

# Diagnosing Delivery Problems in the White House Information-Distribution System

*Mark Nahabedian and Howard Shrobe*

■ As part of a collaboration with the White House Office of Media Affairs, members of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology designed a system, called COMLINK, that distributes a daily stream of documents released by the Office of Media Affairs. Approximately 4,000 direct subscribers receive information from this service, but more than 100,000 people receive the information through redistribution channels. The information is distributed through e-mail and the World Wide Web. In such a large-scale distribution scheme, there is a constant problem of subscriptions becoming invalid because the user's e-mail account has terminated. These invalid subscriptions cause a backwash of hundreds of bounced-mail messages each day that must be processed by the operators of the COMLINK system. To manage this annoying but necessary task, an expert system named BMES was developed to diagnose the failures of information delivery.

**I**n January 1993, the new Clinton administration committed itself to the use of electronic media such as e-mail (and, later, the World Wide Web) for making government information widely available to the public. A collaborative effort between the White House Office of Media Affairs, the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology (MIT), and others quickly created a workable framework for wide-scale distribution of a stream of daily documents originating from the Executive Office of the President. The document stream includes daily press briefings, speeches by the President

and other officials, backgrounders, and proclamations. In addition, the stream of released information includes special documents such as the National Performance Review's reports on reinventing government, the proposed health-care reform legislation, and the yearly budgets.

The Intelligent Information Infrastructure Project at the MIT Artificial Intelligence Laboratory created an information distribution server that functions as the focal point of the distribution chain. Documents are released from the Executive Office of the President through this system; they are sent from this system to a variety of archiving and retrieval systems around the country, most online services (for example, CompuServe, America Online), about 4000 direct subscribers to the MIT server, and a variety of other servers that further redistribute the documents. A survey of people connected to this distribution chain estimated that more than 100,000 people were receiving information through this medium.

Documents released through this service are coded with descriptive terms taken from two taxonomies: The first taxonomy categorizes the type of document (for example, press release versus speech versus press conference); the second taxonomy concerns content (for example, foreign affairs, domestic affairs, economy, taxes). Subscribers to the service specify a personal profile consisting of combinations of the descriptive terms that characterize their interests; it is the server's job to guarantee that subscribers receive exactly

*Table 1. Various Causes of Delivery Failure for E-Mail and the Type of Notification.*

<b>Kinds of Mail Notification</b>	<b>Causes of Delivery Failure</b>
Delivery failure	User not found
Delivery failure but still trying	Host not found
Message received	Mailer configuration problems
Message opened	Temporary mailer resource problem
Message deleted	DNS configuration problem
Vacation notice	User mailbox full

those documents that match their profiles in a timely manner.

Users establish a subscription and modify their profiles by filling out electronic forms (using either e-mail or the World Wide Web) and submitting them to the server. The ease with which users can manage their profiles is an important measure of the quality of service delivered.

## The Problem

The environment just described is open, large scale, and anarchic. The system services thousands of users at hundreds of sites in dozens of countries. Users may establish, modify, and terminate subscriptions at any time. User e-mail addresses registered with the server can become invalid at any time; occasionally, users cancel their subscriptions before this happens, but it is comparatively rare. Also, configuration problems at subscribers' sites make their e-mail addresses temporarily unreachable even though the addresses are valid. Table 1 lists the various causes of delivery failure for e-mail and the type of notification.

Bounced-mail messages are sent to the MIT server informing it of the inability to deliver a message to the invalid e-mail address. Most e-mail systems do not consolidate these bounced-mail messages; if you send two messages to an invalid e-mail address, you receive back two bounced-mail messages. The White House information stream typically includes as many as a dozen documents a day; with a subscription base of 4000 direct subscribers, there is a rather large volume of bounced-

mail traffic each day (more than 100 messages). The failure to handle these messages and update the subscription database accordingly leads to a perception by the administrators of the receiving sites that they are being spammed by the sending site;<sup>1</sup> given that the sending site in this case is the White House, it is unacceptable to ignore the bounced-mail traffic. A second class of problem arises when a user with a valid e-mail address attempts to terminate or modify a subscription without success; in this case, the perception is that the White House is spamming the subscriber personally, an even more unacceptable situation.

