USENIX Association

Proceedings of the 17th Large Installation Systems Administration Conference

San Diego, CA, USA
October 26–31, 2003

Peer Certification: Techniques and Tools for Reducing System Admin Support Burdens while Improving Customer Service

Stacy Purcell, Sally Hambridge, David Armstrong, Tod Oace, Matt Baker, and Jeff Sedayao – Intel Corp.

ABSTRACT

System administrators are under pressure to do more work and provide better customer service with fewer staff members. At the same time, other challenges emerge: constant interrupts, poor morale, career development needs. At Intel Online Services, we use peer certification to reduce system and network administration burdens while simultaneously improving both customer service and staff morale. Intel Online Services (IOS) has teams of system administrators specializing in various areas such as Virtual Private Networks (VPNs), mail, DNS, and firewalls. Before **peer certification** these specialists did all of an area's work, from completing routine changes and handling problem escalations, to doing engineering work. Peer certification was created as a way to add qualified personnel. Specialists certified their peers by having them pass oral content tests and by supervising them doing changes. Tools were created to simplify administration tasks and make them doable by nonspecialists, and varied in complexity and flexibility depending on the expertise needed to do the task. After implementing peer certification, the number of staff certified to make basic changes increased greatly, along with the number of changes made by front line staff, while the number of escalations decreased. Morale improved as interrupts were reduced and staff gained new areas to learn while customer issues and requests were resolved more quickly.

Introduction

In today's current economic environment, system administrators are under pressure to do more work, often with fewer staff members. Even as budgets are cut and staff are laid off, competitive pressures demand that system administrators deliver better customer service. Along the way, other challenges emerge: constant interrupts, poor morale, career development needs. At Intel Online Services, we used **peer certification** to reduce system and network administration burdens while simultaneously improving both customer service and staff morale. This paper describes the process of peer certification, the tools and automation needed to support it, and our experiences implementing it.

Intel Online Services (IOS) has teams of system administrators specializing in various areas such as Virtual Private Networks (VPNs), mail, DNS, and firewalls. Before peer certification these specialists did all of an area's work, from completing routine changes and handling problem escalations, to doing engineering work. In addition, IOS's configuration tools were powerful yet arcane and cryptic, making them difficult to use by nonspecialists. The first sections of the paper describe our problem environment and the requirements that we had for a certification program and for tools.

We created peer certification as a way to add qualified personnel. In addition to procedures and content tests, we also created tools to simplify administration tasks and make them doable by nonspecialists. The next sections of the paper cover our certification process and tool implementation.

After implementing peer certification, the number of staff certified to make basic changes increased greatly, along with the number of changes made by front line staff, while the number of escalations decreased. Morale improved as interrupts were reduced and staff gained new areas to learn while customer issues and requests were resolved more quickly. We examine our results in the next section of the paper. One thing we learned is that peer certification works better in some kinds of environments and for some kinds of problems than others. The final section of the paper discusses where peer certification is most appropriate and what to look for in tools.

Problem Environment

Why did we need peer certification? Intel Online Services (IOS) is a provider of managed web hosting services. IOS has teams of system administrators specializing in various areas such as Virtual Private Networks (VPNs), mail, DNS, and firewalls. Before peer certification, these specialists did all of an area's work, from completing routine changes, handling problem escalations, to doing engineering work. In addition, our system configuration tools had powerful but

arcane command line oriented interfaces [1] or equally cryptic graphical user interfaces.

The complexity, flexibility, and nonintegrated nature of these tools created a number of problems. They were generally arcane enough so that they were unusable by operations/help desk staff and other system administrators not expert in those specialties. Even for specialists, the interfaces to the utilities were complex enough to lead to mistakes. Some changes and troubleshooting tasks required manipulating multiple files and tools, lengthening the time it took to do those tasks as administrators were forced to change from tool or file to tool in order to complete the work. Switching like this also made it easy to make simple mistakes. As an example, let's look at some of the steps necessary to change a domain in DNS.

1. Locate domain file.
2. Open domain file for editing.
3. Add proper records to domain file.
4. Edit domain serial number to reflect date.
5. Save the domain file.
6. Execute ndc reload command with domain name.

Figure 1: Steps to change a domain in DNS.

