Design Guidelines for System Administration Tools Developed through Ethnographic Field Studies

Eben M. Haber, John Bailey
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
+1 408 927 1224
{ehaber, baileyj}@us.ibm.com

ABSTRACT

Information Technology system administrators (sysadmins) perform the crucial and never-ending work of maintaining the technical infrastructure on which our society depends. Computer systems grow more complex every year, however, and the cost of administration is an ever increasing fraction of total system cost – IT systems are growing harder to manage. To better understand this problem, we undertook a series of field studies of system administration work over the past four years, visiting a variety of enterprise and large university sites. One of our most compelling observations was how often the tools used by system administrators were not well aligned with their work practices. We believe that this misalignment was the result of administration tools designed without a complete understanding of the full context of administration work. To promote the design of better tools, this paper describes system administration work in more detail based on examples from our field studies, outlines the dimensions along which enterprise sysadmins differ significantly from other computer users, and provides a set of guidelines for tools to better support how administrators actually work.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: *Evaluation/methodology, interaction styles, style guides, user-centered design,* K.6.4 [**Systems Management].**

General Terms

Management, Design, Human Factors.

Keywords

Ethnography, Design Guidelines, System Administration

1. INTRODUCTION

Information Technology (IT) System Administrators are the linchpin of modern civilization - without their diligent and never-ending work, the technological infrastructure on which we all depend would quickly fall to pieces. As IT systems grow more complex, however, the cost of management has grown to dominate the total system cost [10]. A system with twice as many components can require considerably more than twice the administrative work to account for all the possible interactions [17]. Furthermore, expensive system failures are often attributed to human error [14]. Despite the cost and importance of administration work, little was known about system administrators, spurring us to undertake a series of ethnographic field studies examining sysadmin needs and work practices.

Over the past four years, we made 16 visits across six sites, studying administrators involved in managing web hosting, databases, operating systems, storage, computer security, and data center operations. Our primary approach has been naturalistic observation of administrators at work (usually recorded on videotape), along with interviews, surveys, and collection of various artifacts (diaries, instructions, planning documents, etc.). Ethnographic field studies provide an extremely detailed and accurate picture of what administrators actually do. We've found that administrators' own reports of their activities don't always correspond with what we observed and videotaped - people sometimes don't realize where their time and effort are spent. There are disadvantages to field studies, however: they are extremely time and labor intensive, resulting in a small population and temporal sample (more than once we heard, "you should have been here last week"). It should also be noted that we studied administrators in enterprise/large university settings, and that differences will exist between the work of these administrators and those working in small business or academic environments.

Some of our findings have been reported elsewhere, including a detailed study of troubleshooting activity [13], a discussion of computer security administration [11], and a more general analysis of administration tool use and work practices [3]. Our field studies also informed the development of A1/ATMA, a prototype environment to help administrators create and share small tools that automate tasks and perform monitoring [7][12]. Other studies of administration work are few, the most notable exceptions include studies of tasks and tools [1], workflow and daily activities [5],[9], and coordinated activity [15].

One of the most striking things we observed in our field studies were the cases where administration tools were not well aligned with administrator work practices. As described in [3], we saw many examples where tools failed to support the administrators' activities, forcing them to use clumsy workarounds or self-created tools. Worse, we saw cases where the tools functioned in ways that actually caused problems or significantly lengthened problem resolution. This is not to say that existing tools are completely broken, every day administrators do successfully perform complex operations on complex systems. Yet we witnessed enough problems to believe that administration tools are often created without sufficient understanding of the full context of administration work. Even when administration tools are created using a good User-Centered Design process, it is possible to focus too much on the interaction between a single user and a single tool without taking into account how the user works with other users, how the user works with other tools, and how the different tools interact. It seems that many tools are designed to support individual tasks (which they often do well) without considering broader processes and interactions inherent in system administration. This paper describes the context of administration, illustrated with examples from our field studies, and develops a set of guidelines for tools that better support the way administrators work in the real world. It begins with series of profiles of system administrators from different technical areas, describing their typical day-to-day tools and activities. The next section describes important dimensions along which sysadmins are notably different from other computer users, and lists some of the resulting work practices. The final section describes a set of design guidelines to help administrative tools better support sysadmin work practices, with some real-world examples of what can happen then the guidelines are not followed.

2. PROFILES

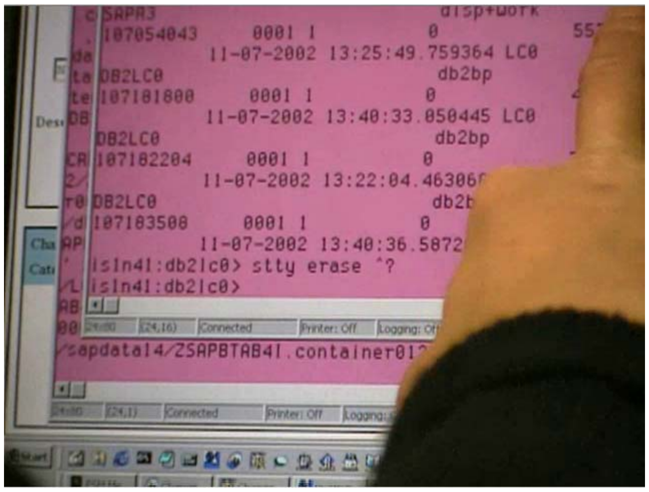
The following profiles are based on real administrators whom we observed, though the names have been changed and some personal details combined to preserve anonymity. Their stories illustrate the range system administration tasks and work environments.

