Improving Problem-Oriented Mailing List Archives with MCS

Robert S. Brewer
Collaborative Software Development Laboratory
Information & Computer Sciences Department
University of Hawaii, Manoa
Honolulu, Hawaii 96822 USA
(808) 956-6920
rbrewer@hawaii.edu

ABSTRACT
Developers often use electronic mailing lists when seeking assistance with a particular language, API, or tool. The archives of these mailing lists provide a rich repository of highly specific problem-solving knowledge. Unfortunately, these archives are inconvenient to use by a developer seeking a quick answer to a problem because they are too voluminous, lack efficient searching mechanisms, and retain the structure of the original conversational threads which are rarely relevant to the knowledge seeker.

In this paper we present a system called MCS which improves mailing list archives through a process called condensation. Condensation involves several tasks: extracting only the messages of longer-term relevance, adding metadata to those messages to improve searching and browsing, and even editing the content of the messages when appropriate to clarify or provide context. The condensation process is performed by a human editor (assisted by a tool), rather than an artificial intelligence (AI) system.

The design and implementation of MCS are described, as are related systems. Experiences with a test database are discussed, along with plans for a wide scale evaluation.

Keywords
Knowledge condensation, mailing lists, archives

1 INTRODUCTION

Modern software development is a complicated task. Over the course of a project a developer may use: a design tool, an editor, a compiler, a debugger, a regression test system, or a packaging tool. The developer may need to use third party libraries or application programming interfaces (APIs) from a variety of sources. On top of all these software development related tools, the developer may be responsible for the installation and maintenance of their computing environment: operating system, hardware drivers, word processing, electronic mail, etc. In this kind of complicated technical environment, problems and questions arise inevitably.

In the case of commercial products, developers can seek out technical support from the publisher. However, commercial support is not always the best venue for a variety of reasons: the cost may be high, the publisher may have a vested interest in not providing certain information (like defect reports), or the level of support desired might simply be unavailable from the publisher. In the case of Open Source products [10], there often is no central source for support.

For these reasons, electronic mailing lists have become a common means for users to exchange information and help each other to solve problems. They can be administered by the producer of the product, but they are often run by a user of the product who wants to create a community for the product's users. These mailing lists often become an essential information source for the product, providing up-to-the-minute information and wise advice from experienced users.

As useful as mailing lists are, they have problems that limit their usefulness. A popular mailing list can have tens or hundreds of new messages daily, but keeping up with that level of traffic is prohibitive for most subscribers. While there is a lot of valuable knowledge available, it can be buried among a seemingly endless stream of beginner questions, off-topic discussions, and sometimes unsolicited advertising. The amount of traffic leads many subscribers to delete or file away messages from the list without reading them simply due to time constraints. When a subscriber has a question, they frequently send it to the list blindly, without knowing whether the answer was just posted recently. This further adds to the information glut.

Luckily for developers, there is another way to find solutions to problems: the mailing list archives. Most mailing lists maintain an archive of all the messages sent to the list, and usually provide some searching capability.

With the rise of the browser, most archives are made available via a web page with a search form, such as the Sun archive of the the "jserv-interest" list [12] (for the discussion of Sun's Java Web Server). These searchable archives provide a way for developers with problems to see if their problem has already been discussed and possibly even solved already.

Unfortunately, mailing list archives are poorly equipped to support this kind of query. All the irrelevant information that was sent to the list is immortalized in the archive, making it difficult to find the useful information. Searchable archives also face the problem that any particular query may return an enormous number of hits. For example, a developer looking for help on how to redirect a web client to different web page with the Java Web Server might do a search for "redirect" on the jserv-interest mailing list archive [12]. As of this writing, that search returns 175 articles, many of which are irrelevant to the developer's goal. The search hits are displayed in chronological order, and this arbitrary ordering doesn't help the developers find the solutions they are looking for. Another problem with conventional archives is that a particular question may have been asked and answered many times with varying levels of accuracy and clarity. A developer might find a message proposing a solution only to miss the follow up message which explains how that solution is flawed.

We have developed a method for improving the archives of these kinds of problem-oriented mailing lists which we call 'condensation'. Condensation involves several tasks: extracting only the messages of longer-term relevance, adding metadata to those messages to improve searching and browsing, and even editing the content of the messages when appropriate to clarify or provide context. The condensation process is performed by a human editor (assisted by a tool), rather than an AI system.

2 CONDENSATION AS A SOLUTION

The goal of condensation is to take the voluminous data stream generated by a mailing list and extract the information which would be useful to future users of the archive. As an analogy, newspapers provide a daily report on current events, but their analysis of which events are accurate or relevant are limited in that newspapers have short deadlines, a broad subscriber base, and other considerations. A story published one day might be amended or retracted the next depending on how events unfold. However, a book describing world events will tend to have a longer deadline which permits more reflection and analysis: a hoax which might occupy weeks of headlines in a newspaper will probably be little more than a footnote in a book (unless the book is about newspaper hoaxes). It is this refinement of information with a long-term perspective that we refer to as condensation.

