Deconstructing User Requests and the Nine Step Model

Thomas A. Limoncelli – Lucent Technologies/Bell Labs

ABSTRACT

How can we improve the process by which System Administrators (SAs) help users? SAs spend much of their time responding to requests from users. Better system administrators use a similar, structured, process. I present the structured process as I have seen and practiced it, examples of each step in the process, and the pitfalls of eliminating various steps. Finally I look at the paper in the larger context of a step towards improving the science of System Administration.

Introduction

In this paper I document and analyze the process for resolving trouble reports that is used by the best system administrators (SAs) I have known and observed. The goal is to improve the process by which SAs repair problems that are reported by users (e.g., "helpdesk requests"). This paper also establishes a base model for use in future studies.

A large part of a SAs workload comes from users that report problems, request improvement, ask questions, and so on. This paper is focused on resolving these requests as efficiently as possible given the resources available. Thus, retain user happiness. The model will identify requests that are out of scope (request for new features, questions, etc.) and offer appropriate responses.

The method to process these "trouble reports" has nine steps, which can be grouped into four phases:

- Phase A: The Greeting ("Hello")
  - Step 1: The Greeting
- Phase B: Problem Identification ("What's wrong?")
  - Step 2: Problem Classification
  - Step 3: Problem Statement
  - Step 4: Problem Verification
- Phase C: Planning and Execution ("Fix it")
  - Step 5: Solution Proposals
  - Step 6: Solution Selection
  - Step 7: Execution
- Phase D: Verification ("Verify it")
  - Step 8: Craft verification
  - Step 9: User Verification/Closing.

In addition to being useful to SAs, I have found that if users understand this model they assist the process. They become more skilled in getting the help they desire. They will be prepared with the right information and they can nudge the SA along through the process if necessary.

I should point out that I don't feel this process is a panacea, nor do I think this process is a replacement for a creative mind or technical experience. However, this gives SAs a common set of terminology and a well tested, effective process, to use when interacting with users. SAs will not magically all become equally productive. Creativity, experience, resources, tools, personal and external management are other influences that contribute to productivity.

Historical Comparison

At the close of World War II, the United States found itself with a huge excess of manufacturing capacity. As a result, companies started producing hundreds of new products that households and businesses never had access to previously. The thousands of returning G.I.'s found jobs selling these new products. The new manufacturing capacity, the new products, and the large number of returning G.I.'s looking for work combined to produce a new era for the U.S. economy. In short, the large manufacturing capacity met the large demand which met the large sales force.

As time went on competition grew. Companies found that it was no longer sufficient to have a large sales force, a good sales force was needed. They started to ask, "What makes the high performing salesmen different from the others?" At the request of industry, business schools began studying salespeople.

Industry encouraged business schools to increase their study of the sales process. They discovered that the better salesmen, whether or not they realized it, had a specific, structured method they employed. It involved specific phases or steps. Mediocre salespeople deviated from these phases in varying ways or performed certain phases badly. The low performers had little or no consistency in their methods.

The method, once identified, could be taught. Thus sales skills went from an intuitive function to a formal function with well-defined parts. Previous sales training mostly consisted of explaining the product's features and qualities. Subsequently, training included exploration of the selling process.

This deconstruction permitted further examination and therefore further improvement. Each step could be studied, measured, taught, practiced, and so on. Focus is improved because a single step can be studied in isolation. Also the entire flow of steps could be studied (a holistic approach).

I imagine that if anyone explained the structured process to the high performing salespeople it would sound strange. To them it comes naturally. It would be like explaining to Picasso how to paint. However, to the beginner, this framework gives structure to a process they are learning.

In recent years, system administration has begun a similar journey. SA previously was a craft or an art practiced by few people. With the recent, explosive growth of corporate computing and intranet/internet applications, the demand for SAs has been similarly explosive. A flood of new system administrators has arrived to meet the need. Quality of their work varies. Training often takes the form of teaching particular product features, similar to when a salesperson's training consisted mostly of learning the product line. Other training methods include exploring manuals and documentation, trial by fire, training by social institutions (IRC, mailing lists, etc.) and professional institutions (SAGE, SANS, etc.).

SA also needs to have its processes understood. Recently there has been a rise in in-school curricula being taught on the topic of system administration. However, most of it has been specific to particular technologies and vendors rather than being non-vendor-specific and theoretical. I hope that more theoretical models will be introduced and popularized in the coming years and this will result in large improvements similar to the improvements made in the sales profession.

The Process

The process described in this paper contains nine steps grouped into four phases. As seen in Figure 1, the phases deal with:

A. how the user reports the problem,
B. identifying the problem,
C. planning and executing a solution, and
D. verifying that the problem resolution is complete.

