The following paper was originally presented at the
Seventh System Administration Conference (LISA '93)
Monterey, California, November, 1993

Guerrilla System Administration: Scaling Small Group Systems Administration To A Larger Installed Base

Tim Hunter & Scott Watanabe
University of Colorado at Boulder

Guerrilla System Administration: Scaling Small Group Systems Administration To A Larger Installed Base

Tim Hunter & Scott Watanabe – University of Colorado at Boulder

ABSTRACT

Many university sites have a large number of machines that consist of a variety of platforms and operating systems. Because of limited budgets, many use a small number of full-time systems administrators supervising a group of undergraduates to administer the campus network. The usual solution is to have the full-time administrators act as managers for a pool of junior and senior undergraduates. The junior administrators learn on their own, with limited help from the senior administrators and managers. This model works well for a small group of undergraduates who can remain in close contact. However, recent increase in demand for system administration requires hiring more undergraduates, causing the small group system to break down – junior administrators do not get as much contact with senior administrators and management, and some lose touch altogether.

This paper presents modifications to the small group model intended to improve training of junior administrators, and associated changes to improve customer service. The driving idea is to give undergraduates as much authority as they can handle responsibly. This means making current undergraduates a part of the interviewing process for new administrators; taking advantage of the benefits of the small group model by forming small groups of junior administrators lead by senior undergraduates, who receive minimal guidance from management; and creating standing orders that extend the authority of a staff member in a crisis situation. These improvements, as a whole, have worked very well. New hires take less time to become part of the team; small groups of undergraduates provide better support for junior administrators, and free full-time administrators from dealing with the details of a large project; competent junior administrators are able to handle most situations without help.

Introduction

UnixOps is a division of the University of Colorado at Boulder's Computing and Networking Services department. Our task is to provide network and system administration services for UNIX systems to paying customers, most of whom are in the College of Engineering. We also receive general fund money to perform a number of campus-wide services including BIND, mail, source access, door card access, and USENET news. We also maintain host, user ID, and mail alias databases for the campus. Our operation is more like a ''real-world'' business than an exclusive service-provider whose income only comes from one source. However, we do operate in both realms.

UnixOps has been in existence in one form or another since 1981. It was created to maintain a few workstations in the Engineering Center, and has grown exponentially since then. About five years ago, there were approximately as many staff members as there were machines to administer. Now we provide full support for six large computing labs and twenty-five smaller labs and stand-alone machines – roughly two hundred machines in all. We provide network support to fifteen other hosts. While most of the machines are Sun workstations, we also administer DECs running Ultrix, Silicon Graphics, and one HP Snake. Most of these machines are in the engineering center, but there are also hosts in several other buildings on campus: math, geology, economics, and physics.

Currently our staff consists of three full-time professionals, who split administrative and management duties between them. We have a systems staff of fifteen students, along with three students to do tape backups and one to assist with administrative work. We further classify the systems students into ''seniors'' and ''juniors'' (not to be confused with that student's status in school). The former have usually worked at UnixOps for more than a year, and can handle nearly all tasks without supervision or assistance. The junior administrators are still on the learning curve – they may have worked nearly as long as the senior administrators, but still require some assistance (or at least assurance that their solutions are valid) on most assignments.

Small Group Systems Administration

Though the staff has recently expanded, the proportions – one professional for every five students

– have remained the same. Until a year ago, our staff consisted of two full-time administrator-managers, between five and ten systems students, two or three operators, and an administrative assistant. The working model that developed, which we refer to as the small group model, has three main areas that apply to systems administration: the training of systems administrators, daily management (or daily survival), and how we get along with our customers. For the most part, the policies and procedures from the small group model are still in use.

Training

UnixOps does very little formal training of new administrators. This is in part because it's difficult to allot large chunks of a staff member's time to prepare and present classes. Another part of the problem is that system administration is not something that can be easily taught in a classroom environment – the problems are too varied and the answers seem to change constantly.

The main thrust of our training is familiarizing a new employee with the mechanics of the job – answering the phone, using Rand's MH and the Trouble MH [1] system for responding to customer requests – modifying campus databases and forwarding phone messages. We also ask that they read a beginning systems administration book – usually Nemeth, et al. [2]. These books don't often help the system administrator deal with day-to-day problems, but they provide the background necessary to understand the solutions presented to them by current staff.