There are a variety of ways that the information could be condensed, depending on the intended use of the archive. Our goal is to provide a searchable archive of information that allows developers to quickly find solutions to specific problems. Since the goal is to help developers find solutions as efficiently as possible, there is little point in preserving the threaded nature of the mailing list conversation. Condensation requires omitting unimportant, contradictory, or inaccurate messages, removing unimportant portions of messages, inserting new text into messages when required for clarification, and adding new messages to the database from scratch when that is the best way to explain something. For this narrow focus on problem solving, we can categorize each message as either a problem or a solution. Each message is annotated with keywords which are actually relevant to the subject of the message. The result of this process is a condensed archive, an archive which does not suffer from the problems of an unabridged searchable one. Since only truly useful information will be put into the archive, the amount of data to be searched is smaller which improves the odds of a search being accurate. Developers also benefit by having a set of standardized keywords which insure that messages using different terms but discussing the same topic will be retrieved by a single search.

3 MCS: A SYSTEM FOR CONDENSATION

To demonstrate the improvements possible through condensation, we have constructed a new software system for condensing mailing list archives. We have named this system (for lack of imagination) the Mailing list Condensation System or MCS [9]. MCS has two main parts: one which is dedicated to taking the raw material from the mailing list and condensing it, and another which stores the condensed messages and allows developers to access them.

One way to perform the condensation would be to implement an AI system that reads the messages and then decides what information keep, what to throw away, and what keywords to assign to each. In order to perform this task adequately, the system would need superb natural language processing capabilities and an in-depth knowledge of the mailing list domain. Such a system is currently at or beyond the state of the art, and would at any rate require a substantial investment of resources to complete.

A practical alternative to an AI system is the employment of human editors for condensation, along with extensive tool support to lower editing overhead to an acceptable level. Humans are quite good at examining textual information and determining what is useful and what is not, while computers are good at queries across

structured data [3]. It is also far more resource efficient because most mailing lists already have a set of 'gurus': subscribers who read all messages sent to the list and who are domain experts. Therefore, in MCS, humans do the editing using the MCS editing program which makes the process as efficient as possible. Only the editors need to use the editing portion; the interface of the end-user portion is simpler and geared towards ease of use.

Requirements

MCS was designed to help users of problem-solving mailing lists by improving the usability of their archives. Making archives more useful not only helps the archive users, it also helps to improve the quality of the mailing list itself because people are less likely to re-request information which is easily available via the archive. To achieve this goal, MCS must be adopted by the user community in preference to the many existing systems for generating and maintaining searchable mailing list archives. To encourage users to adopt the system, the design of MCS takes into account two issues: an explicit domain focus, and the existing list community.

MCS does not attempt to be a general purpose mailing list archive enhancement tool. It only attempts to improve those mailing lists which exist primarily as a forum for resolving problems. Because these lists focus on problem solving, naturally the users of their archives are focused on problem solving. For this reason, MCS-condensed archives only contain messages describing problems or solutions. This means that MCS does not attempt to be a summary of all messages sent to the list since any messages which don't relate to problems or solutions will be discarded. If a user wants a comprehensive archive of all messages sent to the list, he or she should use an existing archive that serves that function. The existence of other "unabridged" archives frees MCS to eliminate any messages or parts of messages that are not worth archiving.

Because MCS receives its input from a mailing list, it is crucial that MCS be designed with the social structure of a user-supported mailing list in mind. Specifically, the mailing list and its community should not be adversely affected by MCS. Any attempt to impose restrictions on how people read or participate in the list (like requiring users to use special software or compose messages in a certain format) would be met with blistering criticism. MCS must stand apart from the mailing list itself, resigned to using messages from the list on an as-is basis. MCS also takes into account the needs of the user community by having very low requirements for the use of the archive. The archive is accessed using a web browser which is presumably standard equipment for most mailing list participants. Furthermore, the web pages themselves are simple; they contain no images, no Java applets (by default), and no JavaScript to insure that even older browsers can be used. The omission and editing of messages is central to MCS, but those actions can reasonably arouse suspicion among list members as to the fairness of editing. To assuage these fears and to assure context, MCS provides a link from each edited message to the original message maintained in a separate unabridged archive.

Functionality

In addition to condensation MCS provides several unique features which facilitate user searching.

Keywords

Messages in MCS are assigned keywords by the editor. The keywords are chosen sparingly such that there are only a few for each message, instead of indexing all the words in each message. These keywords are organized into a hierarchy of categories by the editor. Each keyword and category can be annotated by a description and an URL when appropriate. The maintenance of the keyword hierarchy is a major part of the editor's task. Figure 1 shows one such keyword hierarchy displayed in the editor tool. All screenshots were taken from an MCS condensed archive of the ICEMail (a Java email client) mailing list, see section 6 for more information. While maintaining keywords is time consuming, having the keywords organized in this way provides advantages to the end user. A frequent problem when using conventional archives is figuring out what keyword has been used for a particular concept. For example, "freeze", "hang", and "lock-up" are all words which describe the same concept, but a user might have to try all three in order to retrieve all the problems related to that concept. With the keyword hierarchy, the synonym problem is all but eliminated. Having a relatively small number of keywords arranged in a tree also allows users to browse through the keywords and learn what kinds of topics are contained in the archive. Keywords can be browsed using a system similar to the one used at the Yahoo web portal [15], they can be selected using a Java applet, or they can be typed in directly.

