Getting More Work Out Of Work Tracking Systems

Elizabeth D. Zwicky – Silicon Graphics

ABSTRACT

This paper discusses work initially done for SRI International's Information, Telecommunication, and Automation Division (ITAD). ITAD's computer facility staff originally implemented a work tracking system to avoid the embarrassment of discovering that some important user problem had been brought to their attention and then entirely forgotten. Over the years, the system also began to address more complex tasks, and is now used to deal with some problems before users report them, to better communicate with the users about the amount of work being done, and to get those minor housekeeping chores that kept accumulating done at last. This paper explains how.

Work Tracking Systems

Work tracking systems, also commonly known as trouble ticket systems, are systems that keep track of user problems for a group of people. These have a family resemblance to the systems used to track bugs in software (for instance, they generally allow you to assign responsibility to a person, set priorities, and open and close entries), but differ in some basic assumptions (for instance, bug tracking systems generally assume that a bug can be assigned to a particular product, that it only needs to be fixed once, and that the database of bugs will be searched by naive users, while work tracking systems generally assume that the problem may have multiple causes, that it may recur in other places, and that the database will be searched only by the people fixing things). There are both commercial and public domain systems available, at various levels of complexity and specialization for system administration. At the most complex end, problems are submitted through a special program with a graphical user interface and tracked in a relational database. At the simplest end, problems are submitted through electronic mail and tracked with some pseudo-database.

ITAD's Tracking System

ITAD's tracking system, internally called the ''action queue'', is very much at the simpler (and cheaper) end of the scale. It was developed at ITAD but modeled on the University of Colorado's QueueMH. ITAD's system is discussed in [7] and Colorado's in [2]. Users submit problem reports in e-mail, and the people dealing with them use modified MH commands to set priorities, list outstanding problems, update reports, and close them when the problems have been fixed. This is well suited for ITAD's environment, for several reasons: ITAD considers it important for users to be able to submit problems directly into the work tracking system; MH is already the standard mail system; and there is a need to support users on a wide range of platforms, making it difficult to port beautiful graphical user interfaces. Some of ITAD's users are also strongly resistant to learning new programs, however beautiful, and since they all already knew how to use some mail system, and were accustomed to sending electronic mail requests to ''action'', maintaining this interface kept them happy.

Why Get Fancy?

In late 1992, ITAD's computer facility staff became embroiled in yet another discussion of exactly what it is that the computer facility does and why it takes so long. With four full-time system administrators and only a few problems coming in per day, people didn't understand why problems weren't fixed immediately. The system administrators didn't understand why nobody had yet snapped under the strain, since in addition to the few hundred items that were in the queue, they were dealing with all the maintenance and upgrades that weren't direct user requests, plus all the telephone requests, questions in the hall, and notes taped to doors.

The computer facility developed a new purpose in life; putting absolutely everything that needed to be done into the action queue. When this initiative started, there were a few hundred items in the queue, and 10 to 20 of them were resolved each week. Currently, there are about 1,200 items in the queue, and approximately 150 are resolved each week. (At the queue's peak, there were more than 2,000 queue items.) Even the most skeptical of users is capable of figuring out that this means that the computer facility does a lot of stuff, and has even more stuff left to do.

The change took several months, and required new software and new procedures.

The Software Side

Aside from the action queue software itself, there is a set of programs I call ''complainers''. Each one of these programs detects problems in some particular area; for instance, one of them compares the nameserver's hostname to address mappings with its address to hostname mappings to make certain they match. These programs output messages, separated by a blank line. A separate program, imaginatively named ''complain'', takes messages separated by blank lines as input, compares their subject lines to those of messages currently in the queue, and submits them if they are not already present. This makes certain that any given problem is reported only once. `complain` also introduces a 5 second delay between messages to avoid overrunning the mail queues. (This is a common problem with programs that are automated mail senders, since they tend to be lightweight compared to the mail system. A good, up-to-date sendmail configuration will keep you from drowning the receiving machine, but not from using up all the memory on the sending machine.)