Readers should be forewarned that sometimes certain steps are iterated as required. For example, during Step 4 (Problem Verification) the SA may realize the issue has been misclassified and one must return to Step 2 (Problem Classification). This can happen at any step and require returning to any previous step.

A description of the steps follows.

Phase A: The Greeting ("Hello!")

The first phase only has one deceptively simple step. The user reports the problem.

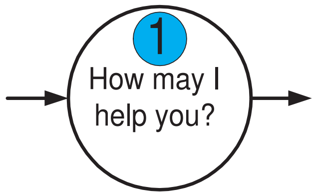

Figure 2: Greeting phase.

Step 1: The Greeting – "The Greeter"

This step is where someone or some thing asks, "How may I help you?" This step includes everything related to how the user's request is submitted. Commonly users can report a problem by calling a customer care center, by walk-up "help desk," or electronic submission. These methods are called "Greeters." Multiple greeters are needed for easy and reliable access. If the user's problem is that they can't send email, reporting this via email is not possible. Having multiple greeters is valuable.

Sometimes problems are reported by automated means rather than by humans. For example, network monitoring tools such as "mon" [Trocki], HP OpenView, and Tivoli can notify SAs that a problem is occurring. This is still the same process, although some of the steps may be expedited by the tool.

Every site and every user is different. Greetings become more or less appropriate based on many factors. Is the user local or remote? Is the user experienced or new? Is the technology complicated or simple? These questions can help a site select which greeters should be used.

How do users know how to find help? There are various ways to advertise the available greeters. Examples include: signs in hallways, newsletters,

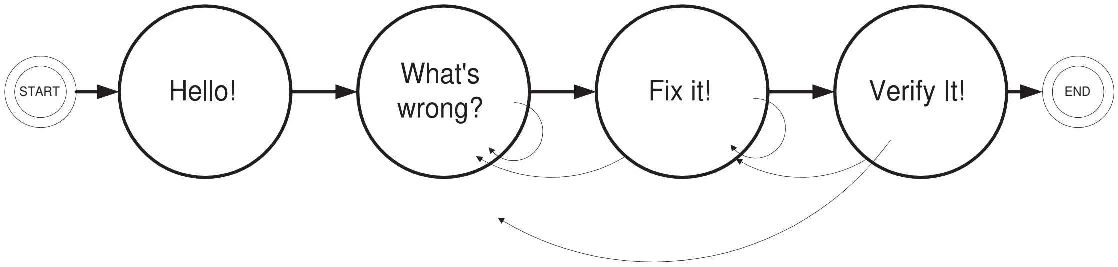

Figure 1: General flow of problem solving.

stickers on computers or phones, and even banner advertisements on internal web pages.

Summary of common methods used to report problems (certainly this is an incomplete list):

- Phone
- Email
- Walk-up helpdesk
- Visiting SAs office
- Submission via web
- Submission via custom application
- Report by automated monitoring system

Phase B: Problem Identification ("What's wrong?")

The second phase is focused on classifying the problem, recording it, and verifying the problem.

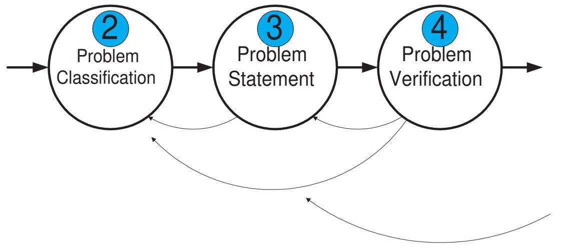

Figure 3: What's wrong?

Step 2: Problem Classification – "The Classifier"

In this phase, the request is classified to determine who should handle the request. This role is called "The Classifier." The classifier may be a human or automated. For example, at a walk-up help desk, staff might listen to the problem description to determine its classification. A phone response system may ask the user to press 1 for PC problems, 2 for network problems, etc. If certain customer groups are helped by certain SAs, their requests may be automatically forwarded based on the requester's email address or employee id number.

When the process is manual, a human must have the responsibility to surmise the classification from the description or to ask the user more questions. A formal decision tree may be used to determine the right classification.

No matter how the classification is done, the user should be told how the request is classified. This creates a feedback loop that can detect mistakes. For example, if a classifier tells a customer, "This sounds like a printing problem. I'm assigning this issue to someone from our Printer Support Group." the user retains greater participation in the process. The user may point out that their problem is more pervasive than just printing, leading to a classification such as a network problem.

If a phone response system is used, the user has classified the request already. However, they may not be the best person to make this decision. The next human that speaks with the user should be prepared to validate the user's choice in a way that is not insulting. Also, the choices given to the user must be carefully constructed and revised over time.

