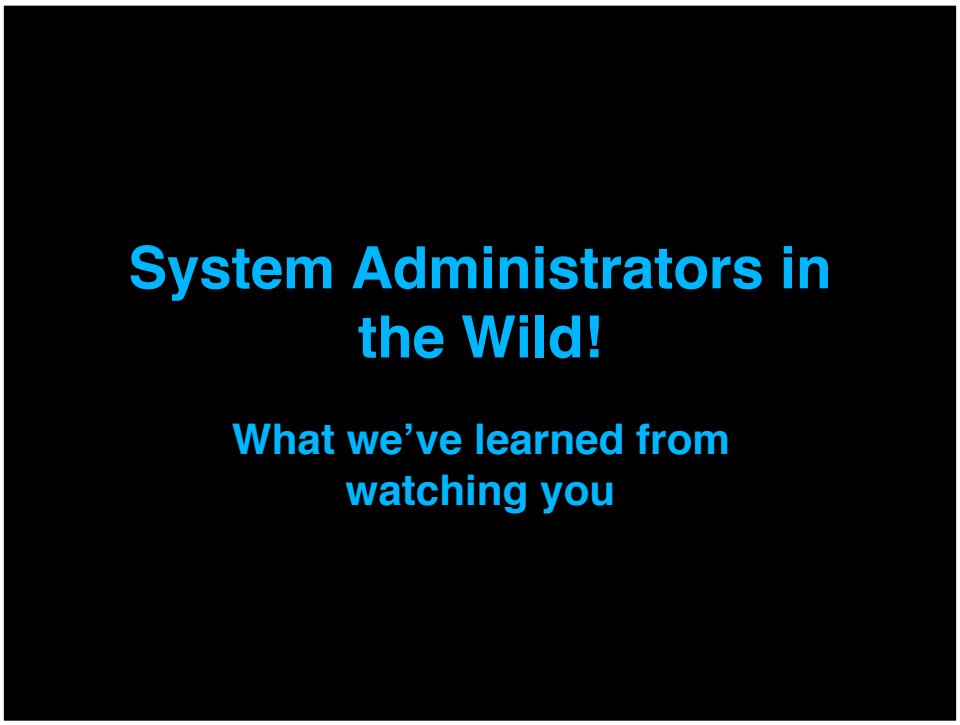
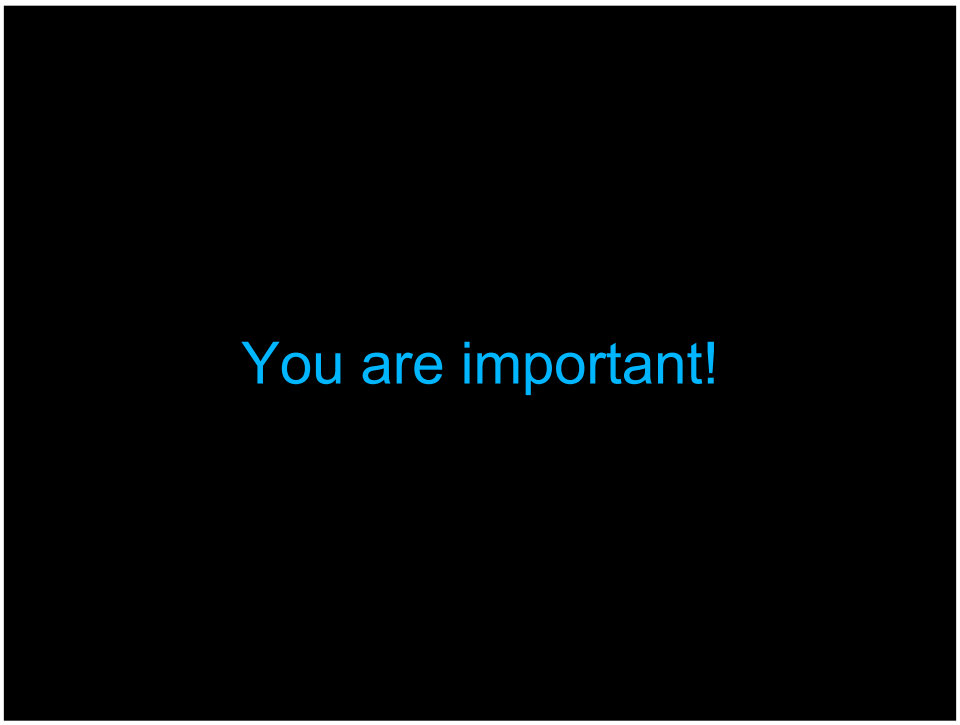
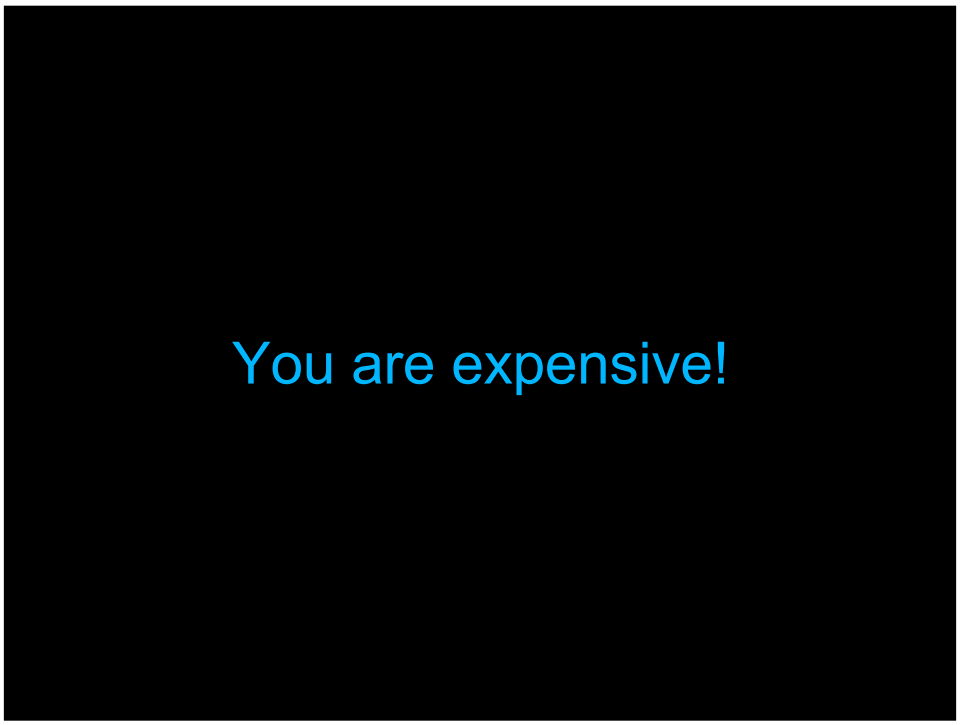


