Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices

Rob Barrett, Eser Kandogan, Paul P. Maglio, Eben Haber
Leila A. Takayama, Madhu Prabaker
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA
+1 214 233 3059
{barrett, eserk, pmaglio}@almaden.ibm.com, ehaber@us.ibm.com,
takayama@stanford.edu, madhukp@berkeley.edu

ABSTRACT

Computer system administrators are the unsung heroes of the information age, working behind the scenes to configure, maintain, and troubleshoot the computer infrastructure that underlies much of modern life. However, little can be found in the literature about the practices and problems of these highly specialized computer users. We conducted a series of field studies in large corporate data centers, observing organizations, work practices, tools, and problem-solving strategies of system administrators. We found system administrators operate within large-scale, complex environments that present significant technical, social, cognitive, and business challenges. In this paper, we describe system administrator tool use in critical, high-cost, labor-intensive work through observational, survey, and interview data. We discuss our findings concerning administrator needs for coordinating work, maintaining situation awareness, planning and rehearsing complex procedures, building tools, and supporting complicated interleaved workflows.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *interaction styles;* Group and Organization Interfaces – *collaborative computing;* K.6.4 [**Management of Computing and Information Systems**]: Systems Management

General Terms

Design, Human Factors.

Keywords

System Administration, Ethnography, Collaboration, Command-Line Interfaces, Situation Awareness.

1. INTRODUCTION

System administrators (sysadmins) design, configure, troubleshoot, and maintain complex computer systems comprised of dozens of components (e.g., database management systems, web servers, application servers, and load balancers), and hundreds of servers that are distributed across multiple networks and operating system platforms. Because the computational infrastructure of everyday life depends on sysadmins performing their work nearly flawlessly, system management costs now account for most of the costs of setting up and running large computing systems, far outstripping that of buying hardware and software [10], [11], [15]. Furthermore, since systems are so complex and so difficult to manage, people are often blamed for failures [15]. Thus, sysadmin work places high cognitive demands on practitioners—as sysadmins troubleshoot systems, making sense of millions of log entries by controlling thousands of configuration settings, and performing tasks that take hundreds of steps—and also places high social demands on practitioners—as sysadmins need organizational and interpersonal skills to coordinate tasks and collaborate effectively with others.

Despite the importance of sysadmins, few HCI studies report on their particular problems and practices (see [1] and [17]). Nevertheless, sysadmins themselves have reported on certain aspects of behavior and tools, including types of work and possible prototype tools [1], and day-in-the-life [4] and workflow studies [6]. Also, work on command line interfaces [20], computer programmers [19], and the relationship between human control and automation [18] are all potentially applicable to problems in system administration.

Because of the lack of in-depth studies of this critical user group, we conducted field studies in large corporate data centers, observing the organization, work practices, tools, and problem-solving strategies of many kinds of sysadmins. Field studies offer insights into work that cannot be found in focus groups, lab studies, or surveys alone (see [7], [12], [14]). When work is examined in context, it becomes clear that people work creatively with technology to support their practices flexibly and adaptively—though systems are often designed inflexibly, people make do, naturally working around limitations and built-in constraints [22]. Field studies also show the ad hoc day-to-day interpersonal interactions not visible in a laboratory setting.

In what follows, first we detail our methods. Second, we provide an overview of our results by describing the typical tasks, tools, and environment of our sysadmins based on observations, diary entries, and survey data. Third, we describe four cases that illustrate some of the major issues we observed. Finally, we discuss our findings on collaboration, situation awareness, planning, tool building, and multitasking.

2. DEFINITIONS AND METHODS

We define *sysadmins* broadly as those who use their technical, social, and organizational skills to architect, configure, administer, and maintain computer systems, including operating systems, networks, security systems, infrastructure, databases, web servers, and applications. In the work reported here, we focused primarily on database and web sysadmins. We used a variety of techniques to gather data, including surveys, diary study, interviews, and naturalistic observations.

2.1 Surveys

We conducted a preliminary survey in mid-2002 to help us begin to understand the system administration domain. In this survey, sixteen web administrators answered a variety of questions regarding their background, tasks, operations, computer environment, and so on.

More recently, we conducted an extensive survey in which we specifically asked about collaboration practices and tool use. We collected data from 101 sysadmins of various specialties (database, web, operating system, network) solicited through newsgroups, mailing lists, local and national sysadmin user groups. We analyzed these data both quantitatively (rating questions) and qualitatively (open-ended questions).

2.2 Diary Study

One of our sysadmin participants kept a log of his daily activities for ten months in 2002-2003. His diary included five to ten items per day, identifying tasks such as meetings, problem solving, or configuration, along with other relevant details, including people he worked with. We analyzed this diary by categorizing each entry as a specific kind of activity, and by noting what tools and people were mentioned for each activity.