Many requests may be transferred or eliminated at this stage. For example, if the user is requesting a new feature they should be transferred to the appropriate group that handles requests for features. That role is often called "service management." Or if the request is outside the domain of work that the support structure does, they might be referred to another department. Or if the request is against policy and therefore it must be denied. The issue may be escalated to management if the user disagrees with the decision. For this reason, it is important to have a well-defined scope of service and process for requesting new services.

Step 3: Problem Statement – "The Recorder"

This is where the user states the problem with full details and this information is recorded. This person performing this role is called "The Recorder" and is often the same person as the classifier. The skill required by the recorder in this phase is the ability to listen and ask the right questions to draw out the needed information from the user. The recorder extracts the problem statement and records it.

A problem statement describes the problem being reported and enough clues to reproduce and fix the problem. A bad problem statement is vague or incomplete. A good problem statement is complete and identifies all hardware and software involved as well as their location, the last time it worked, and so on. Some times not all of that information is appropriate or available.

An example good problem statement is "PC talpc.example.com (a PC running Windows NT 4 SP4) located in room 301 can not print from MS-Word97 to printer 'rainbow', the color PostScript printer which is located in room 314. It worked fine yesterday. It can print to other printers. The user does not know if other computers are having this problem."

It is unreasonable to expect problem statements directly from users to be so complete. They require assistance. The above problem statement comes from a real example where a customer sent email to a SA that simply stated, "Help! I can't print." That is about as ambiguous and incomplete as a request can be. A reply was sent asking, "To which printer? Which PC? What application?"

The reply included a statement of frustration. "I need to print these slides by 3 pm, I'm flying to a conference!" At that point, email was abandoned and the telephone was used. This permitted a faster "back and forth" between the user and classifier. No matter the medium, it is important that this dialog take place and that the final result be reported to the customer.

Sometimes the recorder can perform a fast loop through the next couple steps to accelerate the process. The recorder might ask the typical "just in case" questions such as "Is it plugged in?" and "Have you check the manual or on-line help?" or "Did you receive the memo that said printer 'rainbow' would be decommissioned last week?" In our example the user indicated that there was an urgent need to have the slides printed. Here it might be appropriate to suggest using a different printer that is known to be working.

Certain classes of problems can be completely stated in a simple way. I have found that internet routing problems can best be reported by listing two IP addresses that can not ping each other, but which can both communicate to other hosts and including a traceroute from both (if possible) host to the other.

Large sites often have different people recording requests and executing the requests. This added "hand-off" introduces a challenge as the recorder may not have the direct experience required to know exactly what to record. In that case, it is prudent to have pre-planned sets of data to gather for various situations. For example, if the user is reporting a network problem, the problem statement must include an IP address, the room number of the machine that is not working, etc. If the problem relates to printing one might be required to record the name of the printer, the computer generating the print job, the application generating the print job, etc.

Most sites use some kind of "trouble ticket" software to record the user's report. It can be useful if the software requests different information depending on how the problem has been classified.

Step 4: Problem Verification – "The Reproducer"

This is where the SA tries to reproduce the problem. This role is called "The Reproducer." If the problem can not be reproduced, often the problem being reported is not being properly communicated and one must return to Step 3 (Problem Statement). If the problem is intermittent, then this process becomes more complicated but not impossible.

It is critical that the method used to reproduce the problem is recorded for later repetition in Step 8 (Craft Verification). Encapsulating the test in a script will make verification easier. One of the benefits of command-driven systems like UNIX is the ease in which such a sequence of steps can be automated. Graphical user interfaces make this phase more difficult since there is no way to automate or encapsulate the test.

The scope of the verification procedure must not be too narrowly focused nor too wide, nor mis-aimed. If the tests are too narrow, the entire problem may not be fixed. If the tests are too wide, the SA may waste time chasing non-issues.

It is possible that the focus may be mis-aimed. There may be another, unrelated problem in the environment that is discovered while trying to repeat the user's reported problem. Some problems can exist in an environment without being reported or without affecting users. It can be frustrating for both the SA and the user if many unrelated problems are discovered and fixed along the way to resolving an issue. If an unrelated problem is discovered that is not in the critical path, it should be recorded so that it can be fixed in the future. On the other hand, determining if it is in the critical path is difficult, so fixing it may be valuable. Alternatively, it may be a distraction or may change the system enough to make debugging difficult.

Sometimes direct verification is not possible or even required. If a user reports that a printer is broken the verifier may not have to reproduce the problem by attempting to print something herself. It may be good enough to verify that new print jobs are queuing and not being printed. Such superficial verification is fine in that situation.

However, other times exact duplication is required. The verifier might fail to reproduce the problem on her own desktop PC, and may need to duplicate the problem on the user's PC. Once the problem is duplicated in the user's environment, it can be useful to try to duplicate it elsewhere to determine if the problem is local or global. A lab of equipment for the purpose of reproducing reported problems may make supporting remote users or complicated products easier.

