:)

NEW FEATURE INTERACTIONS IN MOBILE AND MULTIMEDIA TELECOMMUNICATION SERVICES

Pamela Zave

AT&T Laboratories—Research
180 Park Avenue, Room D205
Florham Park, New Jersey 07932, USA
+1 973 360 8676
pamela@research.att.com

Michael Jackson

AT&T Laboratories—Research and MAJ Consulting Limited
101 Hamilton Terrace
London NW8 9QY, UK
+44 171 286 1814
jacksonma@acm.org

7 March 2000

New Feature Interactions in Mobile and Multimedia Telecommunication Services

Pamela ZAVE
AT&T Laboratories—Research, Florham Park, New Jersey 07932, USA

Michael JACKSON
101 Hamilton Terrace, London NW8 9YQ, UK

Abstract. Some previously unidentified sources of feature interaction arising in mobile and multimedia telecommunication systems are presented. We discuss how these feature interactions might be managed by feature developers and service providers. The Distributed Feature Composition (DFC) architecture has proven to be a valuable tool for elucidating these feature interactions, and for describing desirable global behavior without loss of feature modularity.

1. Introduction

The stated purpose of Tsang, Magill, and Kelly's study of multimedia services [5,6] was "to identify classes of service which tend to interact and provide guidelines for service providers on how to avoid feature interaction in the future." They identify the following as major sources of feature interaction in multimedia services:

Voice feature interactions: Most causes of feature interaction in telephone services have natural and inevitable extensions into multimedia services.

Interactions native to other media: In multimedia telecommunication systems, other forms of communication such as electronic mail (text) and Web browsing (images) are brought under the telecommunication umbrella and unified conceptually with telephony. They bring their own native interactions under the umbrella with them. For example, Hall discusses many interactions native to electronic mail [3].

Competition for bandwidth: When the total bandwidth needs of all available services exceed the bandwidth resources, then features must compete for them.

Video conferencing: This is a rich service, with many roles to play and many opportunities to add supplementary features. It is also much in demand. Thus, it is likely to be a major source of feature interactions in the future.

This paper builds on Tsang, Magill, and Kelly's work by identifying additional categories of feature interaction arising in mobile and multimedia telecommunication services. "Additional" means in addition to the first three items above; video conferencing is a service rather than a category of feature interaction.

In Sections 2 and 3 we provide a foundation by presenting our working definition of feature interaction and our descriptive technique for features. Sections 4 through 7 identify new types of feature interaction and discuss how they might be managed by feature developers and service providers.

In performing this research, we found it surprisingly difficult to think of new ways in which mobile and multimedia services can interact. Since our experience with voice is that the variety of services people can dream up is astounding, and the variety of their potential interactions even more so, we rejected immediately the possibility that there are few such

interactions. It is much more plausible that they are as varied as voice interactions or more so, and that the problem is our lack of experience with these services.

This is particularly true of multimedia services, because our experience with non-voice media is mostly confined to situations that do not have the particular richness of telecommunications. We are all used to video, but only in the non-interactive form offered by broadcast networks. We are all used to images seen through a Web browser, but those images are not part of a conversation, as the use of a shared whiteboard would be. We are all used to text in the form of electronic mail, but not so much used to text as part of a real-time two-way connection.

The goal of multimedia telecommunications is to unify all of these media and styles of communication, so that the overall richness and usefulness of communication at a distance is increased. Once large numbers of people are using them, they will, like telephony, change our society. Only then will it become easy to see where the remaining feature interactions are lurking.

2. A perspective on feature interaction

A feature is an optional or incremental unit of functionality. A system specification organized by features usually consists of a base specification and feature modules. The behavior of the system as a whole is determined by applying a feature-composition operator to these modules. A feature interaction is a way in which one feature modifies or influences another in determining overall system behavior.

Two points are important to understand about feature interactions. The first is that they are an inevitable by-product of modularity. The second is that, while many feature interactions are undesirable, many others are desirable or necessary.

Busy treatments provide a familiar example of both these points. Suppose that we have a feature-specification language in which a busy treatment is specified by providing an action, an enabling condition, and a priority. Further suppose that the feature-composition operator ensures that, in any busy situation, the action applied will be that of the highest-priority enabled busy treatment.

In a busy situation where two busy treatments B_1 and B_2 are both enabled, with B_2 having higher priority, these features will interact: the action of B_1 will not be applied, even though its stand-alone specification says that it should be applied. This feature interaction is entirely intentional and desired. It is a by-product of the feature modularity that allows us to add busy treatments to the system without changing existing busy treatments. Without the special feature-composition operator, when B_2 is added to the system, the enabling condition E_1 of B_1 must be changed to $E_1 \wedge \neg E_2$.

From this perspective, effective management of feature interactions would seem to require at least these steps [8]: (1) Feature engineers specify features as if they were independent. This is usually done by imagining that the feature at hand is the only feature being added to the base system. (2) Analysis techniques are applied to discover and understand actual or potential feature interactions. (3) Feature engineers decide which feature interactions are desirable and which are undesirable. (4) Feature engineers alter the specification and/or the parameters of the composition operator to include all desired behavior and exclude all undesired behavior.

Concerning Step 2, there are many approaches to discovering undesirable feature interactions (exemplified by many papers in the proceedings of previous workshops in this series). However, it is equally important to understand and discover potential desirable feature interactions, lest opportunities to serve the customer be overlooked.

For example, here is a potential interaction between Call Forwarding and Voice Mail [8]: If *a* has Voice Mail, *a* is forwarding unconditionally to *b*, *b* does not subscribe to Voice Mail or any other busy treatment, and a call addressed to *a* is failing because *b* is busy, then *a*'s Voice Mail feature should provide a busy treatment. This seems to be good, useful behavior. Because it is a subtle interaction, it would be easy for feature engineers to miss this possibility and design their features so that they do not interact in this beneficial

way.

Another critical goal for any feature-interaction technology is support of Step 4. Feature engineers should be able to specify exactly the desired system behavior without loss of modularity. Ideally, feature composition should be powerful enough to produce exactly the desired behavior, without any feature specification's explicitly cooperating with other features, or even having to be changed because of the presence of other features.

3. An overview of DFC

Distributed Feature Composition (DFC) is a virtual architecture intended for describing telecommunication services in a modular and analyzable way [4]. It plays two roles in this paper. First, we use it in a rather informal way as a means of describing features. Second, we use it to expose and elucidate potential feature interactions, and to show how desired behavior can often be described without loss of modularity. The fact that it plays the latter role rather well suggests that it meets the goals explained in Section 2, at least to a satisfactory degree.

