IOS Press Copyright Notice

Published in: ***Feature Interactions in Telecommunications and Software Systems VIII (ICFI'05),*** 2005

"Categorizing and Prioritizing Telephony Features"

Cite as:

P. Ann Zimmer and Joanne M. Atlee. 2005. Categorizing and Prioritizing Telephony Features. In *Feature Interactions in Telecommunications and Software Systems VIII (ICFI'05),* Stephan Reiff-Marganiec, Mark D. Ryan (Eds.). IOS Press, Amsterdam, pp. 327-333.

BibTex:

```
@inbook{ZimmerAtlee05,
 author = {Zimmer, P. Ann and Atlee, Joanne M.},
 title = {Categorizing and Prioritizing Telephony Features},
 booktitle = {Feature Interactions in Telecommunications and Software Systems VIII (ICFI'05)},
 series = {ICFI'05},
 year = {2005},
 pages = {327-333}
}
```

Categorizing and Prioritizing Telephony Features

P. Ann Zimmer and Joanne M. Atlee
School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
{*pameade, jmatlee*}*@se.uwaterloo.ca*

Abstract. Many approaches to the feature-interaction problem propose to resolve interactions among modular features by feature priority. However, feature-rich services, like telephony, offer hundreds of features, raising the question of whether the task of analyzing and prioritizing features is feasible. This article investigates ordering features by their essential purposes and goals. We discuss how a set of features, collected from customer-service manuals, feature-interaction benchmarks, and feature-interaction contests, might be categorized and how these categories could be prioritized with respect to one another. Features within a category are prioritized according to secondary concerns.

1 Introduction

A key approach to ensuring interoperability among independently developed plug-in features lies in generic architectures, design rules, and protocols that constrain and control how features interact with each other. For example, AT&T's Distributed Feature Composition (DFC)'s pipe-and-filter architecture [7], precedence rules for dispatching input events to features [14], call filters [4], and patterns [12] all resolve interactions by serializing features' reactions to events. Serialization seems to be effective in resolving conflicts over competing features that react to the same event. However, it raises questions as to whether a large feature set can be serialized.

We propose categorizing a feature by its goals or functionality, and then reasoning about how to serialize feature categories. For example, the goal of Terminating Call Screening (TCS) is to block unwanted incoming calls from reaching the subscriber. As another example, a number of features have varying goals to present varying information to the user; but their common functionality is that they present information. Given a collection of feature categories, we can reason about how to serialize categories, to avoid conflicts and to affect desired interactions. For example, features that authorize users should be ordered such that they execute before the user's features execute. The priorities of features within a category may be unconstrained, if the features all react to different events; or they may be ordered to realize the user's or service provider's desired interactions (e.g., giving Call Waiting priority to react to an incoming call to a busy number, so that Voice Mail activates only if Call Waiting is unsuccessful). This approach to prioritizing features realizes worked-out resolutions to unintended interactions between feature categories, and allows the user or service provider to focus on resolving interactions between features within the same category.

To date, little related work has been done on how to prioritize features. Elfe et al.[5] test pairs of features for constraint violations, and (1) reorder and retest any pair of features whose initial ordering violates a constraint; (2) choose the non-violating feature order, if there is one; and (3) prompt the user for a feature ordering if both tested orderings violate constraints. Bruns et al. [2] identify allowable orderings between pairs of features in a similar manner. Tsang and Magill [11] determine feature priorities dynamically, by selecting for execution the enabled feature with the fewest constraint conflicts; a static priority scheme

is used to break ties. These related works could be combined with our work, and used for intra-category ordering of features.

The rest of the paper is organized as follows. Section 2 summarizes background terminology. Section 3 presents some principles used when ordering features. In Section 4, we present the results of our attempt to categorize and serialize 35 distinct telephony features, collected from customer-service manuals, feature-interaction benchmarks, feature-interaction contests, and research papers. We conclude in Section 5.

