *Area Number Calling* feature that Domino's Pizza pays for to serve its customers.
□

The reason for having an upper bound on the number of queries per call is to ensure that call origination/termination treatment will not go into an infinite loop. However, just how many queries per call is sufficient is unclear. Two queries per call may be enough for the above-mentioned situation, but is too low if **A** also has an AIN *Speed Calling* feature that triggers another query to get the dialed number translated.

2.2 SUMC Interactions

SUMC (Single-User-Multiple-Component) interactions form an increasingly important problem of coordination in a telephone network where features accessible to one customer are deployed in different network components. Making sure that features residing in different network components behave correctly is not straightforward; careful consideration must be given to the information that should be shared among the features, and to the order of execution. An interaction arises when the existence of one feature is not known to or has not been considered by designers of features in other network components, resulting in the loss of some feature functionality. The following is one such interaction.

Example 6 *Operator Services* and *Originating Call Screening.*
Operator Services may be handled in a remote switching element that does not have access to the feature subscription profile of every customer who wishes to use the services. Therefore, every call made through *Operator Services* acts like an outgoing POTS call, except that it is operator-assisted. Suppose that **A** has subscribed to the *Originating Call Screening* feature deployed in a local switching element, hoping to screen outgoing calls made to any number in a screening list. However, since the remote switching element does not have the screening list, it is possible for anybody to make an operator-assisted 0+ or 0− call to a screened number by **A**'s line.
□

Timing is also critical to the correct inter-working of features deployed in multiple network components. During call processing, the thread of control[3] can switch from a

[3]We assume that each feature defines a computation thread, and all activated features constitute a group of concurrent threads. The thread of control is in one feature if the system is executing the code corresponding to that feature.

feature controlling one stage of call processing to another feature controlling another stage. When two features are deployed in different network components the transition time from one thread to another may be significant. However, the transfer of control thread may be hidden from the customer. In many cases a customer may send a signal (an off-hook, a "#", or some digit) too early or too late; consequently, the signal is not interpreted by the feature the customer expected. Here is a situation that illustrates this point.

Example 7 *Credit-Card Calling* and *Voice-Mail service.*
Instead of hanging up and then dialing the long distance access code again, many credit-card calling services instruct callers to press [#] for placing another credit-card call. On the other hand, to access voice mail messages from phones other than his/her own, a subscriber of some *Voice-Mail service* such as *Aspen* can (1) dial the Aspen service number, (2) listen to introductory prompt (instruction), (3) press [#] followed by the mailbox number and passcode to indicate that the caller is a subscriber, and then (4) proceed to check messages. However, when a customer places a credit-card call to Aspen, the customer does not know exactly when the *Credit-Card Calling* feature starts passing signals to Aspen instead of interpreting them itself. Suppose that **A** has frequently called Aspen and knows how to interact with Aspen. When **A** places a credit-card call to Aspen, **A** may hit [#] immediately without waiting for the Aspen's introductory prompt. However, the [#] signal could be intercepted by the credit-card call feature; hence it is interpreted as an attempt to make a second call.
□

Timing is not the only factor that contributes to the above problem; limited signaling capability causes the two features to choose the same signal for customer communication. Two obvious solutions present themselves: synchronizing call processing with the user to avoid timing problems, or expanding the signaling capabilities to disambiguate the user's intention. However, even when the signal set is expanded, if features are supplied by many independent suppliers, it will be difficult to resolve timing problems or to make sure that signals are used unambiguously.

Because of the trend toward a distributed architecture for controlling call processing, the number of possible interactions in this category is increasing. One such trend is the deployment of AIN features in SCPs and adjuncts, separate from the switch-based features that reside in local switching elements. Currently, local switching elements serve as the front-end for the subscribers to the telecommunications network. Unless a local switching element can coordinate the execution of any possible subset of features subscribed to by a user,[4] regardless of their distribution, many more SUMC-type feature interactions will occur between local switching elements and SCPs or adjuncts. The following interaction between an AIN feature and a non-AIN feature is related to how a local switching element chooses to query an SCP or an adjunct.

Example 8 *Multi-location Business Service-Extension Dialing (MBS-ED)* and *CENTREX*
The *AIN Release 0 MBS-ED* feature allows a customer to extend a 4 or 5 digit extension dialing plan to locations served by different switching elements. This is accomplished by querying an SCP or an adjunct to translate digit combinations to directory numbers. *CENTREX* features are also based upon a 4 digit extension dialing plan, but are

[4]Other possibilities are to have the coordination done at some other network components (e.g., SCPs), or to implement some distributed coordination mechanism across the network.

served by a single switch. Thus, *CENTREX* features do not query an SCP or adjunct, as the corresponding directory numbers must be local to that switch. When a 4-digit number is received in a switch that supports both *AIN Release 0 MBS-ED* and *CENTREX* features, it is not clear whether an SCP should be queried or not. Assignment of disjoint subsets of the 4-digit numbers to the *MBS-ED* features and the *CENTREX* features could be used by the switch to detect what type of feature is in effect whenever a 4-digit number is received.
□

2.3 MUSC Interactions

MUSC (Multiple-User-Single-Component) interactions can occur when two or more customers access the features associated with a physical line. This type of problem can usually be avoided if the group of customers that share the features carefully work out a consistent use of features. But as the network architecture evolves, new kinds of MUSC interactions will arise. Personal Communication Services (PCS) [5], while providing mobility based on dynamic binding[5] of subscribers to CPEs, introduce a new set of MUSC interactions when features of multiple customers compete for the same physical resources. Examples in this section illustrate MUSC interactions in the present and in foreseeable telecommunication networks.