In this section we attempt the shortest possible summary of DFC. The version of DFC presented is not exactly the version detailed in [4] and summarized in [7]. It has been simplified, to avoid inessential detail. It has also been extended in some modest ways to accommodate mobile and multimedia services.

In DFC we use the term *customer call* for the informal notion of an attempt by one customer to communicate with another. A customer call typically generates and is responded to by a *usage*, which is a dynamic assembly of *boxes* and *internal calls.* A *box* is a module or component, and implements either a line interface or a feature. An *internal call* is a featureless connection between two ports on two different boxes. Figure 1 illustrates a simple usage.

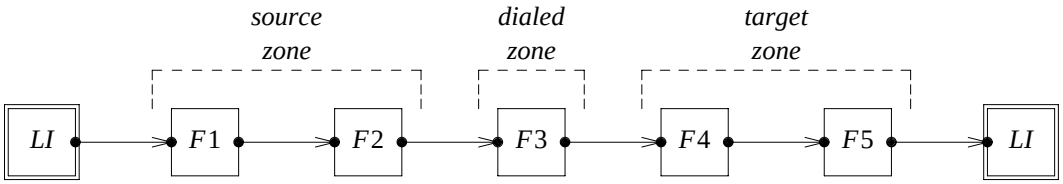

Figure 1. A linear usage.

In Figure 1 a DFC internal call is shown as an arrow from the port that placed the call to the port that received the call. Each internal call begins with a *setup phase* in which the initiating port sends a setup signal to the DFC router, and the DFC router chooses a box and forwards the signal to it. The receiving box chooses an idle port for the call (if any) and completes the setup phase with a signal back to the initiating port. From that time until the *teardown phase*, the call exists and has a two-way signaling channel. The call can also have any number of media channels, each carrying any medium. Because there is no default configuration of media channels in a call (for example, it is not necessary for a call to have a voice channel), each media channel in a call must be opened and closed explicitly by signals on the signaling channel.

Having full control of all the calls it places or receives, a feature box has the autonomy to fulfill its purpose without external assistance. It can behave transparently when no function is needed. It can also behave assertively, re-routing calls, processing media streams, and absorbing/generating signals. To give a simple example, a Call Forwarding on Busy box (box *F4* in Figure 1) monitors the signals coming back to it on its outgoing internal call. If the signals indicate that the target line interface is busy, then it tears down its outgoing internal call and places a new outgoing internal call whose target is the

forwarding address.

The components of the DFC architecture are shown in Figure 2. The line-interface boxes (*LI*) are connected to telecommunication devices by external lines. The feature boxes (*F*) can have any number of ports, depending on their various functions. Internal calls are provided by the port-to-port *virtual network.*

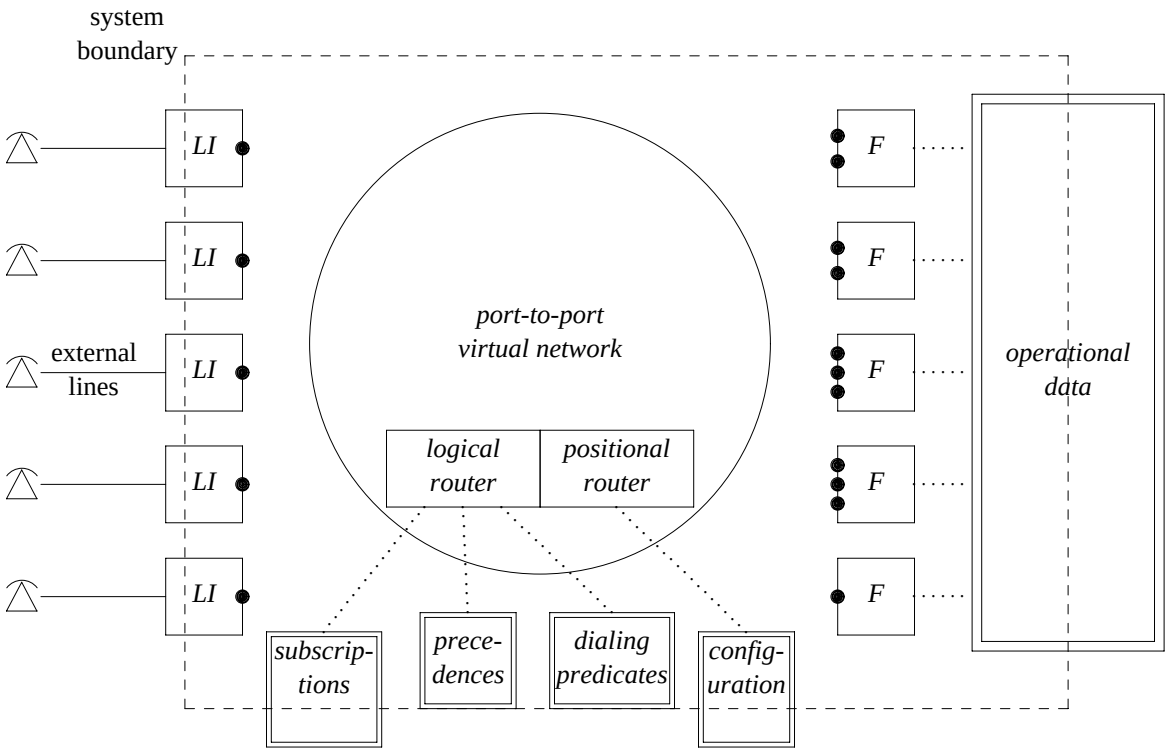

Figure 2. Components of the DFC architecture.

The router of the virtual network is unusual. It not only routes internal calls to the destinations associated with their target addresses, as any network router does, but it also "applies features" by routing internal calls to and from feature boxes. For this reason it needs data on feature subscriptions, feature precedences, and the dialing plan as well as normal configuration data. (All global data is shown in double rectangles in Figure 2.) Routing will be explained further at the end of this section.

Figure 2 also shows global data called *operational data*, which is used by feature boxes. For example, the Call Forwarding on Busy feature box discussed above retrieves its subscriber's forwarding address from operational data. Access to operational data is strictly partitioned by features, so its use cannot compromise feature modularity. In other words, operational data is used only to store data or pass it between multiple boxes that are all implementing the same feature.