The grouping of keywords into categories enables another unique option for users. MCS allows users to perform a *table-based search* by performing simultaneous searches for pairs of keywords. The user selects two categories which contain keywords, and then initiates the table-based search. MCS performs the cross-product of the two categories, and for each tuple of keywords it performs an AND search of the database. The result is a table which shows the coincidence of the keywords in the two categories. Figure 2 shows the results of a table-based search with the categories of "Actions" versus "Providers". The archive this search was performed on has a limited amount of data, but the results give an idea as to which protocol providers have proven prob-

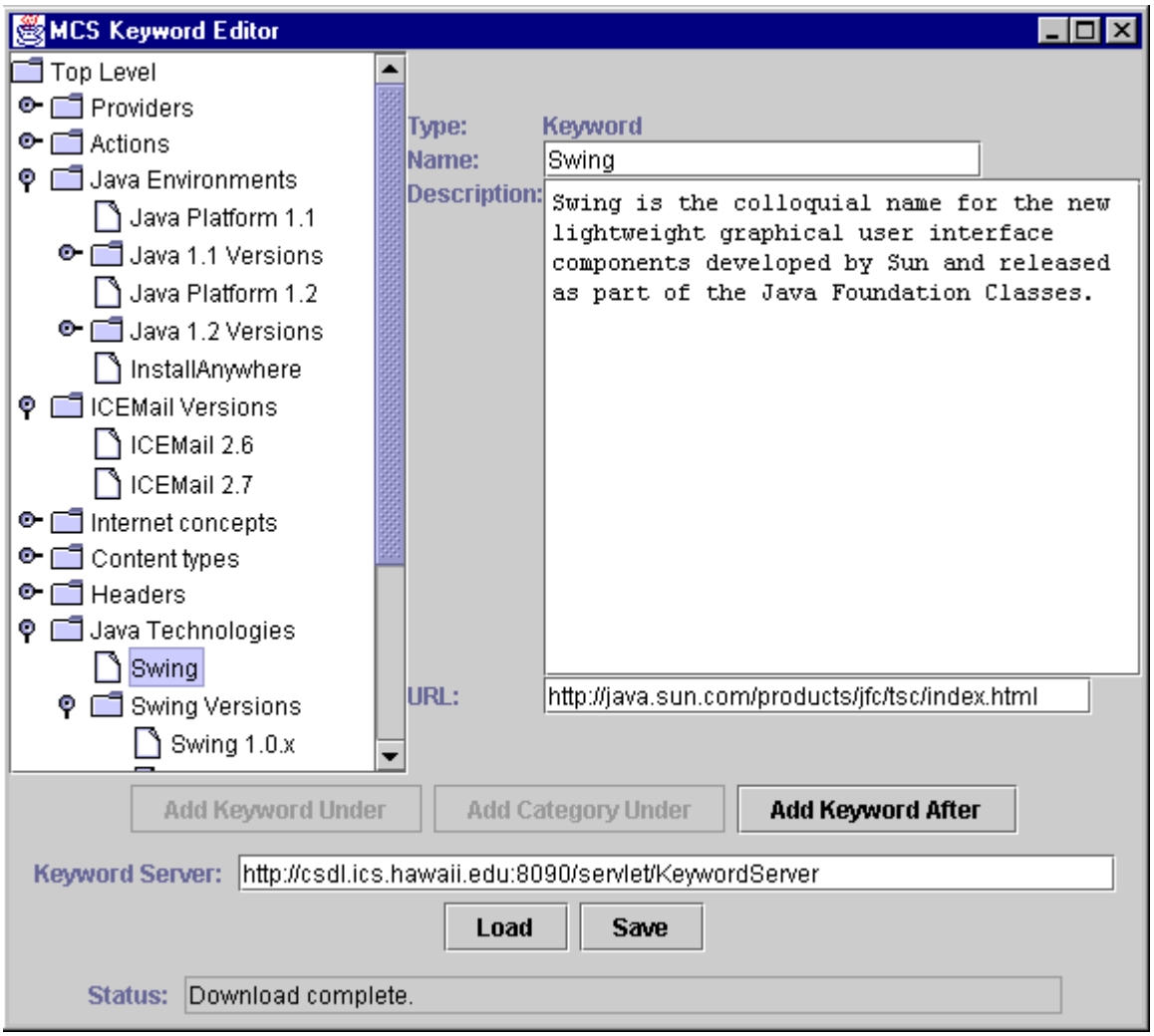

Figure 1: The tool used to edit the keyword hierarchy

lematic with which user actions. As the database grows larger it will allow users to examine trends and perform ad hoc comparisons.