Example 9 *Call Forwarding* and *Originating Call Screening.*
Call Forwarding allows incoming calls to be redirected to another directory number, while *Originating Call Screening* aborts attempts to connect the subscriber to some other directory numbers. It may seem unlikely that anyone would intentionally block calls to a directory number and then forward calls to the same number, but this situation can arise if the forwarding number and the call screening number were supplied by two different people sharing the same physical line. For example, if an adolescent knows that his parents have used *Originating Call Screening* to block all calls to a dial-porno number **X** from their line **A**, he may instruct the switching element to forward all calls terminated at **A** to **X**, and then call himself to get the effect of *Call Forwarding.* Subscribers of *Originating Call Screening* will not be satisfied if a loophole like that exists. Whether the forwarding number is considered a dialed number (to be checked against the screening list) becomes an issue. Supposing that *Call Forwarding* takes precedence over *Originating Call Screening*, calls can be forwarded to the forwarding number despite the fact that the number is also on the screening list.
□

Personal Communication Services are attractive, but the mobility and convenience of PCS also introduce new kinds of MUSC interactions. For one thing, since every customer can choose to associate services with any CPE, resource contention is a potential problem. There could be many more interactions to be resolved. For example, when multiple PCS customers are currently associated with a single CPE, some switch-based features may not work properly based upon the status of the CPE (or the physical line connected to the CPE) alone, as we show in the following.

[5]Part of setting up mobile services is to determine the location of the subscriber/user. Hence, the particular CPE that will receive this subscriber's calls must be determined on a per call basis. This is different from the more traditional static binding, where the location of the CPE designated to receive calls for a given subscriber/user is stored in a table. Once the location is stored, it will be used for all calls, until it is replaced.

Example 10 *Call Waiting* and *Personal Communication Services*
Call Waiting is a feature assigned to a *directory number*. However, *Call Waiting* uses the status of the *line* with which the number is associated to determine whether the feature should be activated: at present in a public switched telephone network, if a non-ISDN line is in use, then it is busy; a second call to the same line will trigger the switching element to send out a call-waiting tone. PCS customers may not all be subscribers of *Call Waiting*. Suppose that **X** and **Y** are both PCS customers currently registered with the same CPE; **X** has *Call Waiting* but **Y** does not. We further assume that **Y** is on the phone when somebody calls **X**. Since **X** has *Call Waiting* and the line is busy, the new call triggers the *Call Waiting* feature of **X**. But is it legitimate to send the call-waiting alert through the line to interrupt **Y**'s call? If not, then **X**'s *Call Waiting* feature is ignored.
□

2.4 MUMC Interactions

MUMC (Multiple-User-Multiple-Component) interactions can occur when two or more users access features supported on multiple network components.

Consider that POTS is a basic communication protocol for telephone users making phone calls, and additional features subscribed to by each user vary the POTS protocol a bit. Since two customers can subscribe to different sets of features, potentially any customer may need to communicate with another customer whose protocol is somewhat different. Identifying and classifying these variations is an important step to designing a software infrastructure that enables these more complex variations of the basic protocol to correctly interoperate. The types of interactions that can arise depend on how a feature alters the logic of call control, or the environment or assumptions needed for another feature to function as expected. Here we begin with some examples that are caused by conflicting assumptions about the use of dialed numbers, directory numbers, or physical lines as the identifier of customers.

Example 11 *Originating Call Screening* and *Multiple Directory Number Line with Distinctive Ringing.*
Originating Call Screening is a feature based on directory numbers; any call placed to a directory number on the screened list will be blocked. Disallowing calls to a directory number prevents connections to the identified line, provided the line is associated with only that directory number. However, services for *Multiple Directory Number Line with Distinctive Ringing (MDNL-DR)* allow more than one directory number to be associated with a single line. Suppose that **B** is a subscriber of the *MDNL-DR* service with two numbers **X** and **Y**, and **A** has the *Originating Call Screening* service with the number **X** in the outgoing call screening list. **A** can still make calls to **B**'s line, if **A** dials **Y**.
□

It can be argued that *Originating Call Screening* is doing what it was designed to, i.e., screening only directory numbers in the list. Therefore, the above example is not a feature interaction. The question raised here, though, is whether customers may misinterpret the intention of the feature when they subscribe to it. Surely, in the above example **B** could have two physical lines instead of *MDNL-DR*, and the scenario will be the same. To avoid any connection to **B**, both directory numbers must be in the screening list. However, in practice a user of *Originating Call Screening* should not have the false expectation that the feature can be used to block any connection to a particular person, because the feature can only screen based on directory numbers; said person may answer calls on some other "unblocked" line, especially if this person is a PCS user.