On the surface, it seems that this problem is amenable to simple automation. However, the open, anarchic character of the Internet makes the problem complex: There are dozens of different mail servers, each with a unique bounced-mail-message format. In addition to the variety of e-mail servers speaking the Internet's native simple mail-transfer protocol (SMTP) (Postel 1982), there are also a large number of other protocol domains bridged to the Internet. These domains include UUCP, Bitnet, X.400, and a large number of proprietary e-mail systems (for example, CC:mail, Microsoft Mail); bounced-mail messages are often reformatted as they cross the bridge between protocol domains, sometimes losing information (and sometimes preserving information that is useless, such as one that directs the recipient to press the F1 key for more information). Within these other mail domains, the format of a mail address might be different from that used on the Internet; bounced-mail messages from these domains often include their foreign-format e-mail address rather than the Internet-format address in our database.

A second set of complications arises from the variability of user's e-mail addresses. Many people have several e-mail addresses, some of which are forwarded to another. Bounced-mail messages in such cases often refer to the forwarded-to address, which isn't in our subscription database. Furthermore, people often subscribe using one address, switch to a second one as their primary address (forwarding the first one to the new address), and then more or less forget about the first address; attempts to modify the subscription using the new, primary address are then unsuccessful because the system is unaware of the new address. Similarly, if the new address becomes invalid, then a bounced-mail message is sent to the server referring to the new address, which is unknown to the server.

In some mail systems (for example, UNIX),

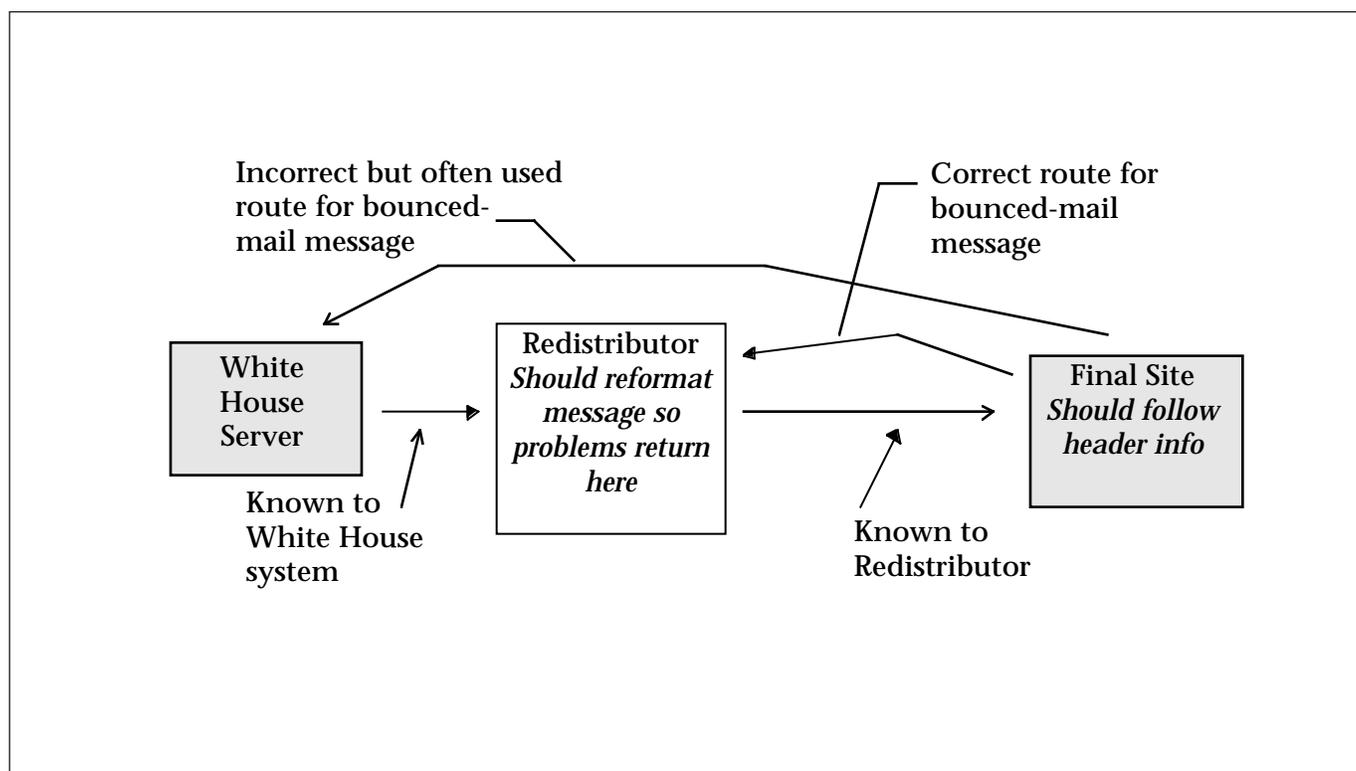


Figure 1. Redistributors Complicate Delivery Notification.

users can direct their mail streams to shell scripts or other programs for processing. *Vacation programs* are a common example; they send back to the sender a message saying that the recipient is away and unlikely to respond soon. This type of message is a courtesy when sent in response to a personal correspondence, but when sent back to a bulk distributor such as the White House server, it shows up as part of the bounced-mail stream. In addition, nothing prevents users from writing new mail-handling programs, including incorrect ones; when such programs fail, the sender of the message (as opposed to the author of the buggy program) is usually sent a bounced-mail message (in principle, the postmaster at the receiving site should be sent this message, but principles and reality don't always correspond in this world).