2 Background and Terminology

We restrict ourselves to features that are implemented as independently developed modules and that are plugged into an architecture, which coordinates the features' executions and communication. A feature may be implemented as several modules; the term **module** refers to the basic entity that can be prioritized, and the term **feature** refers to a feature and the modules making up its implementation. Each feature has two or more communication **ports** by which it sends and receives signals. A **signal** refers to an inter-feature communication, which may be a message, an event, a method invocation, and so on.

A **call** is sequence of communicating modules. The end points of an **established** call are the users, but if the call is in the process of being set up or torn down, then the end points could be modules. On a given call, we distinguish between the user (**caller**) who initiates, or whose features initiate, the call and the user (**callee**) who receives the call. The informal terms, **outgoing call** and **incoming call**, refer to a special call-request signal that establishes a call by incorporating modules into the call one by one.

Using DFC terminology [7, 13], the call is partitioned into the **source region**, which comprises the caller's part of the call (e.g., the caller's device and features) and the **target region**, which comprises the callee's modules. Figure 1 shows an established call: a series of modules from the caller's device, device interface module, and features; then through the callee's features, and device interface module; and ending at the callee's device. A **free feature** is one where a new instance is created for each call. For example, if a user is involved in a call and features have been instantiated for that call, then a second call to that user will spawn new instantiations of the user's free features. In contrast, a **bound feature** is one where there is a static instance for each user, and all calls involving the feature's user are routed through this instance. Any feature that, like Call Waiting, needs to know about all calls to the user is implemented as a bound feature.

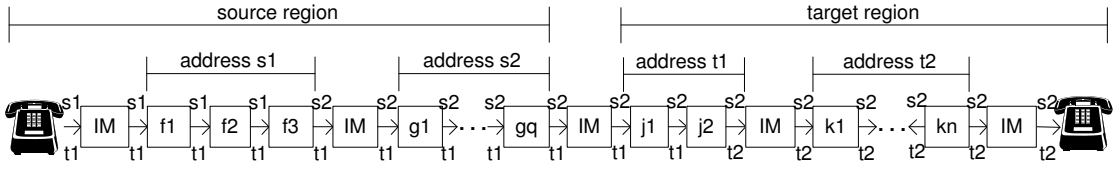

Figure 1: An established call, starting on the left with an initial source address $s1$ and an initial target address $t1$. The arrows between modules are annotated (from above) with the current source address and (from below) with the current target address. Feature f_3 in address space $s1$ redirects the call through a second source address $s2$; and on the callee's side, feature j_2 in address space $t1$ redirects the call to a second target address $t2$.

A call's source and target regions may each be subdivided into multiple addresses, each with its own feature set. For example, there will be multiple addresses in the target region if the initial call request is forwarded to another address. In such a case, the features $j_1, ..., j_m$ associated with the initial target address are included in the call, but only up to the feature that, like j_2 in Figure 1, redirects the call to a new target address; after that, the feature set

$k_1, ...k_n$ associated with the new target address is incorporated into the call. Similarly, there will be multiple addresses in the source region if the caller routes her call through different originating addresses - such as routing a call made from home through a work address. Such a call incorporates the features $f_1, ..., f_p$ associated with the home number, but only up to the feature that, like f_3 in Figure 1, redirects the call; after that, the features $g_1, ...g_q$ subscribed to by the work number are incorporated into the call.

We categorize features by their goals or functionality. A feature's **goals** are user- or service-provider-defined objectives to be achieved by the feature. A feature's **functionality** is its behaviour, and is an attempt to realize its goals.

3 Principles for Ideal Feature Orderings

We have identified a set of principles for ordering features that, when taken together, help to avoid conflicts and achieve desired interactions between disparate features. The principles apply to the set of features associated with a particular address. Although not all of the principles will be satisfied all of the time, we order the feature categories to optimize adherence to these principles.