2.1 Christine - A Database Administrator

Christine is a database administrator for a computer services company, managing client databases as part of outsourcing contracts. The databases are critical to the customers' business operations, and Service Level Agreements (SLAs) exist with strong penalties for system unavailability outside of "change windows" (periods of time set aside for maintenance). Unexpected downtime for longer than permitted by the SLA results in lengthy and tedious root-cause-analysis meetings, and permanent loss of data is considered absolutely unacceptable. Responsibility for different system components is distributed at Christine's site: while she is responsible for databases, other administrators are in charge of the hardware and operating systems on which the databases run. This sometimes leads to disagreements between administrators, such as whether solving a problem requires the computer to be restarted, or just the database. Occasionally the disagreements are sufficiently heated as to require management intervention to resolve.

There are two major components to Christine's work: regular monitoring/troubleshooting, and performing database changes. Monitoring is a continual process of using various tools to ensure that the database is operating correctly and efficiently. These tools might indicate suboptimal performance, leading Christine to investigate possible database changes to alleviate the situation. Unlike ad hoc monitoring, the process of making database changes is highly regulated and regimented. Changes must be approved ahead of time, and must be tested on three increasingly realistic test systems before implementation on the production system, and even then changes can only occur during a specific, customer-approved time window. Christine might spend as much as a week writing scripts and practicing various aspects of a change to make sure that the change process is well understood and that the implementation time window is reasonable.

Christine uses a variety of tools for her work, including email, instant messaging (IM), spreadsheets (e.g., holding collected performance data), and terminal windows to remote machines. She also uses several locally-created web-based tools that perform functions such as monitoring database statistics, and managing the database change process (describing, justifying, and planning the change, and getting approval from all necessary parties). She always uses a command-line interface (CLI) and scripts for performing commands in the database, as existing GUIs don't scale sufficiently to handle the 25,000 tables in the database she manages. Each database table has an eight letter name, leading to many similarly named tables, so she never types a table name directly. Instead she always copies and pastes table names from task instructions to her CLI or script. The CLI is not without faults, for example we witnessed an episode where she accidentally stopped the wrong database process. When selecting a database process ID from a list in the CLI window, each line of the list wrapped around several times. She picked the process ID from the wrong line (Figure 1). For other examples of CLI/script limitations from the same site, see sections 4.2 and 4.3 of [3].


```
SAPR3                                              disp+work
107054043          0001 1              0                    55
                11-07-2002 13:25:49.759364 LC0
DB2LC0                                 db2bp
107181800          0001 1              0
                11-07-2002 13:40:33.050445 LC0
DB2LC0                                 db2bp
107182204          0001 1              0
                11-07-2002 13:22:04.46306
DB2LC0                                 db2b
107183508          0001 1              0
                11-07-2002 13:40:36.5872
isln41:db2lc0> stty erase ^?
isln41:db2lc0>
```

```
/sapdata14/ZSAPBTAB41.container01
```

Figure 1. Christine looking through the command line listing to find the ID number of a database process to that must be stopped.

A typical day for Christine involves a mix of activities. She is in frequent contact with her coworkers via phone, e-mail and IM discussing current system status and planning future changes. She also regularly monitors database operation and performance through web and command-line tools. Sometimes her pager goes off when the customer (or one of the other DBAs) notices a problem, and she might need to put everything else aside until it is resolved. At the same time, a significant portion of her day is spent preparing for the next database change. She might be working on as few as 1-2 changes in a given month, or as many one change per week. The change window is usually on the weekend, so she'll spend much of the week before checking commands, building scripts, and testing each step of the change process on test systems. For critical operations, we saw her spend much of the day working side-by-side with a more experienced database administrator. At some points this appeared to be a way for her to learn about unfamiliar tasks, but at other points the two were on a more equal footing, each contributing knowledge of configuration and past system behavior, and each checking the actions of the other before proceeding. At especially risky junctures, such as entering a command that could damage the system if done incorrectly, they would go back and forth several times, asking each other, "are you sure?" Only once both were satisfied would they go ahead.

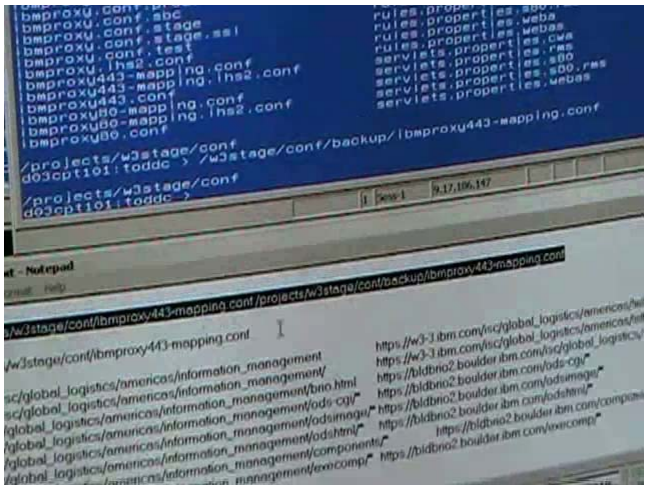
2.2 Bill - A Web Administrator

Bill is a system administrator for a large computing company, working to maintain their corporate web infrastructure. He started his career as a mail server administrator, but got bored and transitioned to a group doing both web and mail. Bill is one of the more junior administrators in a group of 12. He was mentored in web management by Nate, a more senior member of the group, and learned enough to make it his primary focus. Bill still interacts frequently with Nate when he has questions or is involved in particularly difficult operations.

