

ToolBox: A Living Directory For Unix Tools Owned By the Community

Carlos Maltzahn and David Vollmar
Computer Science Department
University of Colorado at Boulder
Campus Box 430
Boulder, CO 80309-0430
{carlosm,vollmar}@cs.colorado.edu

Abstract. Members of a community who work primarily at computer terminals are deprived from distributed cognition as it occurs in many other domains where people work with tools in a shared physical space. This makes it harder for individuals to pick up cues towards useful tools and their utilizations. Traditional help systems were unsuccessful to fill in this gap of information flow. People prefer to consult other people in their community because either their tool knowledge is contextualized appropriately or they know other people who can help. In this paper we describe the design and preliminary evaluation of a system that is actively looking for new tools and is interviewing potential experts. The result of these interviews is presented by weekly newsletters and a hypertext system in World Wide Web format. We specifically address design issues of and experience with the interviewing process and the presentation of the resulting information.

1.0 Introduction

People share tools via networked computers. The design of these systems, however, tend to prevent people from sharing practical knowledge about how to use these tools. Our paper identifies the consequences of this situation and presents a fully functional prototype which supports the exchange of practical knowledge.

The following example illustrates how computers make practical use of tools invisible: Consider a mechanic shop with its abundance of tools hanging from perfboards, lying on work benches or on the floor. A good way to find out about the various purposes of these tools is to watch mechanics using these tools. Compare this situation with a modern multiple room office where people work at networked computers: Without taking a careful look on a person's screen one cannot see any tools. People are punching keys - no matter what tools they use and how they use them. Furthermore, screens are considered to be private areas. It is usually not socially acceptable to stare over somebody's shoulder. Thus, people who are working in a computer environment are less exposed to the practical use of tools.

The latter environment generally leaves three ways to acquire tool related information: (i) one can explore the on-line environment in the hope to stumble across a useful piece of

information, (ii) one can ask another person, or (iii) browse through an on-line or off-line manual. *Exploring* the on-line environment tends to be time consuming and unproductive [13]. Instead, users prefer to consult their colleagues or local experts [7] [9]. *Asking* another person has the advantage of a certain response [8], which contains either contextualized information, i.e., information that is grounded in the domain of the local community [2] [14] [16], or a pointer to another person who might know the answer [11]. The latter has also the advantage that it potentially extends the personal list of social contacts to experts. Asking another person involves, however, “bothering” people, i.e. interrupting their work. Furthermore, asking involves informal conversations. Beside their convenience and social importance their negative aspects are (i) their low effectiveness because only a few members of the community get the knowledge they need, and (ii) their fragility because the knowledge is coupled with the availability of experts [17]. A question also bears the danger of being interpreted as incompetence. Thus, one can view the social context of people helping each other as a “competence market” in which people are trading competence points for answers. This imposes social pressure especially on novices who usually don’t have much to offer but who try hard to become a competent and accepted member of the new community. People in this situation are therefore more likely to *browse* through anonymous sources of information such as manuals and on-line help systems in order to build up some own competence before entering the competence market. These, however, present information in a very general, de-contextualized way which makes it difficult to discover useful tools for a given situation. Thus, the search can be time consuming and unproductive [13]. This is especially true for keyword oriented systems like “man” in Unix¹ since appropriate keyword selection requires knowledge about the overall structure of the corresponding domain [10]. Also, for novices on-line help systems are often as difficult to discover as any other tool.

All of the above ways of information acquisition require (i) the initiative of the user to look for information, and (ii) an idea of where to look for information. Thus, the user can not serendipitously pick up useful information from his/her environment. In this paper we address this problem by presenting **ToolBox** which is a fully functional prototype that attempts to combine the advantages of asking somebody with the advantages of reading manuals or using on-line help systems. Our approach is to provide a community constructed directory for tools, i.e. tools are categorized and reviewed by the community itself. We call our approach a “living” directory because it evolves in a community by actively collecting information about the social and domain-specific context of tools, i.e. who is using which tool for what purpose. The directory is “owned” by the community since every community member can modify the system at convenience.

The idea of a community constructed directory is not new. Perhaps one of the most related systems is described in [8] since it also tries to integrate users into a help systems for Unix tools. However, the author assumes that users voluntarily contribute knowledge and does not address motivational issues, such as benefit of use and the overhead of adding new knowledge. Furthermore, in order to retrieve information the system requires the user to pose a query. However, it is often difficult to come up with effective queries without having sufficient knowledge about the domain structure [10].

1. Registered trademark

Answer Garden [1] relieves the user from asking questions by presenting a branching network of diagnostic questions from which the user can choose the best match to his/her problem. If no satisfactory match exists the user can press an “Unhappy” button and post the new question which is automatically routed to the appropriate expert. The system, however, needs to be explicitly configured in order to provide this routing service, i.e., experts are known a-priori. In the advent of new tools, Answer Garden does not support the discovery of new experts. The authors assume formally assigned experts, i.e., maintaining Answer Garden is part of their job, and do not consider any other motivations.