Each box fits into one of two large categories. A *bound box* is a unique, persistent, addressable individual. Bound boxes are doubled in Figure 1 and all subsequent figures. In Figure 1 the only bound boxes are the two line interfaces. The other boxes in Figure 1 are *free boxes*, meaning that each box is an anonymous, interchangeable copy of its type with no persistence outside its tenure in a usage.

Figure 3 shows a usage in which all the boxes, including the one feature box representing Call Waiting, are bound. The CW_c box is associated with LI_c. The usage was formed because first the customer owning address *c* made a successful call to address *a*. This passed transparently through CW_c just because *c* subscribes to Call Waiting. Later, the customer *b* attempted to call *c*. The internal call generated by his attempt was routed to CW_c on its way to *c*, where it was accepted at the third port.

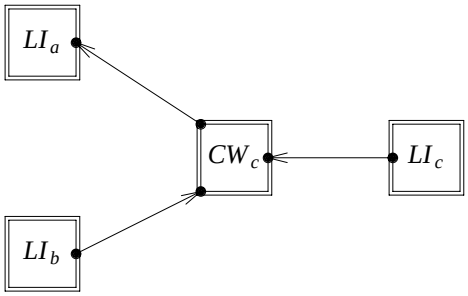

Figure 3. A nonlinear usage.

Acceptance of a call at its third port causes CW_c to spring into action. It first signals back to LI_b that the customer call has reached an alerting state; LI_b will play a ringback tone so that its customer can hear it. CW_c then alerts LI_c by inserting a tone on the voice channel to it. CW_c then monitors the voice channel from LI_c for flash signals. Each time CW_c recognizes a flash signal, it switches its internal voice path between the two possible positions: connecting the ports that connect *c* with *a*, or connecting the ports that connect *c* with *b*.

Figure 3 illustrates the important distinction between a usage and a customer call. There are two customer calls, one placed by the customer owning *c* and one placed by the customer owning *b*. But there is only one usage, representing the joining of those separate communication attempts by the Call Waiting feature. Many difficulties in telecommunications are due to ignoring this crucial distinction.

Nevertheless, in subsequent sections most mentions of "call" refer to customer calls rather than DFC internal calls. We omit the "customer" when there is little chance of confusion.

The key to assembly of the necessary usage configurations is the DFC routing algorithm. It operates on the setup signal that initiates each internal call. The setup signal has five fields of interest to the router: *source: address, dialed: string, target: address, command: {new, continue, update, direct},* and *route:* seq *routing_pair*. Each routing pair has a first component of type *box_type* and a second component of type *zone = { source, dialed, target }*.

We shall explain the function of the router by example, first describing how the usage in Figure 1 was assembled. The setup signal emitted by the line interface on the left had a *source* field containing the line interface's address, a *dialed* field containing the dialed string, and a *command* field containing *new*. The other two fields were empty. Upon receiving the signal, the router first extracted a target line address from the dialed string, and put it into the *target* field.

Next the router, instructed by *new*, computed a new route and put it into the *route* field. The source address subscribes to two feature box types $F1$ and $F2$ in the source zone, so the first two pairs of the route were *(*$F1$*,source)* and *(*$F2$*,source)*. The dialed string matches the triggering pattern of one feature box type *F3* in the dialed zone, so the next pair of the route was *(*$F3$*,dialed)*. Finally, the target address subscribes to two feature box types *F4* and *F5* in the target zone, so the last two pairs of the route were *(*$F4$*,target)* and *(*$F5$*,target)*.

Now the router had finished manipulating the setup signal, and needed to find a box to route the internal call to. It stripped the first pair off the route, and since its box type $F1$ is the type of a free box, it routed the internal call to an arbitrary fresh box of that type.

None of the feature boxes in Figure 1 needed to control the routing in any special way. So when each box prepared a setup signal for an outgoing call, it simply copied the entire setup signal from its incoming call, making sure that the *command* field had *continue* rather than *new*. The continue command told the router not to recompute anything in the

route. The chain unfolded, with one less pair in the route after each box was added to the usage. Finally, in the last internal call, the route was empty so the router routed to the line interface of the target address.

In constructing a route, the router uses a *precedence* relation governing the order in which pairs can occur in a route. It also, of course, uses subscription data. In the most general case a feature has several roles and several box types; an address plays a role in a feature, which means it subscribes to a feature box of the feature in the source zone or the target zone or both. Usually the distinction between a feature box and a feature is not a source of confusion, so we simply say that "an address subscribes to a feature."

The need for some of this complexity is illustrated by Figure 3. A user of Call Waiting must subscribe to *CW in both the source and target zones,* to ensure that every relevant communication goes through the same box. In Figure 3 the customer call from *c* to *a* goes through CW_c because *c* subscribes to *CW* in the source zone. The customer call from *b* to *c* goes through CW_c because *c* subscribes to *CW* in the target zone. Because *CW* is the type of a bound box, and CW_c is the unique box of this type associated with LI_c, both customer calls are routed to exactly the same box.

The two additional routing commands, *update* and *direct,* will be explained where they are used in subsequent sections.

4. Feature interactions caused by media choice

One aspect of multimedia services is media choice: since communication in more than one medium is possible, people can choose in which medium or media they wish to communicate. Media choice is a new telecommunication function, which will undoubtedly be the focus of many new features. These features will exhibit a new class of feature interactions.

Some authors address such issues with the word "negotiation," as if negotiation were an alternative to feature interaction [2]. We do not agree with this viewpoint. A negotiation consists of a sequence of choices, each choice being made by one of the negotiating parties. Each party's choices must either be made manually—by the person himself—or automatically—by an algorithm in software running on behalf of the person.

Manual negotiation is problematic for several reasons. It is likely to be cumbersome and time-consuming. It can also be socially embarrassing. A large ingredient of the popularity of techniques for screening incoming calls is that the caller does not know he has been deliberately ignored. Most people are uncomfortable saying, in any form whatsoever, "I am here and I know you want to communicate with me, but I refuse to communicate with you." Yet it seems likely that manual negotiation would result in exactly that kind of caller awareness and callee embarrassment.

If negotiation is performed automatically, on the other hand, then the negotiation algorithm is a feature. And features can interact with other features, for good or for ill.