The focus of Bill's job is deploying new web applications (or new versions of existing applications) onto the corporate web application servers. The process of web application deployment can be very lengthy, involving extensive planning and implementation. Planning all the timing, steps, and dependencies requires the majority of the time, but even implementation can be time consuming. For example, Bill's most complex application requires 300 to 400 steps to deploy, and his simpler web apps need 20 to 40 steps. In the ideal world, this would take all his time, but in reality he has to troubleshoot when applications fail or don't work as expected. He is also responsible for monitoring certain aspects of the underlying infrastructure (databases, operating systems, and hardware), to make sure that changes (such as new database versions) don't adversely impact web applications. Bill is the primary administrator in charge of all deployments and troubleshooting for four applications, and is backup administrator for another four. Once or twice per year he gets pulled into a "crit sit", a critical situation where the users of an application are sufficiently unhappy that all responsible administrators (web, database, operating system, network, etc.) are brought into a single room to work together to find and fix the underlying problem. These crit sits can last days, weeks, or even months in the most difficult cases.

Bill's environment is divided into a test area and production area, with all applications initially deployed into the test area. Modifications to applications in the production area are only permitted during change windows: minor changes can be made in the evening, more significant changes must wait for the weekend. Locally developed tools exist to help move an application from the test to production areas, but much of the process must still be done by hand.

Bill uses a variety of tools in his work. As with Christine, Bill uses e-mail and IM extensively to communicate with coworkers about system state, and for discussing and planning deployments. IM and phone are the primary tools used during troubleshooting, given the need for immediate responses. He uses a GUI administrative console for monitoring the status of his web applications, but also uses a variety of command-line tools as part of deployments. For command-line work, he keeps of file called "Useful Commands" containing examples of the correct syntax for a number of useful but tricky operations. When preparing for a deployment, he fills a text file with the commands he will need to run, using it as an informal script to reduce typing and sequencing errors during the change window (Figure 2).



```
ibmproxy.conf.abc                       rules.properties.weba
ibmproxy.conf.stage                     rules.properties.webas
ibmproxy.conf.stage.ssl                 rules.properties.cwa
ibmproxy.conf.test                      servlets.properties.rms
ibmproxy.ihs2.conf                      servlets.properties.s00
ibmproxy443-mapping.conf                servlets.properties.s00.rms
ibmproxy443-mapping.ihs2.conf           servlets.properties.webas
ibmproxy443.conf                        servlets.properties
ibmproxy80-mapping.conf
ibmproxy80-mapping.ihs2.conf
ibmproxy80.conf
/projects/w3stage/conf
d03cpt101:toddc > /w3stage/conf/backup/ibmproxy443-mapping.conf
/projects/w3stage/conf
d03cpt101:toddc >
```

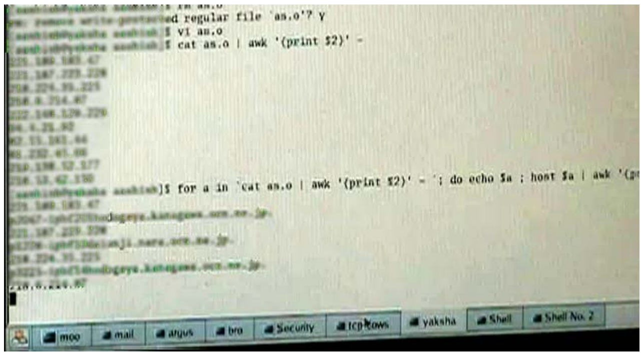
```
t - Notepad
/w3stage/conf/ibmproxy443-mapping.conf /projects/w3stage/conf/backup/ibmproxy443-mapping.conf
/w3stage/conf/ibmproxy443-mapping.conf
sc/global_logistics/americas/information_management      https://w3-3.ibm.com/isc/global_logistics/americas/
sc/global_logistics/americas/information_management/     https://w3-3.ibm.com/isc/global_logistics/americas/
/global_logistics/americas/information_management/brio.html    https://bldbrio2.boulder.ibm.com/isc/global_logistics/
/global_logistics/americas/information_management/ods-cgi/*    https://bldbrio2.boulder.ibm.com/ods-cgi/*
/global_logistics/americas/information_management/odsimage/*   https://bldbrio2.boulder.ibm.com/odsimage/*
/global_logistics/americas/information_management/odshtml/*    https://bldbrio2.boulder.ibm.com/odshtml/*
/global_logistics/americas/information_management/components/* https://bldbrio2.boulder.ibm.com/compo
/global_logistics/americas/information_management/execomp/*    https://bldbrio2.boulder.ibm.com/execomp/*
```

Figure 2. During a deployment, Bill copies commands prepared earlier from a text file (bottom) to the command window (top).

2.3 Aaron - A Security Administrator

Aaron is a security administrator at a large university. Working with his manager and two other security administrators, they are responsible for detecting and eliminating hostile intrusions into any of the several hundred computers in his department (this work is discussed at length in [11]). They also monitor network traffic to ensure that computer use follows established guidelines in areas such as file sharing, downloading of pornography, etc. The department is regularly under attack by "crackers", in part due to its prestige, and also due to its exceptionally heterogeneous installed base of computers, with many hardware vendors, operating systems, software packages and versions. Aaron's group tries to stay abreast of the most recently discovered vulnerabilities, updating exploitable machines before they're attacked. They also perform extensive network monitoring to find and isolate machines under attack, often trying to track attacks back to their source.