DYK (“Did You Know”) [12] allows any member of a community to add new pieces of information to a directory service. The design carefully accounts for motivational issues, i.e., it tries to be an attractive displacement activity by being user-invoked, by presenting small pieces of information and by being engaging. In particular, it provides feedback mechanisms for information providers, an issue that we have not yet addressed in the ToolBox design. To our knowledge, DYK structures its information through a pre-set, non-hierarchical set of categories. It thus does not offer much support for users who look for a particular piece of information.

A nice account of the difficulty of collaboratively structuring a group memory is given in [3] which describes the design and implementation of TeamInfo. The basic idea of TeamInfo is to offer a group version of the various private collections of email folders. The authors report on their experience with trying to collaboratively achieve a consensus on the structure of a group memory. They present a taxonomy of filing habits that illustrates the complexity of developing a group filing habit. We believe, however, that this complexity can be reduced in the following ways. First, consensus on group memory structures is easier to achieve in small, specialized domains, e.g. the domain of Unix tools (see also [17]). However, TeamInfo’s domain is defined as “information of persistent value for the group.” This is not only a large and general domain but also a continually changing one since it is based on the dynamic interests and tasks of community members. Second, the authors describe an attempt to reach an *explicit* consensus through dedicated meetings, i.e. people were asked to explicitly discuss and then agree on a structure. We believe that an explicit consensus is harder to achieve than an *implicit* one, i.e. where people start out with an ad-hoc structure and individually change it as needed. Thus, an implicit consensus with respect to a categorization structure is constituted by the fact that people are using it as a basis for communication without ever explicitly agreeing to it, i.e. the structure is never finished and perfect.

In the following section we will present the underlying ideas of the design of our prototype. We will then describe the various components and present some feedback we got from the members of the Computer Science Department of the University of Colorado at Boulder who were the experimental users of this directory service. The report is concluded with an outlook on further research on collaboratively “grown” information systems.

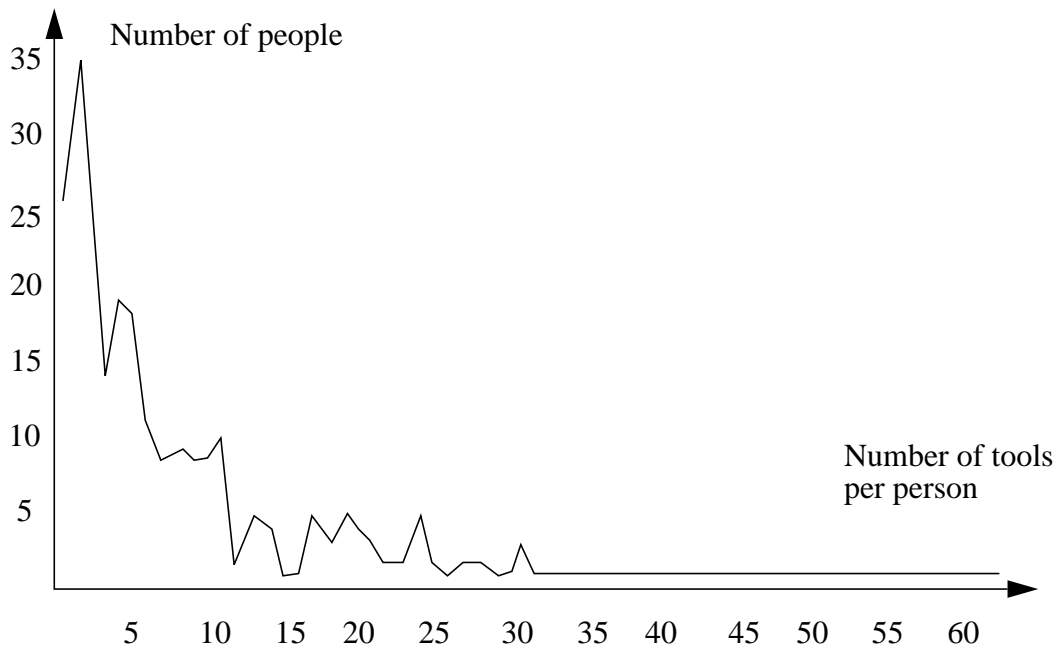
2.0 Design Issues

In this section we describe various design issues we considered particularly relevant for a community constructed directory. After analyzing our target community we identify the problem of motivating community construction. This in turn raises issues regarding the

presentation of the ToolBox browsing space which is discussed in the third part of this section. The last part addresses the problem of the visibility of ToolBox and its services.

2.1 Analysis Of The User Community

The target community of our prototype are the people at the Computer Science Department of the University of Colorado at Boulder who use Unix tools. Based on the number of Unix on-line manual entries and the fact that some of the tools do not have manual entries we roughly estimated the number of tools as 1000. Monitoring the process tables of seven main hosts for one month we came up with the following data: 235 users have used 389 tools. The number of tools each user used ranges from 1 to 86. Most users used two tools (35). The following figure shows the distribution. Although 389 different tools were



used only few people used more than 10 tools. This data seems to confirm studies which conclude that most people in a computer environment are experts for a small set of commands and novices for the rest of it (e.g. [4]). Since this kind of expertise is distributed over a large set of commands it is of great advantage to every member of the community to know local experts.