Example 12 *Originating Call Screening* and *Call Forwarding. (revisited)*
Originating Call Screening blocks calls based on the number dialed; thus, calls to a particular line are blocked only if the dialed number is associated with that line. However, *Call Forwarding* connects to a line other than the one associated with the dialed number. If **B** forwards all incoming calls to the number **X**, connections from **A** to the line identified by **X** will be established, when **A** calls **B**, even if **A** has **X** on the outgoing call screening list.
□

In a distributed environment such as a telecommunications system, lack of a consistent global view of call control among the parties involved in a call often causes interactions as well. The inconsistency may be due to inability to convey the line status, as the next example illustrates; or it may be caused by competing control relationships, as the ones to be discussed in Example 14 and 15.

Example 13 *Call Waiting* and *Automatic CallBack.*
Automatic CallBack is triggered if the called party is busy. But a line with *Call Waiting* appears to be idle to a caller, although it is actually busy. Suppose that **B** is a *Call Waiting* subscriber and **A** has activated the *Automatic CallBack* feature. Will *Automatic CallBack* work for **A**, if **B** is talking to **C** when **A** calls **B**? Note that because of the *Call Waiting* feature that **B** has, **A** will receive a ring-back signal from **B**.
□

Even if a feature is working correctly, confusing situations can still occur when the feature is invoked by both parties to a call. The example below shows how a confusing situation can arise. When two people use the *Call Waiting* feature, complicated call control relationships arise. Although behaviors of the three parties involved in a single *Call Waiting* scenario are specified, nothing is mentioned about the many possible combined scenarios when a chain of people use *Call Waiting* concurrently—a typical case of incomplete specification.

Example 14 *Call Waiting* and *Call Waiting.*
Call Waiting allows a subscriber to put the other party on hold. However, it does not protect the subscriber from being put on hold. Confusion can arise when two parties exercise this type of control concurrently. Suppose that both **A** and **B** have *Call Waiting*, and **A** has put **B** on hold to talk to **C**. While on hold, **B** decides to flash the hook to answer an incoming call from **D**, which puts **A** on hold as well. If **A** then flashes the hook expecting to get back to the conversation with **B**, **A** will be on hold instead, unless either **B** also flashes the hook to return to a conversation with **A** or **D** hangs up automatically returning **B** to a conversation with **A**.

An ambiguous situation arises, when **B** hangs up on the conversation with **D** while **A** is still talking to **C**; there are two separate contexts in which to interpret **B**'s action. Assume that *CW1* refers to the *Call Waiting* call among **C-A-B** and *CW2* refers to the one among **A-B-D**. According to the specification of *Call Waiting*, in the context

of *CW2* **B** will be rung back (because **A** is still on hold) and, upon answering, become the held party in the *CW1* context and hear nothing. But, in the context of *CW1* the termination **B** will be interpreted as simply a disconnection, thus **A** and **C** are placed in a normal two-way conversation, and **B** is idled. The question is: Should **B** be rung back or should **B** be idled?
□

Removing signal ambiguities and contention for resources may initially seem like plausible solutions for solving interactions like that between *Call Waiting* and *Three-Way Calling* (cf. Example 2). However, because of the more complex call control, the possibility would now exist that each party to a call may be involved in two conversations and may be alternating between them. Truly complex situations could arise, as is shown below.

Example 15 *Call Waiting* and *Three-Way Calling (revisited).*
Consider how *Call Waiting* and *Three-Way Calling* might interact in the situations where a user can exercise both features imultaneously on the same line.[6] The call control relationship can now become quite complicated. Suppose that **A** has both *Call Waiting* and *Three-Way Calling*, **B** has *Call Waiting*, and **A** is talking to **B**. Now **C** calls **A**, so **A** uses *Call Waiting* to put **B** on hold and talks to **C**. **A** may decide to have **B** join his conversation with **C**, so he puts **C** on hold, makes a second call to **B**, and after **B** answers the call with *Call Waiting*, **A** brings **C** back into the conversation to establish a three-way call. There are three contexts in this establishment: a *Call Waiting* call and a *Three-Way Calling* call, both established by **A** among **B**-**A**-**C**, and a *Call Waiting* call established by **B** as **A**-**B**-**A**. Now, if **B** hangs up, then according to the contexts established by **A**, the session becomes a two-way call between **A** and **C**; according to the contexts established by **B** though, **B** should get a ring-back because **B** still has **A** on hold.
□

The goals of different features may be in direct conflict with one another. Such conflicts prevent certain combinations of features from being used simultaneously, unless one is assigned priority over another. Next is a simple example of one such conflict, that is presently resolved assigning one feature priority over the other.

Example 16 *Calling Number Delivery* and *Unlisted Number.*
Calling Number Delivery is a call-processing feature that delivers the directory number of the calling party to the customer's premises during the ringing cycle; this assumes that information such as the subscriber's number will be released. *Unlisted Number*, on the other hand, is a directory-service feature designed to allow a subscriber to keep the number private. Conflicts between the goals of these two features arise when a customer **A** with *Unlisted Number* places a call to another customer **B** with *Calling Number Delivery*. If the network allows **A**'s number to be delivered to **B**, then **A** loses privacy; if it doesn't, then **B** gets no information. Either way, one of the features doesn't perform its intended function. Currently, delivery of the number can be blocked by the *Calling Number Delivery Blocking* call-processing feature, which nullifies the *Calling*

[6] The current feature requirements state that *Call Waiting* and *Three-Way Calling* cannot be activated simultaneously on the same line. However, in some switching implementations it is possible for a user, who has set up a three-way call with *Three-Way Calling*, to put the other two parties on hold to answer another incoming call with *Call Waiting*.