Aaron uses a variety of network and computer monitoring tools, most of them CLI-based. Some are running constantly, sending him e-mail when suspicious activity is detected, and others perform specific scans on demand. Alerts come in to his e-mail box constantly, so he checks e-mail every few minutes to evaluate them. Most can be ignored based on his knowledge of the systems involved, though sometimes he will investigate further, going as far as to look up or contact the owner of a machine to determine if certain activity is legitimate. Aaron often creates ad hoc command-line analysis tools (using awk, grep, etc.) for processing the monitoring tool output into more comprehensible forms (Figure 3). Aaron also spends considerable time using a multi-room multi-user chat environment (the "moo") where administrators across the university discuss both work and non-work topics. The security “room” of the “moo” has frequent interchanges between Aaron and his fellow security admins discussing issues such as whether certain network activities are suspicious or not.



```
[illegible]ed regular file `as.o'? y
[illegible]$ vi as.o
[illegible]$ cat as.o | awk '{print $2}' -
[illegible]
[illegible]$ for a in `cat as.o | awk '{print $2}' - `; do echo $a ; host $a | awk '{p
[illegible]
```

Figure 3. In one of Aaron’s many command line windows, he creates ad hoc data processing tools with awk, grep, etc.

A typical day for Aaron consists of alternating between long term tasks, learning, and monitoring/troubleshooting. He monitors the “moo” and his e-mail, responding when necessary to reports of suspicious activity. He does attend meetings, yet he always brings his laptop along so he can continue monitoring. When not responding to reports, he engages in longer term tasks, e.g., scanning all systems to check for vulnerable versions of SSH. He spends the remaining time learning, searching on the web for information concerning the latest vulnerabilities, occasionally downloading and testing exploit code to better understand exactly how it works.

3. DIMENSIONS OF ADMINSTRATION WORK

As Aaron, Bill, and Christine demonstrate, administrators of large IT systems are not a monolithic group - they differ from each other both within and across job responsibilities. As a whole, however, they are distinct from other computer users along several important dimensions, including the complexity and scale of the systems they work with, the scope of their responsibility, exposure to risk, degree of collaboration, and technical ability. We will examine each of these areas in more detail, and describe some of the work practices that administrators have developed around them (for additional detailed case studies and further discussion of work practices, see [3]).

One of the most remarkable aspects of modern system administration is the potential scale and complexity of the systems themselves. Something as conceptually simple as a website might be implemented with a set of different components including HTTP servers, web application servers, authentication servers, content servers, firewalls, networks, and each of these components might be replicated to improve performance or reliability. Furthermore, each of these components could be manufactured by a different vendor, and the particular arrangement of components might be unique in the world. System complexity can lead to specialization, with different experts responsible for each component, but the narrowing of scope can mean that no one individual fully understands the system as a whole. Changes to these complex systems can be very involved: Bill was one of many sysadmins we witnessed engaging in regular tasks that required tens or hundreds of steps affecting different parts of the system. In addition to complexity, sheer size can be an issue. For example, database and storage systems can be massive, with regular operations taking hours or days to complete. One group of storage administrators we observed had so much data accumulating that they foresaw a day within the next few years when the time required to read and back up all their tapes will exceed the media's 10-year lifetime. In addition, admins often deal with vast numbers of different systems. For example, we observed one web administrator undertaking the semi-monthly process of resetting the 120 passwords for the various systems he had access to. Not all sysadmins deal with the largest and most complex systems, but all the admins we saw dealt with systems significantly larger, more complex, and more numerous than other computer users.

Another driving force in administrators' lives is risk and concomitant stress. Many IT systems are mission critical: if they fail then organizational activities come to a halt. System unavailability can cost enterprises millions of dollars per day in lost business or productivity, and data loss can be a true disaster in any environment. Not every administrator works under such a high-risk conditions, yet all the administrators we observed could be certain that a system failure would quickly result in phones ringing and people yelling at the very least. One of the database administrators we observed stated, “If data is lost…that is when you write your resume.” Sysadmins clearly have more at stake than most computer users.

One other important attribute of administrators is their technical inclination. We saw considerable script and tool building activities among sysadmins, and we heard repeatedly of their strong preferences for command-line interfaces. While our observations and surveys found that many lack a formal background in computer science, it is no surprise that people who manage IT systems for a living practice tool building and a prefer CLIs to a greater degree than many computer users.

One of the ways that administrators manage the complexity, scale, and risk that they face is through extensive collaboration. All the sysadmins we observed were parts of larger teams, spending significant time using telephone, instant message, and e-mail communicating with their coworkers. For example, in one 2.5 hour troubleshooting session analyzed in [13] and [3], we found that 90% of the sysadmin's time was spent communicating with other admins about the state of the system and how to proceed, and only 6% of the time was spent gathering information and running commands on the affected system. Responsibility for a system was sometimes spread across several individuals to permit sharing of knowledge, effort, and risk. Admins working together on the same issue can pool experience and double-check each other's actions. As in the case of Christine, we observed database administrators sitting in pairs at a single keyboard while working on critical operations, and only when both were satisfied would each command be run. In addition, complex systems are often managed through distributed responsibility, with experts in different components working together to coordinate their activities and keep the larger system running smoothly. This permits a greater level of expertise for each component, though it can result in no single person having a detailed understanding of the entire system.

In mission critical environments, we saw planning and rehearsal as important tools to manage risk. Rehearsal can be used both to establish the correct sequence, syntax and arguments for all commands, as well as to gain an estimate of the time required to perform the operation (since operations usually must be performed within a limited window of time). Database administrators like Christine were the most strict in their rehearsals, spending as much as a week testing all operations on a series of test systems. Web and storage administrators were not as strict, but they would still check all proposed changes on a test system before touching the active system. Admin-created formal and informal scripts can be part of the planning process, to help ensure consistent and reliable execution.