2.3 Interviews

We conducted 12 interviews with sysadmins, managers, team leads, and others in various roles. Interview questions typically focused on their issues and concerns, challenges in their work, organizational questions, etc. Interviews took place in their offices, usually as they worked, which helped us focus questions on their work and follow leads as issues arose.

2.4 Naturalistic Observations

We conducted six field studies of database and web administrators at large industrial service delivery centers in Colorado, New York, Connecticut, and North Carolina. Two researchers typically participated in each visit, which lasted three to five days. Typically, we followed one sysadmin per day as he or she worked in the office, attended meetings, and so on. One researcher took notes and occasionally asked questions, while the other videotaped interactions with the computer and other activities in the office. We asked our participants to speak aloud while working, which they often did. At the end of each day, we asked clarifying questions about the observations from that day. We collected physical and electronic materials and took pictures of the artifacts in the work environment. In all, approximately 200 hours of videotape were collected, reviewed, and analyzed to varying degrees.

3. TASKS, TOOLS, AND ENVIRONMENT

Over 25 days of observation, we watched 12 different sysadmins of various skills and specializations. When taken individually, each day had rather different characteristics. When taken together however, some aggregate patterns emerge.

3.1 Tasks

Our diary analysis revealed that a significant proportion of tasks (23%) were meetings, nearly the same as found in [4]. These meetings were usually recurring status meetings in which sysadmins and managers discussed changes to computer systems, suggesting that the job is as much social as technical. In fact, even for technical tasks such as troubleshooting and maintenance, our sysadmin reported working with others about half the time.

The second most common category was planning (21%), which was often combined with testing (6%). Production systems had limited time windows when changes were permitted, so sysadmins planned and tested complex operations on testing and staging systems. Maintenance (19%), troubleshooting (11%), and installation (8%) tasks accounted for the remaining time.

3.2 Tools

Most of the sysadmins used laptops at their offices during regular hours as well as off-hours. Off-hours work was usually done at home through a dial-up or high-speed connection. For collaboration and communication, sysadmins relied on a standard set of common applications: telephone, web browser, email, and instant messaging. They also used common productivity tools such as a shared calendar, word processor, spreadsheet, etc. Sysadmins' primary source of information was the web. General purpose web search engines were the primary tool used to find online documentation, discussion databases, problem reports, and user groups. Other web-based tools were used for activities such as timesheet reporting, problem tracking and scheduling.

For sysadmin tasks, we saw tools supplied by systems vendors and third parties, as well as tools created by sysadmins themselves. These tools came in three types: command-line, including basic UNIX utilities; graphical system admin consoles, such as IBM DB2™ Control Center and Microsoft® Management Console; and web-based administration tools, such as BEA™ WebLogic™ Workshop and Oracle® Enterprise Manager Console.

The sysadmins seldom came into physical contact with the machines they managed, rather, they used terminal programs for text-based access and screen sharing tools to interact with remote computers graphically.

3.3 Environment

The sysadmins we observed worked in an environment of significant risk, system complexity, and system scale. Risk resulted from the critical nature of the managed systems. Significant system failures may have serious consequences, such as large sums of lost revenue and termination of employment. To compensate for risk, large installations typically have multiple levels of systems: sandbox systems that allow unlimited experimentation, but have no data; test systems that have sample data and applications; staging servers that permit relatively open access, and are exact replicas of production servers, to which access is highly restricted.

System complexity and scale was exemplified by sites with hundreds of servers that support large scale operations; numerous components such as database systems, web servers, etc. each requiring different expertise; configuration files with thousands of parameters connected by intricate interrelationships; event logs with millions of entries; and complex tasks requiring dozens of steps to perform.

The sysadmins in our studies were organized in teams of 5 to 12 based on specialty (e.g., database, web server, operating system). These teams depended on a team leader (often remotely located) for technical direction, and on a manager for cost issues and performance evaluation. Teams from each specialty worked together to support a number of customer accounts and applications, supported by managers responsible for system changes, system availability, and overall customer relationship.

4. OBSERVATIONS

Through our field studies we discovered patterns of sysadmin work and tool use. In what follows, we describe four cases that illustrate important aspects of sysadmin work practices.

4.1 Case 1: Seven People, One Command Line

Sysadmin George was assigned to create a new web server on a machine outside the corporate firewall and connect it to an authentication server inside the firewall on one of the customer accounts he supported.

His manager sent detailed instructions for the process, which included sample commands for over twenty steps to be performed under a very tight deadline. The first few steps for creating the new web server appeared to go well, but configuring the authentication server to work with the new web server produced a vague error message: "Error: Could not connect to server."