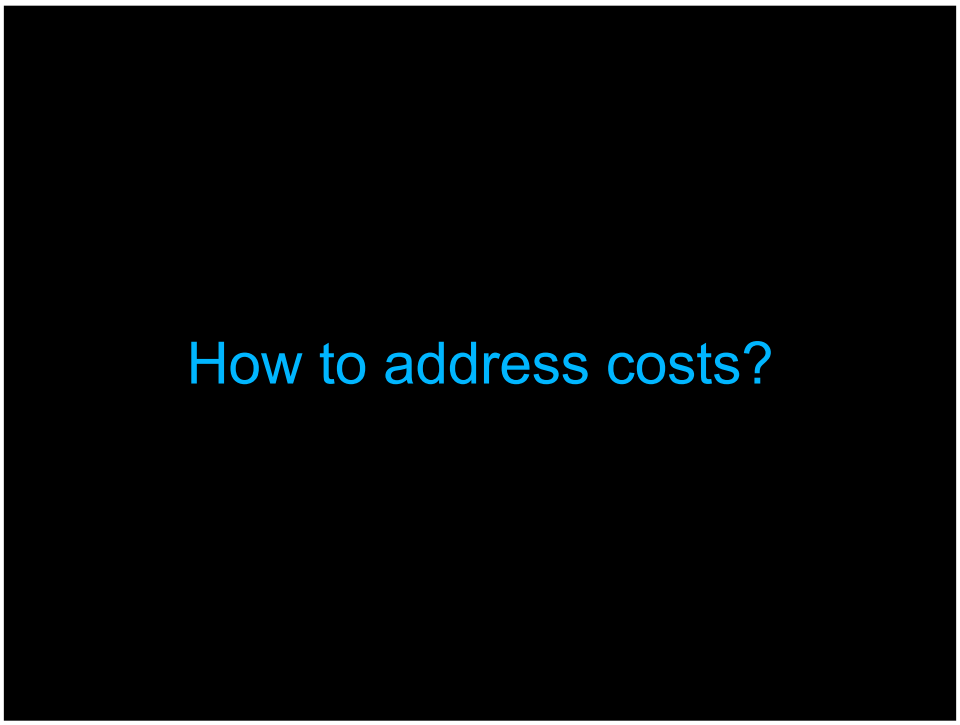
Good Morning! I am really honored to be here as an invited speaker, I've always been impressed by the quality of speakers here at LISA, and I'll do my best to live up to that standard. Thank you for coming to hear me. Today I'm going to talk about you! That's right, you! Why? Because we like you! But more significantly, because…



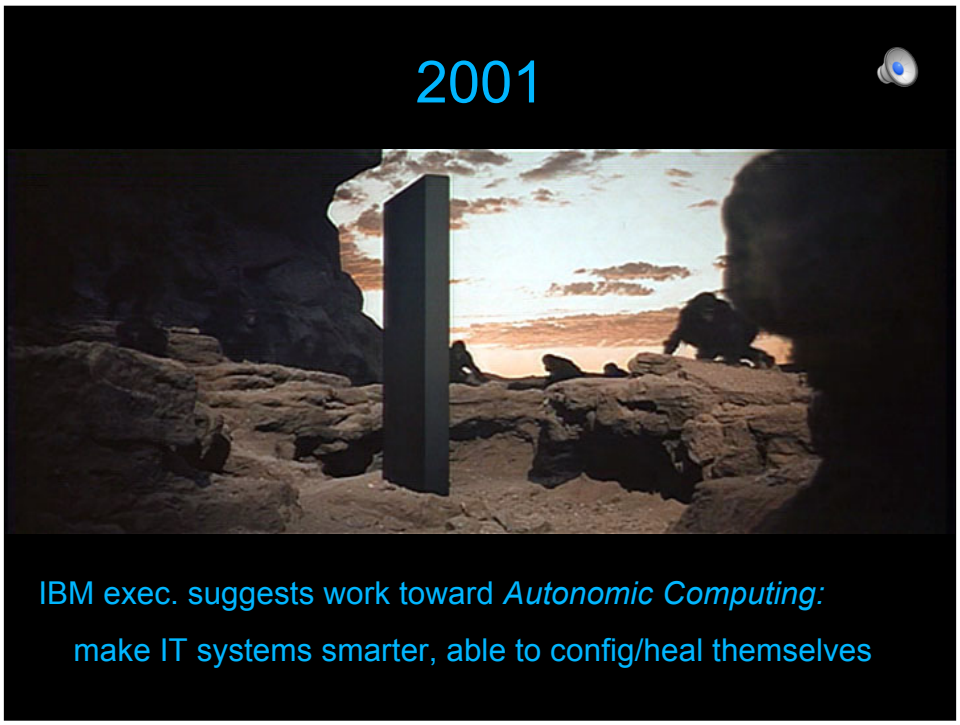
You are important! IT is so pervasive in today's world, that without your diligent efforts, we would lose the technological foundation upon which our civilization rests. You only have to look back at the efforts around Y2K to see how IT is no longer optional. It is what keeps everything going. There's a problem, though…



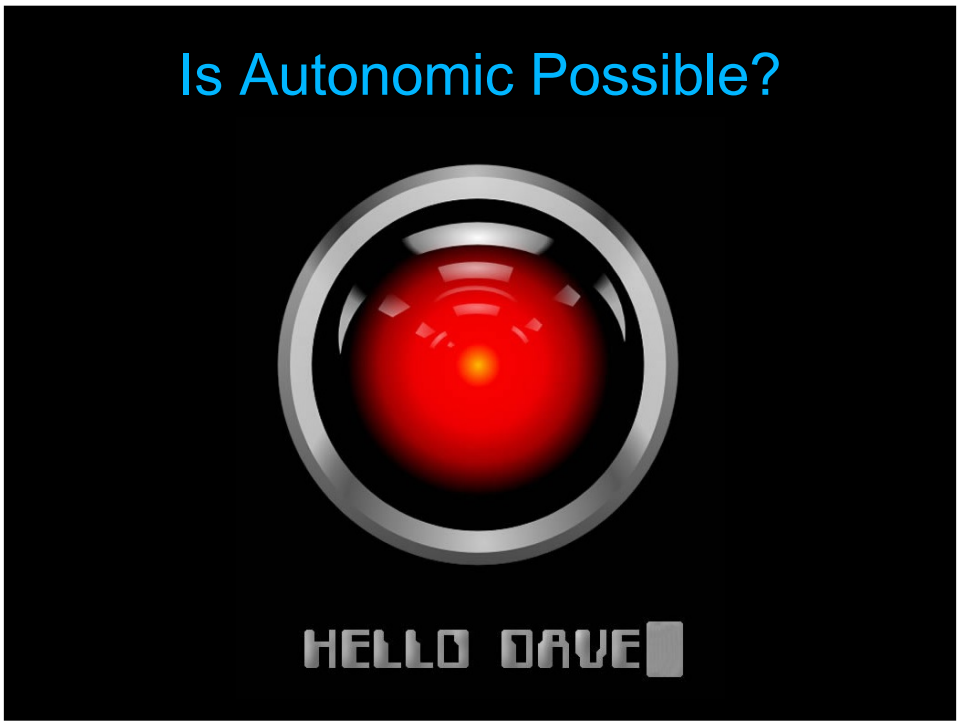
You are expensive! For decades the fraction of total-cost-of-system-ownership taken up by people has been growing. Once upon a time software and hardware dominated, but these days human costs make up over 70% of TCO. IMHO, there are two big causes for this: one is that systems are much more complex. Compare a web server in 1995 to an e-commerce web-site today. The other factor is that computers get faster and cheaper every year, but people don't. Our brains are the same size they were 1000 years ago. We do have better tools, but they aren't keeping up with the tasks at hand.