These programs are run from `cron`, via a small script which has multiple names. It looks for a directory in /usr/facility/complainers that is named after the script, runs all the programs in it, and pipes the output to `complain`. It is run every evening as ''complain-daily'' and once a week as ''complain-weekly''.

Currently, the daily complainers are `bad-aliases`, `bad-time`, `dumps`, `todays-errors` and `user-problems`. The weekly complainers are `hosts-not-hosts`, `missing-man-pages`, `multi-mount` and `no-modules`.

bad-aliases

`bad-aliases` runs through the aliases file, looking for aliases that include files that don't exist, refer to local users that don't exist, or forward mail to remote hosts that return ''User unknown'' for the user we're forwarding to. For local lists, it reads through the files; for remote hosts, it uses a local program called `mailaddr` that contacts the remote host and does an SMTP VRFY. It would be nice to weed out forwards to non-existent hosts, but there is no easy way to distinguish between a host that doesn't exist and a host that isn't reachable at the moment. `bad-aliases` also reads the comments in our files looking for a date, and complains if the date is more than 6 months old; we use this for mailing list aging.

bad-time

Theoretically, all of ITAD's machines run ntp or otherwise synchronize clocks. In practice, ntpd occasionally dies, and some users have root, modify rc.local in odd ways, and fail to notice that the time is wrong until they call up complaining that `make` is behaving oddly (because they're 8 minutes off the time their file server has). `bad-time` checks that all hosts are within 5 minutes of the time on the host running the complainers. It ignores hosts in a different time zone from the host it's running on. `bad-time` is ITAD-specific only in the code that identifies the hosts to run on and the specific message it outputs; otherwise it only needs to be able to run `date` on the remote hosts. Any site that has a simple way to get all remote hosts to provide the time in number of seconds since the epoch, instead of in human-readable form, would probably be much better served to rewrite the program, since that would be simpler and more reliable than `bad-time`'s method of parsing the human-readable date. (It would be a service to the universe if somebody managed to make it standard for `date` to provide a format code for ''I don't want a beautiful date, just give me the seconds''. I have known sites that simply installed a `stupiddate` program for this purpose.)

dumps

`dumps` is another doublecheck. ITAD's backup system sends mail when it's working and when it encounters errors, but this complainer checks the dump frequency field against the data in /etc/dumpdates as a doublecheck. This will catch, among other things, disks that have been assigned to a dump run that is never performed. The basic concept behind `dumps` is applicable to any backup system that uses `dump` as its transport agent and runs on machines with a dump frequency field in fstab, but code changes will be necessary to adapt to the values other sites use in the fstab dump frequency field. It could be adapted to any situation where the information about what should have been backed up and what was backed up is available to a program other than the backup system.

todays-errors

`todays-errors` is one of the most complex and useful complainers. It reads through today's messages in the syslog or equivalent message log on each of ITAD's machines, sorting the messages into three classes; messages it knows are ignorable, messages it knows are bad, and messages it doesn't recognize. Ignorable messages include normal boot messages, informational messages from programs that insist on logging their startup or automated restart (`routed`, `mrouted`, and `named` for instance), and messages about missing services that occur while the servers are down for backup. Messages which are recognized as specific problems with specific advice provided include disk errors, `lpr` complaining about missing printers, and memory errors. In addition, it counts the number of reboots encountered, and sends a tailored message if it exceeds 3. Messages that are not recognized are sorted by the program that logged the message, and one action request is queued for each program.

This finds a number of problems that might not otherwise have been noticed, ranging from dying SCSI disks, which are now usually replaced **before** they completely fall apart and the user has no machine, to a user who believed that the right way to debug his X environment was to power-cycle his machine every time it hung. It also has discovered a number of things were perhaps better left concealed; both the `routed` and the `sendmail` version ITAD currently runs have bugs that cause them to log incorrect error messages, for instance. Then there are the mysteries of the universe, like the machine that logs ''Sun 4/600'' once every few months. That's the complete line. The machine is not a 4/600.