A final complication is that the reported failing address might not appear in our subscriber database. This can be caused by the presence of redistributors, as shown in figure 1. *Redistributors* are people or programs that receive the original message stream and then relay it to a set of subscribers known to the redistributor but not to the primary White House distribution server. Virtually any sub-

scriber can independently decide to act as a redistributor of the document stream (for example, by establishing a mailing list). If an e-mail address on a redistributor's list becomes invalid, the redistributor should be notified; however, the original source of the message (us) is often notified instead. To get the behavior we desire, the redistributor should arrange for the headers of the redistributed message to identify it as the destination for problem reports; however, some redistributors fail to configure their mailer appropriately to achieve this behavior. Bugs in either the redistributor's software or the e-mail server at the ultimate destination (for example, looking at the destination address in the message headers rather than at the message's envelope address) also cause bounced messages that concern addresses that aren't known to subscribers. COMLINK tries to distinguish redistributors from normal users by having different types of database entry for each; redistributor entries include an e-mail address for the administrator of the redistribution list. However, we rely on people to subscribe to COMLINK correctly, and often, people don't.

The problem of managing a large e-

Redistributors are people or programs that receive the original message stream and then relay it to a set of subscribers known to the redistributor but not to the primary White House distribution server.

mail-based distribution system in this environment has been recognized for some time (Westine and Postel 1991). However, to date, the problem has been handled using one of two approaches:

**Don't worry, be happy:** In this approach, bounced-mail messages are ignored. The sender builds up a rather large file of bounced-mail messages that are periodically deleted. The destination sites receive many messages that are bounced, but this happens automatically. All told, a lot of resources are wasted, but nobody really cares because it's largely invisible. To be fair, most maintainers do from time to time examine a sampling of the bounced-mail traffic and attempt to address the problems.

**Aid the administrator with a big bag of tools:** A number of ad hoc tools are built to aid the system administrator in making sense of the bounced-mail traffic (Westine and Postel 1991). These tools help the conscientious list administrator solve difficult problems, but much of the work remains manual.

Given the high visibility of the White House distribution system and its role as an early experiment in using the Internet to improve government services, neither of these approaches was acceptable. Instead, we decided to implement an expert system to aid in the handling of bounced mail and help in managing other problems, such as a user's inability to terminate or modify a subscription.

## Structure of the System

The bounced-mail expert system (BMES) is a component of a larger system, called COMLINK, which is a substrate for building information-distribution and group-collaboration systems using e-mail, the World Wide Web, and other Internet-based transport protocols. At the core of COMLINK is an object-oriented database that includes the following information: (1) *subscribers*, which includes e-mail address, personal name and subscriptions, date subscription started, and date (if any) subscription turned off and identifies whether this user is a redistributor; (2) *network hosts*, which includes subscribers at this host, upward and downward links in the domain name hierarchy, and mail server type; (3) *documents*, which includes descriptive terms, release dates, and subject; and (4) *queued tasks*, which includes time to execute the task, task type, and arguments. BMES draws on this information to help diagnose delivery failures.

BMES is a rule-based diagnostic system driven by a file of bounced-mail messages. Each

message is a symptom of a failure in the delivery system. The user of BMES is the postmaster maintaining the White House COMLINK system. BMES's task is to discover, if possible, the reason why a mail message was bounced and, if diagnosis is not possible, to present meaningful information to the user and help in gathering more information. If diagnosis is successful, then the system rectifies the problem, usually by suspending a user's subscription.

For each message processed, the system follows a standard pattern of processing: (1) classification of the mailer that sent the message; (2) abstraction of the message to hide the syntactic differences between bounced-mail messages; and (3) diagnosis of the cause of the delivery failure, which includes (4) heuristic generation of hypotheses and (5) interaction with administrators at remote sites.

The first task is *classification*, during which BMES matches features of the message against required features in the taxonomy of mailer types. In practice, the classification is done by a rather ad hoc set of rules that search for specific features in the headers and the first part of the body of the message. These features include characteristic substrings within particular headers or in specific locations within the body (usually the first several lines) of the message. These rules were determined based on the authors' observations of the bounced-mail messages.