Personalization: Aliases should be used, when they exist, to present user-related information to subscribers. For example, a subscriber would rather be notified of an incoming call from "Mom", than be presented with the concrete address of the incoming call. Features that record address data and that subsequently initiate calls on behalf of the subscriber should record and use aliases, to take advantage of intervening changes to alias assignments.

Concretization: As the dual of Personalization, features should use concrete addresses when deciding how to process a signal. For example, a blocking feature should operate on fully expanded target addresses, to abort calls to both (translated) aliases and addresses that appear on blocking lists (e.g., the user wants to block calls that incur long-distance charges).

Authorization: A user's identity must be verified before the user can interact with any of his features. A call is not established until the users are authenticated.

Invoicing: For every call to another address, billing information about the current and next addresses must be recorded. Also, the current address is billed for the cost of the subcall to the new address, unless a feature in an alternate address category agrees to accept the charges. For example, when a feature redirects a call from one target address to another target address, the first target address will be charged for the cost of the subcall from its address to the new target address.

Abortion: The subscriber can create blocking lists, for both incoming and outgoing calls, such that calls made to or from numbers that appear on the relevant blocking list are terminated before they are established. As well, no information about such terminated calls is presented to the subscriber or recorded by the subscriber's features.

Accessibility: All of the features associated with the end users' current addresses will be included in every established call. Each end user expects that her full set of features are accessible to her for the duration of the call.

As mentioned above, the principles apply to the features associated with a particular address. Thus, a subscriber whose calls pass through more than one source address might have multiple blocking lists, one for each source address; each of the addresses is billed for the subcall to the next address; and for each of the call's addresses, all of the features associated with that address are included in the call.

4 Classification of Feature Categories

This section presents early work on categorizing features, and ordering categories. We categorize features by their goals and their essential functionality, and then we reason about how to order the categories to best adhere to the ideal feature-rrdering principles. We start this section by showing the results of our partial ordering of feature categories; we subsequently describe the various feature categories we have identified and briefly explain how they are placed within the partial order. We do not claim that our proposed feature categories are complete or that they generalize to other domains. This work merely evaluates, within the telephony domain, whether such an approach to feature prioritization might be feasible.

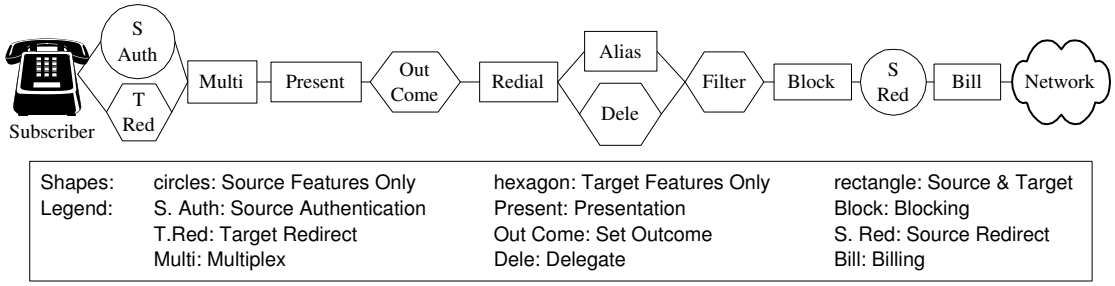

Figure 2: Partial Ordering of Feature Categories.

A **Source Authentication** feature is a source-region feature (represented as a circle in Figure 2) that verifies whether the caller is authorized to initiate a call from or through a particular source address. For example, Remote Access Authentication prompts the caller for a password before allowing the call to pass through its remote address. To satisfy the Authorization principle, we place the Source Authentication category next to the Subscriber, thereby ensuring that none of the remaining features in the address category are applied before the caller is authenticated.