`todays-errors` is highly ITAD-specific, in that not only the code for figuring out which machines to run on, but also all of the regular expressions and tricks it uses to classify lines, are hard-coded into the program.

user-problems

`user-problems` compares /etc/passwd to what it can find of reality. For instance, ITAD has a centrally maintained on-line phone database, which user-problems compares to the phone number and location shown in /etc/passwd. `user-problems` also checks to make certain that users with valid passwords have valid home directories, that their mail is forwarded somewhere, and that they have calendars; conversely, it attempts to assure that users with ''*'' passwords do not have valid home directories, do not receive mail locally, and do not have calendars. (ITAD uses ''*'' to indicate users whose accounts have been removed, and ''**'' for administrative accounts that the program should ignore.)

This serves partly as a check to make certain that people have installed accounts using the programs for that purpose, instead of by hand, and partly as a check on removing accounts, which we prefer not to automate. While it mostly catches botched installs, it also regularly catches people who have changed offices or phone numbers, inadvertent corruption in the online database, and other forms of bit- or brain-rot.

`user-problems` is extremely ITAD-specific, since it knows a great deal of detail about where things are, how they are formatted, and what different kinds of users should have. For instance, it enforces a specific GECOS format, including an expiration date, and cross-checks that against a particular file parsed by column position; it also insists that all users be on one of two specific mailing lists. Other sites are unlikely to find it useful except conceptually.

hosts-not-hosts

`hosts-not-hosts` compares /etc/hosts to the nameservice databases, and also compares forward and reverse data within the nameserver. When it first ran, it found a surprising number of hosts registered in only one place, or registered differently in A and PTR records. Once these were removed, people learned not to do that any more, and `hosts-not-hosts` rarely if ever finds any differences at this point, although there is still no automated program for adding hosts that would avoid these problems. It also compares the NIS ethers file to the hosts in the nameserver. ITAD considers the nameserver the primary source of host information, not the hosts file, and it maintains an ethers file for all hosts, regardless of whether they will ever need to use RARP. This is partly to aid in diskless booting of hosts that normally run dataless but don't at the time have functional operating systems locally, and partly to aid in the identification of hosts from packet traces on the network.

`hosts-not-hosts` is fairly general; it requires a hosts file containing only local hosts, tries to run `named.xfer` on the zone the host it's running on is in, and tries to ypcat the ethers map. If those conditions can be met, only a few strings need to be changed.

missing-man-pages

`missing-man-pages` compares programs in /usr/local/bin to manual pages in /usr/local/man, and complains when programs do not have matching manual pages. With programs like Perl and GNU Emacs, which want to install a version-numbered link, this requires you to link the manual pages as well; I haven't yet figured out whether I think this is a good thing or not. While it might possibly be helpful to users, there's no evidence that they ever notice, and it clutters /usr/local/man with links. The complainer has increased the likelihood that programs will have manual pages (although in practice not the likelihood that the program's installer will put them in place originally), but not to a certainty. Since it walks through all the binaries anyway, it also complains about compressed binaries (the complaint says neutrally that they are compressed and thus useless; some people choose to uncompress them, others to remove them). It should work at any site with minor and obvious changes to the particular binary and manual page directories.

multi-mount

ITAD's environment has an unusually large number of networks, and machines move from one to another with unusual frequency (particularly when you realize that machines are rarely physically moved). This sometimes results in machines receiving NFS service from multiple servers in situations where they could be using a single server. This cannot be entirely fixed by using automounts, since some of the file systems involved are not interchangeable, and furthermore many of the users complained that even though automount would normally give them the closest server, it was not guaranteed to be deterministic; they wanted the server closest in network terms, not currently fastest in responding, and they wanted it to be the same every time. `multi-mount` detects cases in which the user's email, the user's home directory, and the /usr/local mounted on the user's home machine are not all three on the same NFS file server. This is unlikely to even be an interesting question at most sites, but except for the code that determines the host that holds the user's home directory, it's perfectly portable.

no-modules

`no-modules` detects software packages installed in their appropriate central location which do not also have files for use with the Modules package, and which do not have ''.nomodule'' files in their top directory. Modules, described in [1] allows users to add programs to their environment easily. With directory names changed, it should work for most sites using Modules.