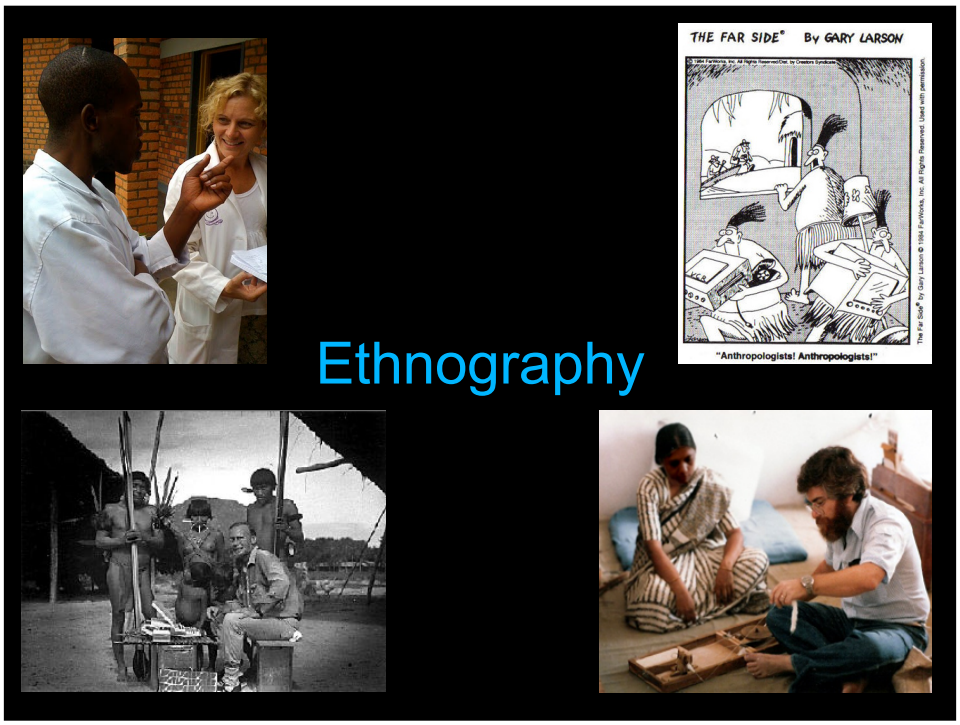
What can we do about this? Outsourcing? That certainly has challenges, though.



One idea came from the year 2001, when an IBM executive suggested that we work toward Autonomic Computing. What if we could make IT systems smarter, able to configure, optimize, protect, and heal themselves?



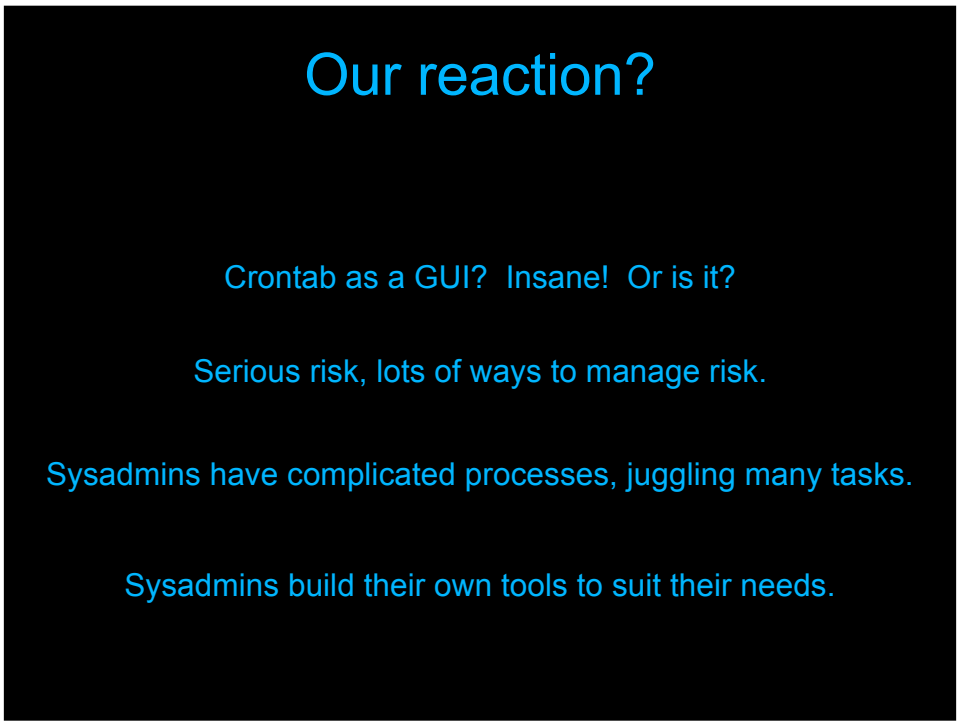
There was considerable debate as to whether this is possible, or even a meaningful concept. Computer science has a history of increasing automation, subsuming more details and permitting users and administrators to interact at higher and higher levels of abstraction, but only when the “black box” is either completely reliable or at least partly transparent. In any case, 9 years on IBM is not selling any “autonomic systems”, though many aspects of its hardware and software have improved. Back in the day, however, we had many questions about what system administration actually entailed, since you can’t automate something unless you know what it is. So a group of us decided to find out.



Our tool was Ethnography, which literally means writing about people, and it's a technique from anthropology for learning about unfamiliar groups by visiting them and observing their day-to-day activities, preferably as a participant. Now we weren't real anthropologists (even if we try to play them on TV), and while we couldn't spend six months or a year living among the natives, we were able to make 16 visits of up to a week across seven sites to observe to tools and works practices of system administrators. If you've ever heard David Blank-Edelman's great talk about the portrayals of sysadmins in popular culture, there are many misconceptions about who you are and what you do. With our field studies we hoped to develop a more accurate understanding of who you really are. …

Ethnography does have its limitations: it's extremely time and labor-intensive, and gives you a small temporal and population sample. Yet everything you see in the field is real, and if you see things often enough there's a good chance it's significant. Now ethnography is about collecting and understanding stories, so I'm going to start with a story from one of our earliest studies.