Number Delivery feature by delivering a number consisting of only 1's.
□

Many features are equipped with parameters that can be (or need to be) instantiated (assigned values). For example, to use *Call Forwarding*, a subscriber must supply the forwarding number. The following simple example illustrates that this can lead to numerous *livelock* situations.

Example 17 *Call Forwarding* and *Call Forwarding.*
A customer with *Call Forwarding* can redirect calls to any number. As a result, any one can "accidentally" forward all incoming calls to his/her own number and create an infinite loop. Moreover, calls for **A** can be forwarded to **B**, only to be forwarded back to **A** by **B** (i.e., a two-number loop); or to **B**, then to **C**, then back to **A** (i.e., a three-number loop); and so on. Detecting the loop during call processing is difficult, if the amount of information (e.g., the numbers having been reached) passed is limited. Currently, SS7 uses a counter, aborting calls that are forwarded more than 5 times.
□

The following example shows how timing can cause interactions in telephone service logic programs, just as it does in many distributed software systems. The following example was first mentioned in [7].

Example 18 *Automatic CallBack* and *Automatic ReCall.*
When a subscriber dials the number X of a line that turns out to be busy, the feature *Automatic CallBack* (AC) can be activated. *Automatic CallBack* automatically redials **X**, when X becomes idle. If **X** is idle, the call is considered completed, and the AC request is removed immediately. On the other hand, if **X** is busy, the AC request is placed on a queue of AC requests in the central office. Call set-up will be attempted again when **X** becomes idle, the subscriber's line is idle, and the subscriber has answered a special ringback. On the other hand, *Automatic ReCall* (AR) automatically returns the latest incoming call for the subscriber, whether the call from some line **Y** was answered or not. If **Y** is idle, the call completes and the AR request is removed. If **Y** is busy, the AR request is placed on a queue of AR requests in the central office. Call set-up will be attempted again when **Y** becomes idle, the subscriber's line is idle, and the subscriber has answered a special ringback, very much like the way *Automatic CallBack* operates. Suppose that **A** has the *Automatic CallBack* feature and **B** has the *Automatic ReCall*. If **B**'s line is busy when **A** calls **B**, it is possible that thereafter the switching element serving **A**'s line keeps initiating new calls to **B** while the switching element serving **B**'s line stays busy returning every call **A**'s AC feature made. Currently, no provisions are made to disallow such behavior based on the assumption that synchronicity of the execution of the two features is highly unlikely. If necessary, the features could be implemented, using an artificial time delay, in such a way that synchronous execution is impossible.
□

2.5 CUSY Interactions

The CUSY (CUstomer-SYstem) category of feature interactions refers to interactions between a customer feature and any system feature for operations, administrative services, or maintenance. One interesting area is the interactions of some customer features with the billing system. Since the divestiture of the Bell system, a simple phone call may now involve more than one service provider or administrative domain. There is also a trend to bill an increasing number of features on a per usage basis. However, it is a general policy that the usage of a specific feature should not be billed, unless the call that uses the feature is completed. This simple view creates confusion, because the definition of a "complete call" becomes ambiguous in multiple administrative domains. The basic question is: if one service provider has completed its part of service for a call but another one did not, does anyone get paid?

Example 19 *Long distance calls* and *Message Rate Charge services.*
Each long distance call consists of at least three segments—two local accesses at each end and one provided by an interexchange carrier in between. Should a customer be charged for the segments that have been successfully completed even if the call did not reach its final destination? Would it be counted as one unit toward the total local units allowed per month for a *Message Rate Charge service*?
□

Some "add-on" billing policies, such as phone access charges introduced by third-party service providers for hotels and resorts, present yet another view of "call completion". If customers assume that they will be billed for completed calls, but not for incomplete calls, then confusion can arise if a complete call is not established.

Example 20 *Calling from hotel rooms.*
Many hotels contract with independent vendors to collect access charges for calls originated from phones in their premises. Without being able to access to the status of call connections, some billing applications developed by these vendors use a fixed amount of time to determine if a call is complete or not—thus one can be billed for incomplete calls that rang a long time, or not billed for very short duration calls (even long distance).
□

When multiple services in AIN Release 0 are invoked on one call, the interface between the SSP (Service Switching Point) and the SCP may not be adequate should services be billed on a per usage basis. The following example illustrates the situation.