A **Target Redirect** feature is a target-region feature (represented as a hexagon in Figure 2) that redirects a call attempt to another target address, with the intent of reaching the original callee. For example, Call Forward on No Answer responds to no one answering at the original target, and redirects the call to another number where the callee might be reached. To adhere to the Accessibility principle, we place the Target Redirect category next to the Subscriber, so that all of the target-address's features are incorporated into the call before a Target Redirect feature redirects the call to a new address. The Target Redirect features can verify that the person who answers the phone is the intended callee before allowing the user to interact with any of the address's features; thus, target zones do not need a separate Target Authentication feature category. Because Source Authentication and Target Redirect categories apply to different regions, their features are never active in the same call, and so there is no constraint on their respective orderings.

A **Multiplex** feature operates as both a source and target feature (represented as a rectangle in Figure 2), and allows the subscriber to be involved in multiple calls simultaneously (e.g., Conference Calling). Multiplex features are bound features, meaning that each subscriber has one static instance of each Multiplex feature, and all of the subscriber's calls pass through these feature instances. This way, a Multiplex feature is aware of all of the subscriber's calls and call attempts, and can coordinate among the calls as per their respective goals. Because Multiplex features are bound and have a single instance that applies to all calls, any nonMultiplex feature that maintains information about the remote user must be placed on the Network side of the Multiplex category. Moreover, if a Multiplex feature is active in a call (i.e., a connection is already established between the Subscriber and the Multiplex category) and it receives the call-setup signal for a second call, it will not progagate the

set-up signal and instead will reuse the existing connection. For example, Call Waiting sends a special notification signal along the voice channel, rather than propagating the call-setup signal. Features that lie on the call path between the Subscriber and the Multiplex features and that react to a call-setup need to be modified to react also to such Multiplex-specific setup signals. To reduce the number of features that lie in this call path and that may need to be modified to accommodate alternate forms of call-setup notifications, we place the Multiplex features as close to the Subscriber as possible: next to the Source Authentication category.

A **Presentation** feature presents information about the call to a user. For example, Call Display will display information about an incoming call, such as the caller's alias. The Personalization principle implies that the Presentation category be placed on the Subscriber side of the Alias features, so that information presented to the subscriber includes any available aliasing information.

A **Set Outcome** feature can assert the outcome of a call attempt, by issuing a signal on behalf of the callee. For example, Make Set Busy (MSB) intercepts all incoming call-setup signals and issues an unavailable (busy) signal in response - regardless of the current state of the intended callee. All features that receive the asserted outcome signal will behave as if the signal came from the callee. Therefore, Set Outcome features must be on the Subscriber side of all categories, such as Delegate, that respond to asserted outcome signals.[1] To adhere to the Presentation principle, we place the Presentation category on the Subscriber side of the Set Outcome category, so that no information about a call terminated by a Set Outcome feature is presented to the subscriber.

A **Redial** feature is used to place a call to a previously recorded address. The most basic Redial feature places a call to the target (source) address of the last outgoing (incoming) call. To adhere to the Personalization principle, we place the Redial category on the Subscriber side of the Alias category, so that if the last dialed number uses an alias, then the redialed call can also make use of the alias. This way, if an alias mapping is time-sensitive, the redialed call is made to the appropriate address. To adhere to the Logging principle, we place the Redial category on the Network side of the Set Outcome category, so that information about incoming calls are recorded even if the call attempt is eventually unsuccessful.

An **Alias** feature allows the user to use an alias, such as a name or a Speed Dial code, to refer to a remote party's address. For outgoing calls, the feature translates the alias into its associated network address, possibly using additional information, such as the time of day. For incoming calls, the feature applies the inverse, translating the source address of the call into its corresponding alias. For example, a Personal Directory (PD) feature maintains a list of mappings from names to addresses, which can be used in either direction. Placement of the Alias category is determined by how other feature categories adhere to the Personalization and Concretization principles.