Given the serious consequences of failure, administrators need to maintain situational awareness of system state. In many cases, however, the monitoring and notification provided by IT tools is insufficient for administrator needs. Furthermore, the heterogeneous nature of many systems means that no single tool exists to monitor everything. As with the blind men and the elephant, each component provides tools to monitor one part of the system, but none gives a view of the whole. In addition, a problem in one part of the system might be caused by a misconfiguration in another part. We observed many instances where administrators created their own tools to provide better situational awareness. In one example, a DBA created a series of dynamically updated web pages displaying important statistics about running databases. In another case we saw a sysadmin spend his own money to purchase a third-party tool that scanned various log files and status web pages for certain regular expressions, e-mailing or paging him when indications of system failure were found. He stated that before he set up this tool, one of their systems had been effectively non-functional for two days before anybody realized. A final example is the "Crit-Sit" (described in detail in [3]), where a group of admins needed to spend several weeks working together troubleshooting an intermittent system failure. They created a number of ad hoc tools for collecting and integrating information from different components, and eventually found the cause to be a subtle interaction between the operation of several components.

The technical inclination of system administrators permits them to build their own tools to manage risk and enhance situational awareness. It also impacts their choice of interface style, both common wisdom and our own observations indicate that system administrators prefer command line interfaces (CLIs). While all the sysadmins we observed used GUI tools for some tasks (e.g., e-mail, web, or certain administrations tools), all preferred using CLIs, especially when performing more critical tasks. To learn more about this preference, we conducted a survey of 101 system administrators from various backgrounds and organizations (recruited through system administration professional organizations). The survey included questions about education, experience, what interfaces they used regularly, and questions about their perceptions of CLIs and GUIs. The vast majority of respondents found CLIs to be faster (89% to 5%), more trustworthy (81% to 4%), more reliable (82% to 5%), more robust (81% to 8%), and more accurate (75% to 11%). Results were more split on likeability, with CLIs somewhat preferred (56% to 31%), and ease-of-use, with GUIs ranked higher by a plurality (47% to 39%). Our observations provide further explanation for the CLI preference. Speed is certainly important in many situations. A GUI is not useful if it takes too long to come up and populate itself with all the data for a very large system, as one admin commented in reference to his GUI: "If the database is crashing, and you have 5 minutes to find the problem and kill the connection before the database crashes, if you have that many connections, forget it." Reliability, accuracy, trustworthiness, and robustness are crucial for maintaining situational awareness, yet we saw cases where GUIs would hang, or show outdated information unless restarted, reinforcing a lack of trust in GUIs. Another factor that probably influences administrator preference for CLIs is tool building: virtually all CLIs support scripting, whereas most administrative GUIs do not. Even without formal scripts, when a sysadmin is following a lengthy procedure, the command line window provides a history of everything that's been done. We often saw sysadmins scrolling back and forth through the window to understand exactly where they were in the process, and what the output was from each step.

While scripting and history are not a standard part of the windows/menus/pointer GUI paradigm, there is no fundamental reason GUIs could not natively support them. Consider the example of SMIT [16], which maintains both a GUI and command line, converting every command in either form to the other. History and scripting are supported through the command-line part of the interface. One can imagine other approaches to supporting history and scripting in a GUI: what is needed is an interactive visual representation for each action and its output. In SMIT the representation was exact, but this need not be the case. Action representations could be shorthand, like navigation breadcrumbs that change the UI when clicked. Implementation would not be trivial, but it would be feasible, and very useful.

System administrators clearly face greater scale, complexity, and risk than most computer users, and they tend to be more technically inclined. These factors influence their work practices, which involve extensive collaboration, planning and rehearsal, and tool building. Existing administration tools often don't support these practices, however. We observed fragmentation - complex systems with many components, each with its own tool providing a different, incompatible view of the system. Configuration and log files were scattered, poorly organized, and often used inconsistent terminology. We didn't see any tools with explicit support for collaboration, shared views of the system, or inter-person workflows, and we saw few tools with support for planning and rehearsal beyond basic scripting. We also saw many clunky GUIs in environments where users prefer CLIs, since they need fast, reliable, accurate, scriptable interfaces. The next section contains a set of guidelines to outline how administration tools could better support the way sysadmins actually work.

4. DESIGN PRINCIPLES

The following design guidelines address the particular characteristics, needs, and work practices of system administrators. The guidelines do not mandate particular UI widgets or interaction styles, rather they specify tasks and organizational attributes that should be supported regardless of the details of the interface. They are not intended to be used on their own, instead they should complement the guidelines already in use for the design process. The first subsection contains guidelines related to dimensions of scale, complexity, and risk. The next subsection contains guidelines related to collaboration and communication, since our studies suggest that virtually all administrators collaborate extensively, and that existing tools provide little support for it. The final subsection contains general guidelines for tools used by technical people. All guidelines may not be appropriate for all users and situations - different groups of administrators fall along different points in these dimensions of scale, complexity, risk, collaboration, and technical ability, the rules in these sections should be emphasized commensurate to the environment and characteristics of the particular users.

4.1 Scale, Complexity and Risk

A real-world installation may be huge, with regular operations requiring hours or days to complete, yet administrators usually have very limited time windows in which to complete changes.

- Provide progress indicators, preferably displaying the amount of time remaining.
- Support forecasting of how long an operation will take before the operation begins. This would help the planning and rehearsal process, as we observed administrators extrapolating execution times from test systems to estimate how long an operation will take on the different hardware and data of the production system.
- Perform operations in an asynchronous, non-blocking manner. For example, an administrator working on a very large database won't use a GUI tool that freezes while filling a list box with the names of all 25,000 database tables. The ability for sysadmins to perform other tasks while long-running operations are progressing is essential.
- Support more flexible management of long-running or resource-intensive tasks. Specifically, permit pausing operations in progress, record and rollback (a.k.a. undo), to permit a system to quickly return to a previous state, and a resume function, so that if a long running operation is interrupted for any reason, it can pick-up from where it left off. Operation restarts as a result of a dropped connection, for example, can be costly and frustrating. Databases already support undo, but generally don't support suspend or resume, and most other IT systems don't support any of these.
- To the greatest extent possible, do not require systems to go offline to perform maintenance operations. Change windows for system downtime are limited, and the more offline operations there are, the harder it will be for sysadmins to get everything done in the change window.