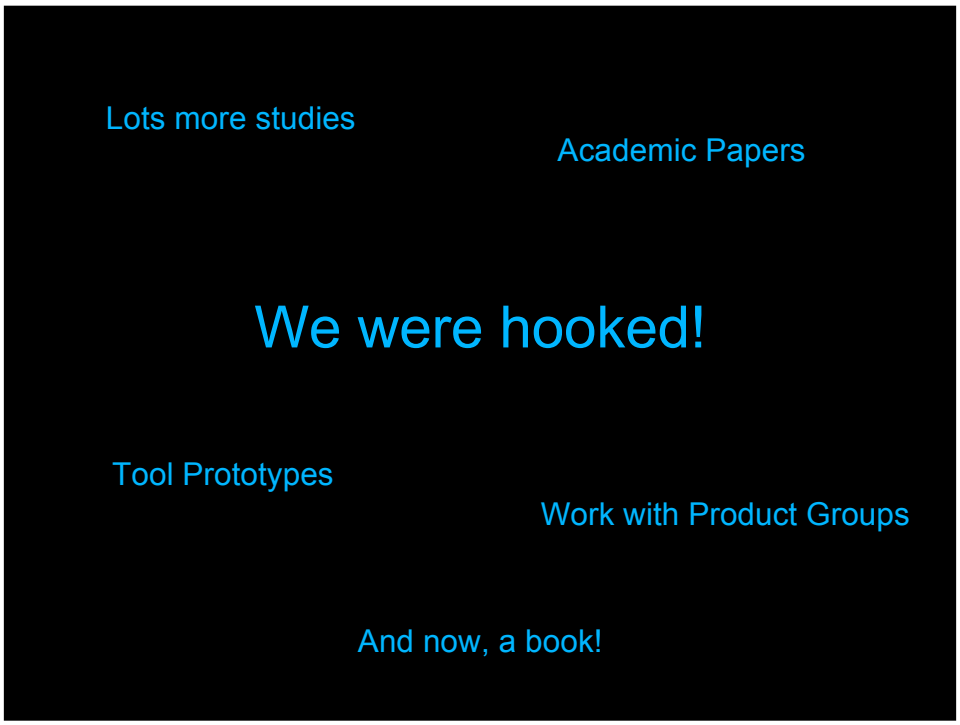
This is the story of Christine and Mike (not their real names), who were preparing to do a database tablespace move for one of their customers. There was only a short change window, so they spent the week before rehearsing the change on various test systems. This video was recorded on Friday, the day before the change window. Christine and Mike were preparing to do an online backup of the database that day, since the customer was still using the database, with the offline backup scheduled for the next day.



Our first reaction on seeing this was, “This is crazy! Crontab as a GUI? What were they thinking? No wonder they almost brought the database down.” With only a single character’s difference between offline and online, it’s easy to make mistakes. Yet on further reflection, this approach seemed better and better. All the common commands for this site were laid out with perfect precision, ready to be executed. And they didn’t, after all, bring down the database, since they had enough time to correct their mistake. And very few GUIs give you time to change your mind when invoking an operation.

We were also struck by the risk involved in this work. You can hear the panic in their voices as they think they might have brought the database down. Because of this risk, these admins had lots of ways to mitigate it, from practicing operations on test machines, to never ever typing a table name manually. They were managing SAP databases, which have 25,000 tables each with an 8-character name - typing a name yourself is way too risky, instead they’d copy the name from a document.

It’s also important to note how these admins had lengthy multi-step processes, with many tasks that they’re juggling at any given time. It seems clear that Christine made her mistake because she was thinking about both the online backup that day and the offline back-up the next.

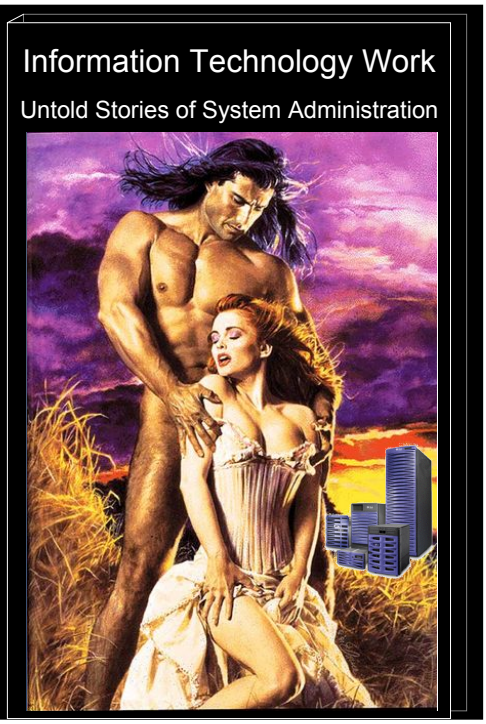


After seeing things like this in the field, we were hooked! We were learning stuff about your work that nobody ever told us, that didn't seem to be written up anywhere, that the people designing middleware tools within IBM didn't know. So we dove in, doing a bunch more field studies over the next few years, publishing some academic papers, producing prototype tools that we thought might help administrators (one of which we published at LISA), and working with IBM middleware product groups to try to improve their tools.

And finally, after many years, we wanted to share everything we've learned with the rest of the world, so we decided to write a book. Something to help designers, academics, hollywood script writers, and even CIOs better understand the work you do and the constraints you are living under. The book is centered entirely around stories that we collected in the field, stories of people like you.

The book is almost complete, so I'm here today to describe our findings, and get your feedback in case we've missed anything important.