A **Delegate** feature redirects an incoming call to an agent acting on the subscriber's behalf (e.g., voice mail services, secretary). Some features, like Call Forward Universal, delegate immediately upon receiving a call-setup signal; others, like Voice Mail, delegate upon receiving a call-attempt-failed signal from the callee, or from a Set Outcome feature. Because the goal a Delegate feature is to redirect an incoming call to a delegate, as opposed to redirecting the call to another number in a continuing attempt to reach the callee[2], there is no need to adhere to the Accessibility principle and try include all of the callee's features into the call. As well, if the callee has delegated a call, then by the Logging principle, there is no need to

[1] In contrast, we place Target Redirect on the Subscriber side of Set Outcome, because Target Redirect features attempt to reach the Subscriber, whereas Set Outcome features explicitly indicate that the Subscriber is not available.

[2] Several telephony features currently serve as both Delegate features and Target Redirect features. For example, Call Forward features can redirect a call to either the callee or a delegate. We recommend that these features be split into two distinctly named features, one for each category.

record the call attempt to follow up with the caller. Hence the Delegate category is placed on the Network side of the Redial category.

A **Filter** feature selectively blocks or redirects calls not meant for the callee's address. For example, Automated Role Identification (ARI) reacts to all incoming calls by offering the caller a menu of options of how to direct the call, and by redirecting the call according to the caller's response. Automated Filter features often query the caller, and the message played may include potentially private information; therefore, the Filter category should reside on the Subscriber's side of, and directly after the Blocking category.

A **Blocking** feature prevents the completion of an outgoing (incoming) call whose target (source) address is found in the feature's blocking list. For example, Originating Call Screening aborts calls that originate from the subscriber or her features that are destined for a target address in the Screening List. The Blocking category dispenses with undesired calls, where as the Filter category handles misdirected or to-be-directed calls; as a second distinction, a Filter feature may be willing to reveal information to a caller, to help the caller complete her call.

For a Blocking feature to satisfy the Abortion principle, it must intercept and react to all outgoing call requests made on the subscriber's behalf. Categories capable of initiating a call include Multiplex and Redial; all incoming call requests must be intercepted before reaching these features. Moreover, Blocking features must receive fully expanded network addresses, as per the Concretization principle, so every outgoing call request must pass through the Alias features before reaching the Blocking features. Therefore, we place the Blocking category on the Network side of the Multiplex, Redial, and Alias categories.

A **Source Redirect** feature is used to route an outgoing call through another source address. A caller might do this to change the caller-identification information that the callee sees, to bill an outgoing call to the new source address, or to access the features associated with the new source address. For example, Remote Access (RA) allows a caller at one location to dial a special access code to link to a remote address. The outgoing call will appear as though it originated from the remote address, and the caller will have access to any features associated with the remote address. To adhere to the Accessibility principle, we place the Source Redirect feature category at the end of the partial order, next to the Network. This ensures that if the call's source address is changed most of the features associated with the old source address will have already been applied, and are not skipped by the change in address zones.

A **Billing** feature records billing information for all call attempts. Billing for the call requires that every address segment in the call have a Billing feature that bills for the portion of the call from this address to the next address, where the "next address" is in the direction of the callee. [3] Thus, a source address is "billed" for redirecting the call to another source address (e.g., from a home address to a work number). This means the owner of the last address in a call's source region pays for the portion of the call from the source region to the target region. The Invoicing principle ensures that the address of the next segment is known to the Billing feature, hence it must lie on the Network side of the Source Redirect category, which may change the address. However, this placement contradicts Section 2, which states that redirected calls stop incorporating features associated with the old address. For Billing to work, the system architecture or the call-routing algorithm must incorporate the Billing features whenever the call is about to leave an address segment because of a redirection. Features from other categories may also charge for their use. Such pay-per-use features are

[3] Alternately, Billing features may change who is billed. For example, a Collect Call (CC) feature reverses the charges incurred by a Billing feature. When CC is invoked, a message is played on the voice channel requesting authorization to reverse the charges. The call will not be completed, or continued, unless authorization is received from the next address segment.

responsible for sending billing information to the billing database directly. However, this aspect of their functionality does not contribute to their classification, hence they are not categorized as Billing features.