Figure 1 shows that a system admin first needs to find the file where the domain is located. If he has appropriate access, he can open the file to add, change, or delete the proper records. Next, he has to edit the domain's serial number in a way that reflects the date. After saving the file, he needs to execute the ndc reload command. Moving from step to step takes time. Several of the steps provide opportunity to make simple errors. For example, text editors typically do not check configuration file syntax. They certainly do not check for non syntax errors such as setting the serial number/date far into the future.

At the same time, customer service suffered as customers had to wait for problems and changes to be handed off to specialists and then wait for results to be communicated back. If a problem or change reached a specialist administrator and more information or clarification was needed, even more time was lost contacting the customer.

This situation hurt morale. System administrators were constantly interrupted by routine changes and simple problem escalations (on call escalation reached up to 50 pages a week). More interesting tasks, such as evaluating new equipment or implementing new systems, were pushed aside by these interruptions. For their part, help desk staff were frustrated that they simply passed on requests and had no way to address them by themselves.

Here is a summary of our key problems:

1. Available tools made troubleshooting and change implementation doable only by limited group of specialists.
2. Available tools made troubleshooting and change implementation time consuming and made making errors easy.
3. Escalation handoffs led to extra time and potential errors.
4. Constant interrupts and escalations affected morale and productivity of admin teams.
5. Lack of ability to make changes proved frustrating for help desk staff and other administration.
6. Customer service, particularly the time to troubleshoot problems and make changes, needed to improve.

Certification Requirements

IOS clearly needed more people who could do routine changes and debug problems while being able to respond quickly to customers. Because of economic conditions, existing staff, usually other system administrators or help desk and operations personnel, was the only source of additional people. We needed a way to get our staff quickly up to a level of expertise that would allow them to quickly and correctly handle the most frequent changes and escalations, and we had to make sure that they were ready before we let them work in a particular area.

To make sure that staff were able to perform necessary tasks, we created a program to certify them as ready. To be effective, our certification programs had to meet a number of requirements. The first requirement was that certified staff could do the set of tasks we needed done and do them correctly. Having more staff but having them make many mistakes would be counterproductive. Also, a person might have a certification from an external entity, but that did not mean that they were able to effectively use the tools in our data centers or that they were effective at all [2].

Next, our certification processes needed to deal with the fact that different tasks have different expertise requirements. Some changes or basic troubleshooting were relatively simple, while others required more knowledge and experience.

Finally, our certification program had to effectively deal with our workload. We knew that there were certain changes and troubleshooting requests that occurred most frequently and took most of the time. That fact implied that the Pareto Principle [3, 4] was at work. Pareto analysis is used as a quality tool through much of Intel. The Pareto principle says that about 80% of all requests (or problems) are from 20% of request types. For example, in the DNS arena, most of the change requests we received were modifications to existing domain or requests to host new domains. Other requests, such secondarying domains or adjusting BIND parameters, did occur, but much less frequently. Thus our certification programs had to cover the 20% of tasks that make up the 80% of requests. The Pareto Principle tells us that if we did not deal that critical 20% of requests, we would not be

reducing our workload significantly. It also implies that dealing with the remaining 80% of request types has diminishing returns. The Pareto Principle became very important in terms of where to invest our development energies and what benefits we would get.

Tool Requirements

It was very clear that the way that we implemented changes and did troubleshooting would have to change. The fact that our certification program would increase the number of people doing changes created new demands. The old way of doing things was to work out of root level accounts. If there were problems caused by a change, we knew that only a few people had that kind of access, and we could contact them.

With the prospect of many more people doing changes, our tools would need to authenticate who was making changes. Once authenticated, the tools should only allow changes that the tool user was certified to make. In case there were problems, we wanted audit trails to track who made what changes, and we wanted a way to back out changes if possible.

To enable staff to get up to speed quickly, our tools had to be easy to use and easy to learn to use. We also had to make sure that that tools prevented errors from being made. Finally, since we did not have a dedicated tools development team, the system administrators in each area would need to develop their own tools. We summarize our tool requirements as follows:

1. User authentication
2. Limiting tasks to those certified
3. Audit trails of changes made
4. Ability to back out changes
5. Easy to use
6. Easy to learn to use
7. Error prevention
8. Implemented by system administrators

Implementing Peer Certification

The first step in implementing peer certification was to partition changes and troubleshooting into distinct specialties. The obvious place to way to group tasks was by system administrator specialty, so we created certifications programs for mail, DNS, VPN, bandwidth management, and firewall access control lists.