Message Types
MCS has a very simple schema for the messages it stores. Each message has a type which currently can be either "problem" or "solution". This typing has a profound affect on MCS. When a developer has a problem, they can search the archive to see if they can find a message describing a similar problem. Once they find a relevant problem in the archive, they can immediately see what solutions have been proposed for that problem. Figure 3 shows the results of a search for the keyword "Swing". Since MCS understands that problems and solutions are associated with one another, it can group together the search results so developers can see related problems and solutions at a glance. The typing of messages also makes the editor's job easier because they only have to worry about messages which are problems or solutions, all other kinds of messages can be discarded.

Searching by Symptom
MCS also contributes the capability to search for problems by symptom. It is common for a developer to be aware of the symptoms of their problem, but unaware as to what the cause might be. While editing messages, the editor may notice that a problem message contains within it a textual pattern which is symptomatic of the problem. The symptom is usually an error message of some sort. The editor can convert the symptom into a regular expression thereby stripping out all the irrelevant parts of the symptom. This regular expression symptom is stored with the message. Later, when a developer encounters a similar problem, they can copy and paste the error message directly into the Symptom field on an MCS web page and initiate a search. MCS will then attempt to match the given text against all the symptom expressions in the archive, displaying the results to the user.

For example, suppose the editor noticed this error text in a message being edited:

```
java.lang.NoClassDefFoundError:
 com/sun/java/swing/tree/TreeCellRenderer
   at com.ice.mail.ICEMail.instanceMain(Compiled Code)
   at com.ice.mail.ICEMail.main(Compiled Code)
```

From his or her knowledge of the domain, the editor knows that this is symptomatic of using the wrong version of the Java Swing class library. So in this case, the relevant portion of this error in regular expression form would be:

```
java\.lang\.NoClassDefFoundError: com/sun/java/swing/
```

This symptom gets at the core of the error because the only two important parts are the type of the error and initial prefix of the mismatched package. If a developer later pasted in the following different error message, MCS will still be able to match it to the correct problem:

```
java.lang.NoClassDefFoundError:
 com/sun/java/swing/text/JTextArea
  at com.ice.mail.MessagePanel.getMsgViewer(Compiled Code)
  at com.ice.mail.MainPanel.getMessagePanel(Compiled Code)
  at com.ice.mail.MainFrame.getMainPanel(Compiled Code)
  at com.ice.mail.ICEMail.instanceMain(Compiled Code)
  at com.ice.mail.ICEMail.main(Compiled Code)
```

4 RELATED WORK

There are a variety of systems and research related to maintaining and searching collective memory. Here we examine several such systems and compare them to MCS. Some of these systems are somewhat informal (like moderated mailing lists and FAQ files), and some are formal research projects. The informal systems are based on the author's knowledge of those systems and generally do not have references because they evolved from common Internet practices.

Moderated Mailing Lists
Some mailing lists address the signal to noise problem by having a moderator or a group of moderators. All submissions to the list are forwarded to the moderator(s) who read the messages and decide whether or not to distribute them to the list. On most lists the moderator(s) do not edit the messages submitted, they just choose whether or not to distribute the message. Also, to allay fears of censorship on the part of the subscribers, usually the criteria used to decide whether to distribute a message are rather liberal, e.g., the message is related to the topic of the mailing list and not an advertisement [11].

While moderation can be useful for maintaining a high signal to noise ratio, it suffers from several problems which MCS does not. Moderation requires a substantial commitment on the part of the moderator(s) to review submissions in a timely manner. Failure to do so halts all traffic on the mailing list and annoys subscribers who have come to expect the short turnaround time that digital media can provide. Moderators also tend to face continual concerns from subscribers as to whether they are moderating in a fair and consistent manner. Since the whole point of moderation is to prevent the distribution of inappropriate material, there is no way for a subscriber to tell whether or not submissions are actually being judged by the stated criteria or whether the moderator(s) are acting on a whim or out of spite. Finally, moderation only partially improves the archives of a mailing list. Moderation will reduce the size of the archive and improve the average quality