5 Conclusions, Limitations and Future Work

We have surveyed many different sources of features [1, 3, 6, 8, 9, 10] and found 35 distinct features, from a selection of over 300 features. That we have been able to classify over 300 features into 12 categories gives us some confidence that the number of categories is not equal to the number of features. The introduction of new features may require the creation of new categories; ideally, we would be able to establish principles for ordering the new categories, and find a partial-order among the feature categories that best adheres to these principles. As anecdotal evidence that new feature categories do necessarily invalidate previous principles and orderings, we have already identified specialized feature categories that deal with device-interface modules, which would be found only in device-address spaces. We have also considered **dual features**, whose modules are distributed among the caller's and callee's address zones, so that a caller's feature can affect the callee's service (e.g., Busy Override).

We are currently formalizing our feature categorizations and are developing proofs to verify that our partial ordering, presented in Section 4, satisfies our Ideal Feature-Ordering Principles and the correctness criteria of each feature category (e.g., that Blocking features block unwanted calls). Future work includes considering features that fit into multiple categories (e.g., perhaps such features can be decomposed into modules that are more easily categorized) and exemption features, which prevent other features from functioning.

6 Acknowledgements

The authors would like to thank Pamela Zave for discussions and feedback on particular feature categories and orderings among them.

References

[1] 3Com®. *IP Telephony Jargon Buster and Glossary Technical Guide*. http://www.3com.com/voip/assets/3com_200251-003.pdf.

[2] G. Bruns, P. Mataga, and I. Sutherland. Features as service transformers. In *Feature Interactions in Telecommunications and Software Systems V*. IOS Press, 1998.

[3] E. J. Cameron, N. D. Griffeth, Y.-J. Lin, M. E. Nilson, W. K. Schnure, and H. Velthuijsen. Feature Interaction Benchmark for IN and Beyond. In *Feature Interactions in Telecommunications Systems II*, pages 1–23, 1994.

[4] D. Cattrall, G. Howard, D. Jordan, and S. Buj. An Interaction-Avoiding Call Processing Model. In *Feature Interactions in Telecommunications Systems III*, pages 85–96, 1995.

[5] C. D. Elfe, E. C. Freuder, and D. Lesaint. "Dynamic Constraint Satisfaction for Feature Interaction". In *BT Technology Journal*, volume 16, pages 38–45, July 1998.

[6] N. Griffith, R. Blumenthal, J.-C. Gregoire, and T. Ohta. Feature interaction detection contest. In *Feature Interactions in Telecommunications and Software Systems V*. IOS Press, 1998.

[7] M. Jackson and P. Zave. "Distributed Feature Composition: A Virtual Architecture for Telecommunications Services". *IEEE Transactions on Software Engineering*, 24(10):831–847, October 1998.

[8] M. Kolberg, E. Magill, D. Marples, and S. Reiff. Second feature interaction contest results. In *Feature Interactions in Telecommunications and Software Systems VI*. IOS Press, May 2000.

[9] Nortel Networks. *Nortel Network's Centrex Feature Library*. http://www.nortelnetworks.com/products/01/centrex/library/voice/.

[10] Northern Telecom. *Meridian Digital Centrex: Voice Features*, 1992. DMS-100.

[11] S. Tsang and E. H. Magill. Behaviour based run-time feature interaction detection and resolution approaches for intelligent networks. In *Feature Interactions in Telecommunications and Distributed Systems IV*, 1997.

[12] G. Utas. “A Pattern Language of Feature Interaction”. In *International Workshop on Feature Interactions in Telecommunications Systems V*, pages 98–114, 1998.

[13] P. Zave. Address translation in telecommunication features. *ACM Trans. Softw. Eng. Methodol.*, 13(1):1–36, 2004.

[14] I. Zibman, C. Woolf, P. O’Reilly, L. Strickland, D. Willis, and J. Visser. “Minimizing Feature Interactions: An Architecture and Processing Model Approach”. In *International Workshop on Feature Interactions in Telecommunications Systems III*, pages 65–83, 1995.