Further training is provided in two ways – the first by occasional seminars presented by a manager or senior student, and the second by one-to-one instruction. The latter takes on many forms. Often we present new administrators with a problem we believe they are capable of handling, let them figure out as much as they can on their own, and then assist them out of dead ends, or confirm the correctness of the solution they have chosen.

Occasionally a senior staff member will walk through a particularly complicated process, such as installing C News, with a junior administrator. For the most part, training consists of making junior admins confident enough to tackle day-to-day problems, and answering any questions they come up with. Sometimes the answer is ''have you read the man page?'' However, one of the few UnixOps policies is that all questions from staff members take priority. The response may be just a pointer to where the answer is found, but there must be a clear, courteous response.

Survival Techniques

Many elements help our day-to-day operations run smoothly. This does not mean that things always run well, just that things would be a lot more chaotic if we didn't have these tools and policies.

Our central tool is the trouble queue – this is the Trouble MH system which has been in use at UnixOps for at least four years. This system allows students to control their own workload, and allows the staff to monitor progress on problems and offer corrections or advice as necessary. It also provides an audit trail, and to some degree, a knowledge base, as all request messages and subsequent responses are logged. Any status report or resolution on a trouble mail message is mailed out to the trouble mail readers. This keeps current work in front of all staff members, allowing them to better answer customer questions, and providing a measure of quality control – if one person makes a mistake, there are ten people who might catch it.

Occasionally a problem will arise with the ''catching'' of mistakes – one student's message highlighting the mistake of another may appear to be more of a personal attack than a professional criticism, regardless of the intention of the sender. However, these problems are usually worked out quickly, either by a face-to-face discussion between the two parties, or by a temporary moratorium on ''flaming''[1] in a trouble mail mistake message.

Our most important policy, that of giving priority to questions asked by employees, has already been mentioned. This does not mean that a customer should be interrupted, but that non-life-or-death work should be put down to answer the question of another staff member. This may mean that senior students and managers often get interrupted, but in the long run more work gets done. The question will have to be answered eventually, and if it is answered immediately, the junior administrator won't sit around idle. This policy also keeps junior employees from being frustrated because they have nothing to do, having reached the limit of their knowledge.

Another implied policy is that of keeping the office home-like. This means keeping a microwave and refrigerator around, along with occasionally bringing in food for everyone (Elizabeth Zwicky mentioned chocolate chip cookies [3] – we find that muffins and bagels tend to keep the administrators from sticking to the walls. We do break down and have donuts every once in a while). It means realizing that school, in the long run, is more important than work, and being flexible to accommodate that. It also means allowing students to use the workstations for schoolwork when they aren't doing office hours, and allowing them to install software needed for schoolwork on work machines. The office is on a card access system, so that students may use the office at any time.

[1] flaming: term defined by USENET to mean excessive or gratuitous abuse.

A refrigerator and a microwave aren't that impressive – many workplaces have such amenities. However, the concepts of using work resources for school and bending schedules to accommodate school are somewhat unusual. The latter is fairly easy to justify – the goal of the university is education, and it wouldn't make sense for our organization to get in the way of that. This is best practiced in moderation – a staff member who is signed up for 18 credit hours and continually bows out of his 20 hour/week office hour commitment needs to either cut back on his office hours or drop a class.

This sort of unrealistic scheduling has not been a problem. Students set their own work schedules at the beginning of the semester, including the total number of office hours they will commit to. It only takes eight students working ten hours per week to have two students in the office during normal business hours. If there are more than eight students, or if those eight students work more than ten hours/week, the week gets covered without the manager having to assign office hours to fill the gap. On the occasion that a gap does occur, there are still full-time staff around to cover it.

Our reasons for allowing students to use office workstations for school are fairly simple. Most of the CPU time that is being used for schoolwork is time that would otherwise be spent idle. There are times when work and school related tasks are occurring at the same time on one machine. However, the school related tasks usually do not add a noticeable load to the machine. It is conceivable that a student might slow down a machine during business hours with a large compilation, but this has not occurred. Part of what makes a good systems administrator is courtesy and a sense of responsibility – someone with these qualities should have sense enough to save the large CPU jobs until after hours, or perform them on a non-work machine. Mis-use of company workstations has not been a problem.