Why Complainers and Not Fixers

People usually ask why ITAD uses complainers and not fixers. In fact, there are fixers for some problems, but many problems are not amenable to automated solutions. A number of the complainers are backups to automated systems that are supposed to have fixed or avoided the problem already, but which occasionally fail or are circumvented. Others involve situations where either of two information sources might be correct (for instance, the password file or the phone database might have the correct phone number) and there is no automated way to distinguish. Some of them clearly are very risky to attempt to fix automatically; for instance, an invalid password entry with an existing home directory should not automatically be fixed by removing the home directory. Occasionally there is a complainer instead of a fixer simply because the complainers are much faster and easier to write, and serve to keep the facility in line while people look for the time to figure out how to fix or avoid the problem.

Complainers also have more applicability for the environment at Silicon Graphics, where the hosts have considerably autonomy, and it's appropriate for system administrators to check for problems, but not for them to actually make changes to other people's systems.

ITAD's Experiences With Complainers

In general, the complainers are an improvement. Problems are detected earlier, more often fixed before users complain about them, and things are generally tidier and more comprehensible. On the other hand, maintaining the complainers can be difficult, and has required some changes to the facility solely to make the complainers work (for instance, the use of * and ** passwords). Furthermore, it has proven to be difficult to get people to deal with complaints consistently and correctly. People have a tendency to remove items that they know are not problematic each time they occur, rather than trying to make the complainers stop complaining. People often fail to check for earlier occurrences of errors in order to merge new complaints with them, possibly because it's an action that's extremely rarely required except for items from the complainers. Real people rarely send in new complaints about the same problem with different subject lines from the original (and when they do, the two complaints are almost never correctly merged together).

There is also an unfortunate tendency to treat the complainers as humans. I've seen many people remove a complaint with an explanation of why the complainer's interpretation of the problem is wrong. (For instance, a complaint about a missing manual page may be removed with the notation that the manual page exists under another name.) Oddly enough, the software does not generally listen to such explanations. The instructions that it provides are not always correct either, and people have been known to take phrases like ''this probably means that ntp is not running'' as gospel (as per the famous saying ''garbage in, gospel out''), leading them into trying to figure out what the `ntp` problem is when the real problem was failure in the complainer to compensate for times that cross an hour boundary, and the clocks were not off. This is despite the fact that the message clearly lists two times less than five minutes apart, with the notation underneath that they are more than five minutes apart.

Underpinnings of the Complainers

Many of the complainers rely on the ability to find all hosts of particular types, which ITAD provides by maintaining a file called Hosts.Status listing various information about our hosts including hardware type, OS type, and a classification into server, dataless client, diskless client, and standalone. This file is built automatically from the nameserver database, using HOST and TXT records to keep the information, and contains a field marking whether the host was ''up'' or ''down'' (in reality, whether it responded to `ping`) last time the file was built. Hosts.Status is built automatically every night, and is used by a number of programs besides the complainers; it actually predates the complainers by several years.

Software Other Than the Complainers

The complainers are not the only programs that insert items into the action queue. Any program that can send mail can insert items into the queue, and ITAD tries to configure things as much as possible so that anything that needs administrative assistance sends mail directly into the queue. In particular Wietse Venema's TCP wrappers [9] mail information about refused TCP/IP connections directly into the queue. This simplifies detection of break-in attempts, at the cost of severely reducing our patience with people who insist on trying failed connections multiple times. Other security programs also send mail directly into the action queue; this allows us to deal with problems which should be looked at, but would be far too frequent and annoying to page a specific security person about.