For example, we now have colleagues who use text conversation (telecommunication connections in a text medium) so frequently and casually at work that they think it impolite to place a voice call to a colleague without first checking the person's availability by means of text. Convinced that workplace customs have changed for good, one of these colleagues might adopt a new screening feature. This feature ensures that the customer's telephone never alerts him of a request for a voice conversation unless a text conversation is already in progress between the same two customers. If the precondition of voice conversation is not satisfied, then the voice request is immediately handled by a coverage feature such as Voice Mail.

In terms of the previous remarks about negotiation, current customs make it acceptable to delay conversation through the text medium ("busy right now—how about this afternoon?") but less so when the voice connection is already established. Furthermore, a person can pretend not to have seen the most recent text transmission, but cannot pretend not to have answered the telephone.

Meanwhile, another person might purchase a new "multimedia telephone" allowing

communication by voice, video, text, and images simultaneously. The assumption of the manufacturers of this device is that voice is the primary medium. All calls entail a voice channel. Once the voice channel is opened, channels carrying additional media can be added if both customers desire it and have appropriate device capabilities. This assumption is deeply embedded in the user interface of the multimedia telephone and in the features provided for use with it.

Clearly our colleague's feature and the features of the multimedia telephone have an undesirable interaction. Both embody rigid automatic protocols for media choice. Since the protocols, in essence, deadlock, users of these two protocols cannot communicate directly with one another.

If the conflicting features were specified in DFC, each feature would be represented by a separate feature box. The program for each feature box would be a relatively simple finite-state machine. Once the features were suspected of interacting undesirably, it would be easy and efficient to create a cluster of processes and queues simulating the usage, and to use a model-checker to detect potential deadlocks. Here the modularity of DFC supports analysis by making it easy to isolate the features of concern, and to analyze their interaction without the added complexity of other features.

5. The issue of shared devices

Mobility has several aspects in telecommunications. In a cellular network, a mobile device can roam without changing network addresses and without the use of features. There is also another form of mobility, in which a mobile address has temporary feature-based associations with successive fixed addresses. Only the latter form of mobility is of interest here, because only the latter form is involved with features and feature interactions.

Speaking in DFC terms, the address of a line interface and its telecommunication device is a *line address;* it is unique and dedicated to this purpose. There are also *mobile addresses*, which have no permanent association with any network interface, line, or device. Any address can subscribe to a set of features in the source zone and a set of features in the target zone. If *d* is a line address, these two feature sets form the *default* features of *d* and of the device *d* identifies.

In systems with mobile and multimedia services, there are likely to be many other feature sets besides default features, used for many other purposes. For example, a customer might have a personal feature set, subscribed to by a mobile address rather than a line address, that he uses from whatever device he happens to be closest to. For another example, a customer might use the same home device for personal communication and for work as a service representative; these two purposes might require very different feature sets. For another example, the same home device might be used by several family members, each of whom has a customized personal feature set.

In such a system, it will often be the case that a device is being shared among several feature sets simultaneously. This will be the source of many feature interactions between sharing feature sets. We regard such feature interactions as different from interactions among features of the same feature set, whether they relate to one medium, multiple media, or signaling alone. It is not that there is a fundamental difference in the ways that features can affect one another, but rather than the grouping into sets causes the interactions to appear at a different level of abstraction.

Interactions caused by device sharing are not entirely new—two examples in Cameron et al.'s benchmark [1] concerning personal directory numbers and distinctive ringing have some of this flavor. However, we provide a much more thorough treatment of the subject here.

To reinforce the idea that a device can be shared among feature sets, consider a customer who likes multimedia services but has no single device that can handle all his favorite media. Fortunately, software features can make several devices function as one multimedia pseudo-device.

The easiest way to achieve this is to regard one device as primary, so that its address is

also the address of the pseudo-device. When a feature of the primary device chooses to add another medium to the conversation, it calls a device able to handle that medium. Such a device is shared between two feature sets, one supporting its use as a stand-alone device, and one supporting its use as a secondary part of a multimedia pseudo-device.

There seem to be two major subcategories of feature interaction arising from shared devices. We discuss them in Sections 6 and 7, respectively.

6. Feature interactions for correct application of feature sets

If a device is shared by several feature sets, then in any situation involving a customer call initiated from or directed to that device, the correct feature set or sets must be applied. This selective application is itself achieved by features. We consider here interactions between feature sets considered as wholes and features for selective application of feature sets. Our primary focus is on positive interactions: How can the appropriate selective application of feature sets be achieved?

6.1. Default and alternate features

Often we want both the default feature set and an alternate feature set applied in either the source or target zone. This is always the requirement when there is no previous agreement between the owner of the shared device and the subscriber to the alternate feature set. In such a case, the alternate feature set should not have the power to bypass or elude the default features of the device, which may include features subscribed to by the device owner for his own protection.

The first question we must ask about alternate feature sets is: How or why are they invoked? Like default feature sets, alternate feature sets must be subscribed to by addresses (in this case mobile addresses). An alternate feature set is applied in the target zone because the caller chose it by dialing the alternate address. For example, a person might have a personal address that subscribes to a personal feature set. He gives the personal address to all his friends, who use it to call him wherever he is currently located.

Figure 4 shows a DFC pattern for applying a default feature set and an alternate feature set in the target zone. The diagram begins with (on the left) an internal call whose target is the alternate address *a*. Previous zones of the route have been exhausted, so the internal call is being routed to the first box of the target zone of *a*.

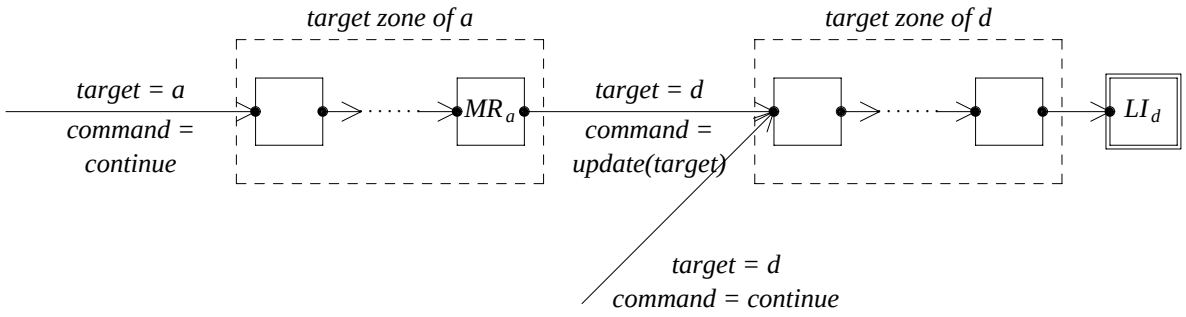

Figure 4. Alternate and default features in the target zone.