2.2 Motivating Community Construction

As a community constructed system ToolBox relies on the active participation of its user community. Thus the system design needs to ensure that each class of users within the target community benefits from its use [6]. In particular, ToolBox needs to offer benefits for the consumer of tool information as well as for the person who adds new information to

the system. Benefits can take many forms: the most trivial one being financial motivations offered by organizations. However, we are interested in benefits that are solely based on the services provided by the system.

In order to identify benefits we need to identify the roles and their goals of system users. In the case of ToolBox these are (i) the consumer, who wants to efficiently find relevant and up-to-date information, and (ii) the tool expert, who mostly wants to work undisturbed (i.e., we assume that the role of a tool expert is not officially sanctioned by the organization). The above user community analysis indicates that tool expertise is distributed, i.e., every member in the community can assume either role depending on the tool in question. In ToolBox we took advantage of this fact by combining the directory construction process with learning experiences. As a result we designed an *interview* during which ToolBox (the interviewer) asks the user (the interviewee) to interact with the representations of ToolBox such that (i) the user reveals information about a given set of tools and (ii) learns about structure and content of the ToolBox information space. However, the user explores this space in the context of the interview, i.e. his/her explorations are task-oriented. There is evidence that task-oriented explorations are more effective than task-free explorations [13]. Important for the acceptance of the interviewing process is how well it fits in everyday practice of the community. For example, users often engage in “displacement activities” [12]. In order to take the role of a displacement activity the interview (i) needs to be brief, (ii) can be aborted at any time and without cost, and (iii) offer some informal contact with the rest of the community.

Another way of increasing the benefit of construction is to reduce construction overhead. We chose strategies of active and intelligent distribution and coordination of the construction load among the appropriate members of the community. More specifically, we use a dispatching strategy that is a combination of optimistic expertise assignment and active soliciting. With *expertise assignment* we mean the process of assuming a user to be an expert with respect to a tool based on his/her tool usage history. An *optimistic* expertise assignment is an assignment based on usage patterns that not necessarily imply expertise. Under *active soliciting* we understand the system initiated process of inviting a potential expert for a construction session. Since our domain are on-line tools every tool user generates inevitably data in the computer network environment. For instance, a workstation which runs Unix maintains a table that list all non-terminated processes. Among other status information this process table also associates each process with the command and the user who initiated it. A subset of these processes represent the set of tools currently used by the community. Monitoring the process tables on all networked computers within the community can be used to discover new tools and their users. For each entry in the process table one can safely assume the following: (i) the tool that corresponds to the command exists, and (ii) the location of the tool is known to the user’s environment. However, it is also possible that the following is true: (iii) the user is able to classify the tool according to a given tool taxonomy, (iv) the user knows where to find documentation, and (v) the user knows something about the usage of the tool within the community’s domain. The assumption of the last three statements (iii-v) is an instance of optimistic expertise assignment. Note, that with this method the directory service will discover *every* program that gets used by at least one member of the user community. Hence, this strategy is particular useful for dynamically changing environments like Unix.

2.3 The Browsing Space

We call the information space of ToolBox *browsing space* since it is intended to offer a “road map” to tool users. The browsing space can be browsed and searched. We will now motivate the following requirements of the browsing space: (i) bridge the gap between the user’s informational needs and the existing tool documentation, (ii) allow for easy capture and easy recall, and (iii) let users modify the browsing space.

2.3.1 Filling the Gap

The current tool directory of our target community only offers information on a keyword basis (i.e., *man* and *apropos*). This is not helpful for users who do not know what kind of support an existing tool collection might provide. A browsing space provides the keywords that are needed to find a tool. Its structure serves as explicating semantic relationships between these keywords such that the user receives some guidance of how to reformulate the problem in order to find the right set of tools. Each location in the browsing space needs to be presented with its semantic context, i.e. the structure in which the current location is embedded, and that the user can easily change location without losing orientation.

Once the user has found a promising category, it is important that the choice can be validated by looking at the category’s list of subcategories and tools. In this situation the user needs to be able to quickly access the potential value of tools to his/her problem. Therefore, each tool should be associated with a one- or two-line description. It is also important that all this information is included into the keyword search space. For example, the user might want to search for tools that are semantically similar to a known tool, i.e., tools which share the category of the known tool or are located in an adjacent category.

In traditional tool manuals the user can be overwhelmed by the technical detail s/he is confronted with once s/he has found a potentially useful tool [13]. Often it turns out that -- after spending too much time in the reading of irrelevant details -- the tool is not appropriate at all. In our design of ToolBox we strived to avoid this situation by providing some kind of bulletin board for each tool. The task of this bulletin board is to post multiple *opinions* about for what the tool is typically used, from where to get good documentation, where it can be found, and who in the community might give helpful information. Thus, we do not want to replace documentation but we want to complement the existing documentation by providing important community specific information.

2.3.2 Costs of Capture and Recall

Our approach is to view the browsing space as a evolving communication forum of the community. It must be easy for users to add new information about tools (minimal capture costs) and retrieve relevant information about tools (minimal recall costs). In the following paragraphs we discuss different ways of structuring a browsing space, i.e., multidimensional spaces, directed graphs and trees.