The next step was to create three levels of tasks within each specialty: basic, intermediate, and advanced. Basic contains the most common and straightforward change and debugging tasks. Intermediate level means that more complex changes and troubleshooting can be done. Advanced is the highest level of competence, typically involving deeper levels of configuration, such as modifying sendmail [5] macro files or altering BIND [6] parameters.

One of our goals in creating these distinct certification areas and levels was to create achievable milestones. While we could have created a basic certification that covered VPN, mail, and DNS, that would have been harder for staff to achieve because of the breadth required.

In a similar fashion, we could have created a DNS certification that covered basic through advanced knowledge, but that would have been equally difficult to achieve. Partitioning certifications in this manner also provides paths with recognized milestones that staff can follow to improve their skills and their careers.

Another key goal in partitioning expertise was to get most of the frequent and easy changes and problems covered in basic level, keeping in mind the Pareto Principle. Creating expertise levels was also our way of dealing with the fact that different kinds of customer changes and debugging required different level of expertise. Figure 2 shows examples of different levels of tasks in the DNS area, and Figure 3 shows examples in the e-mail area.

1. Basic – Edit customer domains, provide zone information to customers
2. Intermediate – Create new domains, secondary domains
3. Advanced – Modify BIND daemon configuration parameters, adjust zone parameters

Figure 2: Examples of different levels of tasks in the DNS area.

1. Basic – Debug common mail problems using the log search utility
2. Intermediate – Set up aliases and virtual mail hosting for customers
3. Advanced – Edit sendmail .mc files.

Figure 3: Examples in the e-mail area.

To pass a certification level, a staff member usually needed to meet three requirements: passing a test on the specialty at the appropriate level, doing a certain number of supervised changes, and being certified at lower levels of certification if any (i.e., you must pass basic and intermediate before doing advanced). Test questions at each level were created to make sure a person being certified was competent at the specialty's area. These questions and all training materials were available online to provide access at any time and to anyone on any shift.

An area's system administrators conduct the tests orally, using the provided questions as a starting point. The testers were free to change the questions or ask more details based on those questions in order to make sure that the test taker really understands the subject and doesn't merely ''parrot'' back answers. This is one key advantage of the peer certification method over more formalized and rigid certification systems. Rather than relying on just a set number of questions, system administrator specialists can add or alter the questions to make sure that their peers comprehend

what is important and relevant. They have incentive to pass people in order to reduce workload, but since they handle escalations, they also have the incentive to make sure that people know their material and can do work correctly.

After the subject matter test, a total of four changes or problem troubleshooting need to be performed. This way, we ensure that the person being certified can actually do useful work. This is analogous to driving or flight experience. To get a drivers license, in addition to passing a test, you need to pass a road test. The changes are supervised by one of an area's specialists. Once again, this allows us to make sure that the person being certified is competent.

When a person passed an area certification, we notified the person's manager and announced that fact to the system administration team for that area and often to the entire data center, and added their name to a web page containing the list of people certified to make changes. This was intended to inform relevant staff that a new person was available to do work in an area. It also served as recognition and a reward for that person's work.

Tool Implementation

The next step was to create tools that would allow people with various levels of skill to safely do changes. We didn't have a dedicated group of tool developers, so all programming had to be done by the area specialists. To make the tools as accessible as possible, most of the tools we built were used via the web as CGI programs. Using of scripting languages such Perl [7] and Expect [8] helped us rapidly develop tools. Sharing code also helped speed up development. Since many of the tools involved using a web interface to modify text configuration files, some tools were converted into others. For example, the DNS zone tool editor originated as the ACL web tool.

Tool flexibility varied with the skill level (see Figure 4). The more expert the level, the more flexibility and choices the tools allowed; the less expert the level, the less flexibility allowed. This is to prevent staff with only basic expertise from making mistakes while granting more flexibility to the more expert.

We had requirements to log changes, provide an audit trail, and backout mechanisms. Users were authenticated using TACACS [9] and Radius [10]. To track changes, the changes, represented as file diffs, are sent to a mailing list of system administrators. The messages contain the ID of the person making changes. In addition, the mailing list is archived using hypermail [11], which makes it easy to browse through changes that have been made. Configuration files are checked into RCS [12] to make sure that we can recover older versions, and the ID of the change implementer is logged in the comment field.