Example 21 *Billing in AIN Release 0.*
In AIN Release 0, the SCP instructs the SSP to generate a billing (AMA) record by sending a call-type code (3 numerics) to the switching element. Because there is room for only one call-type code in the TCAP[7] response, a problem arises when multiple services are invoked by one SCP query. For example, if Area Number Calling is assigned a call-type code of 271 and Originating Call Screening is assigned a call-type code of 275, how will the SCP tell the SSP to generate an AMA record that reflects the fact of both features being used? Of course, one solution would be to assign another call-type code to indicate that both features were invoked. However, with the proliferation of AIN services, call-type codes will soon be exhausted[8].
□

[7]TCAP (*Transaction Capabilities Application Part*) is a protocol in the Common Channel Signaling 7 (CCS7) for a Service Switching Point (SSP) to communicate with a Service Control Point (SCP) in AIN.

[8]Note that this approach would need 2^{10} different call-type codes if only 10 features were deployed, exceeding the 1000 available 3-numerics codes

In an AIN architecture, when a subscriber of any AIN-based service tries to make a call, it is necessary to consult the record with information for that customer in the SCP/adjunct database. Since the provisioning of AIN-based services now involves more than one network component, the way provisioning is carried out is significant.

Example 22 *AIN-based services* and *POTS.*
In the provisioning of some *AIN-based services* deployed in SCPs, updates must normally be made to two network components: the SSP must now know that a trigger should be established to query the SCP databases; and the SCP should have an updated record of customer information. Let **A** be a new subscriber of the *AIN-based services*. Suppose that the provisioning for **A** is in progress, and that (1) once the trigger is set, the SSP will query the SCP database on all calls; (2) the trigger is set for **A** but the customer record is not updated yet in SCP; and (3) **A** has started making a POTS call. In this scenario, the query triggered by **A**'s POTS call will result in a "customer record not found" response from the SCP; consequently, even the POTS call cannot go through.
□

3 Categorization of Causes of Interactions

The examples presented in Section 2 suggest that there are many different causes of feature interactions. These include violation of feature assumptions, limited network support, and problems in distributed systems in general. In the following we address these causes in detail.

3.1 Violation of Feature Assumptions

Features in a telecommunications network need to operate under a set of assumptions. These assumptions may include a particular call processing model, some architectural support, a special billing system, or even how customers perceive the network operates. In a long-lived system such as telecommunications software, the evolution of system architecture or additions to its feature set often create a new environment that violates the assumptions of existing features. We identify several kinds of assumptions that, once violated, can result in some interactions.

3.1.1 Assumptions about Naming

Historically, a telephone number uniquely identified a telephone line. Thus, anyone who placed a call to a number, no matter where the call originated, reached the same telephone line every time. Also, individual customers who were associated with a specific line (e.g., members of the same household) formed a single entity and did not need be further distinguished. New features enabling rerouting or providing personal identification for multiple persons assigned to a specific line have made this "one_number-one_line-one_entity" model obsolete. Still, subscribers who are used to this model may be confused by its violation.

Features such as *800-services*, *900-services*, *911*, and *411* enable calls to be routed based on parameters such as the calling party's location, the time of day, or the day of the week. They alter the one_number-one_line assumption, but still honor the one_number-one_entity view, for entities like emergency unit (*911*) or telephone polls (*900-services*).

The various *Call Forwarding* features allow calls intended for a particular line to be routed to another specified directory number, at the terminating party's request. The called party with *Call Forwarding* features violates not only the calling party's one_number-one_line model (since now dialing different numbers may reach the same line), but also the calling party's one_number-one_entity assumption (since the caller may reach a total stranger).

Other features such as those for *Multiple Directory Number Line with Distinctive Ringing (MDNL-DR)* permit more than one number to be assigned to a line, each with its own distinctive ring. As a result, the one_line-one_entity bond at the terminating end is destroyed, even though the one_number-one_line view at the originating end is still preserved.

The features mentioned above have one thing in common: all of them introduce "aliases", or additional numbers (names) for accessing the same line or the same entity, into the telephone numbering system. The alias may be created explicitly by features like *MDNL-DR*, or implicitly by the various routing features like *800-services* or *Call Forwarding*. Consequently, the intent of a feature F that determines the logic of establishing a call based upon either the originating number or the terminating number can be easily circumvented, if F coexists with any of above features that can provide an alias for the number used as F's parameter. A simple case is between *POTS* and *Call Forwarding*, where a POTS call from **A** to **B** may be forwarded to **C**, who perhaps shouldn't know that **A** calls **B**. Examples 11 and 12 are also typical interactions of this type.

3.1.2 Assumptions about Data Availability

A large amount of data is maintained by the network. Some features may assume that certain kinds of data are widely available; other features, such as *Calling Number Delivery Blocking*, may make the same data private. Network capabilities may also limit the accessibility of some corporate data. Features cannot work properly unless they are able to obtain the data needed by their functionalities. For example, even with the *Calling Number Delivery* feature, the device cannot show a caller's number that is not available (this would happen if the call originated from an area lacking the equipment to transmit it). Whether the device should show an unlisted number is still a subject of debate (cf. Example 16).

3.1.3 Assumptions about Administrative Domain

An administrative domain is a telephone network administered by a single organization. A feature that is defined in one administrative domain may not be usable from another administrative domain. One example of this can frustrate American tourists in Europe. Numbers with an 800 prefix are often the only ones widely known for the services of certain American companies. Unfortunately, these numbers cannot in general be reached from the domain of a European telephone company. Reasons for this are that it may be difficult for European telephone companies to bill those American companies for 800 calls, or that American companies may not want to subsidize expensive overseas calls.