A *multidimensional space* requires every location to be specified by a tuple of fixed length. In a multidimensional browsing space for tools every tool is categorized by its location, i.e. a fixed set of attributes. Thus, tools that are similar with respect to their

attributes are located close to each other. However, in order to place a tool in such a browsing space one has to decide upon values for all attributes. In order to accommodate all Unix tools the number of dimensions of the browsing space needs to be large, i.e., each tool would be represented by a large set of attributes. This requires looking at many attributes that would turn out to be irrelevant. Another problem is to represent such a multidimensional space in an effective manner. Furthermore, adding new dimensions to this space would have an effect on the classification of the other tools and on the presentation of the space.

Semantic relationships are often represented by *directed graphs* (e.g. [15]). However, with sufficient size of the network, orientation becomes increasingly difficult. More recent applications like Mosaic (a World Wide Web client) try to avoid disorientation by providing pointers to the location where one has started, by recording traces such that one can go back step by step, and by private “hot lists” which are unstructured lists of bookmarks. Many hypertext titles have adopted a hierarchical structure, i.e. each page contains pointers to a super page, root page, or to pages on the same “hierarchy level”. The popularity of this structure might be explained by the dominance of hierarchical structure in paper based media.

We therefore looked at *tree* structured browsing spaces. Compared to multidimensional spaces these have the advantage that each category defines its own set of “attributes”, i.e. its super- and subcategories, and that new attributes (i.e. subcategories) can be added without any global effects. Each tool can be characterized by just placing it into a category that resides in a context that makes only the relevant attributes explicit. However, almost all tree structured categorization systems include non-hierarchical cross references among semantically related categories.

To account for non-hierarchical semantic relationships we chose to relax the constraints of tree structures towards directed acyclic graphs, also called multi-hierarchies. This type of structure allows a category to have multiple super categories. Furthermore, we also allow tools to be placed in multiple categories. All the information associated with a tool other than categorization remains however the same.

2.3.3 User Modifiability

Community members can communicate their view on tools by modifying and extending the browsing space. As mentioned earlier, our conjecture is that community members will implicitly agree upon a browsing space as long as they can tailor it to their needs. In order to find out whether these conjectures are true we started out with a prototype that offers unrestricted modifiability to potential tool experts. Potential tool experts interact with the system during interviews. In the interview situation the user can reflect upon the adequacy of the hierarchy since s/he is confronted with the task to place a set of tools. Thus, while interacting with the interviewer the user can add new categories, move categories, and cut and paste tools. We do not allow users to delete categories since this would make potentially large amounts of information inaccessible. However users can move categories into a “trash” category from where they can be recovered if necessary.

2.4 Visibility

One common problem of directory services for tools is that they can only be accessed by running yet another tool. As with all tools, a user needs to know about its existence in order to use it. The dilemma can only be solved if there is some way to actively deliver such information to the user. One very effective delivery system is sending email to users. However, it would not be enough to just mention the existence of ToolBox. What we need instead is a message that catches a user's attention and makes the information instantly applicable. A good candidate for such a message is the kind of message ToolBox sends to potential tool experts in order to ask for an interview: (i) The message is personalized in that it enumerates a subset of the tools the user has used recently. (ii) The user is asked to run a command to start the interview. This is immediately applicable and promises to satisfy the user's curiosity, thereby encouraging displacement activity as mentioned earlier.

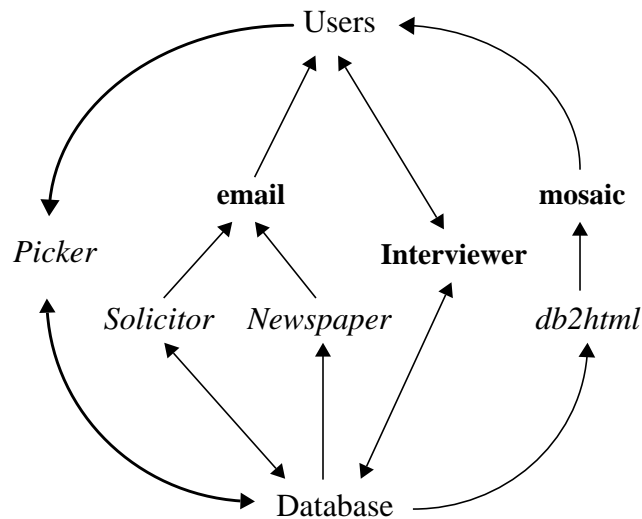
Thus, sending out interview invitations seems to be a promising way to achieve visibility of the directory service. The questions remain, (i) who in the community receives such emails and (ii) whether there is a part of the community that tends to be left out. In order to make sure that everybody finds out about the directory service we designed a service that sends out a weekly newsletter via email. The newsletter reports all tools where ToolBox recently managed to get an interview for the first time and includes the results of these interviews together with instructions on how to use the directory. This newsletter is sent to all members of the community.

3.0 The Components Of ToolBox

ToolBox consists of a *Picker* which picks up new tools, a *Solicitor* which solicits for interviews by periodically sending email messages to users, a *Newspaper* which periodically generates a newsletter, an *Interviewer* which is called by users who are willing to give interviews, and *db2html* which periodically generates a system of World Wide Web hypertext pages that represent the content of the database. The diagram illustrates the interactions of these components. Arrows represent directed information flow. Components in italic fonts are either constantly or periodically run by the system using the timer service of Unix (i.e., *cron*). Components in bold fonts are applications started by the user. The following example scenario should clarify the interactions of the components:

Tuesday: Paul writes an email filter application in order to better manage the daily load of email messages. He is testing his program with a filter application that comes with the elm package. Meanwhile Mary in another part of the building gets finally sick of using t-editor and tries f-editor for the first time which does not help much either. The *Picker*, constantly in the background picks up the filter application of Paul and the f-editor application of Mary because the tools are new to the ToolBox system. Both tool names with their users are stored in a relational database.