Systems may be composed of numerous components that must work together smoothly. Any tool will likely be used with many other tools as part of a larger system. In general, design tools to work well with others, in particular:

- Use standard terminology for reporting configuration and state information. We observed an episode where an administrator wasted 45 minutes when two different tools (from the same vendor) reported the same configuration information using syntax so different that the administrator was convinced that they were reporting different information.
- Use standard configuration and logging formats, wherever available. Log and configuration files from different tools will be searched and correlated together.
- Provide APIs/plug-ins so the tool can be integrated into system-wide monitoring and management meta-tools (e.g. dashboards).
- Follow platform conventions in structuring the tool's installation and data storage file system directories. Administrators familiar with the platform will expect to find data in certain places.
- Provide a clean separation of user and non-user data in file system directories and within individual files. It is very desirable to manage product or administration tool binary files independent of the user-data that a tool generates and manipulates. For example, if a file containing both user and non-user data is updated during application of a patch, the user data may be lost. Likewise, user data is often managed as an asset, and backed-up and/or archived on a regular schedule. Mixing information that is intended to only be read and updated by machine in a file with information that can updated manually by human is likely to cause errors.
- Configuration of multi-component systems can take considerable time and effort. Make sure that errors in configuration are discovered quickly, by checking interactions with other components as soon as possible. For example, the troubleshooting session analyzed in [13] was complicated by the fact that a configuration error in one component was not reported until a second component was configured (even though it could have been checked earlier). This lead to a wild goose chase looking for errors in the configuration of the second component.
- Given the complex, distributed nature of some systems, administrators often access systems remotely. Make sure that administration tools behave well over slower remote connections, and handle inadvertent disconnections gracefully.

Administrators may regularly use many different tools, and won't necessarily be an expert in all of them.

- Provide an easy-to-use interface for users who are technically sophisticated, yet may use the tool infrequently. This does not necessarily mean a GUI or wizard, since sysadmins can be very happy with a well-designed CLI. The following quote illustrates this point - a sysadmin is comparing the GUI and CLI available to him for a particular task: "If the command line were cryptic and not so clear, I would say, all right it makes sense to use the GUI, but it's so simple using the command line. If you don't know something…for example, if you want to list the objects and you don't know how to do that…you type 'object' and it would say 'object' and then give you all the options you could run 'object' with. Then you'd say 'object list' and it would show you all the objects."
- Use conventional and consistent terminology in documentation and interfaces. Being unique or creative does not facilitate positive transfer among tools that sysadmins use.

- Make all documentation available on the web, searchable by outside search engines such as Google™. All the administrators we observed would look up error codes and messages using Google™. They did not use search engines on tool vendor websites, in part because they would then miss out on other sites that discuss the same tool.

Modern IT systems have hundreds or thousands of configuration parameters that may interact in unexpected ways. There exist external tools to help configuration management [4][8], yet existing administration tools could do a better job.

- \If the system uses configuration files, include comments in the configuration files, explaining parameters and interactions, and group related parameters together in the file. As an example of the problems that can arise from poorly organized files, we observed a troubleshooting session involving a component sending and receiving network traffic on two different ports. The port values were specified several hundred lines apart in the configuration file, allowing an administrator to confuse one for the other. Even if the two options could not have been co-located, comments could have referred between the two.

- Provide a means of comparing configuration information, either between systems or for a single system over time. This would allow administrators to compare working and non-working systems, to find which changes cause problems.

- Another approach to managing large numbers of configuration parameters is to group interacting parameter values together into sets of consistent, higher level functional "profiles." Each profile would have a set of parameter values appropriate to a given function, i.e. these parameters should be configured together to complete a function. This abstraction would permit less experienced administrators to make changes without having to understand all of the underlying interactions.

- Configuration parameter values that get loaded during system startup may differ from later operational values. Provide tools that allow a sysadmin to see both values, and to propagate values bidirectionally to prevent errors and improve sysadmins' understanding of the system.

Operations on complex systems may involve many steps. Sysadmins often manage these tasks by creating formal scripts, or by using the command-line window to display a history of everything they've done, showing where they are in the process and the result of each step.

- Both GUI and CLI tools should support scripting of complex processes.

- Some complex tasks would benefit from support for mixed initiative scripting, where parts of the script execute automatically, but other parts require human intervention to complete the action. Generally, complicated processes that can't be completely automated are implemented in an ad hoc manner with different scripts separated by user actions.

- GUI administration tools should also support a history. When there are meaningful chains of discrete steps, provide a view of completed steps and any associated completion codes, status, or other output.

- Administrators often share scripts. Shared tools can be an excellent means for capturing and reusing organizational knowledge. This can be facilitated by implementing central tool repositories, and easy-to-understand scripting languages. For example, see [7][12] which describe A1/ATMA, a prototype sysadmin scripting tool and repository we developed.

Administrators need to establish and maintain situational awareness, including but not limited to complex IT topologies, conceptual models of transaction workload and flows, and IT management processes and governance. Situation awareness is not just an administrator being able to project what will happen next based on information about system configurations and states, rather it is also an understanding of wider IT management related events including help desk, change management, and problem management systems. Situation awareness has a social component as well: knowing what other sysadmins are doing, and where, is an important aspect of managing IT systems.