Security proved to be a major concern, as mistakes in or sabotage of a customer's domain or their firewall configuration could be devastating. We had to safely execute certain functions (such as BIND's ndc program) as root, despite CGI's running under Apache run as user www. To do this, we resorted to setuid C programs that call other programs. Other security precautions include keeping direct input from the user to a minimum and carefully parsing input that we do receive to ensure that no dangerous input is accepted, such as shell escape characters.

The infrastructure needed for our tools was fairly simple, as shown in Figure 5. In each data center, we ran our web tools on a change server that was permitted to access and to make changes to network equipment and servers. This enabled our tools to share configuration information and take advantage of authentication infrastructure. There were separate authentication servers for TACACS and RADIUS. We also had a reporting server that continually took extracts of data and summarized

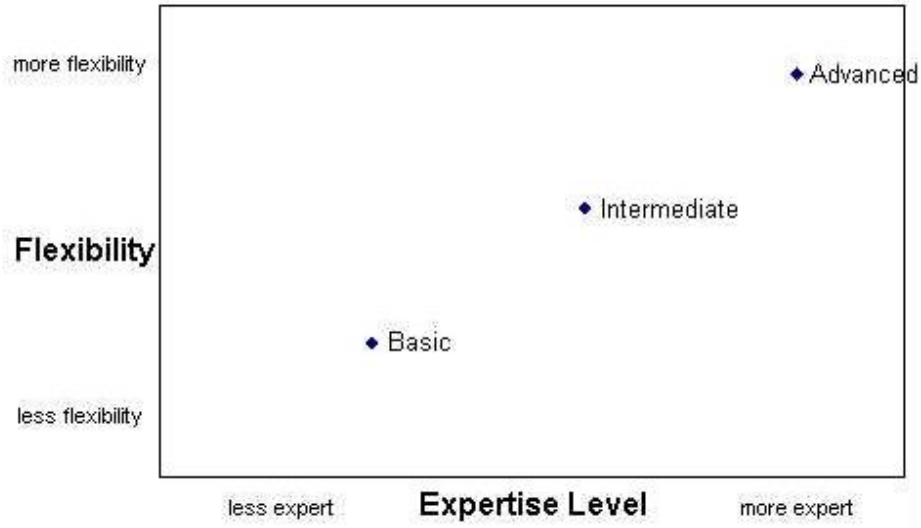

Figure 4: Flexibility vs. skill level.

them. Some tools ran on individual managed servers and were accessed directly through the web.

Results

We created tools to simplify change implementation and troubleshooting, allowing the number of staff certified to make basic DNS, VPN, and firewall changes to increase greatly (tripling in some cases). These tools and our certification program increased the number of problems resolved on the first call (without escalation or handoff). Figure 6 shows the number of escalations taken by the system administrator groups working on Firewall ACLs, DNS, mail, and VPN.

We started peer certification around work week 22 of 2002. Escalations peaked at 50 escalation during work week 26, and the number of pages started falling off as staff became certified and began doing changes and troubleshooting. Customer service simultaneously improved as changes and problems were turned around faster since the need to escalation was eliminated for most changes.

In one particular IOS data center, the percentage of DNS changes made by help desk/operations staff increased from 0% to 70%, close to that predicted by the Pareto Principle. As was experienced in a number of other environments, the certification programs increased morale. Help desk staff could now resolve problems directly and had opportunities to learn new skills and knowledge. They appreciated being recognized by their management and peers when they passed certification levels; this proved to be a cheap but effective reward system. Not only did system administrators enjoy the reduction in routine, simple work, but they also had new opportunities to learn skills in areas new to them.

Our tools generally worked well. We developed tools that did allowed staff with varying skills levels to

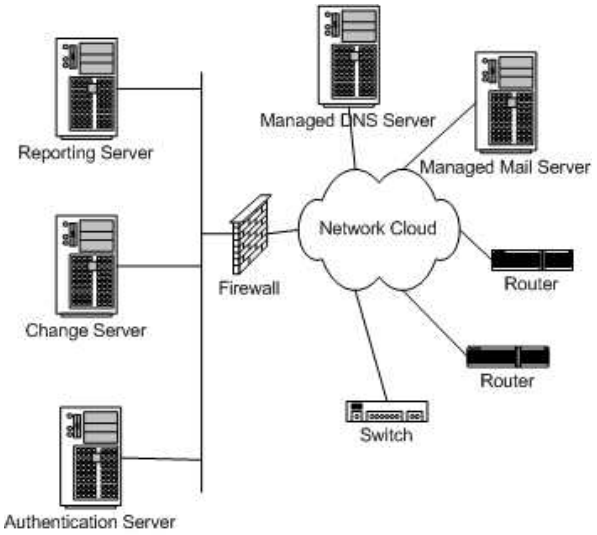

Figure 5: Tool infrastructure.