Billing is one of several OA&M systems affected by the divestiture of Bell system.

A common perception about billing is that a caller will not be billed for a call unless the call is completed. When there are multiple administrative domains, the definition of a "complete call" can be confusing. Regional companies, interexchange carriers, even service providers of hotels and resorts, may all participate in connecting a simple call. Features that are billed on a per usage basis now need to be specific about the exact basis for a charge. Examples 19 and 20 show two such situations, where the involvement of several administrative domains interferes with the desired execution of billing features.

3.1.4 Assumptions about Call Control

Call control refers to the ability of a user to manipulate (e.g., accept, refuse, terminate, or change the status of) a call. Features such as *911* establish a master-slave relationship—once the call is made, only the *911* operator can terminate it. Others such as *Call Waiting* and *Three-Way Calling* allow a subscriber to put other parties on hold. Another type of control, introduced by several advanced features, is to screen (e.g., *Terminating Call Screening*) or selectively forward (e.g., *Selective Call Forwarding*) some incoming calls. Interactions arise when the call control of one feature prevents other features from exercising their control, as seen in Examples 1, 3 and 4. In Example 4, the *Automatic ReCall* feature alters the role of a user from a callee to a caller, which prevents the user from activating terminating features such as *Terminating Call Screening*.

3.1.5 Assumptions about Signaling Protocol

There used to be a simple protocol that provided the status of the called party when a call attempt is in progress to the calling party: a busy signal means the terminating line is in use; a ring-back means the line is available. Many features now can make a line appear to be available (e.g., *Call Waiting*) or in use (e.g., *Make Busy*). Interactions will occur when one feature needs to know the status of the other party, but cannot tell exactly what the status is from the signal received.[9] Example 13 represents one of them.

New signals introduced by advanced features can also create interactions. A well known case where a feature can interfere with its environment is the use of a modem in a line with the *Call Waiting* feature: an unanticipated call-waiting tone signal can disrupt the data connection. If we view a modem as a third-party service, then this is an example of a feature interaction between a switch-based feature and a third-party service.

3.2 Limitations on Network Support

Network components have their own limited capabilities in communicating with other network components or processing calls. As a result, two seemingly independent features may conflict over the reception of the same signal or the usage of the same functionality.

[9] We note that this type of interaction is caused by insufficient availability of *state* data and is in that sense related to the interactions described in Section 3.1.2.

3.2.1 Limited CPE Signaling Capabilities

The set of signals that may be sent from most current CPE's to a switch is limited to *, #, the ten digits 0-9, flashhook, and disconnect. However, with the creation of many new services, the customer needs to send signals for a variety of purposes to the switch. Since there are so few signals available, the same signal is used to mean different things in different contexts. Interactions arise when the interpretation of a signal is ambiguous, as the signal has different meanings in the context of different concurrently active features (see Example 2).

3.2.2 Limited Functionalities for Communications among Network Components

The introduction of common-channel signaling enables various network components to exchange information needed for distributed application processing (e.g., inter-process query and response, or user-to-user data) [11]. Because of the format of the signaling messages and the restricted usage of channel bandwidth for these messages in some network components, features are forced to compete for bandwidth and thus interact. One such instance arises in the realization of AIN features [4]: a switching element must recognize that it needs to send queries to an SCP or an adjunct to trigger AIN features for processing calls. However, a switching element that imposes a "N-query-per-call" constraint on originating queries to an SCP or an adjunct causes interference between different features. This is illustrated in Example 5.

A related problem is the limited information that a SCP can send to a switching element via a TCAP message. A three-digit call type code is used by the SCP to signal the usage of some AIN feature; the switching element then uses the call type code to bill the usage of that feature. When multiple services are invoked by one SCP query, it is not clear how they will be billed (cf. Example 21).

In general, these interactions bring up issues in the protocol and interface design. In the case of launching queries, the protocol for transporting queries to SCPs supports a limited number of queries per call, but the protocol defined by a collection of features requires more; in the case of per usage billing, the interface is not rich enough to carry sufficient information.

3.3 Intrinsic Problems in Distributed Systems

Telecommunications systems are huge, real-time, reactive, distributed systems. Many difficulties in dealing with large distributed systems are also present in managing feature interactions. One obvious case is the problem of *resource contention*. Additionally, the distribution of feature support in the network and the customization of features by each individual can create interactions that require coordination. Some sources of interactions induced by distribution are discussed here.

3.3.1 Resource Contention

The activation of one feature can make other features unavailable. As we mentioned following the discussion of Example 2 on *Call Waiting* and *Three-Way Calling*, the exclusive use of a three-way bridge by one feature automatically disables the other feature.

3.3.2 Personalized Instantiation

Many features allow (or require) each subscriber of the features to assign values to some feature parameters before using the features. A subscriber of *Call Forwarding*, for example, needs to provide the directory number to which incoming calls will be forwarded. For various *Call Screening* features, subscribers have to supply the list of numbers to be screened.