In fact, mis-use of the office in general has not been a problem. Part of the solution is that we think of our office as a learning environment, and encourage students to spend as much time there as they like. In a very real sense, the more time that someone spends on a UNIX workstation, the more she[2] is going to learn about the system – making her a better systems administrator.

Not only does a pleasant and useful office provide more resources for the education of the students working there, and increase their systems knowledge, but it often results in problems being solved after hours – work that we can not afford to pay someone to do. Someone who is working late at night on a project is probably going to be willing to answer the phone, or walk upstairs and spend half an hour fixing a broken card access system, instead of letting the problem sit until the morning. Someone who has stopped by the office to read news might get curious about that persistent ''huge ethernet packet'' message on the netlog[3] and spend time investigating. Someone who can just eat a bag of microwave popcorn instead of having to go home to eat will probably stay late enough to finish a large job that would have taken longer if done in parts. In general, these fringe benefits don't cost much, and the work and expanded knowledge base that is gained would cost hundreds of dollars if paid for by overtime.

Customer Relations

Machines being fixed during off hours doesn't hurt our customer service image, either. However, we do not depend on this, or the fact that happy employees usually make happy customers, to be the basis of our relations with our customers.

Unfortunately, we cannot always keep our customers happy by fulfilling their requests quickly and completely. To do so, we would need three times the staff, and three times the budget we have now. However, we can at least keep them informed – let them know when projects will be completed, how close they are to completion, and when machines will be down. To this aim we have set up a distributed message of the day (motd) system that fits in with our aliases, hosts, and networks distribution system. In this way we can keep all of our customers up to date on network outages and other campus-wide downtime. For some of our customers, being informed is all they need to be happy.

We try to keep individual customers informed by rapidly acknowledging the receipt of trouble mail messages, and by keeping our service definitions available via anonymous ftp. The latter includes our ''Localization Checklist'' – the list of network settings and software that we install on a machine to get it integrated with the campus network. This checklist also resides in the root directory of every machine which we set up, and a customer can read this checklist to see what has and has not been completed. We also maintain an alias on each machine that points to the technical and administrative contact for that machine, so that even if a staff member does not know who owns a machine, he can still inform the owner of any changes made to the system. Another alias, ''diary'', goes to a file on the machine to log important changes for future administrators.

[2] In an attempt to be politically correct and still use proper english, we have tried to make equitable, yet consistent, use of gender-specific pronouns.

[3] We have all machines that we administer report errors to a central host. This host allows anyone logged in as the user ''netlog'' to see these error messages as they arrive.

Finally, we have a set of pagers that rotate among student staff members. The number is listed in our voice-mail message for emergency use, such as a downed card access system or a machine on fire. However, the effectiveness of this system is limited by the fact that we cannot obligate our staff to appear on-site whenever they are paged – doing so would cost too much in overtime. While the staff member carrying the pager is obligated to make a phone call, the decision to go to the workplace is left entirely to the staff member carrying it. Yet most problems can be solved over the phone (especially the ones called in by fellow staff members). By rotating the pagers on a weekly basis, and keeping the numbers somewhat hard to find, (thus limiting the number of people ''crying wolf''), most of the pages get resolved without burning out a particular employee. Most pages are made by junior staff members or operators who encounter problems they can't resolve.

Sometimes user relations also means dealing with uncomfortable situations – most notably the situation of being asked to fix something which was broken by a user with root access, as part of a full-service contract. This is somewhat analogous to restoring a file for a user who has accidentally deleted it, as opposed to restoring a file lost due to hardware problems. Thus our policies are also similar: we will restore for free a file lost due to ''natural'' causes, but charge per hour for a file deleted by a user. We will also fix, for free, a system that we ''broke'', or was otherwise not caused by a user with root access. Yet even for customers who have a full-service contract, we charge per hour to fix other people's messes. Fortunately, this policy has never been enforced.