The Mobile Redirection feature (MR_a) relies on operational data maintaining the current address of the owner of *a*, which is now *d*. This information has many possible sources. It might be supplied to the system manually by the owner of *a*, either through a provisioning interface or, more dynamically, through a feature. Or it might be collected automatically through various types of sensor.

MR places an internal call with *target = d*, an *update(target)* command, and the same route that came with the internal call it received. This will cause the router to replace the target zone of the old route (now actually empty, because MR should be the last feature box in the target zone of *a*) with the target zone of *d*. After this change to the route, the internal call will be routed to the first box of the target zone of *d*.

Although some of the boxes in either target zone might be bound, none of them have to be bound, and we show them in Figure 4 as free.

No port in DFC can ever participate in two calls simultaneously. The purpose of the second arrow going to the first box of the target zone of *d* is to show another possible usage. A customer call dialed to *d* will result in a usage whose target zone is simply the target zone of *d*. If all the boxes of the target zone of *d* are free, then the two usages indicated will have two complete, separate copies of the target zone of *d*. If some box is bound, on the other hand, then the two indicated usages will join at the first bound box.

The two target zones must be ordered as shown in Figure 4, because it takes the MR feature in the source zone of *a* to find *d*. This is indeed fortunate, because the target zone of *d* must stand between LI_d and the target zone of *a*, to protect the device from assault. Although only DFC controls feature priority through position in a usage, any other feature-composition scheme would have to achieve a corresponding relationship between the two feature sets.

It is difficult for a person who places a call from a stranger's device to invoke an alternate feature set such as his own, personal source-zone features. They have no association with the source address, nor with the address of the target he wishes to reach. It seems that the only way to solve this problem is some form of double dialing, in which the caller first dials a special address just for the purpose of invoking alternate features. Later he is prompted for and dials the actual target address.

Figure 5 shows a DFC pattern for invoking a default feature set in the source zone, followed by an alternate feature set in the source zone. Dialable string *z* matches the triggering pattern of the dialed-zone feature Authentication and Reattribution (*AR*). The AR feature will prompt the caller for the mobile address he wishes to use (here *a*) and for proof of his right to use it. Then AR will prompt the caller for the address he wishes to call, here *t*.

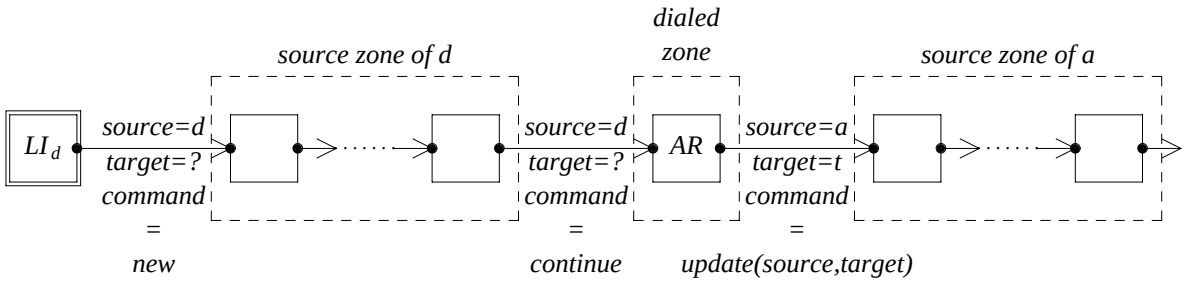

Figure 5. Default and alternate features in the source zone.

The string *z* might or might not look like an address. Assuming that it cannot be interpreted as an address, there will be no targets or target zones in the setup signals of the chain until after the AR box. The setup signal issued by the AR box has a new source and a new target, so both zones of the route must be updated.

The two source zones must be ordered as shown in Figure 5, because there is no authorization of any kind for omitting any of the source-zone features of *d*. Although here DFC controls feature priority by placing default features closest to their device, any other feature-composition scheme would have to achieve a corresponding relationship between the two feature sets. More details of the relationship will be discussed in Section 7.

6.2. Essential and optional or alternate features

When the owner of a shared device and the subscriber of an alternate feature set are willing to cooperate (often because they are the same person), more flexible arrangements of features are possible. In the basic arrangement considered here, the default features of the device are divided into two subsets, *essential* and *optional.* The essential features always apply to customer calls using the device. In addition to essential features, optional or alternate features apply.

Figure 6 shows a DFC pattern for achieving this in the source zone. As in Figure 4, the depiction of two internal calls at the same port is an indication of two different possible usages.

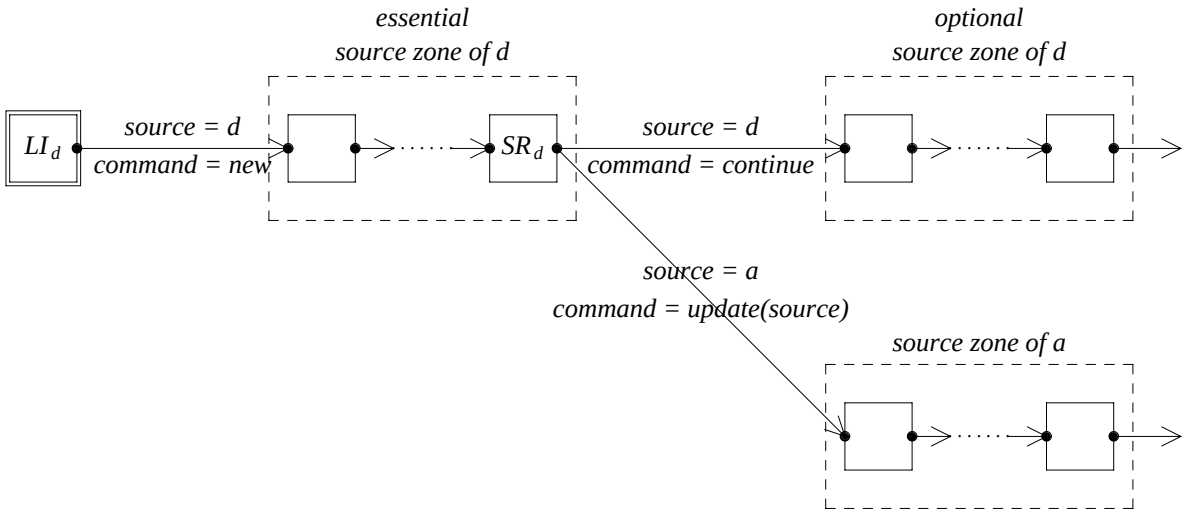

Figure 6. Essential, optional, and alternate features in the source zone.