The assignment of values to feature parameters defines the subset of the numbering plan in which the feature is interested. In many cases a particular set of assignments can make two otherwise independent features collide with each other. Examples 9 and 17 illustrate this type of interactions.

3.3.3 Timing and Race Conditions

Timing (i.e., at what time a particular event occurs, or for how long an event lasts) is always critical in distributed systems. The problem is even worse in telephone services, as the phone network also needs to deal with external human behaviors (e.g., dialing numbers, pushing buttons). As a simple example, how long the switchhook is pushed can determine whether it signifies a flashhook or a disconnection. Also, how soon a customer dials the second digit after a dialed "0" could change an intended *0+-service* call into an *Operator-service* call. Since the control context may switch from one feature to another during the call processing, the interpretation of certain signals may depend upon which control thread is active when the signals arrive at the network component. One such interaction is shown in Example 7.

Communication delays also contribute to the interactions in this category. Since signals cannot travel through the network instantaneously, users as well as network components may not have a consistent view of the system status. Together with the timing of events, communication delays can create race conditions that range from nondeterministic event sequences to infinite loops. Examples 2 and 18 illustrate race conditions.

3.3.4 Distributed Support of Features

In the past, one could assume that all features accessible by a single CPE were supported by (or at least known to) the same network component. Therefore, the network component would be aware of all the feature logic, and could coordinate the execution of various features. Unfortunately, this is not necessarily true for the present network architecture, as different features may be supported by different network components even when the features are used by a single user. Consequently, activating one feature in one component may prevent processing of features in another component. Example 6 is one such case.

The AIN architecture further distributes the support of features. In addition to the switch-based (or non-AIN) features that reside in switching elements, there are now AIN-based features whose service logic programs can be in an SCP, an adjunct, or any other intelligent network component. The switching element not only has to know when to launch a query (and where to send the query) to trigger some AIN-based features, but also needs to anticipate the outcome of such processing to be sure that non-AIN features will not be bypassed. An interaction of this type is shown in Example 8.

Causes of Interactions		SUSC	SUMC	MUSC	MUMC	CUSY
		Nature of Interactions				
Violation of Assumptions	Naming		8	10	11,12	
	Data Avail.				16	
	Admins. Domain					19,20
	Call Control	1,3,4			14,15	
	Signaling Protocol				13	
Limitations on Network Support	CPE Signaling	2	7			
	Func. of Comm.	5				21
Problems in Distributed Systems	Res. Contention	2				
	Instantiation	4		9	17	
	Timing & Race	2	7		18	
	Feature Support		6,8			
	Non-atomic Op.					22

Table 1: A summary of the categorization of the examples presented in this paper

3.3.5 Non-atomic Operations

One consequence of a distributed service logic environment is that provisioning could be a non-atomic (i.e., divisible) operation. For example, AIN-based *900 Call Screening* gives flexibility beyond the switch-based counterpart, offering additional features such as screening only during certain time periods and screening with PIN override. In the provisioning of AIN-based 900 service, updates must normally be made to two network components, the switching element and the SCP (or adjunct). How these updates are carried out in sequence may affect a customer's ability to place a simple POTS call (see Example 22 for details).

4 Discussion

Table 1 summarizes what we have presented in the previous sections. Some examples appear in more than one row, because they may have multiple causes, or they may occur only in the presence of more than one cause. Even though this table is not yet complete, it may be difficult for some empty slots to generate an entry. For example, violating naming assumptions is unlikely to result in SUSC interactions, whereas distributed support of features may cause interactions that only tie in with multiple network components. To provide a benchmark as a common measure for coverage, we welcome contributions from other researchers in the field to complete this table, and appreciate suggestions for improving the categorization as well.

4.1 Approaches to Managing Feature Interactions

Currently, there seem to exist three different groups of approaches to managing feature interactions. The first group of approaches is directed at developing infrastructures for the deployment of features in telecommunications systems. The second group supports the design of features (before they are deployed). The third group supports the resolution of interactions as they occur after deployment of the features. Each of these groups of approaches has to deal in some way with the avoidance, detection, and resolution of feature interactions, although typically one group focuses more on one of these aspects than on the others. The following sections describe these groups in more detail.

4.1.1 Infrastructure for Deployment

There are several possible ways to deal with some of the causes of feature interactions. By dealing with the causes of certain feature interactions, the corresponding interactions can be avoided. For example, a new naming scheme with a precise reference either to generic names or to aliases in each feature could avoid violations of naming assumptions (cf. Section 3.1.1); a richer set of functional signals could help resolving some ambiguities caused by limited CPE signaling capabilities; both a standardized application programming interface and a carefully designed interface protocol are useful means to address communication needs and interoperability issues among network components; and a good distributed system platform could manage problems due to non-atomic operations and the distributed nature of feature support in an advanced telecommunications network. These problems are being addressed in three Bellcore projects: AIN [4], INA [12], and the Touring Machine^{TM} project [3].