Changes to the Underlying Trouble-Ticket System

The trouble-ticket system that was perfectly adequate with 200 items in the queue was not perfectly adequate with 2,000 queue items. We ended up making several after-market modifications. The first was to dramatically enhance the available reporting options, so that as well as getting a list sorted by submission date of the items assigned to you, you could get a list sorted by priority, and within priority have user-submitted tickets before auto-generated tickets, and within that have them ordered by the time since they were last responded to. This corresponds roughly to order of importance. Another report lists goals and the extent to which they were being met. For instance, one goal was to have each ticket assigned to a group and given a priority; another goal was to have no tickets over a year old.

There is also a staggeringly unpopular program called `whine` that sends you electronic mail telling you how many of your action items haven't been replied to recently enough for their priority level. This mechanism works beautifully for people who occasionally have a few overtime items. If you have people who routinely have several hundred, forget it; anyone with any sense simply puts mail filtering in. People who occasionally have several hundred are responsible for the bulk reply and bulk remove features that were added to the trouble ticket system (and are supposed to be used only an autogenerated items, as it's hard to come up with a bulk reply that works reasonably for human-generated items.)

The Human Side

ITAD made two fundamental changes in its procedures. First, everybody became fanatical about making certain that all requests from users went into the queue, no matter how the user submitted the request. Doing this is very difficult; it's hard to remember, it increases the time that short requests take, and it requires that you figure out how to restate the user's question so that it's immediately comprehensible, easy to type, and not insulting to the user. (It also required yet another change to the trouble ticket system to allow tickets to be submitted in a closed state.) On the other hand, it makes all those requests visible. Those little questions can eat away a day, leaving you feeling like you got nothing done whatsoever; the documentation proves that something did happen. When one of the ITAD users complained that the computer facility wasn't responsive enough to his requests, the queue logs proved he submitted more requests than anybody else in the division - as many, in fact, as the next several people added together - and this made management considerably less receptive to his complaints.

Second, everybody entered the things that ought to be done, even though users hadn't requested them. This exercise was enlightening, if not in the end terribly encouraging. Taking the time to consider exactly what needed to be done, independent of current crises, was an enjoyable and fruitful exercise. On the other hand, it eventually became clear that the volume of crises was such that a lot of these projects were never going to get done, no matter how useful they would be. This was not only inherently depressing, but also a major disincentive to put the queue items in, since they merely sat around for a year or so before getting removed. The final compromise position is that the facility staff now enters into the queue only those projects that they consider absolutely critical, not the ones that they consider merely highly desirable. This makes only a moderate difference in the number of them that actually get achieved, but a significant difference in the usability of the statistics in staffing arguments.

Results

This enterprise did have most of the benefits that were hoped for:

- Problems are detected earlier; more pro-active things happen.
- Projects of the system staff have the same weight and visibility as problems reported by users.
- It is easier to communicate the entire appalling scope of the problem to non-staff members.
- When the users try to play ''My queue is bigger than yours'' they always lose.

In addition, it turns out to have several unexpected benefits:

- People find removing items from the action queue inherently rewarding; trivial jobs that have been waiting for years actually get done.
- The action queue is handled by lots of people; since problems are now obvious to people besides the ones that already know how to fix them, more processes are documented.
- Making complainers has required that things be configured more consistently – programs need to be able to figure things out that people could guess at before.
- Developing complainers leads naturally to developing fixers.
- People who had previously claimed that it was extremely important for them to be told every day what was in the queue and what its priority was decided that maybe they really didn't care all that much after all. Similarly, a number of people who had wanted to see every closed action item realized finally that this information was not useful to them. This

noticeably reduced arguments about the titles of items and the exact relative priorities of things with low absolute priorities. (Are you going to do that when hell freezes over enough to skate on, or not until you can build ice fishing huts?)