The Selective Reattribution feature (SR_d) must be the last of the essential features of *d*, since it serves as the branching point: depending on what kind of an outgoing call it makes, the next features applied to the customer call will be optional or alternate ones. To achieve this pattern, of course, *d* must subscribe to SR and the operational data of SR must include the alternate address *a*.

In any of these patterns, any feature set can be empty unless it includes a named feature box—the named features are the necessary ones. Thus in Figure 6 the optional source zone of *d* might be empty, and SR would always change the source to *a*. More commonly, however, the SR feature box would interact with the caller to determine whether the caller chooses optional or alternate features. If the caller chooses alternate features, authentication of privilege may also be required. Depending on the device capabilities and user-interface conventions, signaling between the SR feature box and the caller might be in-band or out-of-band, using specialized signals or standard ones such as DTMF tones.

This pattern is most obviously useful for a device used by a group of people, such as a family. The alternate features could be the customized personal features of a particular family member. Or, the alternate features could be used by a family member when playing a job-related role.

Figure 7 shows a DFC pattern for satisfying the basic requirement of this subsection in the target zone. The caller makes the choice between optional and alternate features by directing the call to target address *d* or *a*.

As with Figure 6, this pattern is useful for sharing a device within a family. It is also useful if the device is being used both as a primary device identified by *d*, and as a secondary member of a multimedia pseudo-device under the address *a*. In the latter case *a* is not made public, as it is only used by the features that make a multimedia pseudo-device

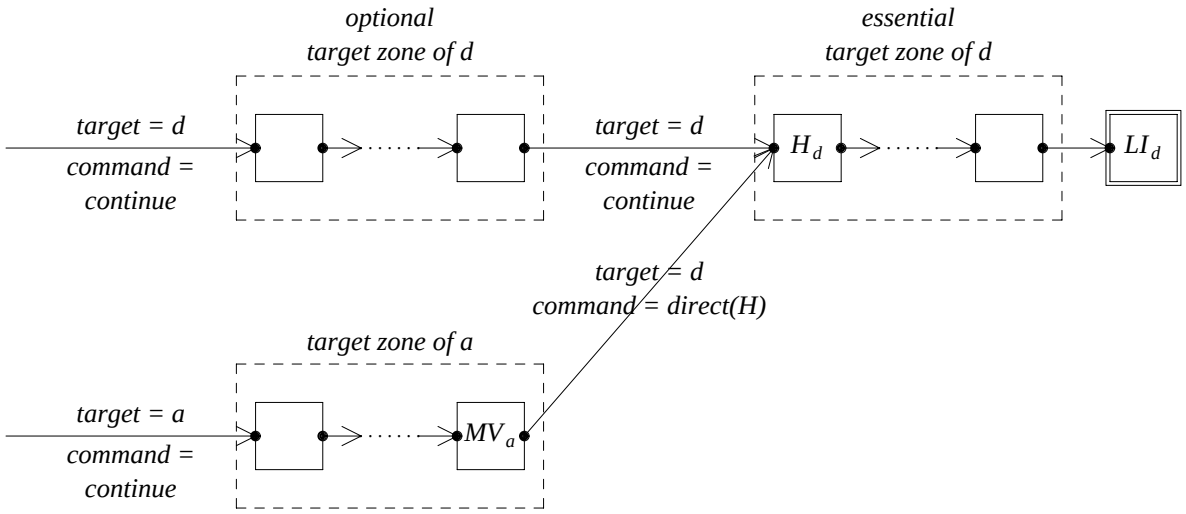

Figure 7. Optional, alternate, and essential features in the target zone.

out of several ordinary devices.

The Mobile Visitor feature box (MV_a) must be the last box in the alternate feature set. The Host feature box (H_d) must be the first box in the essential subset. These two feature boxes are actually part of the same device-sharing feature. Because the device participates in this feature as the host, *d* subscribes to the Host feature box. Because mobile address *a* participates in this feature as the visitor, *a* subscribes to the Mobile Visitor box.

Because these two boxes are part of the same feature, they can have a special relationship without violating feature modularity. Specifically, MV is allowed to name H as the argument of a *direct* routing command. The result of the router's interpretation of this command is a route consisting of the segment of the target zone of *d* beginning at H.

As in Figure 4, all the boxes of the essential target zone of *d* might be free. In this case each of the two usage possibilities shown in Figure 6 would have its own separate copy of the boxes of this zone. We shall discuss other possibilities and other consequences in Section 7.

6.3. Application of feature sets: Summary

The four patterns presented in this section are quite comprehensive, and will suffice for most compositions of feature sets for shared devices. Certainly there are feature refinements that they do not cover, but at least a few additional complexities have been studied and found to fit quite well into this overall scheme. Thus DFC does a good job on this type of feature interaction, describing the various behavioral possibilities straightforwardly and without loss of modularity.

Describing the behavioral options in DFC terms made it easy for us to focus on the important aspects of this type of feature interaction, which include such questions as: When and how are dynamic choices between feature sets made? Are feature sets composed by conjunction or disjunction? Which data, agreements, and subscriptions are needed? Furthermore, the answers to such questions show up rather clearly in the diagrams.

This is in contrast with details that are necessary to implement the features, but less important in the big picture, such as the user interfaces of interactive features such as AR and SR. Using DFC we know that we can fill them in later, so we do not have to grapple with them at the same time that we are trying to answer the more important questions.

7. Interactions between feature sets

We are all familiar with the facts that features within one customer's feature set can interact, and that the source features of one customer can interact with the target features of another customer. Shared devices introduce the possibility that two source-zone features in different feature sets could interact, and also that two target-zone features in different feature sets could interact.

These interactions are often between similar features, or features with related purposes. We present here a representative sample of such interactions.

7.1. Interactions between target zones

Call Waiting is a complex feature for many reasons. Even from the perspective of the customer who subscribes to it, it has both positive and negative aspects. On the positive side, more calls directed to him will get through. On the negative side, conversations in progress will be interrupted.

