Fundamental Nonmodularity in Electronic Mail*

ROBERT J. HALL  
bob-3NonModFIIs@channels.research.att.com  
AT&T Labs Research, 180 Park Ave, Bldg 103, Florham Park, NJ 07932, USA

Abstract. Electronic mail (email) systems have grown in complexity to the point where their reliability and usability are coming into question. The authors of individual email components are widely distributed in both time and space, bound together only by message structure and transfer protocol specifications; consequently filters, autoresponders, and various security features may interact in unanticipated and, often, incomprehensible ways. This paper describes a formalism for modeling and composing specifications of email features, and a feature interaction detection methodology based in part on human intuition and in part on simulation and formal test coverage. The appendix lists 27 interactions found from applying the methodology to ten common email features, a result of independent interest considering the age and relative maturity of the email domain. The paper then proceeds to categorize the interactions according to their impact on the design of the system. This includes a design study of the most natural ways to fix the undesirable behaviors. From this we can infer non-modular dependencies among the features, leading to the result that 9 of the 10 features must be revised (and a custom user interface must be built for them) after they are composed and after feature interactions are detected. This pervasive nonmodularity shows that feature interaction analysis is necessary to optimizing the correctness of an email system design.

Keywords: specification, validation, feature interaction, electronic mail

1. Introduction

Electronic mail (email), one of the oldest and most popular services on the Internet, has grown in complexity since its original governing specifications (Postel, 1982; Crocker, 1982) were published. Unlike the telephone system, whose engineering and evolution have been relatively controlled, email components have been developed by many different entities, not all of whom were aware of each other’s efforts, nor had the same sets of knowledge. Thus, we might expect that when messages traverse components of diverse origins, the system will occasionally behave undesirably; however, this potential has increased dramatically with the advent of aggressive filters and other security features. While each new feature is targeted at a pressing problem, such as message privacy or avoiding unwanted mail (“spam”), the overall effect is to make email less reliable and less comprehensible, because of an increase in unintended behavior due to feature interactions.

This paper’s main thesis is that to adequately design email systems it is necessary to be able to analyze feature combinations and to become aware of requirements violations prior to experiencing them first hand. The first goal of this paper is to provide a semi-automatic technique to do just this and to document 27 undesirable feature interactions found in this

*Significant portions of this paper appeared in the paper “Feature interactions in electronic mail” in the proceedings of the Sixth Feature Interaction Workshop (FIW’00), ©IOS Press, 2000. This paper extends those results with a finer grained analysis and categorization, a design study of fixing the interactions, and results on non-modularity.
way, a result of independent interest considering the age and relative maturity of the email domain.

From that starting point, we move on to study the question of whether modular component-based design can provide a good approximation to a fully correct system behavior within this domain and feature set. The straw man we target is the idea that components can be simply taken “off the shelf”, assembled, and then with relatively little “tweaking” yield a correct system. The result of this study will be to show that in this domain and feature set modular design alone is not viable for producing a correct email system, because correct behavior requires almost all features to break modularity. A feature breaks modularity if it has a special-purpose behavior in order to accommodate requirements induced by the presence of a particular other feature or set of features. I show that this modularity breaking is so pervasive in email that almost all of the features must be revised after all are initially designed, selected, composed, and analyzed for interactions. This is not simply an artifact of poor feature implementations, as a perturbation study shows that even with the least modular features “improved”, the same basic result holds. We even discover evidence that no perfect, interaction-free feature set can exist, which suggests that the goal of design is to optimize the correctness of the system with respect to its stakeholders’ goals, rather than to produce a perfectly correct system. The conclusion of this part of the paper, then, will be that modular design must be augmented with late feature interaction analysis and feature revision in order to optimize correctness of the system.

In summary, the purpose and contributions of this paper are as follows.

- Section 2 describes ten common, modular email features and shows how they and their combination can be formally specified in a way that supports the use of formal analysis tools;
- Section 3 describes an interactive analysis technique grounded in simulation and coverage analysis;
- Section 4 describes applying the technique and summarizes the 27 interactions found, with the full list given in Appendix A; and
- Section 5 studies the viability of modular component-based design in this domain through an analysis of how prevalent modularity-breaking is in fixing feature interactions.

Final sections discuss related work and limitations, as well as speculate on generalizing these results.

2. Email features

For present purposes, a relatively simple model of email and its features will suffice. Basically, a user originates a message, conforming to email message format standards (Crocker, 1982), from an email client program (such as Netscape Mail, Eudora, or numerous others). This message then passes through one or more email feature components (efcs) until it is delivered to the email client of the intended recipient(s). Efcs may physically reside on the client machine of the sender, any one of many server machines in the network or on the client machine of the recipient. The analysis ignores the subtleties of routing and simply