The system currently distinguishes 23 different types of mailer. These types need not necessarily correspond to distinct pieces of mailer software; rather, they correspond to the variety of distinct formats of bounced-mail messages that we've observed. Some mailers have a broad range of configurability, including the format of the bounced-mail message to generate. We have no special knowledge of how the remote sites are being managed; so, if two distinct hosts generate bounced-mail messages that look different, we treat these messages as having been generated by distinct mailers even if not necessarily the case. New mailer types pop up occasionally, but it happens rarely.

The second stage of processing is to *abstract* the message, hiding the syntactic variability between the different formats of bounced-mail messages but preserving their semantic commonality. For example, bounced-mail messages typically contain a transcript, which includes e-mail addresses to which it was impossible to make a delivery and an indicator of the cause of delivery failure. Similarly, most bounced-mail messages contain a copy of the original message that couldn't be deliv-

**Let** the name in the reported address be ?name-1.  
**Let** the host in the reported address be ?host-1.  
**For each** child ?child-host of ?host-1,  
     **If** ?name-1@?child-host is the e-mail address of an active subscriber ?sub-1,  
     **Then** suggest that ?sub-1 is a possible cause of the delivery failure.

*Rule 1. Probable-User-Is-Child-Host.*

ered. The original message includes a set of *received* headers (Crocker 1982), each of which corresponds to a mail server in the chain of delivery. Each received header identifies a host that handled the message; the time of handling; and, in some cases, the user to whom the message was intended to be delivered (note that this address is different from the destination in the *to* header [Crocker 1982] of the message, which is typically a generic address such as *Clinton-distribution*).

Abstraction is effected using the object-oriented programming techniques of CLOS (Bobrow et al. 1988). Once the classification stage has identified the mailer type, BMES constructs a CLOS object whose class corresponds to the type of the mailer. This object mediates the abstraction phase. We established a class hierarchy corresponding to the mailer types and an object-oriented protocol<sup>2</sup> that all mail messages must obey; the protocol consists of about a dozen methods. Each method in the protocol reflects an aspect of the common semantic content that any bounced-mail message must contain. One method in the protocol finds the transcript in the bounced message, and a second one maps over its failure descriptions, calling an action routine with the e-mail address and a canonicalized version of the failure code. There are also protocol methods to locate the message text and then map over the received headers (Crocker 1982) contained in it. We use the class hierarchy to capture commonalities of message structuring. For example, the location within the bounced-mail message and the encoding of the transcript and original message are idiosyncratic to each mailer; however, several different mailers share the idea of partitioning the message body using the multipurpose Internet mail extension

(MIME) standards (Borenstein and Freed 1992) for structuring mail messages, but they can differ about what fields they include. Therefore, different classes implement the protocol methods differently, but where there is commonality, it is captured by CLOS inheritance. All mailers that use MIME encoding, for example, are represented as subclasses of the common MIME-structured message class.

The power of this approach is that it abstracts away the syntactic variability exhibited by the variety of bounced-mail-message formats but highlights their semantic commonality. Higher levels of the system can expect any mail message to contain standardized information and behave in standard ways, without having to be concerned with the underlying syntactic variability.

The next stage of processing is *diagnosis*, which involves deciding whether the failure is permanent and whether the recipient is actually known to the COMLINK system. If the address in the mail message is found explicitly in the COMLINK database, the failure is the result of the user's account being closed out (as opposed to a transient error), and the user has an active subscription, then BMES cancels the subscription.

However, sometimes the bounced-mail message reports an invalid address that is not present in the COMLINK database. At this point, the heuristic generation phase is entered. A small collection of heuristic candidate-generation rules is used to suggest candidate addresses that are in the database and that might have led to mail being sent to the address reported in the message. For example, the message might report a problem with foo@ai.mit.edu; in this case, if foo@w.ai.mit.edu or foo@mit.edu are in the database, they would be good candidates for possible causes

*The addition of some form of reverse mapping of MX records would help to identify an address on the distribution list based on an address as determined from a bounced message.*

of the failure. A rule called possible-user-at-child-host suggests the first. A second rule called possible-user-at-parent-host suggests the second. An English paraphrase of the first rule is shown in rule 1.

Such candidate-generation rules work by traversing COMLINK's map of the portions of the Internet-domain name space for which it has subscribers. Rules suggest the superior domain (for example, mit.edu is the superior of ai.mit.edu), any inferior domains (for example, w.ai.mit.edu is an inferior of ai.mit.edu), and any sibling domains (for example, lcs.mit.edu is a sibling to ai.mit.edu) that the system knows about.