Wednesday: At 6am the Unix timer service starts the *Solicitor* which once a day sends email messages to the users of tools that were newly discovered during the past 24 hours. Each email message contains a short list of tools that were individually compiled for the corresponding user. The message also asks each user to call a tool called *Interviewer* that helps to classify each tool and to answer some questions about usage, documentation, and



advisor status. At 7am the timer service starts *db2html* which compiles the content of the databases into a system of mosaic pages.

In the morning Paul reads his email. One of his email messages has the subject line “ToolBox needs your help”. He reads the message and cuts and pastes the command that calls the *Interviewer*. The email message has already supplied the command with the right arguments. He executes the command and is confronted with a full screen ASCII interface that presents the hierarchy and a set of commands that allows him to navigate through the hierarchy, modify or extend it. He drops each tool of the list that was provided as arguments to the command into categories he thinks are appropriate and modifies or extends the hierarchy along the way. Whenever he drops a tool into a category the *Interviewer* asks him about usage, source of documentation and whether he wants to be a listed advisor, an anonymous advisor, or not an advisor at all. He can also indicate that he would prefer not to be solicited again.

Mary receives a similar message that allows her to let loose of her frustrations about the f-editor. However, while reading the email message she also found out about the ToolBox Mosaic page. She fires up Mosaic and visits the ToolBox page. Her goal is to find out about new editors. Since she already knows the t-editor she tries to use the Mosaic search function in order to get to the editor section of the browsing space faster. In the same section she discovers l-editor. She clicks on the tool name which takes her to the l-editor tool page. There she looks through a list of local comments and learns about the users in her community who use l-editor. In one of the statements someone mentions a newsgroup for l-editor. She then re-visits the ToolBox home page and clicks on “return to outline” which takes her to the editor category within the categorization structure. She finds related categories next to the editor category and she spends some time “grazing” interesting bits of information.

Monday: At 8am the Unix timer service starts the *Newspaper* which generates a new newsletter and sends it to the whole community. Among other things it contains the result of the interviews with Paul and Mary since they were the first ones interviewed about the filter application and the f-editor.

The next subsections describe the tools more in detail and the important issues related to them.

3.1 Picker

The Picker is a constantly running process that periodically scans the process tables of a specified set of networked computers. Its task is (i) to maintain the community database which relates to each tool the list of users who are using it, (ii) to discover tools that are either new or “forgotten”, and (iii) to collect users of unknown tools. To achieve (ii) and (iii) it maintains a database of known tools and of newly discovered tools with their users. Known tools are time stamped and can be forgotten. Forgetting is important since a tool is not really discovered as long as no tool expert has been successfully interviewed; users are free to ignore any ToolBox messages. If a specified period of time (e.g., half a week) passes and the tool has not received enough responses (e.g., at least two) the entry in known tools expires and the tool can be newly discovered, i.e. the Picker looks for new users of those forgotten tools.

3.2 Solicitor

The task of the Solicitor is to send email messages to those users that have used new tools. To minimize disturbance the Solicitor mails only one message per user and in order to keep interviews short it limits the maximum number of tools that a user is asked about to a specified constant (e.g., five). Tools that do not fit in mail messages are wait-listed for the next initiation of Solicitor (e.g., the next morning). As a basic principle, the Solicitor will never ask a user about the same tool twice.

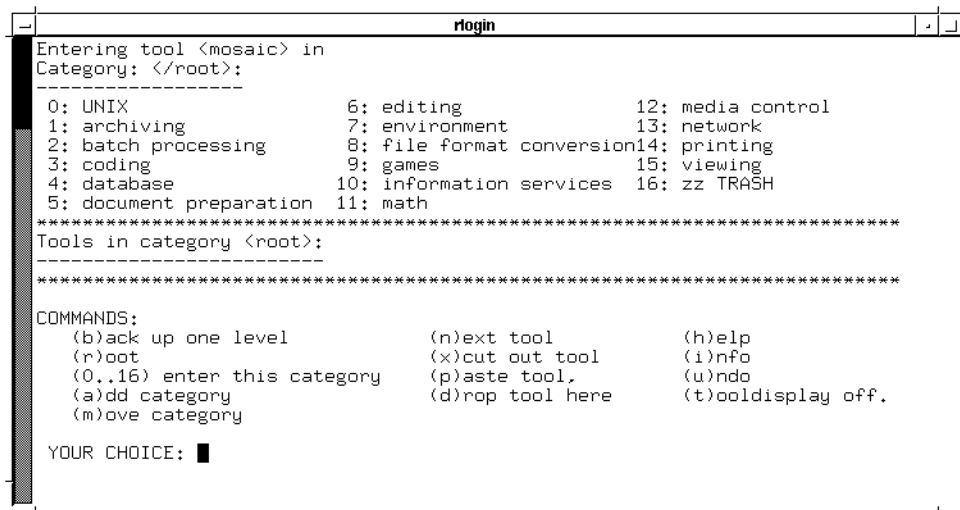
Generally, the messages consist of three parts: (i) The introduction which includes a shortcut to directly execute the interviewer command without reading the rest of the mail and the list of tools about which ToolBox wants to interview the user, (ii) the instruction part where the user receives an overview of what the interview will look like, and (iii) a trailer which gives information about the project, the authors, and the location of the mosaic page. Users who have not been solicited before receive with the soliciting message a more elaborate introduction to the ToolBox service.