- Situation awareness has been described as a process that progresses through three stages – perception, comprehension, and projection [6]. Administration tools should be designed to support development and maintenance of projection-level situation awareness [2].

- Provide alerting tools to help automate monitoring. Alerts should support customizable, progressive thresholds, selectable destinations (e.g. pager, email, console), and be suppressible.

- Provide visual representations that selectively layer physical and logical representations with configuration and operational state information.

- Log both who launches an operation on the system and also an optional comment from the administrator describing why. In an internal development study of problem determination, after checking for a "pulse", the next thing a troubleshooting sysadmin usually does is find if someone has changed something. Knowing what changed, who made the change, and why can greatly help diagnose problems.

- Whenever changes are made to a system, automatically log all the parameters/settings that changed. When a system goes from a working to non-working state, administrators spend considerable time determining what changed to cause the failure. A log recording everything that changed, and when, would greatly help troubleshooting.

Administrators practice planning and rehearsal to manage risk during complex operations.

- Tools should provide formal support for migration of scripts/operations from test to production environments, for example, encapsulating an operation in such a way that it could be moved from one machine to another without modification. As described in [3], we observed practices of writing scripts, executing them on one machine, then copying and editing them to run on the next machine. Every time edits occur, errors can be introduced.

- Allow scripts to be checked before execution, so the user can be sure that they will run, and know how the script will change the system, and approximately how long the script will take to run. This would let a sysadmin create a script to perform a certain operation, and know ahead of time that it will run as expected in the given time window.

- When errors in script execution occur, display to the user whatever changes have been made to the system, so the resulting system state is clear to the user. We witnessed instances where errors in scripts or other operations left the system in an unknown state.

In risky environments, administrators often must respond quickly to signs of impending failure.

- Tools must start up and execute sufficiently quickly to respond to critical situations.

- Permit users to record or specify the normative baseline of system operation, then automatically issue warnings when there are significant departures from the norm. This would signal potential future problems and provide sysadmins more time to react.

4.2 Collaboration and Communication

System Administrators will work together, whether or not this collaboration is assisted by their tools. Sysadmins will often work on the same problem from different computers.

- Provide a means for sysadmins to share views of system state, so they can see and discuss the same thing. We observed some admins using screen-sharing software, but it would be better if administration tools formally supported shared views. For example, just as instant messaging tools allow users to know whether their buddies are logged in, administration tools could let users know who else is logged in, what they're doing, and let them share views of the system. Alternately, there could be something like a URL that could be passed via IM or e-mail to let one sysadmin show another exactly what is on the screen.

Complex activities may involve handoffs between sysadmins responsible for different components.

- Tools should support being part of a larger workflow. Shifts in responsibility could be encoded into scripts, with a new sysadmin automatically notified and brought to the right part of the interface when it's their turn.

As activities such as problem determination escalate, more people are pulled into the activity for their particular technical expertise or experience, creating a spreading activation effect.

- Provide support for grounding new participants as quickly as possible when they join the activity. We've observed that a lot of time is spent "bringing the new person up to speed" as to what has already been tried, and the results of attempted resolutions or investigations

4.3 Interfaces for technically oriented people

System administrators are technically inclined people responsible for configuring, maintaining, and fixing computer systems. Our observations suggest that in their work, they must not only determine what actions to take, they must also understand *why* to take certain actions. The process of troubleshooting is often concerned with determining the state of a system, but it can also be about fixing a system administrators incorrect understanding of how the system works.

- Administration tools should present information in such a manner as to help the sysadmins understand how the system works. This is a rather general point that may be aided by the following specific example. In the troubleshooting session analyzed in [13], the administrator was using a tool to configure one server to permit communication between two additional servers. An error message appeared, saying, "Cannot reach server: Error 1231A". Given that there were three servers and numerous network ports involved, the error was vague to the point of uselessness. If, however, the error message had reported, "Cannot reach server 'foo.bar.com' on port '7234'", then the administrator could have very quickly isolated the cause. As it was, the administrator spent several hours trying to isolate problems with the wrong server based on a faulty assumption. Administrators are technical people trying to understand and solve problems, and tools must give them the detailed information to do so.

- As stated above, visual representations can help administrators understand complex system state and behavior, especially when multiple components are interacting.

Given their technical inclinations, it is not surprising that system administrators express a preference for command-line interfaces. We believe, however, that this preference is founded on the needs of system administrators for fast, reliable, trustworthy tools for performing complex operations on large, complex systems in a risky environment. To be successful, any administration tool, whether it has a GUI or CLI, should have the following attributes:

- Speed – the tool must launch and respond quickly enough to deal with emergencies.

- Scriptable – it must be possible to drive the tool from a human-readable pre-defined script.

- Reliable/Trustworthy/Transparent – from our observations, many GUIs are deemed unreliable or untrustworthy because they crash or hang and display outdated information. The continually running nature of a GUI means that the data it shows could be out of date if an internal problem has stopped data collection. A CLI tool, running on demand, usually shows correct information, or nothing at all. GUIs could overcome some of these problems by providing progress and provenance information. For example, they could stamp displayed information with the time it was collected, or provide a visual indication showing that the tool is still running and collecting up-to-date information.

- As mentioned earlier, GUI's often do not scale across hundreds or thousands of resources. In some cases there will be no substitute, expert sysadmins will need the power that CLI and scripting provides to perform operations targeted at large collections of resources in "batch mode". GUIs must be tested against the largest conceivable system size, and if they don't perform then a CLI must be provided.