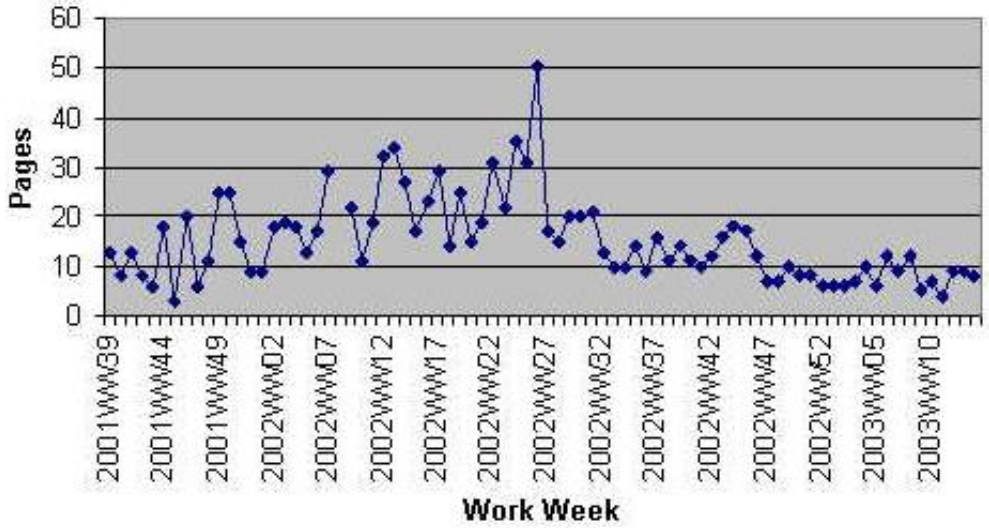

Figure 6: Escalation pages per work week.

do things like change VPN passwords, change customer DNS domains, and do firewall ACL changes. System administrators created all of these tools, and this proved advantageous for several reasons. First, we didn't have a dedicated tool development team or funding to outsource their creation, so system administrators were the only people who could create them.

Second, this sped development, as communication loop between developer and requester, along with the opportunities for misunderstandings and misinterpretations of requirements, was eliminated. The people who understand the problem the best were creating the tools. Also, using scripting languages like Perl allowed more than just a small select group of people to look at the tool source code and identify and fix problems. This would have been more difficult if all of the tools were built in C, C++, or Java. Finally, as mentioned above, the system administrators had a strong incentive toward making this work correctly, since they would be escalated to if there were problems.

Our certification processes and tools worked well for the most part, but there were a number of issues and problems that did come up. One surprising occurrence was that more people did not take the opportunity to become certified. Figure 7 shows who did basic DNS changes in our largest data center during from December 2002 through April 2003.

Staff Type	Number of Changes	Percentage
Core DNS staff	14	18.2%
Help Desk staff	43	55.8%
Other administration specialists	20	26.0%

Figure 7: Breakdown of DNS change ownership.

Figure 7 shows that 55.8% of changes were done by first line staff. We thought that help desk staff would try for every certification available, but we found out that motivation varied from person to person and from shift to shift. While first call resolution improved overall, there were certain shifts that did not have coverage for the basic certification in all areas. As a result, management in some data centers required their staff to become certified.

In other cases, some data centers seemed quite content to simply pass on requests. To some of our staff, the testing phase proved extremely intimidating. We had staff with test phobias, and while we knew that they could probably pass some of the tests, they elected not to take them. As a result, they were not certified. On the other hand, specialists in other systems administrator areas tended to take advantage of the certification programs. In general, they seemed to have a much higher motivation to learn new things and grow – a pattern that we saw in other data centers. Figure 7 has good news for the core DNS staff – where they once did 100% of DNS changes, they only had to do 18.2%. Ideally they would have to do 0%, but we encountered the obstacles mentioned above.

