Feature Interactions in the Global Information Infrastructure

(Panel Session Introduction)*

Alfred V. Aho
Department of Computer Science
Columbia University

Nancy D. Griffeth
Bellcore
Morristown, NJ

Abstract

The international telecommunications system is the world's largest distributed computing system, offering a wide range of services and service features. Telecommunications service providers are eager to offer new services on this infrastructure, but the development of these new services is hampered by interactions of the service features among the different services. Undetected and undesirable feature interactions cause confusion and dissatisfaction among the users of services and add delay and expense to the development and deployment of new services. This panel reviews the progress that has been made in anticipating and controlling undesirable feature interactions in telecommunications services. This paper sets the stage by discussing the impact of the feature interaction problem on the different phases of the software development lifecycle.

1 Introduction

Two or more features of a telecommunications service are said to interact if one changes the functionality of the others. In developing new telecommunications services, it is essential to determine and manage the interactions of features among the new and existing services. Customers affected by unexpected and unwanted feature interactions will become confused and ultimately dissatisfied with the performance of the new services.

Feature interactions impact all phases of the software lifecycle. Timely mechanisms for resolving untoward feature interactions at all stages of the software lifecycle must be part of the new service development process. Otherwise, feature interactions will complicate system development and maintenance, causing delay in the introduction of new services to the market[2].

We view the feature interaction problem as a software quality problem. As with all defects, it would be most desirable to eliminate undesired feature interactions at the specification stage, but this may not always be possible because feature interactions may not be discovered until a new service is deployed in conjunction with services and features offered by other service providers. The problem becomes one of determining the most cost-effective techniques for detecting and removing undesired feature interactions during the lifecycle of new service development.

2 Sources of Feature Interactions

Since the first International Workshop on Feature Interactions in Telecommunications Software Systems held in December 1992, the community has developed a much more comprehensive understanding of the nature and causes of feature interactions in telecommunications systems. Cameron et al.[3] categorize the nature of feature interactions according to the kind of features involved, the number of users affected, and the number of network components in the interaction. A prototypical example of a single-user single-component interaction occurs between call waiting and answer call. Suppose a subscriber has both of these features. If the subscriber is on a call when a second call arrives, should he receive a call-waiting

* Author's addresses: Alfred V. Aho, Department of Computer Science, 500 West 120th Street, Columbia University, New York NY 10027, email: aho@cs.columbia.edu; Nancy Griffeth, Bellcore MRE 2Q-382, 445 South Street, Morristown NJ 07960. Alfred Aho did this work as a consultant at Bellcore.

tone or should the second call be forwarded to the answering service?

Since networks involve multiple components, interactions can occur if a service in one network component is either unaware of or incompatible with features of a service in other components. One example of such an interaction occurs when a subscriber with the unlisted number feature (desire to keep his number private) calls a person with calling number delivery. Here one or the other of the features must be compromised. If the network delivers the subscriber's number to the called person, the caller's privacy is compromised. If the called person doesn't receive a calling number, her feature does not work.

The causes of feature interactions are varied. Perhaps the most common is violation of the assumptions inherent in the original feature definition. Features were created to operate under a set of assumptions about their environment – how network resources are named and accessed, what kind of data is available, how calls are controlled, what signaling protocols are used, what administrative processes are in place and so forth. As the network and its services evolve, these assumptions may be violated and the services may no longer work.

Violation of an assumption also causes feature interactions in non-telephony software systems. Using the assumption that the window size was based on a fixed screen size, early versions of the visual editor 'vi' did not scroll properly when an X window (run as an 'xterm') was resized. Subsequently, SIGWINCH (signaling that the window size has changed) has made it possible to fix this problem.

Another cause of feature interactions stems from limited signaling capabilities and limitations on network components. The current telephone handset can send only 14 signals to the switch: *, #, the ten digits, flashhook, and disconnect. For some services, the length of time the switchhook is pushed determines whether it signals a flashhook or disconnect. Thus limitation of signaling capabilities makes it difficult to disambiguate requests.

Since the telecommunications network is a large, distributed system, many of the problems dealing with large reactive systems are present. Resource contention, timing, and race conditions are typical problems. Another non-telephony example involves contention for entries in a frame display buffer. In a windowing system, contention for entries in an 8-bit frame display buffer limits the colors that can be used by a window that shares the global colormap; but using a local colormap for the window creates strange effects when it's not the active window. Inadequate communication between different systems can also hinder resolution of an interaction. For example, 'vi' still cannot handle window resizing if the user is logged in remotely.

3 Approaches

A number of promising approaches for attacking the feature interaction problem have been proposed and are being investigated. These approaches are well-represented in three collections of papers[7, 6, 1]. Techniques and tools have been proposed for avoiding, detecting, and resolving interactions. These techniques and tools can be embodied in processes that improve software quality in the service development lifecycle.