Phase C: Planning and Execution ("Fix it")

In the previous phase the problem was identified. In this phase it is fixed. This involves planning possible solutions, selecting one, and executing it.

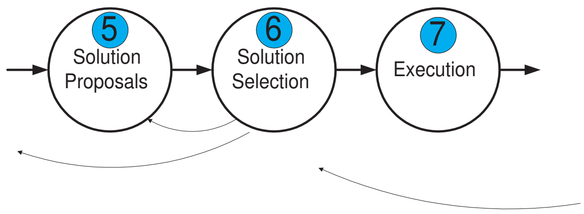

Figure 4: Flow of repair.

Step 5: Solution Proposals – "Subject Matter Expert"

This is the point where the possible solutions are enumerated. This role is performed by the "Subject Matter Expert" or SME. Depending on the problem, this list may be large or small. For some problems the solution may be obvious and there is only a single proposed solution. Other times there are many possible solutions. Often verifying the problem in the previous step helps finding possible solutions.

The "best" solution varies depending on context. At a bank, the Help Desk's solution to a client-side NFS problem was to reboot. It was faster than trying to fix it and it got the customer up and running quickly. However, in a research environment, it would make sense to try to find the source of the problem, perhaps unmounting and re-mounting the NFS mount that reported the problem. In our printing example, since the user indicated that they needed to leave for the airport soon, it might be appropriate to suggest alternative solutions such as recommending a different printer that is known to be working. If the user is an executive flying from New Jersey to Japan with a stop-over in San Jose, it might be reasonable to transfer the file to an office in San Jose where it can be printed. A clerk could hand the printout to the executive while he waits for his connecting flight at the San Jose airport.[1]

Some solutions are more expensive than others. Any solution that requires a desk-side visit is generally going to be more expensive that one that can be handled without such a visit. This kind of feedback can be useful in making purchasing decisions. Lack of remote support capability affects the total cost of ownership of a product. There are tools (commercial and non-commercial) that add remote support to such products.

If the SA does not know any possible solutions the issue is escalated to other, usually more experienced, SAs.

Step 6: Solution Selection

Once the possible solutions are enumerated, one of them is selected to be attempted first (or next, if we are looping through these steps). This role is also performed by the "Subject Matter Expert" or SME.

Selecting the best solution tends to be either extremely easy or extremely difficult. However, solutions often can not be done simultaneously so possible solutions must be prioritized, usually with the help of the user.

The user should be included in this decision. The user has a better understanding of their own time pressures. If the user is a commodities trader, she or he will be much more sensitive to downtime during the trading day than, say, a technical writer or even a developer (provided they're not on deadline). If solution A fixes the problem forever but requires downtime, and solution B is a short-term fix, the user has to be consulted as to whether A or B is "right" for his or her situation. It is the responsibility of the SME to explain the possibilities. Some of this the SA should know based on his or her environment. There may be predetermined service goals for downtime during the day. SAs on Wall Street know that downtime during the day can cost millions, so sort-term fixes may be selected and a long-term solution may be scheduled for the next maintenance window. In a research environment, the rules about downtime are more relaxed and the long-term solution may be selected immediately.[2]

When dealing with more experienced users, it can be useful to let them participate in this phase. They may have useful feedback. In the case of inexperienced users, it can be intimidating or confusing to hear all these details. It may even unnecessarily scare them. For example, listing every possibility from a simple configuration error to a dead hard disk may cause the user to panic and is a generally bad idea. (Especially when the problem turns out to be a typo in CONFIG.SYS)

Even though the user may be inexperienced they should be encouraged to participate in determining and choosing the solution. This can help educate the user so future problem reports can flow more smoothly or even help them solve their own problems in the future. It can also give the user a sense of ownership, the warm fuzzy feeling of being part of the team/company, not a "user." That can help break down the "us vs. them" mentality that is common in industry today.

Step 7: Execution – "The Craft Worker"

This is where the solution is attempted. The skill, accuracy, and speed at which this step is completed is dependent on the skill and experience of the person executing the solution.

The term "craft worker" refers to the SA, operator, or laborer that performs the technical tasks involved. This term comes from other industries. For example, the foreman at a construction site plans what is done when, the craft workers (carpenters, plumbers, etc.) do the physical work. In the telecommunications industry, while others have received the order and planned the provisioning of the service, the craft workers run the cables, connect circuits, etc. In a computer network environment, the Network Architect might be responsible for planning the products and procedures used to give service to customers, but when a new ethernet interface needs to be added to a router, the craft worker installs the card and configures it.

Even the user might become the craft worker. This is particularly common when the user is remote and is using a system with little or no remote control. In that case, the success or failure of this step is in the hands of this user.