3.3 Newspaper

Based on the contents of the database the Newspaper generates a newsletter which contains the following: (i) General statistics about tool usage, similar to the data we presented about the target user community earlier on in this paper. This includes which hosts are scanned and an acknowledgment of all the users who added information to the ToolBox. (ii) New tools with the results of their first interviews, and (iii) general information about the Toolbox.

3.4 Interviewer

The Interviewer is invoked by the user with a list of tools. For each tool the interviewer presents the browsing space and asks the user to “drop” the current tool in an appropriate category. During this part of the interview the user can browse and modify the browsing space, i.e., move categories, cut and paste tools, and get information about tools s/he finds in categories. After the user drops a tool in an appropriate category the Interviewer asks the user for: (i) short descriptions, experience and opinions, (ii) good sources of documentation, and (iii) preferred advisory status (“Not available for advice”, “Willing to advise, but anonymously”, “Willing to advise”, and “Don’t want to participate in ToolBox”). Since the Interviewer is called by the user it has complete access to the users environment. Therefore, it can automatically derive the location of a tool. Furthermore, it is able to check whether the user has access to Mosaic. If the user does not, it can interactively help the user to set up the necessary environment variables for Mosaic.



```

rlogin
Entering tool <mosaic> in
Category: </root>:
-----
0: UNIX                      6: editing                   12: media control
1: archiving                 7: environment              13: network
2: batch processing         8: file format conversion  14: printing
3: coding                   9: games                    15: viewing
4: database                10: information services   16: zz TRASH
5: document preparation    11: math
*****
Tools in category <root>:
*****
COMMANDS:
(b)ack up one level          (n)ext tool                  (h)elp
(r)oot                      (x)cut out tool             (i)nfo
(0..16) enter this category (p)aste tool,               (u)ndo
(a)dd category              (d)rop tool here           (t)ooldisplay off.
(m)ove category

YOUR CHOICE: █

```

The Interviewer is the most critical component of ToolBox since its design has decisive influence on the quality and quantity of information offered by the ToolBox service. People have to like interacting with the Interviewer in order to achieve a collaboratively and timely maintained directory.

3.5 db2html

This program takes the information stored in the shared database and compiles it into a system of World Wide Web hypertext pages which can be viewed by Mosaic. db2html creates a home page which essentially represents the browsing space including the hierarchical structure and the list of tools which are associated with a short abstract similar to the apropos database. Within this page, hypertext links from categories to tools and back facilitate efficient browsing. Furthermore the document can be searched for any word using the built-in search function of Mosaic. db2html also creates an additional page for each tool which is linked to the home page. By clicking on the tool name in the home page one visits the tool page which contains (i) multiple statements about use, documentation, and

location, (ii) a list of advisors with instructions on how to mail questions, (iii) a list of users that have used the tool since the directory service was initiated. Mosaic allows the annotation of each page for personal use (see Appendix for screendumps of the hypertext pages as presented by Mosaic).

4.0 Feedback and Lessons

The availability of a structured list of available tools with a list of willing advisors seemed very attractive to our target community. Due to the high visibility of ToolBox it took us almost no effort to get a lot of feedback. In this section we summarize the issues that were raised. Note, that most of our feedback comes from people who are not involved in this or any closely related project and thus have no other interest in using ToolBox than the interest motivated by the service ToolBox offers.

Some people did not believe that the expertise assumption is a useful design concept, i.e. just because users are using a tool does not mean that they are able or willing to offer information about it. Also, a user can use a tool quite often without knowing much about it, whereas the same user could be an expert on another tool that s/he is currently not using at all. We agree that the expertise assumption is not an *accurate* measure for locating expertise. However, even if the frequent user of a tool knows little about the tool, the frequency of use itself suggests the possible value of the tool. A novice might profit from knowing about such frequently used tools. Whether the user is willing to offer information is based on complex social factors similar to those we mentioned earlier when we discussed the risks of asking people. ToolBox does not force anybody to offer any information.

People did not like the selection of tools they were solicited for. Since the Picker blindly picks up any new command users were solicited for (i) tools that they thought were too insignificant (e.g., *login*), (ii) private tools that were not supposed to be available to the community, and (iii) tools they never heard of because they were invoked by the login procedure or by another tool they use. These instances proved to be very bad for the image of ToolBox. We are therefore working on alternative ways of evaluating process tables. One idea is to maintain an exclusion list of insignificant tools. Another approach is to analyze the parent/child dependencies of processes in order to catch automatically invoked tools.