Most mailers attempt to deliver a message for several days when possibly transient problems are encountered; they deliver a failure message only after this elapsed time. Because of this long latency, bounced-mail messages can continue to arrive for several days after a user's subscription has been canceled. If a bounced-mail message refers to an e-mail address whose subscription has already been canceled, then the user of BMES is not bothered because the problem has already been handled; the message is presumed to have arisen during the period between the time the e-mail address became invalid and the time COMLINK was informed of this problem. Thus, COMLINK must maintain an entry for users whose subscriptions have been canceled for a period of time after the cancellation; when BMES cancels a subscription, COMLINK creates a queued task entry in its database with a firing time of one month in the future. When this queued task runs, it completely removes the user's account from COMLINK's database. However, during the intervening period, BMES can tell that it knows about this account and that it knows that it has already canceled the account's subscription.

Most messages are handled by the simple processing described previously; however, there is usually a residual of harder problems. One cause of the residual problems is that many of the mailers provide minimally useful information in their bounced-mail messages. In other cases, there is information provided but the bounced-mail message refers to an e-mail address that isn't in the COMLINK database, and none of the prior heuristics leads to a known address either.

In almost all cases, this situation arises when the failing address is reached through an *indirection*: Either the address is on the mailing list of a redistributor, it is the target of a forwarding entry for some other e-mail address, or an MX<sup>3</sup> record (Partridge 1986) is

involved. In these cases, completely automatic processing isn't possible; not enough information is available to BMES to form a full diagnosis of the problem. Some of the required information is at a remote site and can be obtained only by communicating with an appropriate person at the remote site. It is a further complication that we don't actually know what remote site does have the information we need.

BMES can help make an educated guess: If it can find the original message included in the bounced-mail message and if there are received-from headers in the original message, then the server mentioned in the header might have relevant information. In particular, any user at this server who is marked as a redistributor in the COMLINK database is a particularly useful candidate. Redistributor entries contain an e-mail address for the administrator of the redistribution list; BMES formats the first draft of a standard e-mail message to the maintainer asking if the failing address is known to the administrator of the list and, if not, requesting help in figuring out what else might be going wrong (the user is then offered the option of further editing the text of this message). Another heuristic is to look for e-mail addresses similar to the failing one at each of the sites mentioned by the received-from headers and then send, to the postmaster at each of these sites, a message explaining the problem and asking for help.

Some techniques that we use manually today are subject to automation. One is used when a small number of users at the site bounced the mail, but it still isn't possible to make a definitive identification of the invalid address (either because the bounced message doesn't contain an address, or it contains one that doesn't match any entry in our database). In this case, we generate one message for each user in our database known at the site; this message explains that we are having delivery problems and asks for the user's help if possible. There are two useful outcomes: (1) one of the users knows what's going on and helps us fix it and (2) one of these messages bounces, but because the bounced message has the specific user's address in it (which our normal messages lack because they are sent to the whole subscription list), we are now able to determine which address is invalid. This technique is analogous to techniques used in model-based troubleshooting where a new and maximally informative test is generated.

## Application Payoff

This application is not a commercial venture, so payback in monetary terms is not a relevant metric for evaluation. BMES was created as a support tool within a collaboration between a research group at MIT and a line organization in the White House Office of Media Affairs. Each partner in this collaboration had its own goals: The participants from the Executive Office of the President wanted to make information routinely and reliably available to the public and demonstrate the viability of the Internet as a model for the future National Information Infrastructure. The research group at MIT wanted to explore issues in computer-supported collaborative work and intelligent management of information. For both groups, management of the bounced-mail problem is a necessary supportive task but one that cannot be allowed to consume valuable resources; in particular, neither group has substantial personnel to devote to the task. Therefore, the relevant metric for evaluating the payback of the investment is in terms of the reduction of personnel contributions from the two groups, which, in turn, directly translates into the effectiveness of the system at handling bounced-mail messages.

We have been collecting data on the effectiveness of BMES since early in its lifetime. Figure 2 shows these data for the bulk of calendar year 1995. During this period, 63,091 bounced-mail messages were received. BMES was capable of automatically processing 48,031 of these messages, or 76 percent of the total. As can be seen from table 1, there is a great deal of temporal variability in the system's performance. It simply seems to be the case that some weeks we run into problems with sites whose mail servers provide less information; these weeks have lower overall performance. However, it is also noticeable that there is a long-term trend of improvement in the system's performance, which is probably the result of a combination of two factors: (1) Over time, we have confronted most of the mailer types that exist and have built up useful heuristics for dealing with them. (2) Over time, there has probably been a stabilization of technology in the community and a switch to more robust and informative mailer software.