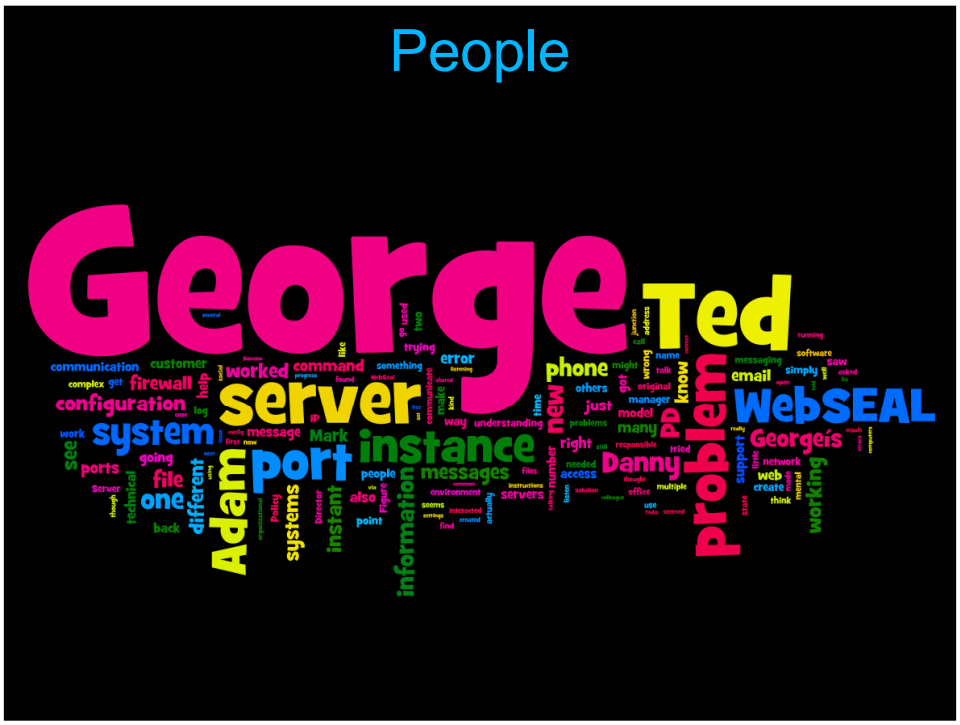
Our Book



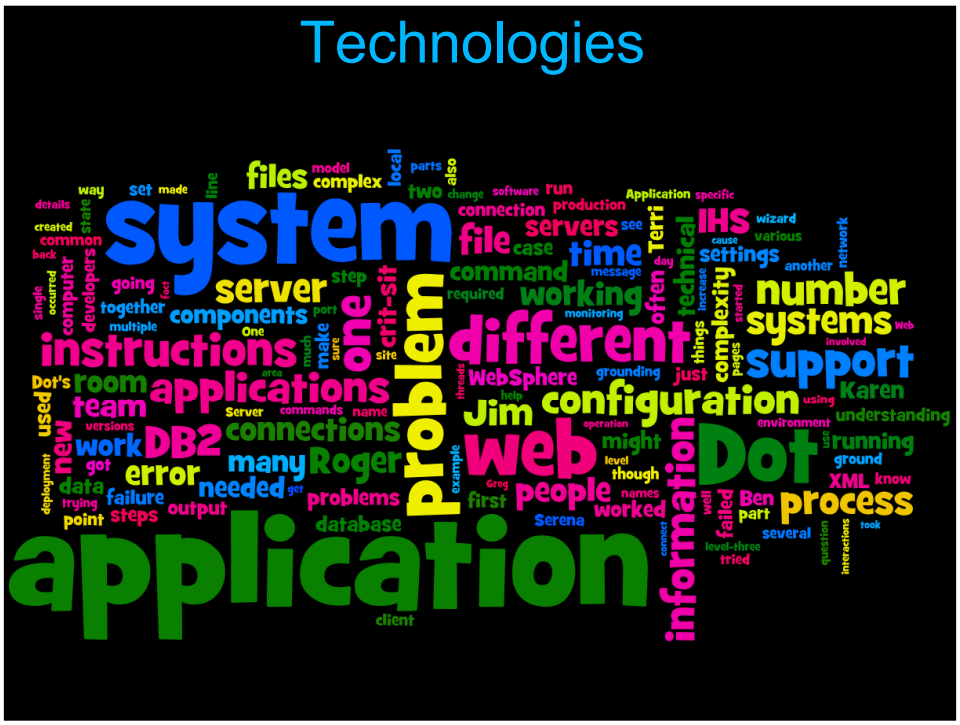
So, we have a book contract with Oxford Univsersity Press, and with the current schedule the book should be out some time in 2011. The working title is “Information Technology Work”, with the subtitle and cover art still up for discussion.

Our book has chapters highlighting different important aspects of your work:

People, Technology, Methods, Tools, Organizations, Communities, and a summary called IT Work.

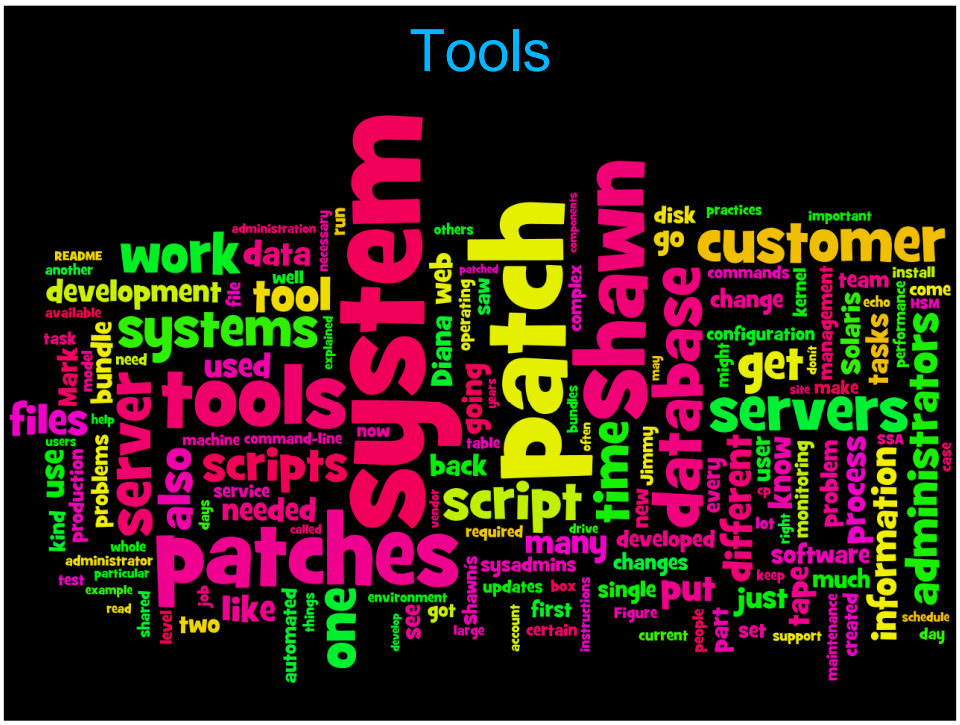


The People chapter is about interpersonal complexity, the intense collaboration, communication, and coordination that is a necessary part of IT Work. Systems are often too complicated for a single person to understand in detail, so teams of specialists work together to keep a system running. The chapter focuses on the story of an sysadmin called George, and one of his worst days ever, some of which I'll cover later in this talk.



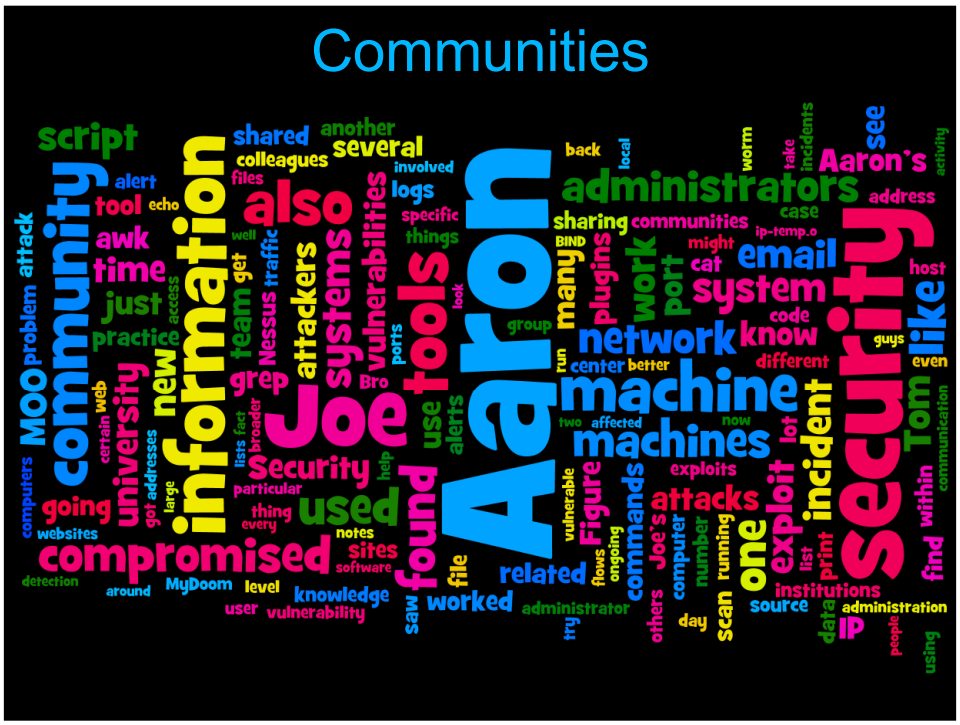
The Technology chapter describes the extreme technical complexity of modern IT systems. The prime example here is the IBM practice of a “crit-sit”: when a system’s performance reaches an unacceptable level, a team of responsible people are brought together and put into a single room and told to stay there until the problem is solved. Admins hate this, but it is often the only way the root cause can be found. In one case that we followed, a team spent over 8 weeks analyzing and fixing a problem with a web application. Technology is way too complicated when we’ve reached this point.

The Methods chapter is all about the practices that system administrators develop for managing in their risky and complex environment. Through various stories of DBAs Christine and Mike (who we saw earlier), we provide examples of all the methods and practices they used to ensure that their work went as smoothly as possible.