For the next few hours, George was involved in increasingly intense troubleshooting. Through telephone, e-mail, instant messaging, and in-person conversations, he worked with seven different people, including his manager, the network team, his office mate, the architect of the system, a technical support person, a colleague, and a software developer. Each asked him questions about system behavior, entries in log and configuration files, error codes, and so on, and each suggested commands to run. Each sought his attention and trust, competing for the right to tell him what to do (see [13]).

We refer to this collaboration pattern as "Seven People, One Command Line," as various people participated in troubleshooting, but only George had access to the troubled system (Figure 1). His manager wanted to know when the problem would be fixed and whether others should be redirected to help him complete the task on time. The support person wanted to resolve the problem ticket and end the call as quickly as possible. His colleague wanted to help within the limitations imposed by his own responsibilities. The system architect wanted to know if there was any problem in the overall design without being mired in the details. Other specialists waited for instructions to manipulate the subsystems they were responsible for.

The problem was eventually found to be a network misconfiguration. George misunderstood the meaning of a certain configuration parameter for the new web server (ambiguously labeled as "port") to be for communication *from* the web *to* the authentication server, when in fact it was the opposite. The former would have been permitted by the firewall, but the latter was not.

George's misunderstanding affected the remote collaborators significantly throughout the troubleshooting session. We witnessed several instances in which he ignored or misinterpreted evidence of the real problem, filtering what he communicated by his incorrect understanding of the system configuration, which in turn greatly limited his collaborators' ability to understand the problem. George's error propagated to his collaborators. The solution was finally found by the one collaborator who had independent access to the systems, which meant his view of the systems was not contaminated by George's incorrect understanding (see [13]).

4.2 Case 2: The Lost Semicolon

Christine, a database a

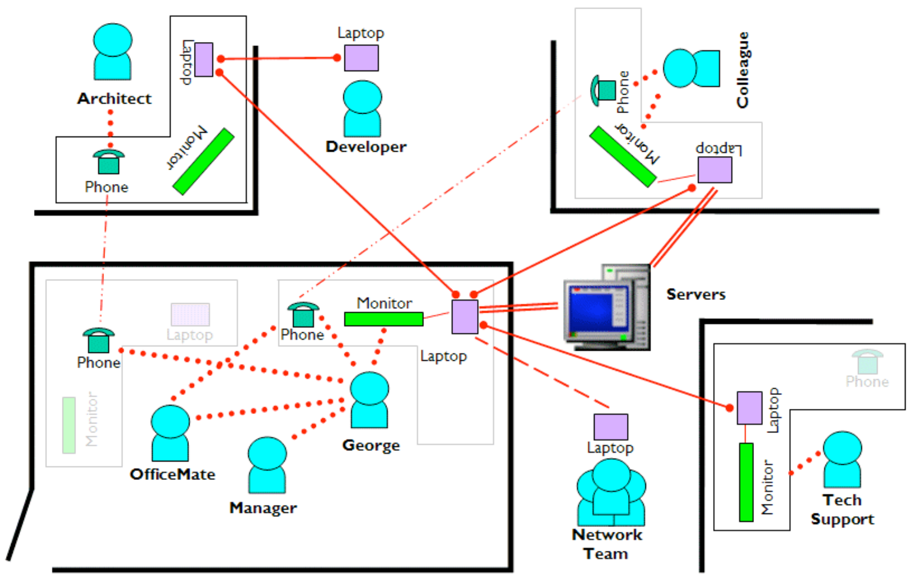


Figure 1. Throughout the long hours of troubleshooting, our admin George engaged with at least eight different individuals or groups using various means of communication. In this figure, instant message communication is shown in single solid lines; e-mail communication in dashed lines; phone conversations in dotted-and-dashed lines; and face to face communication in dotted lines. Also shown here in double solid lines is access to the problematic server, which only George and his colleague had during the session.

complex database operation that she had never done before. The task included moving database tables to a different file system on the production server to free up disk space. Because she had no experience with this task, her colleague Mike agreed to help. He had notes and executable scripts from the last time he had performed this operation. As the task involved production servers and a limited maintenance time window, they rehearsed the operations on three test servers first. Mike sat with Christine during rehearsal and verified each operation as she followed his instructions. The instructions included specific commands to run as well as notes such as "Check that the tables were created properly" at various points. After the commands and scripts were run on each test system, they were manually edited in a text editor to change such details as server names for the subsequent test.