We found it difficult to find the right compromise between a short interview and an interview which asked for substantial information which would result in a greater number of more specific questions. As mentioned earlier, we were striving for a short interview in order to make it attractive as an displacement activity. In particular, people did not know what to answer to the first question of the interview which asked about an opinion of a tool. Often people thought they were asked to give a brief description for a tool of which they believed was already well documented. They therefore regarded this question as redundant. The following attempts to fix this problem might illustrate the problem of finding a short question that is easy to answer and does not require any explanations. One version of the question was “Usage:” which proved to be too hard to answer. Another version said “For what do you use this tool?” which users answered with very general statements like “for editing” (in an interview concerning the tool ‘emac’). The same applied to still another version asking “What do you think of this tool?” which caused even less specific

statements like “It’s great!” Users complained about the sequential nature of the interview and suggested a form based approach that would serve as an overview of the interview and examples instead of instructions. We are working on a complete redesign of the Interviewer that will account for these complaints. We learned from this experience that users need a lot of context information in order to answer a question successfully. For an interview the context should include the overall structure of the interview and example answers to questions.

Most people only knew ToolBox from their interactions with the Interviewer. Using Mosaic proved to be too difficult for the target community: most people did not know how to use Mosaic. We expect that this will change in the near future. However, it raises the issue of how much “bootstrap” information a novice needs to acquire in order to successfully use ToolBox. In this case, a novice needs to learn the basics about how to use email, the Interviewer and Mosaic. Experience showed that we could rely on the fact that people knew how to use email and how to call a command (this knowledge was necessary for starting the Interviewer). However, this was not true for Mosaic. We think that Mosaic will become more known but to rely on this would be a mistake. Instead we plan to extend the Interviewer in such a way that it (i) helps the novice user to use Mosaic, and (ii) does not prevent expert users to use Mosaic efficiently.

In an initial test phase people sometimes received two messages a day which they found obnoxious. Sending messages to users once a day seems to be a maximum.

Concerning the issue of privacy, we got conflicting feedback. Some people did not care at all about being on the user community list as long they were not swamped by questions. Others were opposed to being on a list that was publicly available over which they had no control.

5.0 Observations

In this section we present some observations we took during the first month of ToolBox. We started ToolBox with an ad-hoc categorization of tools which was roughly based on the tools we came across ourselves. We populated this categorization with about 150 tools we randomly picked out of the man pages. Within the first few days about six other users added another 39 comments about tools. After a month we had 17 users (including the authors). ToolBox then contained 249 comments about tools (of which the authors contributed 23). During the first few days the categorization we provided was rapidly extended and also considerably restructured resulting into 65 categories. After a month only 6 more categories were added. In one case a user added a semantically equivalent category with a different name (e.g., “document preparation” while “editing” already existed). The Solicitor mailed an average of 10 messages a day with an average of three tools per message. Over the month the selection of tools discovered by Picker improves as the database becomes “smarter”. Although, the user base is still very small and there is so far no evidence for the benefit of use, we think that these preliminary results are encouraging.

6.0 Conclusions And Future Work

In this paper we presented a directory service for Unix tools that is designed in such a way that its timely collaborative maintenance does not need to be externally motivated. We are now at a stage where a continually increasing number of users *want* to use this service and give us feedback about the various obstacles and shortcomings they encounter. We therefore consider our project as mild success since its service appeared beneficial to users who were not involved in this project at all. However, as the feedback shows, many problems still need to be solved. Since ToolBox was started a little bit over a month ago it is yet too early to make definite conclusions about the viability of two basic assumptions: (i) the expertise assumption which states that a person who uses a tool knows enough to substantially help a novice, and (ii) the assumption that the structure of a user modifiable browsing space will stabilize over time. As shown above, preliminary results do not discourage these assumptions.

Future research will focus on three areas: (i) Improving the quality of the Picker: in particular we want to investigate the applicability of text analysis methods, (ii) Integration of the present Interviewer with Mosaic, and (iii) Supporting collaborative integrity maintenance.

Ideally, the Interviewer would be integrated with electronic mail (see for instance the Active Mail approach in [5]) and Mosaic such that the user does not need to call an extra application when receiving a message from the Solicitor and can take advantage of the navigation facilities and graphical user interface of Mosaic.

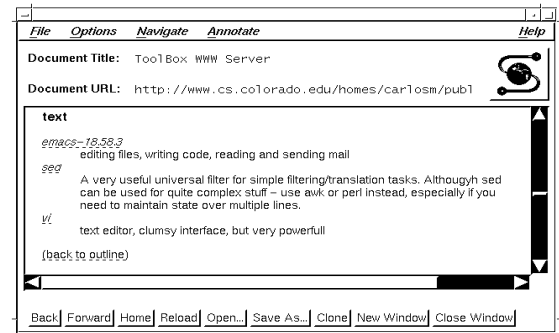
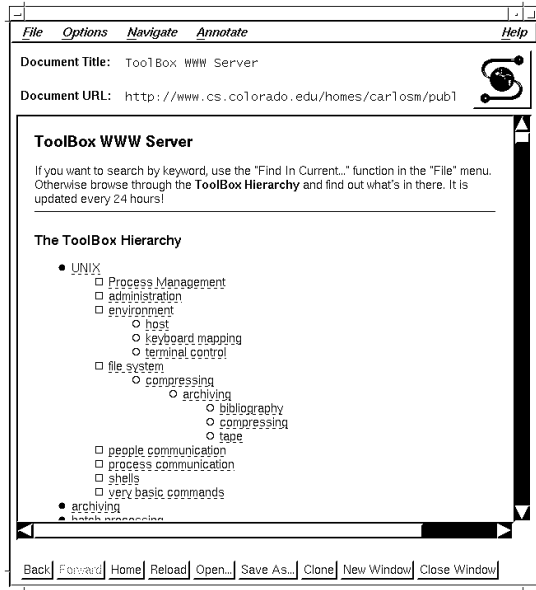
7.0 References