Over the whole lifetime of the project, the time each day put into bounced-mail handling has declined from nearly 3 hours a day in calendar year 1993 to about 30 minutes a day now. We would certainly like to drive this number down further, but the transfor-

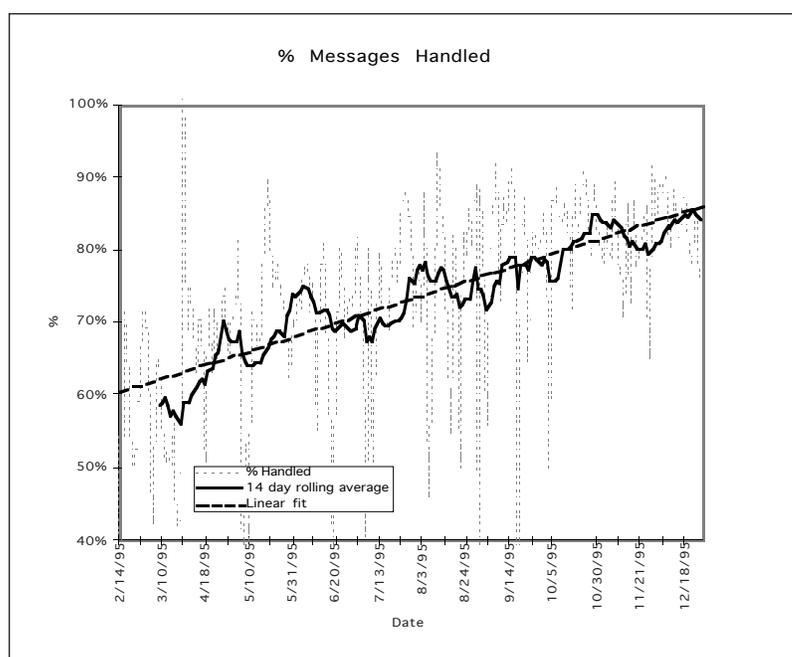


Figure 2. Effectiveness of BMES versus Time.

mation to date has been a qualitative one: The three hours a day required at the start was simply not viable; today, the task is annoying but well within scope.

## Implementation

Both COMLINK and BMES are implemented within the Symbolics GENERA environment, which runs both on Symbolics hardware and Digital Equipment Corporation ALPHA AXP workstations (using the OPEN GENERA emulator software from Symbolics). BMES is integrated with GENERA's ZMAIL<sup>4</sup> mail client, which is built on an extensible substrate for complex mail-handling applications. Much of the system relies on this substrate for low-level processing such as mail file and header parsing, pattern matching, and string searching. BMES itself is implemented in JOSHUA (Rowley et al. 1987) and makes extensive use of its protocol of inference to reason about the contents of the mail messages. BMES itself is invoked as a ZMAIL command that is applied to the mail file containing the bounced-mail messages. When mail messages need to be sent to postmasters or users at remote sites, this process is facilitated by use of ZMAIL's programmatic interface. Table 2 shows the component files in the system, including number of characters and lines of source text and number of definitions (rules, Lisp functions, methods, and so on).

Table 2. Code Distribution in BMES.

File name	Chars	Lines	Defs
New-db-interface	2,180	69	10
User-rules	10,662	292	21
Zwei-msg	2,798	79	12
Understanding-bounced-mail	41,067	1,106	104
Zmail-commands	25,450	655	20
Mailer-vanilla-unix	10,308	256	11
Mailer-smailer	3,820	101	2
Mailer-compuserve	4,090	95	2
Mailer-mime	6,212	148	9
Mailer-mmdf	7,482	176	14
Mailer-pmdf	4,789	125	4
Mailer-mime-pmdf	5,810	163	6
Mailer-uucp	3,603	89	1
Mailer-uucp-warning	2,814	70	1
Mailer-ibm	3,352	86	3
Mailer-vines	3,130	79	2
Mailer-microsoft	4,894	124	4
Mailer-minos	2,724	71	3
Mailer-local-delivery-agent	5,242	127	6
Mailer-undeliverable	3,139	77	3
Mailer-cc	2,514	63	3
Mailer-aol	3,497	93	6
Mailer-lisp	4,825	122	4
Mailer-mercury	3,266	84	3
Mailer-ctstateu	3,324	80	2
Mailer-smtp	4,049	104	4
Mailer-ksgbbs	3,405	86	4
Bounced-mail-complaint-reply	1,882	51	5
Check-recipient	2,541	61	3
Simple-redirection	5,491	140	5
Relay-zmail-command	25,987	702	65
Total	214,347	5,574	342