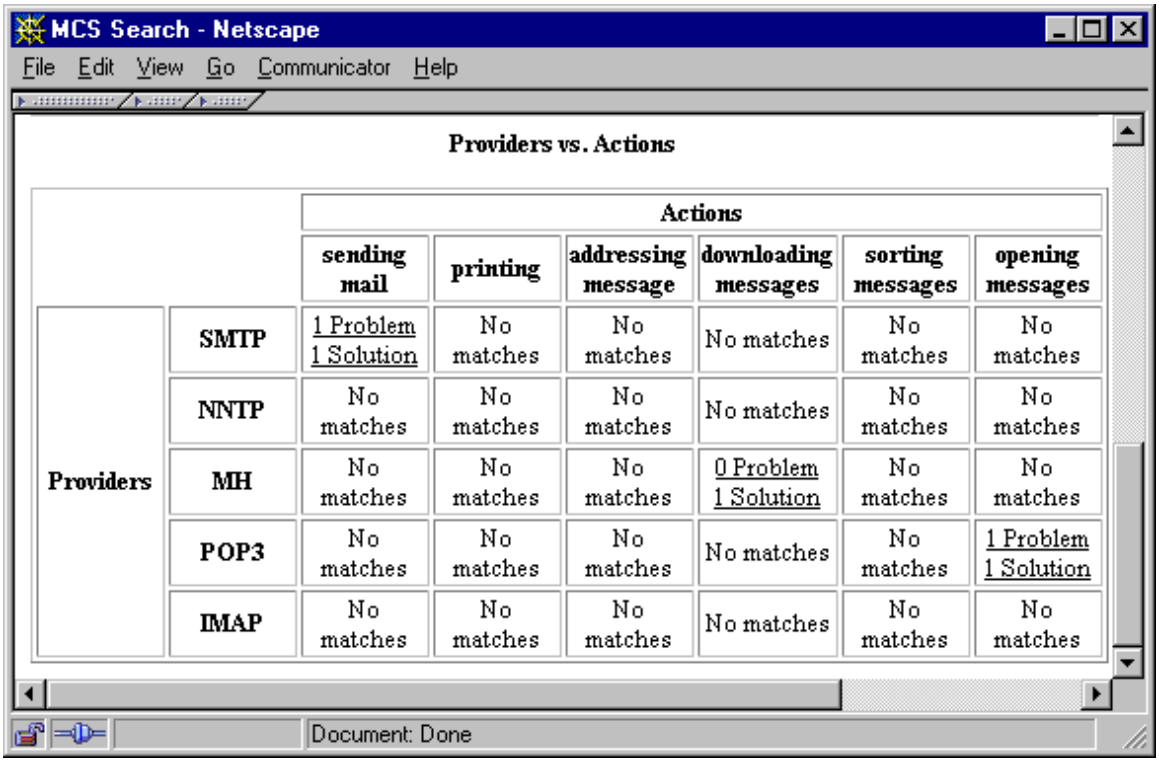

Figure 2: A table-based search of Actions vs. Providers

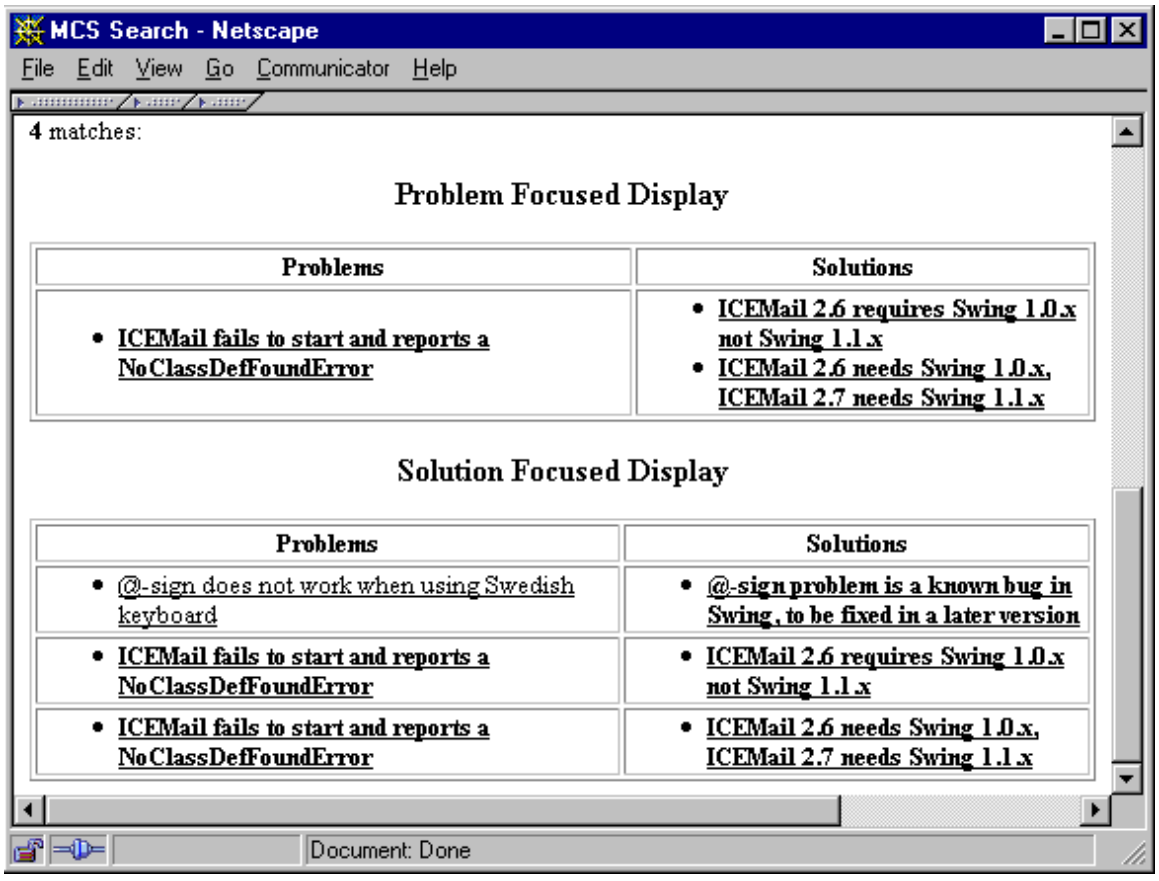

Figure 3: Results from a search for keyword "Swing"

of a message in the archive compared to the archives of non-moderated lists. However, moderation does not solve retrieval problems, and due to the time pressures faced by moderators they rarely have time to do more than a cursory check of submissions.

