Turning the Corner: Upgrading Yourself from "System Clerk" to "System Advocate"

Tom Limoncelli – Lucent Bell Labs – Murray Hill, NJ

ABSTRACT

Two (of the many) types of System Administrators are identified: The "System Clerk" simply performs clerical tasks as she is told to do them (creating accounts, installing software) while the System Advocate proactively works with users to develop and evolve the environment.

Transitioning from Clerk to Advocate involves a change of attitude and "making time" for advocacy. Ways to develop more "spare" time for pro-active work are also spotlighted. I also explain ways to avoid the "system administrator visibility paradox": nobody notices the system administrator on days when things are working well.

Introduction

Here's a test for system administrators: Do users walk into your office with software to install that they procured, licensed, and had shipped to them? When a new employee is hired, are you asked to create her login after she has arrived for her first day? Does your salary come from the same budget as the clerks and the mail-room, not a technical budget?

If you said "yes" to any of these questions, it may be a warning sign that you are being treated as clerical overhead rather than as part of the main business processes of your company. This paper explains how someone can make the transition from System Clerk to a proactive role that I call the System Advocate. Making this transition is not easy. Advocacy requires having "spare" time which usually means both automating processes that are currently performed manually and avoiding time-wasters. This transition is key to improving the service you provide to your users. As a result, it can help your career or at least keep you from being bored.

Changing a habit is one of the most difficult things a person can do. This paper is about changing many, many habits. If you choose to make this transition, you will discover that it is not easy. One should be realistic and expect the changes to happen slowly and be uncomfortable at first. Eventually you will see the rewards. Once the new, good habits have replaced the old, it will be difficult to break them!

What Is Clerical?

If someone hand-writes a memo, it can be given to any clerk/typist to be typed. Clerical SA tasks should be automated and documented (e.g., creating accounts) or simple enough (e.g., installing packaged software[1]) that anyone can do them.

[1]When clerks run into problems installing software they call the toll-free number listed in the manual instead of reverse engineering the install script and proceeding manually.

You can be more valuable to your company and enhance your career by making the move from System Clerk to System Advocate. The value to the company is obvious: by being part of the software purchasing process, you knew to purchase additional disk capacity at the same time as the new software. Result: Your users were not waiting extra weeks while the order for additional disks goes through purchasing. By being part of the new employee orientation planning, you were informed in advance that she was coming and can ensure her account was created and ready when she arrived. Result: the new employee has a better impression of your company (which is important, she took a big risk to move here). She feels welcomed because everything was ready for her when she arrived.

Your career is enhanced because you are providing better value to your company, and a fair and ethical company rewards that. You also feel better about yourself because you are now a part of your environment, not doing tasks that are considered "afterthoughts." You are also less likely to be bored because less of your time will be spent on menial tasks.

What Is A System Advocate?

A System Advocate is a system administrator who completes clerical tasks quickly (usually through automation) so she can focus on finding ways to make her users more productive and advocating for the users' needs to management. How do you know what makes users more productive? Ask them. Be ready with ideas, but let them take the lead.

Example 1:

A user says she could get more done if her computer was faster. You watch her work for a while and hear her swap disk "going nuts." After verifying the problem with `pstat` and other tools, you conclude her machine needs more RAM. A day later you walk into her boss's office with the user and a quote for more RAM. You've educated the user so she can state her case, but you are there if she stumbles or if the boss has technical questions. The boss agrees to purchase the RAM. Soon you have a more productive user and a feather in your cap.

Example 2:

A programming shop would be more productive if it had debugging tools like PureAtria's Purify or CenterLine's CodeCenter. You get sales literature from both companies and bring them to lunch with the lead programmer. Soon you are arranging sales presentations and evaluations with both companies. You and the lead programmer work up a quote for the software, disk space, etc. and present it to your boss together. She buys it, the programmers are more productive and happy: you all win!

A System Advocate is pro-active, a partner of the users who can help sell ideas to management. The clerical tasks have been automated and become a minor part of your brain-time.

The System Advocate Attitude

An advocate has "the right attitude." I'm surprised how often I have to remind system administrators that their users are not "lusers" or "pests with requests." They are the reasons you have a job. You are there to serve, and more importantly to collaborate, and advocate. You are part of the team.

If you mentally think of them as "customers"[2] whom you serve, instead of "pests" that make requests all day, your attitude can change dramatically. However, a System Advocate can get into trouble if one adopts the attitude that "the customer is always right." You could end up doing people's jobs for them if you do not remember to help users help themselves. It is a balancing act.