## Deployment and Maintenance History

Work on BMES was begun in spring 1993 as an adjunct to a predecessor system to COMLINK (called FORUM) that represented the first collaboration between the MIT AI Lab and the White House Office of Media Affairs. The bulk of BMES was completed by summer 1993. As COMLINK's development proceeded, a second version of BMES was developed by modifying the first version to take advantage of the extra information maintained by COMLINK. For a few months, COMLINK and FORUM were run in parallel, and users were encouraged to

switch their accounts over. During this period, both versions of BMES were run to manage problems from the two streams. The final cutover to COMLINK was completed in early 1995. Since this time, new features have been added to BMES as necessary.

It is interesting to note that BMES was literally developed and deployed simultaneously; it was an experience in the evolutionary design of a complex software system. As soon as there was useful function, it was deployed and then enhanced during its ongoing operation.

BMES is an unusual application: It is a component of the COMLINK system, which supports thousands of users, but there is only one user

of BMES itself. This user is also the developer and maintainer. Currently, the bounced-mail processing is done at MIT; however, we anticipate complete hands off of the COMLINK system in the near future, at which time, personnel in the Executive Office of the President will assume responsibility. As with much else about this application, a crisp definition of deployment is not easy. A large population has received information from the White House for several years now, and the management of e-mail delivery problems has been automated substantially as part of the task. It is true that the system is still operated by its developers, but this situation was anticipated at the outset. Routine sustainable operation has been achieved, which has enabled other aspects of the project to proceed without undue drain on scarce personnel resources.

## Future Work

Although BMES greatly reduces the effort required to process the mail backwash from a bulk electronic-mail distribution, there is room for improvement. The addition of some form of reverse mapping of MX records would help to identify an address on the distribution list based on an address as determined from a bounced message. The domain-name system does not provide such a mapping, so one would have to be constructed by iterating over all mail sites in the distribution database and doing a domain MX lookup for each one. Because of changes to the distribution database and the DNS (Internet domain-name system [Mockapetris 1987]), this reverse mapping would need to be updated regularly.

As it is currently implemented, BMES is difficult to extend as new mailer types are discovered and existing ones change because the work of identifying mailer type is distributed over a number of ad hoc parsers. As one adds a parser to recognize a new mailer type, one must be careful that this parser does not also recognize the messages of previously implemented mailer types. Perhaps reimplementing the parsers using a rule-based parser generator would simplify the definition of mailer types.

The ideal solution to the problem of handling bounced mail would be the universal adoption of standards that specify how mail delivery status information is reported. If delivery-failure notifications explicitly stated the reason for failure and the failing address, as well as any addresses from which it might have been derived, then BMES could be

replaced by a much simpler tool. Only one simple parser would be needed to extract the information from the bounced message. The system would require fewer, simpler rules for identifying the problem subscription. Recognizing the problem of numerous bounced-mail formats, the Network Working Group of the Internet Engineering Task Force has recently proposed a set of standards (Moore 1996a, 1996b; Vaudreuil 1996a, 1996b) that specify how mailers should report delivery status. As sites upgrade their mailers to ones that adhere to these standards, fewer and fewer bounced messages will require a system such as BMES to interpret.

## Notes

1. *Spamming* is a colloquial term, now common in discussions about the Internet, that refers to the practice of filling up somebody's electronic mailbox with unwanted material, often advertisements, complaints, or flames. The origin is unknown.
2. Here, we use the term *protocol* in the same sense as in the metaobject protocol (MOP) or the Joshua protocol of inference (Rowley et al. 1987), not in the sense of an Internet protocol such as SMTP (Crocker 1982). Fortunately, the object model used here doesn't use the message-passing metaphor, or we would also have confusion between mail messages and messages being sent to objects.
3. MX records are part of the Internet domain name system; the MX record for a host specifies which machine should actually receive mail addressed to the original host.
4. There are several other products named ZMAIL, which are not related to the one included in the MIT Lisp Machine software systems and its commercial offshoots such as Symbolics' GENERA.