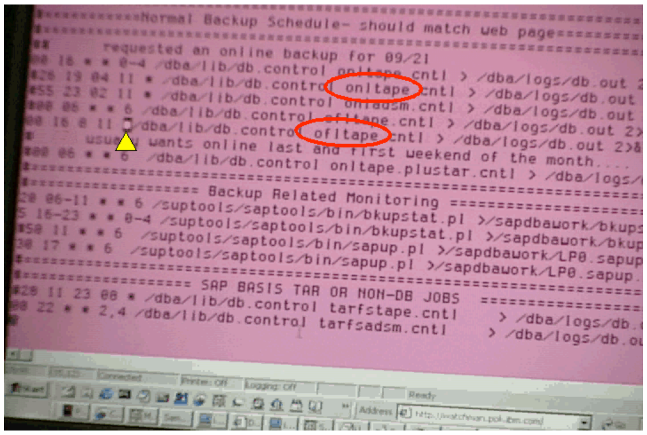
In the final rehearsal, errors appeared during the execution of one of the scripts because a semicolon that separated consecutive commands had been deleted accidentally when Christine edited the script. The script was aborted manually, but not before several commands in the script had run. In fact, they thought the script had created an incorrect database table, though it had not. When they tried to delete this (nonexistent) table, they received error messages that they interpreted as syntax errors. They looked up documentation and manually executed many different (and risky) commands to delete the table. It took them quite a while to realize that the table had not been created in the first place.

4.3 Case 3: Crontab as User Interface

Jeanette and Bob, both database administrators, needed to perform an online backup operation to store all database contents while the production database system continued to serve requests. As such backup operations take quite a while to run, they developed the practice of performing this kind of task using the UNIX *crontab* utility, which is normally used for executing regularly scheduled tasks as background processes that continue to run even after the terminal connection has been closed.

As usual, Jeanette edited the crontab file, which contained the list of scheduled commands they frequently ran. She uncommented the line for the backup command, and set the execution time to be exactly one minute after the current time. She then saved the file and the crontab utility automatically started running scheduled tasks. As soon as she saved the file, she realized that she had accidentally uncommented the line for an *offline* backup, which would shut down the database system, rather than an *online* backup, which would not (Figure 2). She immediately opened the crontab file again to remove the wrong command and save it within the short one-minute time window before the backup was scheduled to start. Having done this, she needed to check to make sure the incorrect backup process had not in fact started. She did so by rapidly executing various process status commands and by checking the server logs on Bob's suggestion.

This time, Jeanette was lucky; the offline backup process had not started. She told us that she usually gave herself more than one minute before issuing such commands to allow time to catch these sorts of errors. In this case, the problem was that the commands for offline and online backup were right next to each other in the crontab file, and the two commands differed only by one character (*onltape* vs. *<u>ofltape</u>*). In addition, Jeanette and Bob had been discussing a different offline backup task at the same time, possibly leading to confusion.



```
#************Normal Backup Schedule- should match web page==========
#*******************************************************************
##         requested an online backup for 09/21
00 16 * * 0-4 /dba/lib/db.control onltape.cntl > /dba/logs/db.out 2
#26 19 04 11 * /dba/lib/db.control onltape.cntl > /dba/logs/db.out
#55 23 02 11 * /dba/lib/db.control onladsm.cntl > /dba/logs/db.out
#00 06 * * 6 /dba/lib/db.control ofltape.cntl > /dba/logs/db.out 2>
00 16 0 11 * /dba/lib/db.control ofltape.cntl > /dba/logs/db.out 2>&
#      usually wants online last and first weekend of the month....
#00 06 * * 6  /dba/lib/db.control onltape.plustar.cntl > /dba/logs/
#===================================================================
#==================== Backup Related Monitoring ====================
20 06-11 * * 6 /suptools/saptools/bin/bkupstat.pl >/sapdbawork/bkups
5 16-23 * * 0-4 /suptools/saptools/bin/bkupstat.pl >/sapdbawork/bkup
#50 11 * * 6  /suptools/saptools/bin/sapup.pl >/sapdbawork/LP0.sapup
30 17 * * 6  /suptools/saptools/bin/sapup.pl >/sapdbawork/LP0.sapup.
#===================================================================
#==================== SAP BASIS TAR OR NON-DB JOBS =================
#20 11 23 08 * /dba/lib/db.control tarfstape.cntl    > /dba/logs/db.
00 22 * * 2,4 /dba/lib/db.control tarfsadsm.cntl    > /dba/logs/db.ou
```

Figure 2. Jeanette's cursor was on the wrong line of the crontab file, setting execution time for offline backup rather than for online backup, which was located two lines above.

4.4 Case 4: The Crit Sit

"Crit sits" are *critical situations* that are initiated when a customer is extremely unhappy with the service level of a system. All responsible parties are brought together either physically and/or virtually to work on the problem until it is fixed. We observed a crit sit for an intermittent problem with a web application that occurred across multiple systems when the number of connections to the application reached a certain level. In this case, the crit sit would be considered resolved if the system ran for three consecutive days with no problems.