Requests from users are opportunities to do a fantastic job, one you can be proud of. When you integrate this attitude into your habits, your users will notice the difference in the service you provide!

When I receive email reporting a problem, my reply now ends with a sincere note of thanks for reporting the problem and that "it helped me fix the problem and look for ways to prevent it in the future." If a user makes a request often, I look for ways to empower the user to help themself (maybe `sudo` or other `suid` binary that lets them do the task) or to find a more permanent solution.

Sometimes the solution is as simple as keeping the spare toner cartridges by the printer so users can change them if you aren't around. If a user accidentally deletes files often and you waste a lot of time each week restoring files from tape, you can invest time in helping the user learn about "rm -i" or use other "safe delete" programs. Or, maybe it would be appropriate to advocate for the purchase of a system that lets users do their own restores. If you maintain a log of the number and frequency of restore requests, management can make a more informed decision. (or decide to talk to certain users about being more careful.)

[2]"Whither The Customer?" Kevin C. Smallwood, ;login: and counterpoint by Rob Kolstad

"Spare" Time

You can't make the transition without investing time. "But I have zero free time!", you say. Then first you must invest time in automating tasks so that later you will have more "spare" time.

Creating Spare Time

It takes a long time to manually create a user account. Investing two hours to automate this task will have a dramatic payoff. Much of the reason you save time will be that you will no longer spend time fixing mistakes that are inevitable with manual processes. These mistakes make you look sloppy, even if they only happen once in a long time.

Automation opens doors to better ways of doing things. Without automation, you might choose to put off all account creation requests until a certain day of the week, then do them all in "batch." Users see extremely poor turn-around time. If the automation makes creating accounts simple, you can do them "on demand."

The more you automate a process the better service you can provide. I worked at a site that did OS loads and upgrades manually. Each time they had to "from memory" remember what links to make, files to change, etc. This was followed by 2-3 days of fixing the inevitable mistakes as users reported problems. The result was a network of machines that never changed their OS after the initial load and never had security patches or bug-fixes loaded because it was too much work. Another site had a manual process (requiring you to lug a CD-ROM player to the machine) but loading/upgrading the operating system was one of two simple procedures followed by running a script that did everything else. In one evening an admin could upgrade 15 machines and new machines could be loaded in an hour. Within three months of a SunOS release the entire network was upgraded. Currently I work at a site that uses Sun's AutoInstall for Solaris. By typing "`boot net - install`" at the console, a machine boots from a `bootp` server, loads/reloads its OS from scratch, installs our favorite patches, and installs our local modifications; lately we don't even stick around to make sure the procedure worked because it always does. At night machines check for new patches in our patch library, load them, and reboot if required. Our entire network upgrades itself nightly. Users now have machines ready to use 30 minutes after we've unpacked it and all machines in our system have consistent environments.

To find potential areas of improvement, it may help to keep a log of what you do each day for a week. Either write down what you do as you do it, or have your computer beep every 15 minutes as a signal to log whatever you are doing right now. At the end of a week, look for tasks to automate. When automating a task, do not get bogged down in creating the ultimate system. A simple script that assists you with "the common case" may be more valuable than a large system that automates every possible aspect of a task. The script I use to configure our `bootp` server to accept a new NCD X-terminal simply outputs the commands I should type to create the proper symbolic links. If I approve, I cut and paste them to a shell prompt. The script was easy to write because it doesn't have to deal with asking, "Are you sure?" or special cases. If the NCD requires something special I can use the output as the basis for the commands I actually type.

Finding "Hidden" Spare Time

Where do you find the spare time to create the automation that will give you more spare time? Spare time is hiding all over. Look at these places for some:

- Most businesses have a "light" time of the year. Software shops with a new release every four months usually have a 3-4 week "slow period" after each release. You might have spare time then, or this may be the busy time for you as you get things ready for the users' busy period. You may find that during their busy period nobody bothers you.
- Stop reading Usenet. Period.
- Remove yourself from the two busiest mailing lists you are on. (But make sure you are subscribed to `cert-announce` of course!)
- Take advantage of mail filtering software such as `procmail`[3] to put mail from mailing lists into one folder which is read weekly.
- Take advantage of the early morning: Come in an hour earlier and handle many small, daily tasks when nobody is around to interrupt. Don't waste this time cleaning your email box!
- Have your boss send you to a 1-day time management class. Painful as they sound, they do teach some amazing time-saving tricks. I feel like I have gained an extra two hours each day when I use the techniques I learned.
- Invest in training workshops or books that will enable you to automate tasks (write code) in less time. If you don't know `perl` you are working too hard. Don't overlook `make` either.
- Hold weekly or monthly meetings with your chief customer (a manager or department head) to set priorities and eliminate superfluous items.
- Hire an assistant. If your boss reads this paper and agrees with the philosophy, she may agree to getting you a part-time assistant to take care of some of the clerical tasks. Possible candidates: A local high school student, a technically inclined secretary, the person that covers for you when you are away, a summer intern, an interested programmer. Local students are particularly useful because they are cheap and are looking for ways to gain experience.

Now you have the spare time you need to automate the boring, error prone stuff that eats your time. Remember: "Investing time in automating tasks doesn't cost, it pays."

A Single Success

Now you have some "spare" time, pick one situation to "advocate." Start with something small that, if successful, will be visible. Your goal is to help a user be so successful that she tells others. The best advertising is word of mouth. This also means not biting off more than you can chew. Bust your butt to make sure you follow through and make this a success for the user.

Now repeat. You may have to do this a number of times before you create a success that gets the visibility you hoped for.

Finding places to be helpful can be difficult. I recommend eating lunch with different users every day and keeping your eyes open for opportunity. Avoid the temptation to say "yes" to everyone. Your first attempt requires your undivided attention. Don't say "yes" to to more than one simultaneously task until you've got a feeling for the work involved.

Nobody likes an overbearing System Advocate. Warning signs to look for include: you dominate the conversation, you are loud, you always have something to say no matter what topic comes up, you try to "one-up" other people's anecdotes with better ones.

Your Personal Positive Visibility

The System Administrator's Visibility Paradox is that you are only noticed if something breaks. Six months of 100% uptime takes a huge amount of behind the scenes work and dedication. Management may get the impression that you aren't needed because they do not see the work required. Then a server crashes every four hours until a controller is replaced. Suddenly you are valuable. You are the hero. You are important.

This is not a good position to be in because users will get the impression that you are doing nothing 95% of the time because they don't see the important, behind-the-scenes work you do.

[3] `procmail` by S. R. van den Berg, <ftp://ftp.informatik.rwth-aachen.de/pub/packages/procmail>

A bad alternative is to not maintain a stable system so you are always needed. Bad idea.

A better idea is to make sure people know you do a lot in the background to keep things running smoothly, but do it without being overbearing. Remember:

YOU are responsible for your own
Personal Positive Visibility!
No one does it for you!

How can you do that? Here are three techniques that not only improve your personal positive visibility, but are good pro-active things to do anyway!

Technique #1: Be Your Best

None of the other techniques will work if you aren't providing good service to your users.

Technique #2: The System Status Web Page

Everyone loves surfing the web. During a recent major network reorganization at Bell Labs we maintained a web page with an easy-to-type URL that listed current status:

SYSTEM STATUS
No known problems at this time.
Mon Jul 2 10:20:13 EDT 1997
Please report problems via email to "help".

Coming soon:
- Wed, July 10: 10am-noon
  - Aisle 4D5 scheduled power outage
- Thu, July 11: 12:15pm
  - Router reboot (2 minutes)

Changes recently completed:
- Fri, June 28:
  - System "londo" and "gkar" upgrades complete
- Fri, June 30:
  - System "elbows" converted to flame-net

You might configure users' browsers to go to this page when users click on the "Home" button and provide a number of useful URLs at the top of the page so there is less temptation to change the default.

At times the system status was changed to simply, "Machine narn is down, we're working on it." This says to users "we care" and "we are working on the problem." In the absence of information users often assume you are out to lunch, ignoring the problem, or ignorant of the problem.

Obviously you care about the problem and are working on it, but taking 10 seconds to let your users know this creates the Personal Positive Visibility you desire.

As users become accustomed to checking this URL before complaining, the number of phone calls you get while trying to fix the problem is reduced. There is nothing worse that being delayed from working on a problem because you are busy answering the phone calls of users that want to "help" by reporting the problem. Users will not develop the habit of checking this URL until you consistently update it.

You might also consider putting a chalkboard at the entrance to your machine room and putting status messages there. Low tech, but it works.