The Tools chapter discusses examples we saw of administrators creating and using scripts, web pages, cheat sheets, tool repositories, and even locally shared tools to ensure that their tasks could be executed consistently and reliably. We think that this creativity is one of the most important aspects of administrative work, allowing sysadmins to handle the indosycracies of their local systems.

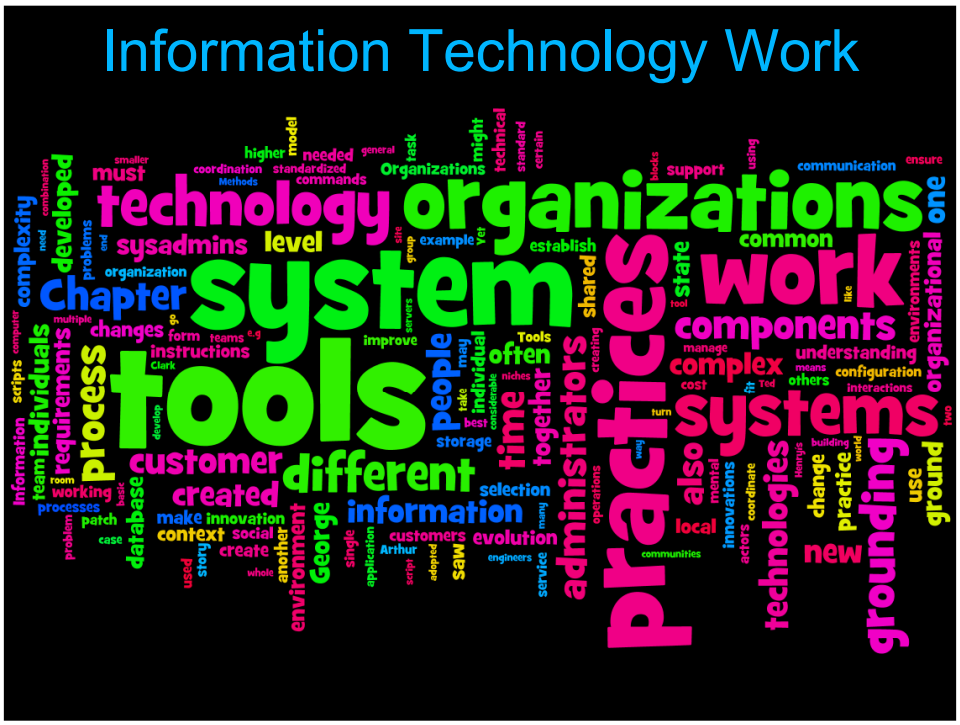
The Organizations chapter discusses how IT often relies upon careful coordination between a wide variety of organizations, from sales people to transition managers, technical support to account managers, architects to operators on the floor.



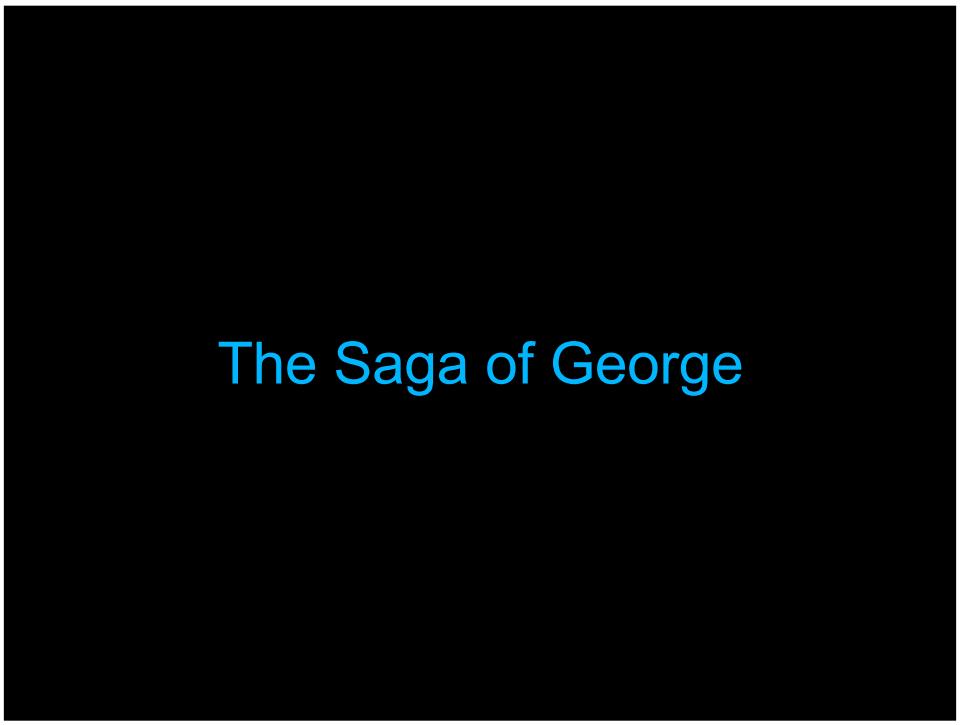
The Communities chapter discusses the importance of communities of practice in developing tools, sharing information, and solving broad problems. The examples in this chapter come from a study of security administrators, who are a truly fascinating bunch - they really do have people out to get them, and they wage information warfare trying to share and keep appropriate information and tools within their community while the attackers are doing the same.

For them, security is really a community activitiy, tracing attacks from site to site, sharing information from honeypots, finding and sharing vulnerabilities in attackers' tools and so on.

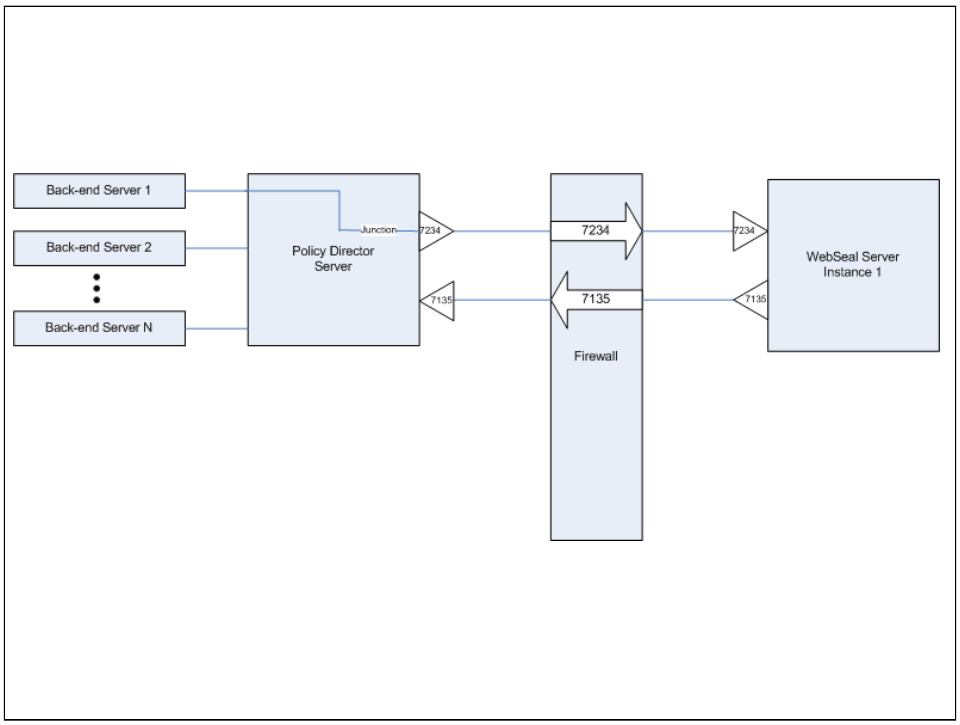
The development of open source administration tools is another important example of community activity.