5. CONCLUSIONS

IT System Administrators are a crucial population who are not always well served by the tools at their disposal. We believe that administration tools can be improved by supporting the larger context of administration work, addressing the scale, complexity, and risk faced by system administrators. Of course, different sysadmins have disparate environments and practices, so the guidelines presented here must be applied based on the particular characteristics of the users.

Future work will be needed to evaluate the best ways to satisfy the guidelines, and to evaluate their impact and cost. Many of the guidelines are instances of known good practices, so the question must be considered as to why they aren't followed more often. We recognize that many of these guidelines could be difficult to realize. For example, an area as straightforward as improving error messages can be difficult due to the demands of the development process, where user-visible text often must be frozen before any other part of the interface, and generic messages are preferred since they may be reused many places. In addition, it may be difficult to balance some of the different guidelines, e.g., a feature such as collaboration support could increase complexity and reduce reliability. No doubt other barriers exist due to marketing and business considerations involved in creating admin tools. In the end, however, administration tools will be better and administration less costly if developers remain aware of the particular needs of system administrators while developing their tools, even if only some of the guidelines can be met. As one developer commented after seeing a presentation containing some of our video and conclusions about sysadmin needs, "Without this kind of information, we will be doomed to endlessly deliver beautiful, sparkling tools that are totally inappropriate to the jobs our customers need to use them for."

6. ACKNOWLEDGMENTS

The huge labor involved in performing and analyzing this series of ethnographic field studies was the work of many beyond the authors, including Rob Barrett, Christopher Campbell, Steve Farrell, Eser Kandoğan, Cheryl Kieliszewski, Paul Maglio, Madhu Prabaker, Joe Ryan, Leila Takayama, and Anna Zacchi. Credit for the sysadmin survey and analysis goes to Eser and Leila. Thanks also to Jeanette Blomberg for her valuable advice on conducting ethnographic studies for those of use who hadn't studied Anthropology. Special thanks go out to all the system administrators who took time out of their busy schedules to answer our questions, fill out our surveys, and who put up with being followed around with video cameras.

7. REFERENCES

[1] Anderson, E. Researching system administration. Ph.D. Thesis. University of California, Berkeley, 2002.

[2] Bailey, J., Etgen, M. & Freeman, K. Situation awareness and system administration. In Barrett, R., Chen, M., & Maglio, P. P. (Eds). *System Administrators are Users, Too: Designing Workspaces for Managing Internet-scale Systems*, CHI 2003 Workshop.

[3] Barrett, R., Kandogan, E., Maglio, P. P., Haber, E. M., Takayama, L. A., Prabaker, M. "Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices." Proc. CSCW 2004.

[4] Burgess, M., ''A Site Configuration Engine,'' *Computing Syst*ems, Vol. 8, Num. 1, p. 309, MIT Press, Cambridge, MA, Winter, 1995.

[5] Dijker, B., A Day in the Life of System Administrators, SAGE, http://sageweb.sage.org

[6] Endsley, M.R. (1996). Automation and situation awareness. In R.Parasuraman & M. Mouloua (Eds.), *Automation and human performance: Theory and applications* (pp.163-181). Mahwah, NJ: Lawrence Erlbaum.

[7] Haber, Eben, Eser Kandogan, Allen Cypher, Paul P. Maglio, and Rob Barrett, "A1: Spreadsheet-based Scripting for Developing Web Tools." Proc. USENIX LISA 2005.

[8] Hagenmark, B., K. Zadeck, ''Site: A Language and System for Configuring Many Computers as One Computer Site,'' *Proceedings of the Workshop on Large Installation Systems Administration* III, p. 1, USENIX Association, Berkeley, CA, 1989.

[9] Halprin, G. The Workflow of System Administration. In Proceedings of the 6th Annual Conference of the System Administrators Guild of Australia (SAGE-AU '98) (Canberra, Australia, July 6-10, 1998)

[10] IBM, "Autonomic Computing: IBM's Perspective on the State of Information Technology"; http://www.ibm.com/ industries/government/doc/content/resource/thought/278606 109.html

[11] Kandogan, Eser, and Eben M. Haber, "Security Administration Tools and Practices." Security and Usability: Designing Secure Systems that People Can Use. Ed. Lorrie Faith Cranor and Simson Garfinkel. Sebastapol: O'Reilly Media, Inc., 2005, pp357-378.

[12] Kandogan, Eser, Eben Haber, Rob Barrett, Allen Cypher, and Paul Maglio, "A1: End-User Programming for Web-based System Administration." Proc. ACM UIST 2005.

[13] Maglio, Paul P., Eser Kandogan, and Eben Haber, "Distributed Cognition Analysis of Attention and Trust in Collaborative Problem Solving." Proc. Cognitive Science 2003.

[14] Patterson, D. et al. Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies, Technical report CSD-02-1175, Computer Science Dept., Univ. of California, Berkeley, 2002.

[15] Sandusky, R. J. Infrastructure Management as Cooperative Work: Implications for Systems Design, In Proceedings of the ACM Conference on Supporting Group Work (GROUP'97) (Phoenix, Arizona, November 16-19, 1997), ACM Press, New York, New York, 1997, 91-100

[16] Vetter, Scott, and Susan Segura, "System Management Interface Tool SMIT." IBM RedPaper, http://www.redbooks.ibm.com/abstracts/redp0105.html.

[17] Woods, D. D., Decomposing Automation: Apparent Simplicity, Real Complexity. In Parasuraman, R., Mouloua, M., (Eds). *Automation and Human Performance – Theory and Applications*, Lawrence Erlbaum Associates, New Jersey, 1996.