Technique #3: User Meetings

Another technique is to host regularly scheduled meetings with users: Users' Group meetings can be an excellent forum for bidirectional feedback. They can also be a disaster if you are not diplomatic. Be careful. The focus must be on their needs, not yours.

I suggest you use a format similar to this:

Welcome (2 minutes) – Welcome the group and thank them for attending. It is a good idea to have the agenda written on a white board so people know what to expect.

Introductions (5 minutes) – Each attendee introduces themself.

User Feedback (20 minutes) – Getting feedback from the users is part art, part science. You want users to focus on their needs. You may consider taking a class on meeting facilitation if you feel weak in this area. What works for me is to ask open-ended questions like:

"If one thing could be improved it would be . . ."

"The single best part of our computing environment is . . ."

"The single worst part of our computing environment is . . ."

"My job would be easier if . . ."

Keep a log of what users suggest. A huge pad of paper on an easel is best so everyone can see what is being recorded. Don't write full sentences, just key phrases like "faster installation new C++" or "add CPUs to server5." Once a page is filled, tape it to the wall and go to the next question.

Do not reject or explain away any requests. Just record what people say. To get the best responses, people must feel they are safe. People do not feel safe–and will stop talking–if every suggestion is answered with your reason why that "just isn't possible" or "we can't afford it." However, be clear that recording an idea does not guarantee it will be implemented.

Be careful of permitting one talkative person to dominate the meeting. If this happens, you might want to go around the room having each person answer the question in sequence or use phrases like, "Let's hear from the people in the room that have not yet spoken."

Don't get stuck in an infinite loop. If you hit your time limit, cut off discussion politely and move on. People have busy schedules and need to get back to work.

Review (10 minutes) – Review what you've recorded by reading the items out loud. Consider reviewing these lists with your boss afterwards to prioritize (and reject) tasks.

Show and Tell (30 minutes) – This is the main attraction. People want to be entertained and the best way to entertain technical people is to have them learn something new. Ask a salesperson to present information on a product, have an internal person present some information about something they do or found useful, or you yourself might want to focus on a new feature of your network. This may be a good time to explain a big change that is coming soon, or describe your network topology or some aspect of the system that you find users don't understand.

Meeting Review (5 minutes) – Have each person in turn say how they felt the meeting went (less than one sentence each).

Thank everyone (2 minutes) – First ask for a show of hands to indicate if they thought this meeting was useful. Then remind people that you are available if people want to drop by and discuss these issues further. Then (and this is important) thank people for spending time out of their busy schedule.

Your Boss

It is a good idea to show your boss this article before you begin. A good boss will support the gradual change to System Advocate because she understands how this will create value to your company. A really good boss will help you set the pace, make suggestions, and mentor you. Warning: a good boss will set a pace that is slower than you want. Trust her advice.

A bad boss will say this paper is trash and you'll be forced to use these techniques independently until you have made dramatic improvements in your users productivity.

Conclusion

Transforming a clerical role to a pro-active "System Advocacy" role is a change in attitude and working style that can dramatically improve the service you provide to your users. It isn't easy. It requires hard work, and requires investing time to "create" free time. Achieving your first success can be very difficult. Taking ownership of your Personal Positive Visibility not only enhances your career, but provides opportunities to serve your users better. To lower cost of ownership, vendors are automating the clerical side of system administration but no vendor can automate the value of a System Advocate.

References

Love Your Job!, By Dr. Paul Powers & Deborah Russell, 1st Edition August 1993, O'Reilly and Associates.

The Macintosh Way, By Guy Kawasaki, September 1989, Scott Foresman Trade.

Programming Perl, 2nd Edition, Larry Wall, Tom Christiansen and Randal L. Schwartz, 1996, O'Reilly and Associates.

System Performance Tuning, By Mike Loukides, 1st Edition November 1990, O'Reilly and Associates.

When You Can't Find Your UNIX System Administrator, By Linda Mui, 1st Edition April 1995, O'Reilly and Associates.

"Whither The Customer?" Kevin C. Smallwood, *;login:* and counterpoint by Rob Kolstad

Author Information

Tom Limoncelli is a MTS at Bell Labs, the R&D unit of Lucent Technologies, where he is chiefly concerned with the architecture and operation of the data network for much of Research. Tom started doing system administration on VAX/VMS systems in 1987 and switched to Unix in 1991. He holds a B. A. in C. S. from Drew University, Madison, New Jersey. His homepage is http://www.bell-labs.com/user/tal; he can be reached at <tal@lucent.com>.