Crit sits are very costly, as at least one member of every relevant specialty team is required to participate. In this case, it involved seven to ten people at various times, and each person also had backup support. Typically these sysadmins converge on a central location and sit together in a "war room" for the duration of the problem. One sysadmin even flew in from his regular location on 8 hours notice. Of course, for practical reasons, not everyone can be in one location all the time, so conference calls and instant message sessions or "chat rooms" are set up to keep everyone in contact. The "all hands" chat room was used for general awareness as the situation developed, such as reporting the current number of connections at all times.

A number of times we observed all sysadmins stopping to focus on collecting more data when the connection reached the maximum and the application stopped responding. To automate this process and to analyze the data, the sysadmins decided to write a script that would report the number of connections and the time. They planned to use this report to correlate connection data with log data from other systems. Though this was a rather simple script to write, it took longer than expected and eventually turned into a competition among sysadmins as to who could write it most quickly and most elegantly. Interestingly, everyone had difficulty creating the desired output where they tried to put the time and the connection count on the same line in the report.

Throughout the crit sit, multiple discussions went on simultaneously, with occasional interruptions when the connections "maxed out" and the different sysadmins performed tasks in their own areas of expertise. At a number of points, finding the right person to do the tasks became an issue as backups and primary sysadmins traded places and joined in and dropped out of the "all hands" chat room or the conference call. In one case, a sysadmin wanted to test his hypothesis and needed a database administrator to shutdown the database server. He first called her name on the conference call, when no one replied he then tried the chat room. Fortunately, her backup was found eventually and instructed to perform the shutdown.

Troubleshooting intermittent multi-system problems is a time-consuming task, as it is extremely difficult to collect all the information from all the systems involved at the right time and collectively make sense of it. This crit sit continued for two weeks after we left.

5. DISCUSSION

We observed that available sysadmin tools do a relatively poor job of supporting sysadmins in several important areas. Specifically we saw (1) collaborative work hampered by lack of tools for sharing system state; (2) problems caused by poor situational awareness, as complex systems require awareness of different levels of detail at different times; (3) lack of support for planning and rehearsal causing problems or delaying problem resolution; (4) both graphical user interface (GUI) and command-line interface (CLI) tools exhibiting deficiencies in supporting sysadmin work, requiring sysadmins to build their own tools; and (5) sysadmins spending hours maneuvering through complex information, tools, and human barriers to get the job done. We discuss each in turn.

5.1 Collaboration and Communication

The first case, Seven People and One Command Line, demonstrated how sysadmins rely extensively on human-human interaction and communication to accomplish their jobs. We analyzed this case in detail, and found that 90% of the troubleshooting time was spent in human-human interaction (via telephone 44%, instant messaging 23%, and face-to-face 22%) and only 9% of the time was spent in human-computer interaction (via the command line 6%, and web browser 3%). Furthermore, more than 20% of the time when people were communicating, they were talking about *how to communicate* with one another (e.g., "I'll send you that information via e-mail, what is your address?" or "You should call tech support number at 1-800-…").

Collaboration was one way to manage risk, system complexity, and system scale. No single individual understood every aspect of systems, so having a group of experts helped. In addition, systems required 24×7 coverage, so handoffs had to be made with shift changes. Lines of responsibility were carefully defined so that different people have control of different subsystems. For example, database administrators controlled the data schema, indices, and storage, but application administrators controlled the data content, and OS administrators controlled the server machine. There was also a division in focus in which the sysadmins were responsible for technical details, managers were responsible for schedules and customer satisfaction, architects were responsible for overall design, and so on. Finally, collaboration was required even among those with similar skills and responsibilities because a "second pair of eyes" was often needed to help build an accurate view of the problem and its solution, as seen in both Seven People and Once Command Line Case and the Lost Semicolon cases.

Tools provided uneven support for sysadmins' collaboration and communication needs. We saw that telephone, instant messaging, and email were heavily used, and we were told that pagers, cellular phones, and two-way radios were also used at times. These tools were quite dependable, but there were three areas that were clearly lacking. First, no single medium worked for all situations. We commonly observed sysadmins changing media; for example, moving from instant messaging to telephone when important information needed to be communicated rapidly with emotional force, shifting back to instant messaging when a cryptic error message needed to be conveyed precisely, and then shifting to email to send a large log file. However, the different media were not well integrated: phone numbers were sent by instant messages, telephones had limited channels and were sometimes busy, and electronic meetings required further coordination of meeting codes and passwords.