- [1] M.S. Ackerman and T.W. Malone. Answer garden: A tool for growing organizational memory. In *Conference on Office Information Systems*, pages 31–39, Cambridge, MA, 1990. ACM.
- [2] L. Bannon. Helping users help each other. In D. Norman and S. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*, pages 399–410. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.
- [3] L.M. Berlin, R. Jeffries, V.L. O’Day, A. Paepcke, and C. Wharton. Where did you put it? issues in the design and use of a group memory. In *INTERCHI ’93*, pages 23–30, April 1993.
- [4] S.W. Draper. The nature of expertise in Unix. In B. Shakel, editor, *INTERACT ’84*. 1984.
- [5] Y. Goldberg, M. Safran, W. Silverman, and E. Shapiro. Active mail: A framework for integrated groupware applications. In D. Coleman, editor, *Groupware ’92*, pages 222–224. Morgan Kaufmann Publishers, 1992.
- [6] J. Grudin. Groupware and social dynamics: Eight challenges for developers. *Comm. ACM*, 37(1):92–105, January 1994.
- [7] S.R. Hiltz. *Online communities: A case study of the office of the future*. Ablex,

Norwood, NJ, 1984.

- [8] M. Kurisaki. Help systems: An information-sharing approach. In D. Diaper, editor, *INTERACT '90*, pages 529–534. Elsevier Science Publishers B.V. (North Holland), New York, 1990.
- [9] K.N. Lang, R. Auld, and T. Lang. The goals and methods of computer users. *International Journal of Man-Machine Studies*, 17:375–399, 1982.
- [10] N. Miyake and D.A. Norman. To ask a question, one must know enough to know what is not known. *Journal of Verbal Learning and Verbal Behavior*, 18:357–364, 1979.
- [11] C.E. O'Malley. Helping users help themselves. In D. Norman and S. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*, pages 377–398. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.
- [12] D. Owen. Answers first, then questions. In D. Norman and S. Draper, editors, *User Centered System Design: New Perspectives on Human-Computer Interaction*, pages 361–375. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.
- [13] J.F. Rieman. Learning strategies and exploratory behaviour of interactive computer users. In *Tech. Report CU-CS-723-94*. Dept. of Computer Science, University of Colorado at Boulder, 1994.
- [14] L.L. Scharer. User training: Less is more. *Datamation*, pages 175–182, July 1983.
- [15] J.F. Sowa. *Principles of semantic networks: exploration in the representation of knowledge*. Morgan Kaufmann, San Mateo, CA, 1991.
- [16] L.A. Suchman. Office procedures as practical action: Models of work and system design. *ACM Transactions on Office Information Systems*, 1:320–328, 1983.
- [17] L.G. Terveen, P.G. Selfridge, and M.D. Long. From "Folklore" to "Living Design Memory". In *INTERCHI '93*, pages 15–22. ACM, April 1993.

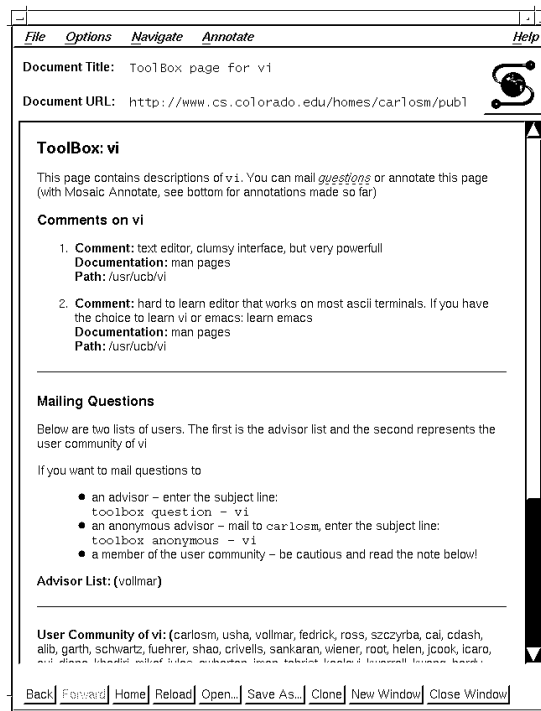
8.0 Appendix



The content of the category “text”. This is a different section in the same document that contains the hierarchical structure.

“Back to outline” leads back to the “text” category in the hierarchical structure. Clicking on the tool names leads to the corresponding tool pages.

Part of the hierarchical structure of the ToolBox Home Page as it is generated by db2html and viewed by Mosaic



Example of the tool page of “vi”. There are two comments about vi and one advisor. Part of the local user community is displayed at the bottom.