One problem that we encountered was getting enough changes and troubleshooting opportunities for people to get certified. Sometimes this was because of our own success. Certified front-line staff would do all of the changes and not leave any changes for people being certified. Shifts and time zones caused problems also. Some shifts often had few or no available area specialists to supervise changes, particularly night shifts at locations with most of the system administrators. In these cases, special arrangements had to be made for certification purposes. In some cases, the lack of changes was simply because of the breadth and depth of areas that we tried to create certifications.

Going by the Pareto principle, 80% of the changes and problems would be taken up by some 20% of the total types of changes and problems. Once we had covered those 20%, getting the remaining 80% would be distributed among 20% of the incoming requests. In our zeal to create certifications, we didn't always keep the principle in mind, and thus our time was poorly spent creating certifications in areas that had relatively few requests.

For some tasks that did not require a lot of upfront knowledge, our certification process was not used. The VPN group created a very simple interface for updating passwords. There was not a lot of knowledge needed to use it, so the tool was deployed without requiring certification. The VPN administrators gave a brief training session to prospective users, issued user IDs, and gave permission to start using the tool. Similarly, a tool was created by our network group to move systems from one Virtual LAN (VLAN) segment to another. The interface was very simple, and all the change implementer needed to know was the current and destination VLAN number. Access was also given after a brief training session.

For the most part, our tools worked well, but there were a number of areas that proved to be problematic. One area was the problem of ''leaky abstractions'' [13]. Our tools (and indeed, many automation tools) abstract and hide all of the mechanics of a change. For example, our DNS change tool accepts changes to a zone, edits a zone file, and then forces a reload of that zone file. To the help desk staffer making the change, all the change he or she does is bring up the domain, fill out a web form, and click on a ''propagate'' button. DNS zone changes have been abstracted to simply filling in a form, and the user assumes that the reload was effective. If there were a problem with loading the zone, that information would never make it back to the change

implementer. This is one example where the abstraction broke, and problems leaked through.

We had problems where our tools were too flexible, despite the precautions we took. By "too flexible" we mean that the tools allowed enough choices and ambiguity to cause problems. Our DNS change tool allowed the addition of DNS records without a record name. The default record name for an entry in BIND zone files is the previous record, and the zone if no records have been specified. Because of the way that many of our zone files looked, some change implementers assumed that the default record name of such entries was the domain name. This assumption caused a number of problems, which could have been avoided if the tool was less flexible and did not permit the addition of entries without record names.

Another tool problem we experienced involved the impact of our tools. We developed tools for our mail relays to measure and graphs the load average and mail queue length, and the top destinations for which mail is queued. We also developed a tool for browsing mail server logs. These tools were available via the Web in order to make them accessible to help desk staff and other system administrators. The problems occurred when the mail relays became heavily loaded. That was the time when we were very interested in the load average, mail queue length, and what were the top destinations in the mail queue.

Analyzing mail queue contents and checking mail logs caused significant additional load and made loading problems even worse. The lesson we learned here was that debugging tools should not make problems worse. A better implementation of the tools would have moved data off the servers where they could have been analyzed without impact.

Finally, we had the opportunity to push changes even closer to the customer by creating tools for them to make their own changes. Unlike staff, we couldn't certify our customers, so these tools had to be as simple and foolproof as possible. With end users, the flexibility vs expertise chart looks like Figure 8.

Our best success was a narrowly focused tool that allowed the customer to make specific DNS changes: changing an MX record and adding a certain CNAME record. The tool only allowed the customer to select a zone and then push buttons that did operations. This tool saved IOS staff from doing several changes a week. We had less success with a tool for converting DNS zones to use a distributed content vendor. The tool had multiple functions, such as browsing zones, and converting zones. It accepted domains in some forms and entire fully qualified domains names in other forms, and this caused confusion and questions from the customers. Again, excessive flexibility and ambiguity in interfaces caused problems.

During mid 2003, Intel left the web hosting business, moving all of the customers and the current environment to another provider. Peer certification made transition control of the environment much easier. In order to create the peer certification process in a system administration area, we had to document our environment and tools, create training for use of the tools, and have the documentation and training on-line and ready for anyone who wanted to learn.