Another problem with available tools was that they did not provide for easy sharing of system state and other context. As seen in the Seven People and One Command Line case, remote collaborators received all their information about the problem from George verbally, so their information was filtered by George's misunderstanding of the system's operation. This case clearly indicates that although improved networking and communications technology increased the social interactions in the work place, misunderstandings are still commonplace [16]. If the system administration tools allowed others to actually view and interact with problematic systems (with proper approval and authentication), they would have been better able to assess the problem. Likewise, it was not easy to gather all relevant data for a problem together and share it. Logs, configurations, commands, and errors, each had to be found, composed, and sent. No sysadmin tools we observed had such a "work together" feature.

5.2 Planning and Rehearsal

As the Lost Semicolon case showed, planning and rehearsal were important work practices. Sysadmins worked with production systems that could not go down except during narrow time windows of scheduled maintenance. Though brief system failures might have been tolerable, loss of data was never acceptable. We observed that most actions were carefully planned and rehearsed before they were performed on production systems. Database administrators had the most extensive planning and rehearsal procedures, but we observed web admins also doing considerable planning before making system changes. Rehearsals not only gave sysadmins opportunities to demonstrate correctness of operations, but also practice at solving problems and timing steps so they could be sure the task would be accomplished during the allotted time window.

How well did existing interfaces and tools support rehearsal? We did not observe the use of any tool that seemed particularly suited for planning and rehearsal. As the Lost Semicolon case showed, manually editing command-line scripts as they move from system to system was hazardous. The database tools provided no means to avoid manual editing and, even worse, provided no way of syntax checking a script without running it against the database. Vendor provided graphical tools in this case were not designed for replicating actions on different systems.

An important part of rehearsal was logging each step of the procedure so that system output could be compared between rehearsal and production runs. Recording the time of long-running steps was important both as a check on correctness and for estimating the time of the production run. Rehearsal tools ought to help track timing and output information, both within a given activity as it works its way through rehearsal and across activities so that similar operations done later can take advantage of previous runs.

5.3 Situation Awareness

System complexity put substantial cognitive load on sysadmins as they were troubleshooting systems by coordinating information from many sources and many people. The Crit Sit case is an excellent example of sysadmins dealing with dynamic and complex processes at many different levels of abstraction. Radical co-location setups such as this allow participants easy and committed access and to coordinate activities around shared events and work artifacts [23]. In this case, participants had to be aware of systems that were not only complex, but that also changed frequently. Furthermore, sysadmins had to share situation awareness across shifts and areas of responsibility. Sysadmins necessarily had incomplete mental models of the complete systems they managed. As one sysadmin put it, "If understanding the (whole) system is a prerequisite for operating the system, we are lost."

The Seven People and One Command Line case provided another example of the need for situation awareness. In that case, situation awareness depended on understanding the interaction between several components in an overall system. Each system had its own management interface and so gaining overall awareness was very difficult. George managed this complexity by rapidly moving among multiple management tools and working together with many experts, but there was no single view of the entire system. A simple drawing of the configuration might have made the situation clear and avoided hours of troubleshooting.

The Lost Semicolon case provided further examples of problems caused by faulty situation awareness. The sysadmins tried to delete a database table they thought they had incorrectly created, but which in fact they had not. In this case, their command-line environment did not aid situation awareness. In contrast, a graphical tool for managing the database would likely require clicking on a table icon to delete it; if the icon had not been there, it would be obvious that there was nothing to delete. Of course, in an environment containing thousands of tables, presenting so many icons would not be simple, which is a separate but related issue in attaining situation awareness. There must be a dynamic middle ground between lack of information and information overload.

5.4 Tools and Tool Building

The very existence of sysadmin-authored scripts might seem to be evidence that the supplied tools were at times inadequate. However, customization and automation seem a normal part of sysadmin work. Professional tool designers simply cannot foresee all possible tasks, needs, and requirements. Many tools failed to support sysadmin needs in the areas of scale, complexity, and risk of the operations, as noted in the Lost Semicolon and Crontab as User Interface cases. Sysadmins often applied long-running operations to very large numbers of objects, making automation and scripting crucial. Most GUIs we observed fail to support this. CLIs offer more power, but with less ease of use and situation awareness.

One side benefit of sysadmins building their own tools is that they know and trust the resulting tool: Sysadmins know that the *ps* command in their scripts correctly reports the process status whereas a "running" status light on a graphical interface might only mean that the monitoring process crashed, as we observed in a number of cases.

At one site, we observed a database administrator who had developed a set of monitoring scripts that periodically gathered data from a large number of databases, creating web pages with status reports and triggering alarms when certain criteria were met. These scripts were used across the organization, but responsibility for maintenance rested with the one administrator. A web administrator at another site had configured a similar system: a program that regularly checked various servers, sending e-mail or pager messages in case of errors. However, most of the sysadmins we observed did not have the skills or time to build such an environment.