Two target zones can be composed. What if they both contain Call Waiting? Our first concern, in minimizing negative interactions and maximizing positive ones, must be to ensure that there are no disastrous runtime conflicts or inconsistencies. Our second concern should be to maintain a fair balance between call completion and call interruption.

If the target zones are composed as in Figure 4, any customer call can interrupt any other customer call, although the dynamic priorities can vary depending on whether the interruption takes place in CW_a or CW_d. Because there is no prior agreement between *a* and *d*, there is nothing we can do to alter the priorities, but at least DFC feature composition ensures that there will be no disasters.

If the target zones are composed as in Figure 7, and if CW_d is placed among the optional features, then we have the interesting behavior that calls directed to *a* can only be interrupted by other calls to *a*, and calls directed to *d* can only be interrupted by other calls to *d*.

Overall, we would recommend the arrangement shown in Figure 8, in which only *d* subscribes to Call Waiting, and Call Waiting is an essential feature. This will allow completion and interruption on the fairest and friendliest basis. Note that Figure 8 is an exact likeness of a single DFC usage, in which there are two copies of the free box type H, and the two branches of the usage join at the three-port Call Waiting box.

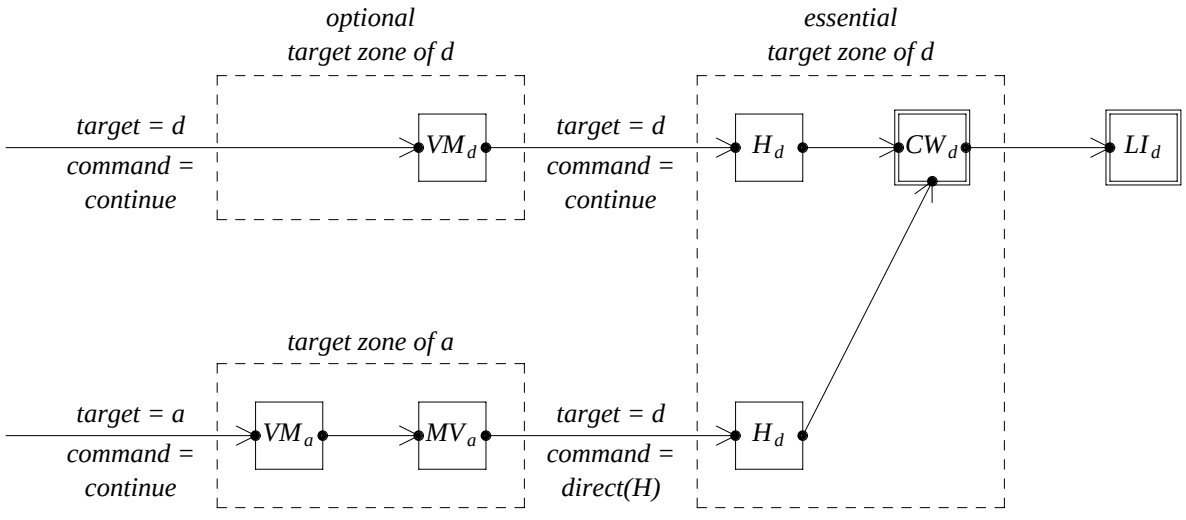

Figure 8. A good composition of Call Waiting and Voice Mail features.

If desired, it would also be possible to achieve an asymmetric arrangement in which one class of calls has clear priority over another class. This would be similar to the Emergency Break-In feature [4].

Either *a* or *d*, or both, could subscribe to a Voice Mail feature. It is highly desirable that an uncompleted call directed to *a* be served by VM_a, and that an uncompleted call directed to *d* be served by VM_d.

In Figure 4 this desirable feature interaction is difficult to achieve. A busy signal from LI_d will reach VM_d (if any) before VM_a (if any), and the first Voice Mail box to receive a busy signal will process and absorb it. VM_d will ignore a busy condition and propagate it to VM_a only if it is specifically programmed to do so, and it can only be programmed to do so if some indication is preserved that the call was originally directed to *a*.

This would be a loss of modularity, but the problem seems to be exposed by DFC more than caused by it. Indeed, it is difficult to imagine why a feature subscribed to by *d* should make an exception for *a* when there is no prior agreement between their owners.

In Figure 7 the desirable feature interaction is easy to achieve, as shown in Figure 8. A busy signal on either path will originate at the CW_d box and travel backward along the path to the correct VM box. This works even if there are separate CW_d and CW_a feature boxes, provided that each is later in its zone than the corresponding VM box.

7.2. Interactions between source zones

Concerning interactions between source zones, the most interesting features are related to dialing. We shall exemplify the issues with Speed Calling and Call Restriction.

Speed Calling enables a user to dial a short code which is translated by the feature to a full address. The feature maintains a database for performing this translation. Call Restriction prevents the placing of calls to certain addresses. The feature-interaction issue raised by Call Restriction is always that of scope: Is there any way to circumvent the restrictions through the use of other features?

In a form more specifically relevant to our topic, the questions are: Can Speed Calling in the source zone of *a* circumvent Call Restriction in the source zone of *d*? Should it be able to?

In answer to the latter question, it is arguable that Call Restriction need not apply to the source zone of *a*. Calls placed from the source zone of *a* have *a* as their source and, very likely, as their billable address. Or *a* might be used only by an adult in the family that shares device *d*, and the Call Restriction feature is intended only to apply to the calls made by children.

On the other hand, it is equally plausible that children in the family can have their own personalized feature sets, yet must adhere to all call restrictions. This would be a requirement in the context of cooperative source-zone composition, as in Figure 6. Figure 9 shows how the requirement can be satisfied in that context.

Obviously Call Restriction must be applied after Speed Calling has performed its address translation (if any). In DFC this means that Call Restriction must have a later position in the source zone than Speed Calling. If there are personalized SC boxes in both the optional and alternate source zones, then there must also be CR boxes in both zones. Even more cooperation is required between the owners of addresses *d* and *a*, as both must provision their CR boxes with the same restriction lists.

Figure 9 shows a blocking scenario on the lower of the two possible paths. The caller has dialed the code *c*, and uses SR_d to choose to use the alternate source-zone features. SC_a translates *c* to the forbidden target address *t*, so the usage is not continued by the CR_a box.