MCS reduces or eliminates all these problems with moderated mailing lists. One way of thinking about MCS is a form of moderation of the archives of a non-moderated list. Since MCS relates to the archive and not the list itself, the issue of timeliness is much less crucial: if you need to know what happened today on the list you should be reading the list itself, not the archive. Also, since MCS does not affect the list distribution itself at all, most concerns about censorship should be eliminated. MCS provides a link from each edited message to the original unabridged message so users can easily see what was edited out or changed in any particular message. A truly suspicious user could even compare the MCS-condensed archive to other unabridged archives of the list since MCS archives are designed to exist in parallel with traditional archives. Finally, an MCS editor can remove or rewrite parts of a message long after the message is sent to the list when necessary to make the message more useful which is not done in a traditional archive even of a moderated list.

Frequently-Asked Question Files

Most frequently-asked question (FAQ) documents attempt to provide a similar service to MCS: a condensed version of important and useful information that came from a mailing list or newsgroup. There are several important differences between the two systems. FAQ files are usually maintained without specific tool support so they require extensive effort on the part of the maintainer to create and update. FAQ files are generally created with the intention of easy distribution either as plain text or HTML. Because of this requirement, FAQ files are mostly limited in size to a few hundred kilobytes and they are laid-out to be easy for humans to read. Since FAQs cannot be of arbitrary size and complexity, they must omit useful information.

MCS does not have these limitations. Since it is not intended to be distributed by FTP or by posting to a mailing list or newsgroup it can be as large as is necessary. A sophisticated query system is an integral part of MCS, so it is not necessary that the underlying data be structured in an easily understandable human format. Because MCS lacks these two restrictions, it need not limit the archives it creates to merely "frequently-asked" topics, it can contain any information that would be useful regardless of how broad its appeal.

The Internet FAQ Consortium [8] maintains an index of many FAQs and has some outlines of a plan to write a book on FAQs.

FAQ Finder

FAQ FINDER allows users to quickly find answers to questions by searching a database made up of FAQ documents posted to Usenet [4]. The user enters his or her question into the system in natural language. First the system uses standard information retrieval techniques to determine which FAQs in the database are most likely to contain the answer to the question. It presents the top five FAQs to the user, who can select the most likely candidate. Then the system uses a combination of lexical and semantic similarity checks between the asked question and the question-answer pairs in the FAQ file. It then presents the 5 most likely pairs for user consideration. A live version of the system can be found at the University of Chicago web site [5].

While FAQ FINDER is an interesting system, it is attempting to solve a different problem than MCS. FAQ FINDER assumes that there exists a large number of FAQ files which are already organized in question-answer format, and from those files it attempts to help users find the answer to their questions. The designers of FAQ FINDER explicitly chose not to implement any domain-specific knowledge into their system because their intended dataset is a large number of unrelated FAQ files. MCS attempts to create a FAQ-like body of knowledge from a mailing list, and then present the condensed information in useful, possibly domain-specific ways. In this way MCS attempts to solve the problem of getting the information into an FAQ-like state, which is already presupposed in FAQ FINDER. Once the MCS archive is available, it might be possible to create a "stub" FAQ which FAQ FINDER could index, and if the user's question is a good match, FAQ FINDER would just send the user to the MCS-created archive.

Answer Garden

The Answer Garden system is designed to provide an "organically" growing database of answers to questions by end-users[1]. Users interact with the system by answering a series of diagnostic multiple-choice questions which lead them through the tree of answers already in the system. If users find that their questions are not answered in the database, they can enter their questions into the system and it will be forwarded to an appropriate expert via email. When the expert answers, the result is sent back to the original question-poser and also inserted into the tree for future retrieval.

Answer Garden's goal in life is to answer questions. Like MCS, it uses human input to decide what questions and answers should be in the database. However, Answer Garden is really only suited to the task of answering questions. A user who just wants to browse information either has to answer the diagnostic questions or guess where on the tree the information might be located. It

also requires a group of experts to be responsible for answering the questions posed by users. In an organization where certain people's job function is answering the questions of others in the organization, this works well because users get answers efficiently and experts don't have to answer the same questions over and over. However, the assumption that there is a pool of experts who are required to answer questions falls down in a volunteer user community where nobody is required to do anything. In MCS, experts can answer questions posed to the list at their whim; only the editor is required to work in order to keep the system functional.