A dialog is required between the SA and the user to make the solution work. Has the user executed the solution properly? If not, are they causing more harm than good?

The dialog has to be adjusted based on the skill of the user. It can be insulting to spell out each command, space, and special character to an expert user. It can be intimidating to a novice user if the SA rattles off a complex sequence of commands. Asking, "What did it say when you typed that?" is better than "Did it work?" in these situations. Bi-directional communication is critical and the skills related to this can be a unique specialty in our industry. Training is available. Workshops that focus on this area often have titles that

[1]This is a true story which happened at a previous employer. The printer was a very expensive plotter. Only one such plotter was at each company location.

[2]Personal communication with Josh Simon.

include the buzzwords "Active Listening," "Interpersonal Communication," "Interpersonal Effectiveness," or simply "Advanced Communication."

At this point it is tempting to think that we are done. However, we aren't done until the work has been checked and the user is satisfied. That brings us to the final phase.

Phase D: Verification ("Verify it")

At this point the problem should be remedied but we need to verify that it really has been. This phase isn't done until the customer agrees the problem is fixed.

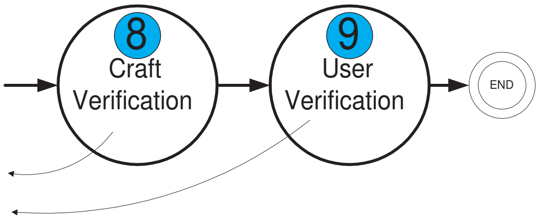

Figure 5: Verification flow.

Step 8: Craft Verification

This is the step where the craft worker that executed Step 7 (Execution) verifies that the actions taken to fix the problem were successful. If the process used to reproduce the problem in Step 4 (Problem Verification) is not recorded properly, or not repeated exactly, the verification will not properly happen. There is potential that the problem still exists, but verification fails to demonstrate this, or the problem may have gone away but the SA does not know this.

If the problem still exists, return to Step 5 (Solution Proposals) or possibly an earlier step.

One tool that is useful in this step is "diff." Capture the output generated when the problem is reproduced. During craft verification first one may "diff" the captured output against the new output. Alternatively, one might copy the output that demonstrates the problem and edit it to be the way it should be on a working system. Or one might have a working system to generate a sample "good" output. Either way, "diff" can then be used to compare the current output with the corrected output.

Variations on this theme are many. Once a user was able to provide me with a sample TeX file that processed fine in his previous department's TeX installation but not on ours. Since I had an account on the computers of his previous department, I could establish a basis for comparison. This was extremely useful. Eventually I was able to fix our TeX installation through successive refinement of the problem and comparison on both systems.

Some problems do not generate output that is well suited to "diff," but perl and other tools can pare down the output to make it more palatable to diff. Once we were tracking reports of high latency on an ISDN link. The problem happened only occasionally. We set up a continuous (once per second) "ping" between two machines that should demonstrate the problem. We recorded this output for a number of hours and observed consistently good (low) latency except occasionally there seemed to be trouble. We built a filter in awk that would extract pings with high latency (where latency was more than three times the average of the first 20 pings) and would reveal missed pings. We noticed that no pings were being missed, but every so often a series of pings took much longer to arrive. We used a spreadsheet to graph the latency over time. Visualizing the results helped us notice that the problem occurred every five minutes within a second or two. It also happened at other times, but every five minutes we were assured of seeing the problem. We realized that there are protocols that do certain operations every five minutes. Could a route table refresh be overloading the CPU of a router? Maybe there was a protocol that overloaded a link? By repeating the ping test between smaller and smaller portions of the path, we were able to isolate which router was introducing the latency. Its CPU was being overloaded by routing table calculations, which happened every time there was a real change to the network or every five minutes. This agreed with our previously collected data. The fact that it was an overloaded CPU, not an overloaded network link explains why latency increased but no packets were lost. Once we fixed the problem with the one router we used our ping test and filter to demonstrate that the problem had been fixed.

Step 9: User Verification/Closing – "The User"

The final step is for the user to verify that the issue has been resolved. If they aren't satisfied, the job isn't done. This role is performed by the user themselves.

Presumably if the craft worker positively verified that the solution worked (Step 8, Craft Verification) this should not be needed. However, often users report at this point that the problem still exists. This is such a critical problem at some sites that the author chose to emphasize it by making it a separate step.

User verification can reveal mistakes made in previous phases. Communication problems include the user not properly expressing the problem, the SA not understanding the user, or the SA not properly recording the problem. Errors may have crept into the planning phase. The problem that was verified in Step 4 (Problem Verification) may have been a different problem that also exists or the method that verified the problem may have been incomplete. The solution may not have fixed the entire problem or may have turned the problem into an intermittent one.