As shown by the Crontab as User Interface and the Crit Sit cases, sysadmins used scripts to automate monitoring of system health, to perform operations on a large number of systems, and to try to eliminate errors on common tasks that take many steps. Other sysadmins we observed kept a directory of short scripts and commands called something like "my favorite commands" to rerun commands that once worked for them. Typically, such commands and scripts were shared with other sysadmins who modified them to fit their particular environments.

One difficulty in script writing was error handling. When handled incorrectly, errors may leave the system in an inconsistent state. The Lost Semicolon case described a procedure punctuated by instructions such as, "Check that the tables were created properly." It was left up to the sysadmin to *know* what commands to use to check the tables. Furthermore, when asked about this, we were told that that was when sysadmins would "show their stuff" because errors called for ingenuity and creativity.

Another difficulty with script writing was managing input and output. Most scripts we observed were command-line tools that took few if any parameters. Scripts were not flexible; each did one particular task using several hard-coded constants, as in the Lost Semicolon case. Output was often nothing but the concatenated output of each command in the script, which was often so verbose that errors might not be observed.

Within the groups of sysadmins we observed, there were various levels of expertise, including some "script writers." Though most sysadmins could put together a handful of commands into simple shell scripts to execute a series of commands at once, script writing requires understanding basic programming concepts, which our survey and interview data show is not necessarily a skill shared by all sysadmins (only 35% reported having a bachelor's degree in computer science). Moreover, most software development tools are meant for programmers and are intended to support large scale development, not the less formal style of sysadmins. The sysadmin work environment we saw was very different from that required for the design-development-test cycle of software developers. For the sysadmin, rebooting and starting over was usually not an option.

5.5 Multitasking and Diversions

Because of the nature of their environments, the sysadmins we observed had a complex interleaved workflow with multiple tasks conducted in parallel, yet their workflow was often diverted because of missing information, unfulfilled prerequisites, broken tools, or required expertise. Multitasking was particularly an issue for sysadmins as they maintained a large number of long-running tasks, while trying to be very efficient overall. When tasks were loosely related, multitasking seemed to work without much trouble. When tasks were too close, however, problems such as the offline versus online confusion in the Crontab as User Interface case occurred.

Multitasking was nicely supported in terminal sessions, which can run tasks on multiple open sessions simultaneously and support quick switching among tasks. The history of a terminal session was of further help for sysadmins, reminding them of the context of their tasks as they could see previous commands along with their output. The GUI tools we observed were less supportive of multitasking since most were not designed for rapidly switching between different system contexts, and none displayed a history of past commands. For example, the database management GUI did not allow multiple simultaneous system views, but required multi-step navigation to switch between viewed components.

Diversions were a common and expected part of the sysadmin work. Our analysis of computer sysadmins solving problems during routine work suggests that much troubleshooting centered on tools, infrastructures, environments, and other people that were not directly related to the problems at hand, but that had to be dealt with nonetheless. That is, while solving specific computer system problems, administrators often solved problems that arose outside the scope of the initial problems themselves. For instance, when trying to fix a misconfigured web server, we observed an administrator needing access to the server machine, which in turn required finding the person responsible for controlling access and convincing that person to grant permission, as also shown in the Crit Sit case. Though the original problem concerned software configuration parameters, the solution required dealing with systems and people that were not in the space of configuration parameters. And this was not an isolated incident: Observational data from three troubleshooting episodes showed that about 25% of time was spent on these sorts of diversions.

6. CONCLUSION

Our studies focused on people and practices to find opportunities for supporting work through appropriate design and technology. As illustrated in the cases, our main findings are (1) collaboration was a primary activity among sysadmins; (2) sysadmins worked in an environment that was very complex, both technically and socially; (3) sysadmins spent significant time planning and rehearsing; (4) sysadmins were at once system users, builders, and repairers who relied on technical, social, and organizational skills; (5) sysadmins often found themselves "off the trail", diverted from their tasks because of missing information, broken tools, or needed expertise. Furthermore, the tools available to sysadmins do not support their work practices in these areas.

How can we design more effective tools for sysadmins? This question becomes really critical as new paradigms such as Autonomic Computing are being put forward as solutions to the manageability of complex software architecture (see [10]). First, an important task will be to structure our understanding of these workers and their work in many different settings [1]. Our studies focused on web and database sysadmins in large corporate information technology departments, and included analysis of tool use (see [3]) and collaborative problem solving activities (see [13]). We are well aware that the situation is different in smaller businesses where a single sysadmin has to manage many different kinds of systems. And there are probably many things to learn about storage administrators, network administrators, operating system administrators, and hardware administrators who work with physical rather than virtualized technology.

Second, we must translate our findings into concrete guidelines that transform real technologies and processes that help sysadmins. There are two clear dangers here. The first is that sysadmins already have many more tools than they can manage. The products they use come with tools. Third parties produce alternative tools. Home-brew tools add to the mix. In fact, every new script they write might be considered yet another tool. Adding lots of little tools will probably not help. We expect that rethinking the larger structure of sysadmins' relationship to the systems they manage in an integrated environment will lead to the most useful results.