Applying Peer Certification Techniques in Your Environment

The applicability of peer certification to other system administration environments depends on the conditions in those environments. The process of doing peer certification can be a weighty one, requiring testing and monitoring of a certain number of changes. From our experience, tasks that require some knowledge before changes are best served with a

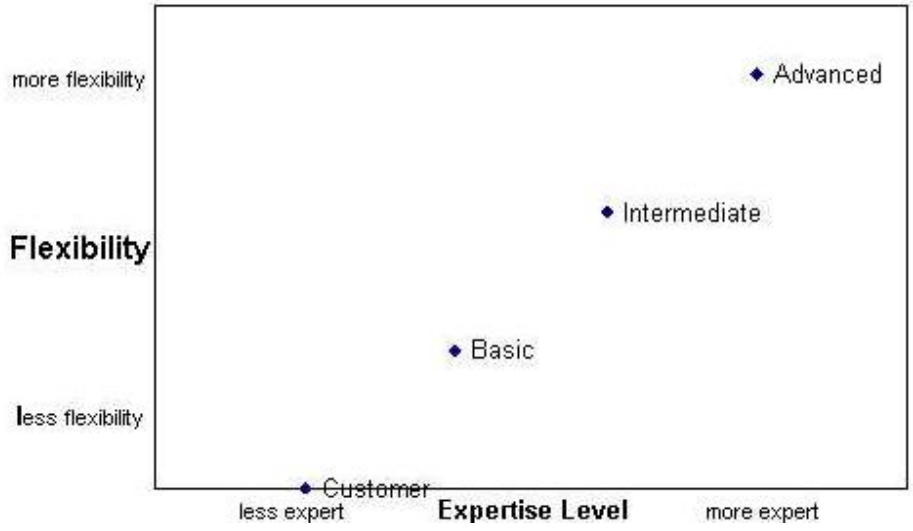

Figure 8: Flexibility vs. expertise level.

certification process. For example, no matter how easy a DNS zone modification web interface may be, the change implementer still needs to know what DNS is and the characteristics for different records. Mail debugging requires specialized knowledge, as do firewall rule changes. On the other hand, a task like doing password changes is fairly simple, and peer certification would be overkill.

Staffing environments are another consideration in applying peer certification. Environments where there are a lot of handoffs of problems or changes – either moving tasks to someone's work queue or escalating problems – are ideal for peer certification. These situations often involve help desks or call centers and present an opportunity to reduce handoffs and escalations, improve the time to service calls, and improve working conditions. On the other hand, a very small shop where a small staff handles almost everything would not benefit from peer certification, since most everyone would know everything anyway and the certification efforts would not gain much.

A very large environment would most likely benefit from peer certification, although certification work would have to go on continuously with staff turnover. Peer certification in a large environment would encounter some of the challenges we experienced, particularly dealing with many different zones and shifts. System administrators would have to be prepared to occasionally work at odd hours to supervise certifications, but at the same time, peer certification could be used to train other system administrators to do this part of the process.

To deploy peer certification effectively, web accessible tools that are easy to learn and use are essential. While tools we developed at Intel are not publicly available, there are a number of extensible tools that do similar functions, such as the Los Task Request System [14], Webmin [15], and Linuxconf [16]. We were not aware of these tools (a common problem [17]) before writing our own. Before using these kind of web based system administration, you will need to make sure that they have proper authentication, access controls, and audit trails. In addition, extensibility is very important, as you will likely need to write tools or modules to fit your own environment.

No matter what size the staff, some tool development and analysis techniques we used are usable in any environment. Pareto techniques are a great way to maximize the effectiveness of any automation effort. Charting what problems or changes occur most frequently and take the most time reveals where efforts will be most effective. Automating multiple step changes or functions into a single step is a time saver and can even persuade experienced system administrators to use automation tools.

Having system administrators choose or develop their own tools using scripting languages is an excellent strategy. It eliminates communication delays and misunderstandings that might occur between system administrators and a separate developer. Making tools and documentation available from a web browser opens up access, and should be coupled with making tool flexibility vary depending on the ability of the tool users. Any tool that implements changes needs to authenticate users and allow them to change what they are permitted, and there needs to be backout mechanisms. Finally, any monitoring tool should avoid negatively affecting what it is monitoring.