Finally, the information in Answer Garden only grows as the system is used, while the information in MCS grows as long as there is useful traffic on the mailing list.

Answer Garden 2

Answer Garden 2 is a refinement of the Answer Garden system discussed above. It improves on Answer Garden by adding a system of gradual escalation for questions input into the system (thereby providing more context to the person answering the question), and a subsystem for collaboratively "refining" the information in the database [2]. All of this is built on a set of versatile and configurable components which allow the system to be tuned for a particular environment.

This system appears to implement many of the features required for our system. The system which inputs data into the system (CafeCK) provides a mechanism for capturing mailing list messages, and the "refining" system called Co-Refinery allows collecting, culling, organizing, and distilling information. The Co-Refinery system seems particularly close to MCS's requirements. Answer Garden 2 is not available for public distribution, so the actual implementation was not available for use as a foundation for MCS [Ackerman, personal communication].

Faq-O-Matic

This system was designed to allow a user community to create a dynamic WWW-based FAQ. Any user can browse through the web pages and make changes or additions as necessary. This provides an easy way to maintain an FAQ since any member of the community can volunteer help. However, there is no access control, so it relies on a cooperative user community. Since there is no centralized authority in charge of the FAQ, pieces of potentially incorrect or mutually conflicting information can be posted. Furthermore, new additions have to written from scratch by contributors. Documentation on Faq-O-Matic is provided through an FAQ maintained using Faq-O-Matic [6].

5 IMPLEMENTATION

MCS consists of two subsystems: the server side which stores the archive and provides a World-Wide Web interface to end-users, and the editing tool which allows editors to submit edited messages to the archive. MCS has been implemented entirely in Java which means both the server and editor can be run on a wide variety of platforms.

The server side consists of several Java servlets and support classes. Servlets are a way to extend the functionality of server, particularly web servers. The MCS servlets conform to version 2.0 of the Sun Servlet API, which means that they can be used with any of the many web servers which support servlets. The servlets are responsible for: storing the condensed messages in a simple flat file database, accepting user queries, presenting search results and messages to users. There are also servlets which interact with the editing tool to allow updates to the database.

The editor side makes use of the fact that the data source for condensation is email. The messages to be condensed are stored in normal mailbox folders on an email server. The Java email client ICEMail [7] has been extended for use as the MCS editing tool. Editors use ICEMail to contact the email server using IMAP (Internet Message Access Protocol), and use the standard ICEMail interface to read and delete messages from the list. When the editor encounters a message which needs to be condensed for inclusion in the archive, he or she accesses the MCS extensions to ICEMail which allow the editing and annotation to take place in a separate window. When the condensation of the message is complete, the editor can upload it to the MCS database with the press of a button.

End users of the MCS-condensed archive access it using their web browser. They go to a particular URL, and the MCS servlets dynamically produce the HTML which is rendered by the user's browser. The servlets produce only standard HTML to enable almost any browser to use the archive. An optional Java applet interface is also provided which allows users to browse through the keyword hierarchy and initiate searches more interactively.

6 EVALUATION

MCS is currently in the early stages of evaluation. Now that the implementation is complete, an initial evaluation is being performed with a small mailing list. The mailing list selected is the "icemail@gjt.org" list for the discussion of the Java email client called ICEMail. The list currently has 166 subscribers, has been in existence since September 1998, and has roughly 170 messages in its existing archive. It took 479 minutes to condense the existing 170 messages into 39 MCS messages. On average it took 2.8 minutes per message to edit.

The condensing of the existing archive has only recently been completed, so feedback from users is limited. This

initial test is designed primarily to get some real people using the system so obvious defects can be found and fixed before moving on to a larger and higher traffic list.

Future Evaluation
We plan to evaluate MCS on three criteria: adoption, preference, and editing effort. By adoption we mean determining whether a significant percentage (10%) of the list subscribers use the MCS-generated archive. By preference, we seek to determine whether those end-users who make use of the MCS-generated archive prefer it to the existing traditional archives. By editing effort we mean assessing whether or not list communities will be willing to expend the effort required to condense the archives.

Adoption
To measure the adoption percentage defined earlier, we need two pieces of information: the number of list subscribers and the number of users of the MCS archive. We intend to determine the number of subscribers with a query to the list maintainer (which should be readily provided since it does not give us any private information about subscribers). The other measure required to assess adoption is the number of people using the MCS archive. We will obtain an estimate for this number by surveys and analysis of log data from MCS. Note that the adoption percentage as we have defined it is an imperfect measure since we will not be able to positively determine whether the users of MCS are actually subscribers of the mailing list.

Since HTTP is a stateless protocol, it is difficult to extrapolate from the number of HTTP requests to the number of visitors. There are several rules of thumb used in industry which we will apply [13] [14]. We will avoid techniques that might be considered invasive (such as 'cookies') since the goal is maximal adoption. Online surveys can also provide a means for counting visitors.