A second danger is falling into the CLI vs. GUI debate. Many sysadmins are quick to proclaim that CLIs are good and GUIs are bad. However, this oversimplifies the issues. After all, web browsers, email, instant messaging, and even terminal emulators are valued GUI applications. From our studies, we have begun to develop a list of qualities that good sysadmin interfaces have. Whether GUI or CLI or both, successful tools will be fast, truthful, scalable, and scriptable. There is much to be done to develop interface components that scale to the complexity, risk, and number of computer systems that sysadmins manage. We must think carefully about fundamental issues, such as recall vs. recognition, reversibility and complexity of actions, immediacy of feedback, and presentation of information.

7. ACKNOWLEDGMENTS

Thanks to Christopher Campbell, Steve Farrell, and Anna Zacchi for their help in conducting field studies and the sysadmins and their managers for their time and support. We also thank Jeanette Blomberg, Beverly Harrison, and Christine Halverson for many helpful comments on an early draft.

8. REFERENCES

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

[2] Barrett, R., Chen, M., & Maglio, P. P. System Administrators are Users, Too: Designing
Workspaces for Managing Internet-scale Systems, *CHI 2003 Workshop.*

[3] Barrett, R., Maglio, P. P., Kandogan, E., Bailey, J., Usable Autonomic Computing Systems: the Administrators' Perspective, Proc. ICAC'04 – International Conference on Autonomic Computing, 2004.

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

[5] Fitzpatrick, G., Kaplan, S., Mansfield, T. Physical Spaces, Virtual Places and Social Worlds: A Study of Work in the Virtual, Proc. CSCW '96 – *Computer Supported Cooperative Work*, 1996, pp. 334-343.

[6] Halprin, G. The Workflow of System Administration, *SAGE-AU '98*, Canberra, Australia, 1998. http://www.sage-au.org.au/conf/sage-au98/

[7] Halverson, C. A., The Value of Persistence: A Study of the Creation, Ordering and Use of Conversation Archives by a Knowledge Worker, Proc. 37th Annual HICSS'04 – Hawaii

International Conference on System Sciences, 2004, pp. 40108.1

[8] Hrebec, D. G., and Stiber, M. A Survey of System Administrator Mental Models and Situation Awareness, *Proc. ACM SIGCPR – Computer Personnel Research*, 2001, pp. 166-172.

[9] Hutchins E. (1995). *Cognition in the Wild*. Cambridge, MA: MIT Press.

[10] IBM, "Autonomic Computing: IBM's Perspective on the State of Information Technology"; http://www.ibm.com/industries/government/doc/content/resource/thought/278606109.html

[11] Kephart, J. O., Chess, D. M. The Vision of Autonomic Computing, *IEEE Computer*, January 2003, 41--51.

[12] Luff, P., Hindmarsh, J., Heath, C. (1999). *Workplace Studies: Recovering Work Practice and Information System Design*. Cambridge, MA: Cambridge University Press.

[13] Maglio, P. P., Kandogan, E., & Haber, E. (2003). Distributed cognition and joint activity in collaborative problem solving. In *Proceedings of the Twenty-fifth Annual Conference of the Cognitive Science Society*. Boston, MA. LEA.

[14] Orr, J. E. (1996). *Talking About Machines: An Ethnography of a Modern Job*. Ithaca, NY: Cornell University Press.

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

[16] Rogers, Y. (1992) Ghosts in the network: distributed troubleshooting in a shared working environment. Proc. CSCW'92, pp. 346-355.

[17] Sandusky, R. J. Infrastructure Management as Cooperative Work: Implications for Systems Design, *Proc. ACM Group '97 – Conference on Supporting Group Work*, 1997, pp. 91-100.

[18] Sheridan T.B., *Telerobotics, Automation and Human Supervisory Control*, MIT Press, Cambridge, MA, 1992.

[19] Shneiderman, B. Empirical studies of programmers: the territory, paths, and destination. *1st Workshop on Empirical Studies of Programmers*, 1986, pp. 1-12.

[20] Shneiderman, B. Designing the user interface: Strategies for effective human-computer interaction, Chapter 8, Addison Wesley Longman, 1998.

[21] Sproull, L. and S. Kiesler, *Connections: New ways of working in the networked organization*. 1991, Cambridge, MA.: The MIT Press.

[22] Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge: Cambridge University Press.

[23] Teasley, S.D., Covi, L., Krishnan, M.S., & Olson, J.S. (2000). How does radical collocation help a team succeed? Proc. *CSCW'00*, pp. 339-346.