Expanding the Scale of Small Group Administration

These policies and concepts worked quite well until about a year ago. On the whole, customers were reasonably pleased with our performance, and the staff felt that their jobs were important and enjoyable. However, after a complete changeover in management, the doubling of student staff drove the organization a lot closer to chaos. The two new managers became swamped with learning how things worked, and the students who still needed guidance got lost. The time it took to respond to customer requests started going up, and some difficult problems were put aside indefinitely.

Hence the need to expand upon the small group model – the main reason being to provide better support for the junior staff, who by now outnumber the managers and senior students combined. There are two primary goals for this reorganization: the first is to make the organization a cohesive unit, as it was before the expansion. The second goal is to train new administrators faster – to accelerate the normally quite slow process of bringing new administrators to the point where they can handle most problems without help. Our main intention is to expand the small group model, as opposed to replacing it altogether. This should keep the changes minimal enough to be easy to implement while still accomplishing something.

Additional Survival Techniques

The biggest change is the addition of another level of responsibility that formalizes the status of senior students as mentors: more accessible than the managers, and able to answer most questions. Each senior student is placed in charge of a small group of (three to four) junior students. The senior student becomes a team leader – someone who not only acts to organize and lead the group, but also takes a fair share of the work. This frees up time for the managers, provides leadership experience to the senior students, and increases dissemination of knowledge to the junior administrators. The team leader does not replace the managers, but is instead added to the junior administrator's quick list of people to turn to for help.

Taken as a unit, these groups are small enough to re-create the small group conditions. The team leader takes the responsibility of not only planning and working on the team projects, but also increasing the knowledge of her team members. Our belief is that systems administration cannot be learned in a classroom. Even small seminars only serve to give the attendees an introduction to the topic. Ninety percent of the knowledge about that topic comes from hands-on exploration. (with the manual or a knowledgeable party by his side) on the part of the learner. Thus a senior assigned to four junior administrators should serve to increase the amount of time a team member can be learning directly (or indirectly) from another person. We also assume that, as in most tutoring situations, the tutor will receive some knowledge from the tutored.

The team leader not only learns from her team members, but also learns about her team members, and is thus able to help her team members choose tasks that are challenging, but not so difficult that the junior administrator will either fail to complete it, or require hand-holding the whole way through. This should also speed up the junior administrator learning process.

The second aspect of our improvement program involves extending the ''staff empowerment'' aspects of small group administration. The premise is that if a student is responsible enough to be a systems administrator, she is also responsible enough to select tasks that she will be able to complete, and assign her own office hours. Large group administration also assumes that the student knows enough about systems administration to know what qualities, aside from prior knowledge, make a good systems administrator – and thus we have made staff interviews part of our hiring process.

We hope that this will benefit us in two ways. First, staff interviews will allow current staff to determine how likely the interviewee is to fit in well with the organization. If a manager is still undecided about a particular applicant, the impressions of current staff can be very important. The second foreseeable benefit is that, if hired, the applicant will take less time to fit in and to feel comfortable interacting with the rest of the employees.

Another aspect of staff empowerment is the creation of ''standing orders'' – policy statements which give student administrators permission to bypass normal policies in an emergency. For instance, we cannot add a user to a machine without the permission of the owner of that machine. However, one of the standing orders is that in an emergency (for example, if the machine was advertising bad routes and disrupting traffic on the campus backbone) an administrator may add himself to the machine with root capabilities in order to log in and fix the problem. Standing orders rely on the discretion and responsibility of the student administrators, and enable them to cut through political ''red tape'' when absolutely necessary.

The final new ''survival tactic'' is the idea of the ''three Cs'' for dealing with a crisis situation: stay cool, calm, and collected. Most of our junior administrators can handle things after only a few months of training. After that, it is only a matter of making them realize it. We try to stress the idea that no problem will be helped by panic.

These ideas are not particularly new or innovative – the innovation is their application to a small university group staffed almost entirely by students. It's easy for students to look at this job as just a student job to earn money. However, our aim is to make this organization as professional as possible – to come as close to ''real world'' standards as we can with university funding and a staff that averages only fifteen work hours per week.