In either case, if the user does not feel the problem is fixed, there are many possible actions. Obviously, Step 4 (Problem verification) should be repeated to find a more accurate method to reproduce the problem. However, at this point it may be appropriate to return to other steps. For example, the problem could be re-classified (Step 2, Problem Classification) or re-stated (Step 3, Problem Statement), or escalated to more experienced SAs (Step 5, Solution Proposals). If all else fails, one may have to resort to escalating the problem to management.

It is important to note that "verification" isn't to verify that the user is happy, but that the user's request is satisfied. Some users are never happy. In a perfect world, this step would be where the customer thanks the SA, but we know we can not always expect gratitude. Sometimes gratitude takes odd forms because users may not understand what is "hard" and what is "easy." The typical example is the user that hardly blinks when SAs work overtime to resolve a major network issue but send compliments to the SA's management after being so impressed that the SA fixed a problem where the user couldn't log in (the caps lock key had been pressed).

Once user verification is complete, the issue is "closed." If a tracking system is used, the "ticket is closed." Lastly, and possibly only in a perfect world, the customer is told to have a nice day.

Perils of Skipping A Step

Each step is important. If any step in this process is performed badly the process can break down. It is my experience that many SAs skip a single step either due to lack of training or honest mistake. I find many stereotypes about bad SAs are the results of SAs that skip a particular step. I have assigned Seinfeld-esque names to each of them and list possible ways of improving the SAs process. Reading this paper should also help improve their process.

The Ogre: Grumpy, caustic SAs are trying to scare users from Step 1. They are preventing the Greeting from happening. Suggestion: Management must set expectations for friendliness. Also, it is important to set expectations with users.

The Mis-delegator: If you've called a large company's technical support line and the person that answered the phone refused to direct your call to the proper department, you know what its like to deal with a Mis-delegator. They skip Step 2. Suggestion: A formal decision tree of what issues are delegated where.

The Assumer: I've never seen anyone habitually skip Step 3, but I've seen SAs assume they understand what the problem is when they really don't. Suggestion: An "Active Listening" class usually helps this kind of SA.

The Non-Verifier: A SA that skips problem verification (Step 4) is usually off fixing the wrong problem. Recently I was panicked by the news that "the network was down." In reality, a non-technical user couldn't read their email and reported "the network is down." This claim hadn't been verified by the newly hired SA who hadn't yet learned that certain novice users report all problems as "the network is down." The user's email client was misconfigured. Suggestion: Teach SA to replicate problems, especially before escalating them.

The Wrong Fixer: Inexperienced SAs sometimes are not creative, or are too creative, in proposing and selecting solutions (Step 5 and 6). But skipping these steps entirely results is a different issue. After being taught how to use an Ethernet monitor (a network sniffer), an inexperienced but enthusiastic SA was found dragging out the sniffer no matter what problem was being reported. He was a Wrong Fixer. Suggestion: Mentoring or training. Increase the breadth of solutions with which the SA is familiar.

The De-Executioner: Incompetent SAs sometimes cause more harm than good when they execute incorrectly. How embarrassing to apply a fix to the wrong machine. However, it happens. Suggestion: Train the SA to check what they have typed before pressing RETURN or clicking "OK." It can be useful to include the hostname in one's shell prompt.

The Hit-And-Run Sysadmin: This SA walks into a user's office, types a couple keystrokes and waves as he walks out the door saying, "That should fix it." The users are frustrated to discover that the problem was not fixed. In all fairness, what was typed *really should have fixed the problem* but it didn't. Suggestion: Management needs to set expectations on verification.

The Closer: Some SAs are obsessed with "closing the ticket." Often SAs are judged on how quickly they close tickets. In that case, they are pressured to skip the final step. I borrow this name from the term used to describe high-pressure salespeople who are focused on "closing the deal." Suggestion: Management should not measure performance based on how quickly issues are resolved but on a mixture of metrics that drive the preferred behavior. Metrics should not include time waiting for customers when calculating how long it took to complete the request. Tracking systems permit a request to be put into a "customer wait" state while waiting for them to complete actions, etc.

Improving The Process

With the process broken into specific steps, each grouped into distinct phases, improvements can be made by focusing on each step. Entire books could be written on each step. This has happened in other professions that have similar models (Nursing, Sales, etc.).

In addition to focusing on improving each step, one may also focus on improving the entire process. Transitioning to each new step should be fluid. If the user sees a staccato hand-off between each step, the process can look amateurish or disjointed.

Every hand-off is an opportunity for mistakes and miscommunication. The fewer hand-offs, the fewer opportunities for mistakes. A site small enough to have a single SA has zero opportunities for this class of error. However, as systems and networks grow and become more complicated, it becomes impossible for a single person to understand, maintain, and run the entire network. As a system grows hand-offs become a necessarily evil. This explains a common perception that users have: larger SA groups are not as good as smaller ones. However it shows an area for improvement: when growing a SA group one should focus on maintaining high quality hand-offs. Or, one might choose to develop a ''single point of contact'' (SPOC) or user advocate for an issue. That results in the users seeing a single face for the duration of a problem.