There are several techniques that can be used to avoid or reduce feature interactions. These include specifying and developing open systems architectures with well-defined interfaces that can be used in the implementation of services [10]. Effective quality guidelines that can be incorporated into the service-creation development process are clearly beneficial. These architectural guidelines can play a significant role in making explicit what assumptions are being made in the definition of new features and in removing interactions that stem from arbitrary choices made during implementation.

Better methods for specifying new services in a platform-independent way are vital. The current method is often an informal natural language specification that is inconsistent and incomplete. More rigorous specification techniques are desirable and a number of the formal methods have been proposed for specifying services including finite-state machines, transition systems, or variants of temporal logic[11, 5]. The current formal methods, however, have substantial practical limitations. People require extensive training to use them, the tools that support them are often difficult to incorporate into the standard production software development environment, and many of the methods cannot be used in an incremental fashion to incorporate changes or modifications. understood by wide classes of users and fully automated methods for verifying that an implementation satisfies a specification are still not adequate to cope with the complexity of telecommunications services. Nevertheless the use of tools to define and verify the properties of features is an important area of investigation.

For detecting feature interactions from the specifications, a wide variety of verification, validation, and simulation tools have been developed. The ATR Communications Systems Research Laboratories in

Japan have used this approach [9]. They have developed a general framework for specifying services using finite state machines. They describe feature interactions in terms of behaviors on states, such as a state that has no succeeding transitions, or as a state with multiple transitions for a single event. They have developed tools incorporating these methods to test a large number of telecommunications services for feature interactions.

Feature interactions also make integration testing difficult because of the explosion of test cases. One method of addressing this provides tests that examine multiple interactions simultaneously[4].

In a competitive environment with multiple service providers, resolution of feature interactions is a thorny problem. It would be desirable if every service and feature worked in a uniform, simple, easy-to-understand manner, no matter who is the service provider. Good documentation and customer interaction are essential. Techniques that have been proposed for on-line resolution of feature interactions include feature-interaction managers, negotiation, and event-based mechanisms.

Because the number of potential interactions increases considerably faster than the number of features, feature interaction management will not be feasible without general rules for resolving interactions. One simple, general method for resolving interactions is to prioritize features to determine which feature controls a call when more than one is active. A more flexible approach is to automate negotiation between different parties to a call in order to determine which call is set up when different parties have different preferences. An easily-implemented method of negotiating is described in [8].

4 Conclusions

As the feature interaction problem is becoming better understood, we are in a position to construct

- more rigorous specifications of services and features
- tools for analyzing and generating new services and features
- tools for detecting and resolving interactions
- tools for experimenting with new features

The major challenge is to integrate these tools into the service-creation software development environments.

References

[1] L. G. Bouma and H. Velthuijsen, editors. *Feature Interactions in Telecommunications Systems*. IOS Press, Amsterdam, 1994.

[2] T.F. Bowen, F.S. Dworak, C.-H. Chow, N.D. Griffeth, G.E. Herman, and Y.-J. Lin. Views on the feature interaction problem. In *Proceedings of the 7th International Conference on Software Engineering for Telecommunications Switching Systems*, pages 59–62, London, July 1989.

[3] E.J. Cameron, N.D. Griffeth, Y.-J. Lin, M.E. Nilson, W.K. Shnure, and H. Velthuijsen. A feature interaction benchmark in IN and beyond. In *Feature Interactions in Telecommunications Systems*, pages 1–23. IOS Press, Amsterdam, 1994.

[4] D. M. Cohen, S. R. Dalal, A. Kajla, and G. C. Patton. The automatic efficient test generator (AETG) system. *Proceedings of the Fifth International Symposium on Software Reliability Engineering*, pages 303–309, November 6–9 1994.

[5] A. Fekete. Formal models of communication services: a case study. *Computer*, pages 37–47, August 1993.

[6] N. Griffeth and Y.-J. Lin, editors. *Communications*, volume 31. IEEE, August 1993.

[7] N. Griffeth and Y.-J. Lin, editors. *Computer*, volume 26. IEEE, August 1993.

[8] N. Griffeth and H. Velthuijsen. The negotiating agents approach to runtime feature interaction resolution. In *Feature Interactions in Telecommunications Systems*, pages 217–235. IOS Press, Amsterdam, 1994.

[9] T. Ohta and Y. Harada. Classification, detection and resolution of feature interactions in telecommunications services. In *Feature Interactions in Telecommunications Systems*, pages 60–72. IOS Press, Amsterdam, 1994.

[10] R. van der Linden. Using an architecture to help beat feature interaction. In *Feature Interactions in Telecommunications Systems*, pages 24–35. IOS Press, Amsterdam, 1994.

[11] P. Zave. Feature interactions and formal specifications in telecommunications. *Computer*, pages 20–31, August 1993.