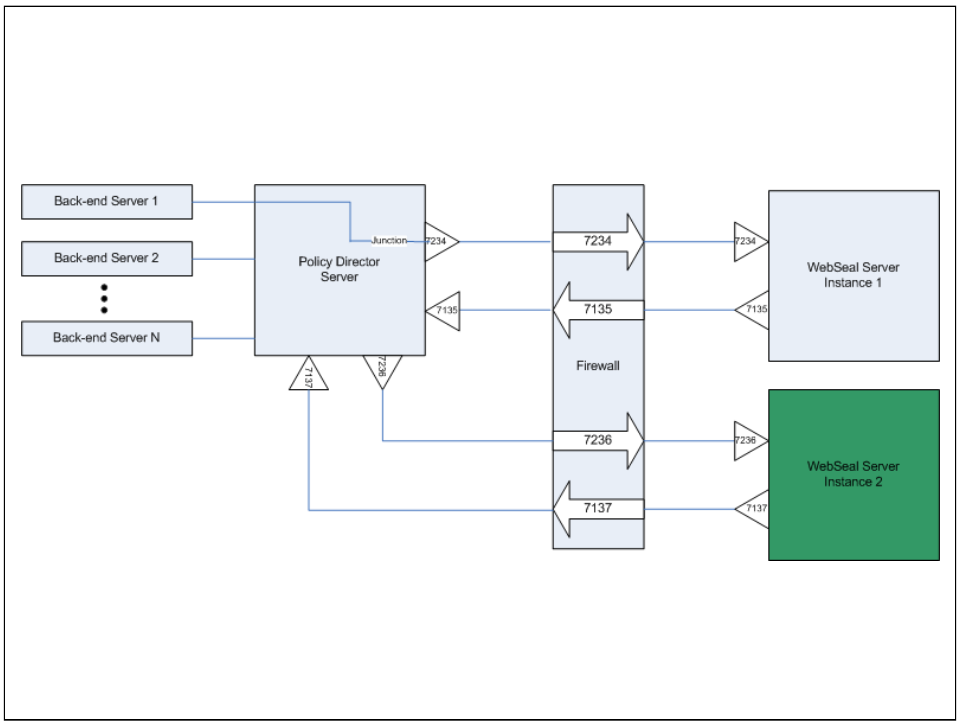
In the last chapter, we tie together all the different threads of the book. We discuss the different actors: people, organizations, and communities of practice, and the driving force of technology, continually changing, growing more complex, forming the landscape in which this work is played out.



For the rest of this talk I'm going to go over a set of episodes from the People chapter, though it highlights many aspects of our book. This is the Saga of George. I'll let George himself describe what he and his colleague Thad were up to…

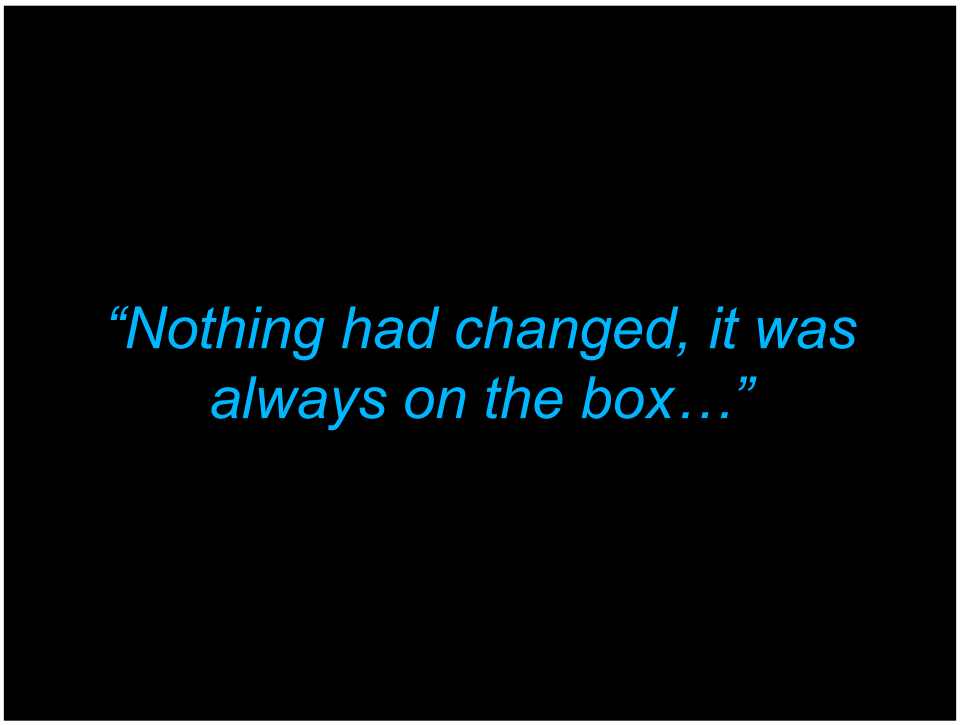


From his description, this is a diagram we created to understand what he was doing. This is the existing system, with a front-end WebSeal webserver, a firewall, the PD authentication server in the middle, providing fine-grained access to various back-end content servers.



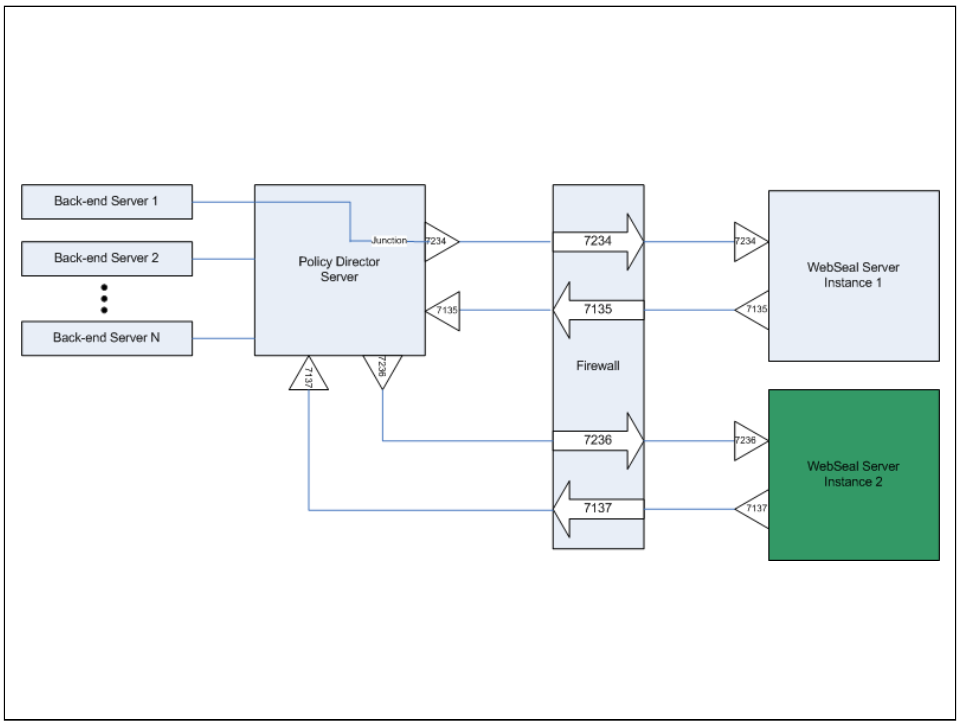
In this case, he needed to create a new front-end web server instance, get ports opened in the firewall, and configure the authentication server to allow the new front-end to connect to a back-end mail server.