Preference
Assessment of the users preference of MCS over traditional archives can only be easily determined in a qualitative way through user surveys. We expect that many list members will have attempted to use the traditional archives of the list since the URLs to the archives are often included in the footer which is appended to each message distributed to the list. Therefore questionnaire respondents will be able to compare the MCS archives with traditional ones. In addition to online surveys, some over-the-shoulder guided interviews will be performed to get more in-depth information from users.

Editing Effort
In this research the actual editing will be done by a researcher and not by a list member. Unlike traditional archives, an MCS archive requires human effort to maintain its usefulness. After the research is complete, the list community will need to take over the task of archive maintenance. List members not involved with the search (thereby lacking an important motivator) may find the time and effort required to be an editor is too great. To get a handle on editing effort, the editing tool will record how much time is spent in the tool and how much time is spent on each message. These metrics will allow estimation of how much time is required to condense an existing traditional archive and ongoing maintenance of the MCS archive. This will provide an indication of whether the use of MCS outside the realm of research is possible.

7 FUTURE WORK
MCS is only just now emerging from its shell, leaving a wealth of opportunities for evaluation and extension.

Editor Recruitment
The editor(s) obviously play(s) a crucial role in the operation of MCS. Without continual updating, the database becomes of only historical interest. For foreseeable future, the author will be acting as the sole editor. In order to ensure the continued survival of the condensed archives, it will be necessary to recruit other editors. If the archive is useful enough and the editing tool is easy to use, it should be possible to get volunteers from the list to step forward as editors.

Open Source Distribution
As part of the growing Open Source movement [10], we plan to release the MCS source code to the public under the GNU General Public License. In some sense this isn't that uncommon in the academic world, but we believe that easily downloadable source (and binaries for that matter) will encourage others to adopt the system for their mailing lists, and spur other researchers to build on the MCS framework.

Adoption by Other Mailing Lists
Convincing other mailing lists to use the software for their archives would be the final stage in moving the software out into general use. This adoption process may be more difficult because it requires the mailing list's community to embrace the system and it also requires recruitment of one or more editors from the mailing list. It might be necessary for me to target another mailing list to which I subscribe so that I could jump start the process by acting as editor for some period of time. Once a few mailing lists are using the system, word of mouth should attract other mailing lists to adopt the system.

Scalability Improvements
MCS was designed as a research system, and as such, many decisions were made in favor of speed and ease of implementation. However, if MCS grows to serve large archives, work will be required on its scalability. MCS does not use a backend database, so many operations

will slow down when there are many users and/or many condensed messages. MCS also assumes that there is only a single editor for an archive, which obviously does not scale well to high-traffic lists where editors will want to share the daily workload.

Expansion into Technical Support Market
Mailing lists are used extensively both internally and externally by those who provide technical support. In this kind of environment, MCS could be used to do a sort of "data mining" on old email archives, turning them into valuable knowledge bases which can in turn reduce support costs. With the potential of lowered costs, it would make sense for corporations to support editors either within their company or even external editors.

ACKNOWLEDGMENTS

This research would not have been possible without the help of Philip Johnson, and all the members of the CSDL research group. Your encouragement is very much appreciated. Thanks also to Yuka Nagashima for her steadfast support throughout.

REFERENCES

[1] M. S. Ackerman and T. W. Malone. Answer garden: A tool for growing organizational memory. In *OIS90*, Filtering, Querying, and Navigating, pages 31–39. ACM Press, 1990.

[2] M. S. Ackerman and D. W. McDonald. Answer garden 2: Merging organizational memory with collaborative help. In *Proceedings of the ACM 1996 Conference on Computer Supported Work*, pages 97–105, New York, Nov.16–20 1996. ACM Press.

[3] F. P. Brooks, Jr. The computer scientist as toolsmith II. *Communications of the ACM*, 39(3):61–68, Mar. 1996.

[4] R. D. Burke, K. J. Hammond, V. A. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the FAQ finder system. Technical Report TR-97-05, Department of Computer Science, University of Chicago, June 20 1997. Mon, 23 Jun 1997 21:02:34 GMT.

[5] FAQ Finder web site. Available at <http://faqfinder.cs.uchicago.edu/>.

[6] Faq-O-Matic web site.

[7] ICEMail a Java based email client. Available at <http://www.ice.com/java/icemail/>.

[8] Internet FAQ consortium. Available at <http://www.faqs.org/>.

[9] MCS: Mailinglist Condensation System. Available at <http://csdl.ics.hawaii.edu/Research/MCS/MCS.html>.

[10] Open Source web site. Available at <http://www.opensource.org/>.

[11] R. C. Pedersen. Reviewing Internet mailing lists. *The Serials Librarian*, 30(2):27–33, 1996.

[12] Sun
Java Web Server interest list archives. Available at <http://archives.java.sun.com/archives/jserv-interest.html>.

[13] B. Winett. Tracking your visitors. Available at <http://www.hotwired.com/webmonkey/98/16/index2a.html>, 1998.

[14] B. Winett. Troubles with tracking. Available at <http://www.hotwired.com/webmonkey/98/25/index4a.html>, 1998.

[15] Yahoo web portal. Available at <http://www.yahoo.com/>.