Conclusions

By increasing the number of qualified personnel, peer certification reduced system administrator burdens at Intel Online Services while improving customer support and increasing staff morale. Implementing peer certification requires defining tasks appropriate to differing levels of expertise, creating tests that measure competence, and creating automation tools that simplify and safeguard the process of making changes or debugging problems. Peer certification can help the most in environments where tasks have many handoffs before being done and those tasks require detailed knowledge in order to accomplish them. Areas to carefully consider when using peer certification are dealing with differing motivation levels, preventing leaky abstractions in tools, and making tools with appropriate flexibility.

Author Information

Sally Hambridge has worked for Intel Corporation since 1984, where she was a librarian, a database administrator, and a system administrator. She was with Intel Online Services from 1999 to 2003 as a Firewall Engineer and was team lead for the ACL team. She is reachable at sallyh@ludwig.sc.intel.com.

Stacy Purcell graduated from the Georgia Institute of Technology with a BS-CS in 1992. He joined Intel Corporation in Folsom, CA immediately after graduating where he has been employed in several roles including System Administrator, Network Engineer, and manager over the course of the last 11 years. Reach him electronically at stacy.p.purcell@intel.com .

Jeff Sedayao is a network engineer for Intel. Between 1987 and 1999, he architected and ran Intel's Internet connectivity, and after that, worked in Intel Online Services. His primary interests are in Internet performance, security, and policy implementation. Jeff can be reached electronically at jeff.sedayao@intel.com .

Tod Oace is a UNIX and networking systems engineer for Intel. Over the past 20 years he has worked with a wide range of computer platforms from micro to mainframe, and is proficient in several programming languages and networking protocols. Tod worked in the Firewall Engineering group of Intel Online Services. His email address is tod.r.oace@intel.com .

David Armstrong is a Network and Firewall Engineer. From 2000 to 2003, he worked in at Intel Online Services in the Firewall group, specializing in Firewalls, VPN connectivity, and DNS.

Matt Baker is a technical marketing engineer with Intel's Platform Networking Group. From 1998 through early 2000, Matt led Intel's broadband and VPN technology trials, designing and deploying one of the earliest large scale corporate xDSL and VPN remote-access networks. At Intel Online Services, Matt focused on the concept of the Datacenter, a place where internet service provider and enterprise issues can frequently converge and how these issues affect VPN connectivity/performance, network security design, and AAA systems design. He can be reached at Matt.W.Baker@intel.com .

References

[1] Hambridge, Sally L., Tod Oace, Jeff Sedayao, and Charles Smothers, "Just Type Make! Managing Firewall Using Make and Other Publicly Available Utilities," *First Usenix Conference on Network Administration*, Santa Clara, CA, April 1999.

[2] Vaas, Lisa, "IT Paper Chase," E-Week, http://www.eweek.com/article/0,3658,s%253D703%2526a%253D13923,00.asp , September 3, 2001.

[3] Donnini, Alex and Alan Miller, "Relieving the Burden of System Administration through Support Automation," *LISA 14 Proceedings*," December, 2000.

[4] Slater, D., "Call Center Management," *CIO Magazine*, http://www.cio.com/archive/040100_numbers.html , April, 2000.

[5] http://www.sendmail.org/ .

[6] http://www.isc.org/products/BIND/ .

[7] http://www.perl.org/ .

[8] Libes, Don, *Exploring Expect,* O'Reilly and Associates, Inc., Sebastopol, CA, 1995.

[9] Cisco Corporation, *TACACS+*, http://www.cisco.com/en/US/customer/tech/tk583/tk642/tech_protocol_family_home.html .

[10] Rigney, C., S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," *RFC 2865*, June, 2000.

[11] http://www.hypermail.org/ .

[12] Tichy, W. F., "RCS – A System for Version Control," *Software Practice & Experience*, Vol. 15, Num. 7, July, 1985, pp. 637-654.

[13] Spolsky, Joel, "The Law of Leaky Abstractions," *Joel on Software*, http://www. joelonsoftware.com/articles/LeakyAbstractions.html , November 11, 2002.

[14] Stepleton, Thomas, "Work-Augmented Laziness with the Los Task Request System," *Proceedings of LISA 2002: Sixteenth Usenix System Administration Conference*, November, 2002.

[15] Cameron, Jamie, et al., "Webmin," http://www.webmin.com/.

[16] Solucop, et al., "Linuxconf," http://www.solucorp.qc.ca/linuxconf/ .

[17] Burgess, Mark, "System Administration Research," *;login*, June, 2000.