7.3. Signaling problems

Source-zone features have signaling dialogues with callers through the calling line interface and device. Target-zone features have signaling dialogues with callees through the called line interface and device. Device sharing does not exactly introduce new

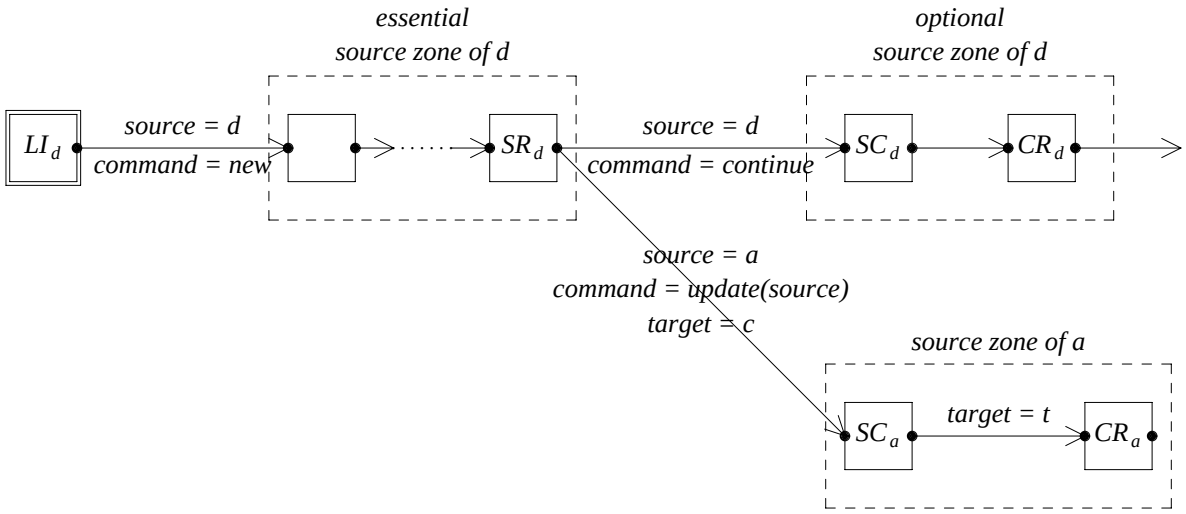

Figure 9. A good composition of Speed Calling and Call Restriction features.

signaling problems, but it certainly exacerbates two old ones.

The nicest user interfaces are designed with close coordination between the device—which generates and displays signals—and the features—which generate and respond to signals. Features designed without knowledge of the devices from which they will be controlled must use a restricted and often awkward signaling vocabulary, such as DTMF tones and in-band announcements. Mobile feature sets fall into this latter category, because they are intended to be usable from almost any telecommunication device.

Another old problem that gets worse with shared devices is signal ambiguity. If both default and alternate feature sets contain similar features, it is likely that both features will respond to the same signal. In this case it is inherently difficult to determine which feature the signal was meant for, and to ensure that the intended feature is the only one that receives it.

Since alternate feature sets must use restricted, standard signals, there is an additional incentive for default feature sets to use special, customized signals. Not only does it provide a better user interface, but it also reduces the danger of signal ambiguity when the device is shared.

Ultimately, the only way to guarantee the absence of signaling problems is to anticipate all possible combinations of feature sets and analyze the compositions. This is not an easy task, but at least DFC feature composition provides a structure within which the analysis can take place.

8. Conclusion

In this paper we have presented three new categories of feature interaction in telecommunication services: (1) interactions among media-choice features in multimedia systems, (2) desirable interactions that achieve selective application of feature sets, and (3) interactions between multiple feature sets sharing a device. The latter two categories of interaction can be found in both mobile and multimedia services.

Throughout the paper we have used the Distributed Feature Composition (DFC) architecture to expose and elucidate issues and problems. We have also used it to describe various systems in which the features have all the desired interactions and no undesired interactions.

Several properties of DFC make it work particularly well for this purpose: (1) Feature composition in DFC is structured so that the most blatant feature-interaction problems,

such as outright inconsistency, are eliminated by the architecture. Features can be checked automatically for conformance to the basic rules. (2) The desired system behavior can always be described in DFC, and almost always without loss of feature modularity. (3) Many important aspects of feature composition can be represented graphically. (4) The architecture makes it possible to separate protocol and user-interface concerns from the concerns of routing, addressing, and feature application. Many issues concern one kind of complexity, but not both. (5) DFC also lends itself to a higher-order abstraction in which we can talk about feature sets as wholes almost as easily as we talk about features, including issues of application and interaction.

Because of these properties, DFC has been an invaluable aid in anticipating difficulties and opportunities in the telecommunication services of the future. We know of no other scheme for feature description and composition that can do as much.

Acknowledgments

Our colleagues Greg Bond, Eric Cheung, Paul Fuoss, Hal Purdy, and Chris Ramming have contributed greatly to our understanding of these issues.

References

[1] E. Jane Cameron, Nancy D. Griffeth, Yow-Jian Lin, Margaret E. Nilson, William K. Schnure, and Hugo Velthuijsen. A feature-interaction benchmark for IN and beyond. In L. G. Bouma and H. Velthuijsen, eds., *Feature Interactions in Telecommunications Systems*, pages 1-23. IOS Press, Amsterdam, 1994.

[2] Nancy D. Griffeth and Hugo Velthuijsen. The negotiating agents approach to runtime feature interaction resolution. In L. G. Bouma and H. Velthuijsen, eds., *Feature Interactions in Telecommunications Systems*, pages 217-235. IOS Press, Amsterdam, 1994.

[3] Robert J. Hall. Feature interactions in electronic mail. In this volume.

[4] Michael Jackson and Pamela Zave. Distributed feature composition: A virtual architecture for telecommunications services. *IEEE Transactions on Software Engineering* XXIV(10):831-847, October 1998.

[5] Simon Tsang, Evan H. Magill, and Bryce Kelly. The feature interaction problem in networked multimedia services—present and future. *British Telecom Technology Journal* XV(1):235-246, January 1997.

[6] Simon Tsang, Evan H. Magill, and Bryce Kelly. An investigation of the feature interaction problem in networked multimedia services. In *Proceedings of the Third Communication Network Symposium,* pages 58-61. Manchester, England, July 1996.

[7] Pamela Zave. Architectural solutions to feature-interaction problems in telecommunications. In K. Kimbler and L. G. Bouma, editors, *Feature Interactions in Telecommunications and Software Systems* V, pages 10-22. IOS Press, ISBN 90-5199-431-1, 1998.

[8] Pamela Zave. Systematic design of call-coverage features. Submitted for publication, 1999.