Better Customer Relations

A large part of being professional is treating your customers well. This is very easy to forget in a university environment, where most services have only one provider. This is almost, but not quite, the situation we occupy, and so we forget that the ''users'' are also the customers. For instance, about six months ago a professor called to ask about the X window system, and when the student on the other end put the phone down the professor heard the student refer to him as a moron.[4] Our improvements to customer relations are intended to remind the staff that UnixOps exists to serve the customers, and that customers are not annoying faceless entities. Another goal is to remind and/or teach the staff that not everyone is as comfortable with computers as they are. The obvious first step is to increase direct interaction with the customers. When a customer calls asking about an X question, a student should walk up the customer's office, instead of trying to debug the problem over the phone. The resulting personal contact changes the interaction from administrator-and-user to one person helping another. The same change is accomplished by increasing the number of office hours that student administrators hold in the workstation labs, and dividing the hours up among several administrators.

The second improvement once again relies on student responsibility. The general idea is to eliminate the manager's role as intermediary between someone who wants work done and the administrator who actually does the work. If a client asks a manager to upgrade her machine, the manager will assign a student or team to the task, and then have the student or team leader contact the client for the specific details. This should result in faster response on customer requests, and reduce the load on the management.

Another improvement to customer response time should result from a particular student or group taking responsibility for handing requests and problem reports from a few specific clients. If a student is already familiar with that customer's specific machine or lab, the customer does not have to continually re-explain unique setup or software, and the student should be able to find the problem faster. This is actually something that happened informally several years ago, when we had fewer customers and a larger senior staff. It worked very well then. However, this direct contact would be lost after the UnixOps employee graduated, leaving the customer back to explaining everything again. Yet, if an entire team is aware of the general setup of all the team member's contact machines, the loss of knowledge that occurs every spring should be significantly reduced.[5]

Results of Large-scale Improvements

Training and Survival Techniques

These ideas have been slowly phased in over the last six months. The last idea that we implemented was that of permanent sub-groups, about a

[4] This particular anecdote might be seen as a contradiction of the idea of the responsible student administrator. We feel that one rude remark does not necessarily indicate that a student is irresponsible – though that particular student certainly needs to work on his tact.

[5] There is one potential downside to this practice. If a customer only contacts one administrator, that customer will have trouble getting a response when his contact is busy with other work or school. However, this problem is easily resolved if the contact simply forwards the problem on to the trouble queue or notifies the client of the her temporary unavailability.

month ago. However, many large projects over the summer were handled by temporary sub-teams. The temporary teams worked largely as expected. They eased the load on management and got the job done faster due to teamwork. The teams did not work perfectly, as one team leader completely monopolized her project, leaving no work for her team members. However, one of the recently created sub-teams managed to upgrade a two-machine office in three days, involving all three team members. The upgrade went as planned – which is saying a lot in terms of the plans of systems administrators.

Possibly the most successful aspect of these changes has been the combination of paying more attention to selecting tasks for junior administrators (as carried out by the managers, as opposed to team leaders) and the idea of staying cool, calm, and collected. Three students – two who were hired at the beginning of the summer and one who moved up from doing backups at about the same time – have blossomed over the summer. They are now at the point where they can work independently, needing little or no help even with obscure problems. One of these students, with minimal supervision, installed four SMD drives and a color board on a Sun 3, two months after starting work.

Student staff participation in the interviewing process has been less successful. At the beginning of the summer five new systems administrators were hired as a result of a month-long hiring process which included staff interviews of all the applicants. Those hired integrated with the rest of the staff much faster than did previous new hires. However, the ease that these new staff members felt may have hurt as much as it helped. Often, the unease felt by a new employee in his new job will prompt him to work hard at learning and performing his occupation. Without this unease, two of the new hires spent a lot of time doing very little, and another spent a lot of time on the phone with personal calls. The former problem has been solved by putting the two less-motivated individuals in a sub-group whose leader is one of the managers, who has focused on keeping them busy. The latter problem seems to have abated with the start of school.

In general, things have improved, now that the sub-groups have been formed. Staff members are not getting lost in the shuffle and problem reports are receiving faster replies. Our general response time is still not as good as it could be. And while the new staff have fit in well, we are still not operating as a cohesive entity.