## References

- Bobrow, D. G.; DeMichiel, L. G.; Gabriel, R. P.; Keene, S. E.; Kiczales, G.; and Moon, D. A. 1988. Common Lisp Object System Specification. *Sigplan Notices* (Special Issue) 23.
- Borenstein, N., and Freed, N. 1992. MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies. Internet RFC 1341.
- Crocker, D. H. 1982. Standard for the Format of ARPA Internet Text Messages. Internet RFC 822.
- Hurwitz, R., and Mallery, J. C. 1995. The Open Meeting: A Web-Based System for Conferencing and Collaboration. In Proceedings of the Fourth International Conference on the World-Wide Web. Boston: MIT Press.
- Kiczales, G.; des Rivières, J.; and Bobrow, D. G. 1991. *The Art of the Metaobject Protocol*. Cambridge, Mass.: MIT Press.
- Mockapetris, P. 1987. Domain Names—Implemen-

# MSL-96 : Proceedings, Third International Conference on Multistrategy Learning

*Edited by Ryszard S. Michalski and Janusz Wnek*

The theme of the workshop, multistrategy learning, concerns theoretical and empirical issues in the development of learning systems that employ multiple inferential and/or computational strategies. The study of such systems draws on the achievements in all other research subareas of machine learning and constitutes a major new research challenge for this field. Because humans are multistrategy learners, multistrategy learning has a natural connection to cognitive studies of learning and provides an excellent opportunity for cross-fertilization of these two areas. Because of their versatility and the ability to integrate complementary strategies, multistrategy learning systems have a potential for solving more complex learning problems than monostrategy systems, which have so far been the main focus of machine-learning research. Multistrategy learning workshops serve as a forum for presenting and discussing research progress in this area.

Papers in this volume present a sample of the recent research on multistrategy learning conducted at major research laboratories in Australia, Austria, Belgium, France, Germany, Italy, Japan, New Zealand, Poland, and the United States. Major topics of the workshop include the study of interrelationships among learning strategies and paradigms, cognitive models of learning and their relationships to methods and paradigms of machine learning, and the development of multistrategy learning systems and their practical applications. The papers have been grouped into four categories, according to their primary themes: 1) theoretical issues, 2) cognitive models, 3) methods and systems, and 4) special topics and applications.

ISBN 0-1-57735-010-3, 348 pp., index. \$40.00 softcover

**The AAI Press • 445 Burgess Drive • Menlo Park, CA 94025-3442 USA**

(415) 328-3123 <http://www.aaai.org>

tation and Specification. Internet RFC 1035.

Moore, J. K. 1996a. An Extensible Message Format for Delivery Status Notifications. Internet RFC1894.

Moore, K. 1996b. SMTP Service Extension for Delivery Status Notifications. Internet RFC1891.

Partridge, C. 1986. Mail Routing and the Domain System. Internet RFC 974.

Postel, J. B. 1982. Simple Mail Transfer Protocol. Internet RFC 821.

Rowley, S.; Shrobe, H.; Cassels, R.; and Hamscher, W. 1987. JOSHUA: Uniform Access to Heterogeneous Knowledge Structures (or Why Josting Is Better than Conniving or Planning). In Proceedings of the Sixth National Conference on Artificial Intelligence, 48-52. Menlo Park, Calif.: AAI.

Symbolics. 1993. Editing and Mail Manual, Symbolics Inc., Woburn, Massachusetts.

Symbolics. 1993. Genera Concepts, Symbolics Inc., Woburn, Massachusetts.

Vaudreuil, G. 1996a. Enhanced Mail System Status Codes. Internet RFC1893.

Vaudreuil, G. 1996b. The Multipart-Report Content Type for the Reporting of Mail System Administrative Messages. Internet RFC1892.

Westine, A., and Postel, J. 1991. Problems with the Maintenance of Large Mailing Lists. Internet RFC 1211.

**Mark Nahabedian** is a member of the research staff at the Massachusetts Institute of Technology's (MIT) Artificial Intelligence Laboratory. He received a B.S. in computer science from MIT in 1982. He's been employed as a Lisp programmer in various capacities ever since, with experience ranging from the application level to Lisp Machine microcode.

**Howard Shrobe** is assistant director of Intelligent Systems and Software Technology and chief scientist of the Defense Advanced Research Projects Agency (DARPA) Information Technology Office. He directs programs in the areas of software engineering, information survivability, and AI. Shrobe is on loan to DARPA from the Massachusetts Institute of Technology (MIT), where he has been a principle research scientist in the Artificial Intelligence Laboratory since 1979. From 1982 to 1992, Shrobe split his time between MIT and Symbolics Inc., where he served as a technical director and vice-president of technology. Shrobe received his M.S. (1975) and Ph.D. (1978) from the Artificial Intelligence Laboratory at MIT, where he was a cofounder of the Programmer's Apprentice Project, and his B.S. (1968) from Yale College.