In addition to improving the individual steps or the flow, one can take a holistic view to seek improvements. No man is an island, and no single trouble report is an island either. The flow from problem report to problem report is an area that should be studied. Does a user report the same issue over and over? (Why is it recurring?) Always in a particular category? (Is that system badly designed?) Are many users reporting on the same issue? (Can they all be notified at once? Can that problem receive additional priority?) All of these scenarios can be identified and become areas of improvement for a SA organization.

For example, during a major network outage, many users may be trying to report problems. If users report problems though a automatic phone response system (''Press 1 for... press 2 for...'') usually such a system can be programmed to announce the network outage before listing the options. ''Please note the network connection to Denver is currently experiencing trouble. Our service provider expects it to be fixed by 3 pm. Press 1 for... press 2 for....'' This kind of ''global announcement'' can be easily provided in any of the first three steps.

If the users talk to a different person every time they call for support, there is less chance for the SA to become familiar with the users' particular needs. There are ways of rectifying this. For example, subteams of the SA staff may be designated to particular groups of users, rather than based on which technology they support. If the staff answering the phone is extremely large they may be using a phone ''Call Center'' system where users call a single number and the call center routes the call to an available operator. Modern call center systems can route calls based on caller id. They can use this functionality to, for example, route the call to the same operator they spoke to last time if that person is available. This means there will be a tendency for users to be speaking to the same person. It can be comforting to be speaking to someone that recognizes your voice.

A better educated user can be a better customer. If a user understands the nine steps that will be followed, they can be better prepared when reporting the problem. They might have more complete information about the problem being reported when they call because they understand the importance of complete information. In gathering this information, they will have narrowed the focus of the problem report. They might have specific suggestions on how to reproduce the problem. They may have narrowed the problem down to a specific machine or situation. Their additional preparation may lead them to solve the problem on their own! Training for users should include explaining the nine step process to facilitate interaction between users and SAs.

Some things hurt the process. For example, an ill-defined delineation of responsibilities makes it difficult for a ''classifier'' to delegate the issue to the right person. Inexperienced ''recorders'' don't gather the right information in Step 3 (Problem Statement) which makes further steps difficult and may require contacting the user unnecessarily. A list of standard information to be collected for each classification will reduce this problem.

Architectural decisions may impede the classification process. The more complicated a system is, the more difficult it can be to identify and duplicate the problem. Sadly, some well accepted software design concepts are at odds with this, such as delineating a system into layers. For example, a printing problem in a large UNIX network could be a problem with DNS, lpd on the servers, lpr on the client, the wrong version of lpr, misconfigured user environment, the network, BOOTP, the printer's configuration or occasionally even the printing hardware itself. Typically many of those layers are maintained by separate groups of people. To diagnose the problem accurately requires the SAs to be experts in all of those technologies, or that the layers crosscheck each other.

Team of One

The solo SA can still benefit from using the model to make sure that users have a well-defined way to report problems, that problems are recorded and verified, solutions are proposed, selected and executed, and that both the SA and the user has verified the problem is resolved.

Problems can be escalated to vendor support lines. Often the solo SA's site is part of a larger company that has a larger IT organization.

Future Work

I feel that deconstructing and analyzing the things that SAs do is the most fruitful way to improve our profession and turn our practice into a science. I hope to see other processes analyzed this way. I also look forward to competing models that describe what I have presented here. Such an academic debate would only help our profession. I would also like to see extensions to the model or exploration of ways to perform particular stages.

The System Administration Maturity Model (SAMM) presented in [Kubi93] establishes a maturity model for IT that is similar to CMU's Software Maturity Model. It would be fruitful to explore how SAMM and the process described in this paper can complement each other.

This paper does not discuss metrics. A system of metrics grounded in this model might detect areas needing improvement. The model can be instrumented easily to collect metrics. However, developing metrics that drive the right behaviors is difficult. For example, if SAs are rated by how quickly they close tickets, one might encourage "The Closer" behavior described above.

As SAs pro-actively prevent problems, reported problems will become more serious and time consuming. If average time to completion grows, does that mean minor problems were eliminated or that SAs are being slower at fixing all problems?