Customer Relations

Our customers also seem to be happier. The increased lab time is working very well – over the summer two undergraduate labs were almost completely upgraded, and now each lab has a server running BSD-based UNIX (SunOS 4.1.3) with several clients running System V based UNIX (Solaris 2.2). Given the potential for confusion between the two operating systems, we have not had any complaints, and the users seem to be handling the change with a minimum of difficulty, thanks to the in-lab advisors. Staff members are leaving the office more often to fix printers or check out problem reports, speeding up the resolution of those problems.

Conclusions

While things are a lot less chaotic around the office these days, it is still too soon to judge the overall success of the large group model. Subgroups seem to be the most promising innovation, but there is still one pitfall to overcome: group dynamics. At least at this university, undergraduates are not taught how to work well in groups. The only classes that require group effort are senior-level, final project courses. As such, we may have to establish our own curriculum for promoting teamwork, or find a suitable course outside the university.

Involving current staff in the interviewing process is another promising idea, but it has not worked as well as we hoped. Our first results have proved to be somewhat mediocre, and the schedule juggling required to achieve those results leaves the system, as it now stands, too costly and counterproductive to use. With the juggling to schedule an interview with the managers and the wait for a time when at least three staff members were in the office, the whole process took too long. It took so long that one good candidate dropped out of the running.

We have not abandoned the idea completely, however. The hiring process needs to be streamlined. Possibilities include forming an interview committee to handle the entire interviewing process, or pre-selecting a group of students to represent the student staff, and arranging a regular time when this group can meet with prospective employees.

For now, we have gone back to the manager-only interviews, as we are still expanding our staff. The third professional (non-student) manager started work a month ago, and we recently hired two operators and an administrative assistant. The five student administrators who started at the beginning of the summer are, as a group, only now becoming confident and effective. It may be six more months before the people who make up our organization feel comfortable working with one another.

The temporary sub-groups and the training of our new administrators over the summer have generated more observations that may prove useful in further expanding the large-group model. Assigning a new employee to a mentor from the current staff, at least for a week or two, brings the new employee up to speed much faster than the aforementioned book-and-project method. We are also thinking about a ''localization checklist'' for new employees

– a checklist of things for a mentor to teach his novice how to use: MH, voice mail, pager, microwave, stereo, etc.

We want to instill a sense of pride in doing the job, and in doing it right the first time. In the past, UnixOps has had a reputation for effective and robust solutions. Many administrators who graduated from this organization have gone into system administration jobs in the ‘‘real world’’ and continued this tradition. Our ultimate goal is not only to continue to provide the best service possible, but to give students the skills necessary to succeed in any professional endeavor to which they apply themselves.

Acknowledgments

We would like to acknowledge Bob Coggeshall and Lynda McGinley, who (knowingly or not) created the small group model.

Author Information

Tim Hunter is in his fifth year of the Electrical and Computer Engineering program at the University of Colorado at Boulder. He is in his fourth year as a student administrator at UnixOps. He recently spent the summer working in the ‘‘real world’’ at Odyssey Research Associates in Ithaca, NY. He can be reached via electronic mail at tim@colorado.edu.

Scott Watanabe is a Systems Manager at UnixOps at the University of Colorado at Boulder, where he also received his B.A. in micro-biology. He provides the students at UnixOps with moral support, technical guidance, and chocolate pudding. He can be reached at watanabe@colorado.edu.

Both authors can be reached at the University of Colorado, UnixOps/CNS, Campus Box 455, Boulder, Colorado 80309-0455.

Bibliography

[1] T. Hein, E. Nemeth, and T. Galyean, ‘‘Trouble-MH: A Work-Queue Management Package for a >3 Ring Circus’’, USENIX LISA Fall 1990 Proceedings.

[2] Nemeth, Evi, Garth Snyder, Scott Seebass, *UNIX System Administration Handbook*, Prentice Hall, Englewood Cliffs, 1989.

[3] ‘‘System Administration Tools Your Vendor Never Told You About: The Chocolate Chip Cookie’’, Elizabeth Zwicky, *;login:*, Vol. 18, No. 1, p. 10.