Futures

As originally written, the complainers rely heavily on ITAD's environment. Furthermore, they have no options, no configuration files, and a depressingly small number of comments; changing what they do involves changing the Perl code. This is acceptable for small ones, where the code is rapidly comprehensible, but not for the larger ones, where the intent becomes buried and it is difficult to maintain the rule sets. Every complainer that needs to talk to multiple hosts contains the code for identifying hosts that meet specific criteria and running programs on them, allowing the complainers to either have common bugs that therefore have to be fixed multiple times, or even more fascinatingly to all have different bugs.

I am currently engaged in rewriting the complainers for a less centralized environment (notably, one where someone who is not me needs to be able to maintain them). The rewrite involves a single program which takes a rule file to specify how to find the host list, which ones to run on, and what conditions map to what complaints, looking at files or the output of single command lines. This master complainer will not handle every condition; it's appropriate mostly for checking for full disks, error messages in logs, and the like. It won't replace either bad-aliases or user-problems, so the architecture of having numerous small programs and the occasional intruding monolith will be maintained.

Related Software

There are a number of other packages aimed at monitoring networked systems to proactively detect problems, including [3], [8], [5], and [4]. These systems are in general considerably more beautiful than this one, but are also in general aimed at immediate notification (via graphs, beeping computers, or beeping beepers) and are not integrated with trouble ticket systems. This system, while unbeautiful, is highly flexible and requires little or nothing from the machines that are being monitored. Trouble ticket systems that have been documented include [8] and [6] in addition to the previously mentioned [7] and [2].

Author Information

Elizabeth D. Zwicky is a senior system administrator at Silicon Graphics, where she supports the company's internal system administrators. She prefers to program in languages that begin with a ''P'' and work for companies that have three-letter abbreviations containing the letter ''S''. Reach her via US Mail at Silicon Graphics, Mail Stop 730, 2011 N. Shoreline Boulevard, Mountain View, CA 94043-1389. Reach her electronically as zwicky@corp.sgi.com .

References

[1] John L. Furlani, ''Modules: Providing a Flexible User Environment'', Proceedings of the Fifth USENIX Large Installation Systems Administration Conference, 1991

[2] Tinsley Galyean, Trent Hein, and Evi Nemeth, ''Trouble-MH: A Work-Queue Management Package for a >3 Ring Circus'', Proceedings of the Fourth USENIX Large Installation Systems Administration Workshop, 1990

[3] Stephen E. Hansen and E. Todd Atkins, ''Automated System Monitoring and Notification with Swatch'', Proceedings of the Seventh USENIX Systems Administration Conference (LISA VII), 1993

[4] Darren Hardy and Herb M. Morreale, ''buzzerd: Automated Systems Monitoring with Notification in a Network Environment'', Proceedings of the Sixth USENIX Systems Administration Conference (LISA VI), 1992

[5] Richard W. Kint, ''SCRAPE (System Configuration, Resource And Process Exception) Monitor'', Proceedings of the Fifth USENIX Large Installation Systems Administration Conference, 1991

[6] David Koblas and Paul M. Moriarty, ''PITS: A Request Management System'', Proceedings of the Sixth USENIX Systems Administration Conference (LISA VI), 1992

[7] Bryan McDonald, ''QMH: A Problem Tracking System'', Proceedings of the World Conference on System Administration and Security, 1992

[8] James M. Sharp, ''Request: A Tool for Training New Sys Admins and Managing Old Ones'', Proceedings of the Sixth USENIX Systems Administration Conference (LISA VI), 1992

[8] Carl Shipley and Chinyow Wang, '' Monitoring Activity on a Large Unix Network with perl and Syslogd'', Proceedings of the Fifth USENIX Large Installation Systems Administration Conference, 1991

[9] Wietse Venemaa, ''TCP Wrapper: Network Monitoring, Access Control and Booby Traps'', Proceedings of the Third USENIX UNIX Security Symposium, 1992