Many other questions need further research:

- What are all the ways to greet users? How do they compare by cost, by speed (faster completion), by user preference? Is the most expensive method the one that users prefer the most?
- How can classification be improved?
- Some problem statements can be stated concisely, like the routing problem example in Step 3. Given various situations, what is the shortest problem statement that completely describes the issue?
- Are there times not to use these steps? For example, if a router has lost power, there is no need to go through the steps. One simply turns the power back on!
- Diagnostic tools that integrate well with this model.
- What is the best way to communicate status to a single user? To many users?
- Which tools are good matches to this model? What tools are missing?
- Some SAs feel that after a problem is fixed, one should reboot the host and verify that the problem doesn't reappear. Other operating systems are known to have most common problems fixed via a reboot. How do these situations fit into the model?

Conclusion

I have presented a model that deconstructs the process of users requesting and receiving support in hopes of making the process repeatable, easier to teach, and easier to improve and manage. The process has four phases: "The Greeting," "Problem Identification," "Planning and Execution," "Fix and verify." Each phase has distinct steps.

By following this model the process becomes more structured and formalized. The process is something highly akin to the scientific process: observe, hypothesize, test, repeat.

Phase	Steps	Role
Phase A "Hello!"	**Step 1: The Greeting**	Greeter
Phase B "What's wrong?"	**Step 2: Problem Classification**	Classifier
	Step 3: Problem Statement	Recorder
	Step 4: Problem Verification	Reproducer
Phase C "Fix it"	**Step 5: Solution Proposals**	Subject Matter Expert
	Step 6: Solution Selection	
	Step 7: Execution	Craft
Phase D "Verify it"	**Step 8: Craft Verification**	Craft
	Step 9: User Verification / Closing	Customer

Figure 6: Overview of problem solution phases.

Analyzing the execution of each step as well as viewing the entire process holistically are fruitful sources for improving the way user requests are handled in an organization. In addition, having a process makes measurement possible.

The nine steps should be integrated into training programs for SAs. If all SAs used the same terminology to describe their processes it would help communication between SAs. While knowledge of the model can improve a SA's effectiveness by leveling the playing field, it is not a panacea; nor it is a replacement for a creativity, experience, having the right resources, etc. Users that understand these steps can be our best customers because they become part of the process.

Deconstructing the process has permitted a deeper analysis of this important portion of our field. Other parts of system administration could benefit from similar analysis.

Acknowledgements

I would like to thank Eric Anderson, Josh Simon, Tommy Reingold and Jay Stiles for their editing, feedback and suggestions.

References

[Arch93] Archer, Barrie, "Towards a POSIX Standard for Software Administration," Systems Administration (LISA VII) Conference, Monterey, CA, pp. 67-79, 1993.

[Bent93] Bent, Wilson, "System Administration as a User Interface: An Extended Metaphor," Systems Administration (LISA VII) Conference, Monterey, CA, pp. 209-212, 1993.

[Hunt93] Hunter, Tim and Watanabe, Scott, "Guerrilla System Administration," Systems Administration (LISA VII) Conference, Monterey, CA, pp. 99-105, 1993.

[Kubi92] Kubicki, Kubicki, "Customer Satisfaction Metrics and Measurement," Systems Administration (LISA VI) Conference, Long Beach, CA, pp. 63-68, 1992.

[Kubi93] Kubicki, Carol, "The System Administration Maturity Model – SAMM," Systems Administration (LISA VII) Conference, Monterey, CA, pp. 213-225, 1993.

[Mani87] Maniago, Pierette, ‘‘Consulting via Mail at Andrew,’’ Large Installation System Administrators Workshop Proceedings, Philadelphia, PA, pp. 22-23, 1997.

[McNu93a] McNutt, Dinah, ‘‘Role-based System Administration or Who, What, Where, and How,’’ Systems Administration (LISA VII) Conference, Monterey, CA, pp. 107-112, 1993.

[Ment93] Menter, E. Scott, ‘‘Managing the Mission Critical Environment,’’ Systems Administration (LISA VII) Conference, Monterey, CA, pp. 81-86, 1993.

[Scha92a] Schafer, Peg, ‘‘Is Centralized System Administration the Answer?,’’ Systems Administration (LISA VI) Conference, Long Beach, CA, pp. 55-61, 1992.

[Trocki] ‘‘mon’’ by Jimi Trocki, Service Monitoring Daemon, http://ftp.kernel.org/software/mon/ .

[Zwic90] Zwicky, Elizabeth D., Steve Simmons, and Ron Dalton, ‘‘Policy as a System Administration Tool,’’ LISA IV Conference Proceedings, Colorado Springs, CO, pp. 115-124, 1990.

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, and in 1996 decided to focus on networks, not operating systems. He holds a B.A. in C.S. from Drew University, Madison, New Jersey. Reach him via U.S. Mail at Lucent Technologies, Room 2T-408, 600 Mountain Ave, PO Box 636, Murray Hill, NJ 07974-0636. Reach him electronically at <tal@lucent.com>. His web page is http://www.bell-labs.com/user/tal .