4.2 Design Support

A second group of approaches typically concentrates on the detection of feature interactions during the design phase. The difficulty of detecting interactions varies. Some interactions can be detected easily by checking the assumptions of a feature against characteristics of the environment in which the feature will be deployed. The detection of interactions caused by limited CPE signaling capabilities is also easy, simply by identifying the set of features that could respond to the same signal at the same time. Others, such as the interactions caused by timing and race conditions, may require sophisticated formal techniques for specification and reasoning [6, 10]. Techniques for protocol engineering, including design, specification, verification, validation, and testing, will be very useful in detecting interactions in this area.

4.3 Run-time Resolution

Some causes of feature interactions may be difficult to deal with at the feature design phase. Because of diverse preferences of customers, no single policy governing the availability of data could be satisfactory, nor could a set of rigid precedence relations exist for resolving conflicting call controls. Instantiations of feature parameters may be so unpredictable that dealing with all possible cases before the deployment of features could be infeasible. Differences in objectives also create barriers to any kind of agreement among administrative domains. A run-time resolution scheme based upon techniques developed in the area of distributed artificial intelligence is a possible approach [8].

5 Concluding Remarks

The feature interaction problem arises in many phases of service creation and deployment. However, while talking to individuals with different experiences and involvements in the process of service creation, we were surprised to learn that the problem conveys so many images: viewed by electrical engineers with a signal processing background, feature interactions are as simple as "signal ambiguities", and can be resolved easily using "functional signaling"; viewed by requirements writers, feature interactions

^{TM} Touring Machine is a trademark of Bellcore.

are inevitable consequences of ambiguous or incomplete specifications/requirements of telecommunications services, and tools to aid in the creation of concise, accurate requirements are crucial; viewed by software developers, the problem is similar to software engineering issues such as *extensibility*—the needs of new services have not been anticipated by the designers of the existing system; viewed by distributed system researchers, how to manage the issues of naming, control strategies, and race conditions is critical; and for artificial intelligence researchers, feature interactions occur as conflicting customer needs and/or policy objectives that can in many cases be resolved by negotiation.

While these viewpoints all represent fundamental concerns in the creation of interworking features, our categorizations and examples provide a tangible "check list" of specific issues to be dealt with.

A successful advanced service platform depends upon identifying adequate ways of managing feature interactions. This in turn requires an innovative resolution framework and a powerful interaction detection mechanism. A comprehensive understanding of the causes of adverse feature interactions is both fundamental and essential.

Acknowledgments

The authors thank Frank S. Dworak and Gary Herman for many insightful discussions. We are also grateful to Ron F. Baruzzi, Ralph B. Blumenthal, Ritu Chadha, James E. Katz, Martin Krupp, Shoshi Loeb, Ashok Ranade, and Brian S. Whittle for useful comments.

References

[1] *LATA Switching Systems Generic Requirements (LSSGR)*, Bellcore TR-TSY-000064, FSD 00-00-0100, July 1989.

[2] *Proceedings of the 3rd Telecommunications Information Networking Architecture Workshop (TINA-92).* Narita, Japan, January 1992.

[3] M. Arango, P. Bates, G. Gopal, N. Griffeth, G. Herman, T. Hickey, W. Leland, V. Mak, L. Ruston, M. Segal, M. Vecchi, A. Weinrib, and S.-Y. Wuu. Touring Machine: a software infrastructure to support multimedia communications. In *MULTIMEDIA '92, 4th IEEE COMSOC International Workshop on Multimedia Communications*, Monterey, CA, April 1–4 1992.

[4] Advanced Intelligent Network Release 1 Proposal. Special Report SR-NPL-001509, Issue 1, Bellcore, November 1989.

[5] Generic Framework Criteria for Wireless Access Communications Systems (WACS). Framework Technical Advisory FA-NWT-001013, Issue 3, Bellcore, March 1992.

[6] E.J. Cameron and Y.-J. Lin. A Real-Time Transition Model for Analyzing Behavioral Compatibility of Telecommunications Services. In *Proceedings of the ACM SIGSOFT '91 Conference on Software for Critical Systems*, pages 101–111, December 1991. New Orleans, Louisiana.

[7] F.S. Dworak, T.F. Bowen, C.H. Chow, G.E. Herman, N. Griffeth, and Y.-J. Lin. Feature Interaction Problem in Telecommunication Systems. In *Proceedings of the Seventh International Conference on Software Engineering for Telecommunication Switching Systems*, pages 59–62, July 1989. Bournemouth, United Kingdom.

[8] N.D. Griffeth and H. Velthuijsen. The Negotiating Agent Model for Rapid Feature Development. In *Proceedings of the Eighth International Conference on Software Engineering for Telecommunications Systems and Services*, March 1992. Florence, Italy.

[9] A. Lee and D. Carrington. Formalizing Extensions and Modifications to Telecommunication Software. In *Australian Conference on Telecommunications Software (ACTS)*, pages 205–210, 1991.

[10] Y.-J. Lin. Analyzing Service Specifications Based upon the Logic Programming Paradigm. In *Proceedings of the IEEE GLOBECOM '90*, pages 651–655, December 1990. San Diego, California.

[11] A.R. Modarressi and R.A. Skoog. Signaling System No. 7: A Tutorial. *IEEE Communications*, pages 19–35, July 1990.

[12] H. Rubin. Tutorial on Information Networking Architecture. In *Proceedings of the IEEE GLOBECOM '91*, 1991.