George was new to this particular task, so he did research before proceeding, looking at online docs and the configuration file to make sure he understood what was going on.



This debugging was not a atypical - there were too many moving parts, and each round of debugging revealed a new level of something causing a problem. We don’t know exactly how they solved this one, later on they were talking about running the PD server as root until they worked out the issue with SeOS. In any case, by the time we resumed observations of George a few days later, they had found some solution so George could create the new WebSEAL instance, but then some new problems showed up.

As an aside, the instructions that George found suggested that this was the correct command for creating the new instance, so he pasted it on the command line and filled in the values for instance and internal port. There wasn't a definition of those options, so he filled them is as best he could.

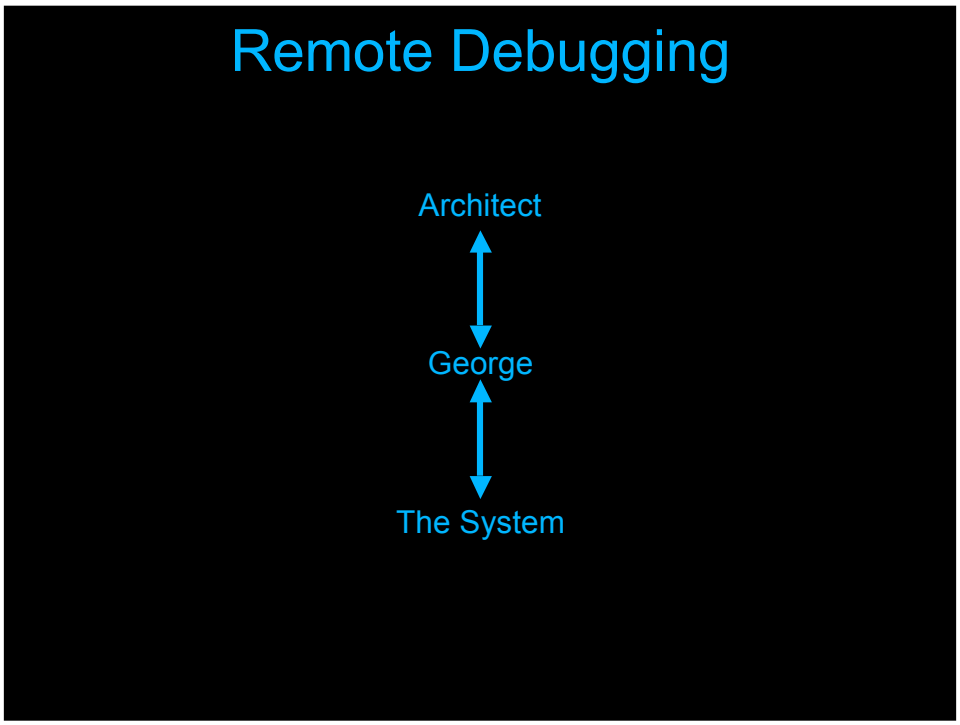


Once George had created the new server instance, he thought the architecture looked something like this. In the next clip he started working with the PD server to allow the WebSeal server to talk to the back-end mail server.

In these clips we have a few snapshots of George trying to configure the new server. The syntax is new to him, so he's trying lots of combinations, and isn't sure whether the syntax is wrong or there's another problem. The error messages aren't helpful. Finally he discovers that he can configure the old webserver instance, but not the new one. He sets it up, and lets his manager know that the customer at least has something to work with.

George's manager suggested that George work with the application architect to figure out what the new instance wasn't working right. In the next clip, George is talking over the phone with the architect, bringing him up to speed on the issue.

One of the first things they do while working together is to get more information about the error. Like just about everybody, George's first approach is to paste it into Google. It takes a few tries to get the search terms right, but eventually he finds the page listing the errors. But the information is not very helpful!

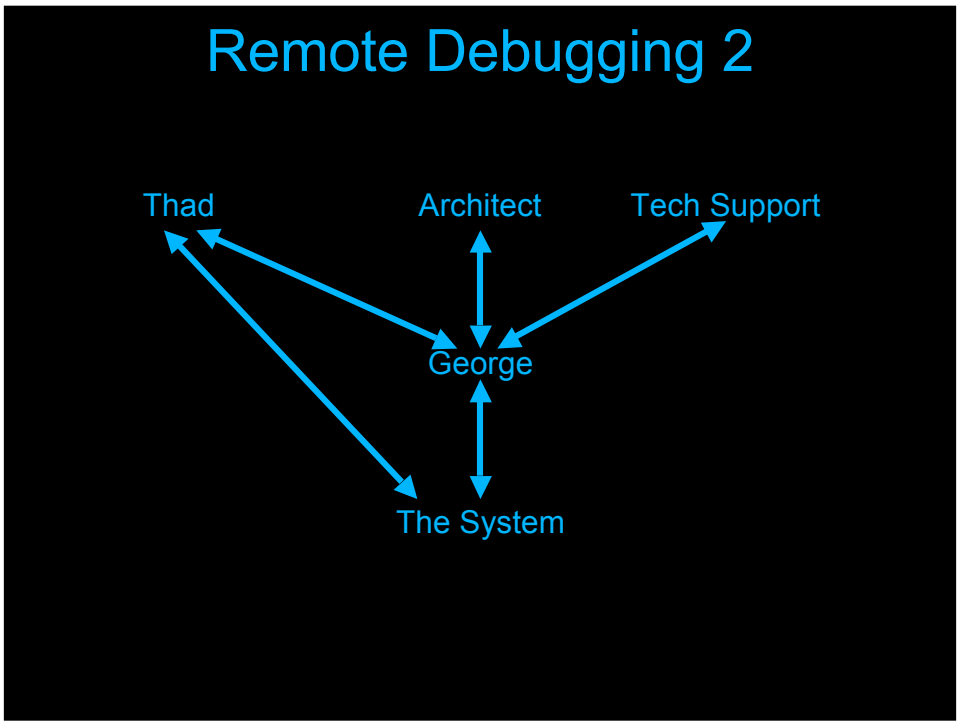


For the next 40 minutes or so, the architect worked with George to figure out what was going on. They recreated the new instance, looked through configuration files, and went on a wild good chase when they noticed that the instance creation tools called the instance a different name from the PD server. Eventually they discovered that the two tools simply used a different syntax.

All the while, everything was mediated through George - the architect did not have access to the production machine, so he would make suggestions, ask for information, and George would do it.

Eventually the architect couldn't figure it out, and suggested George call tech support. George wasn't optimistic about this course.

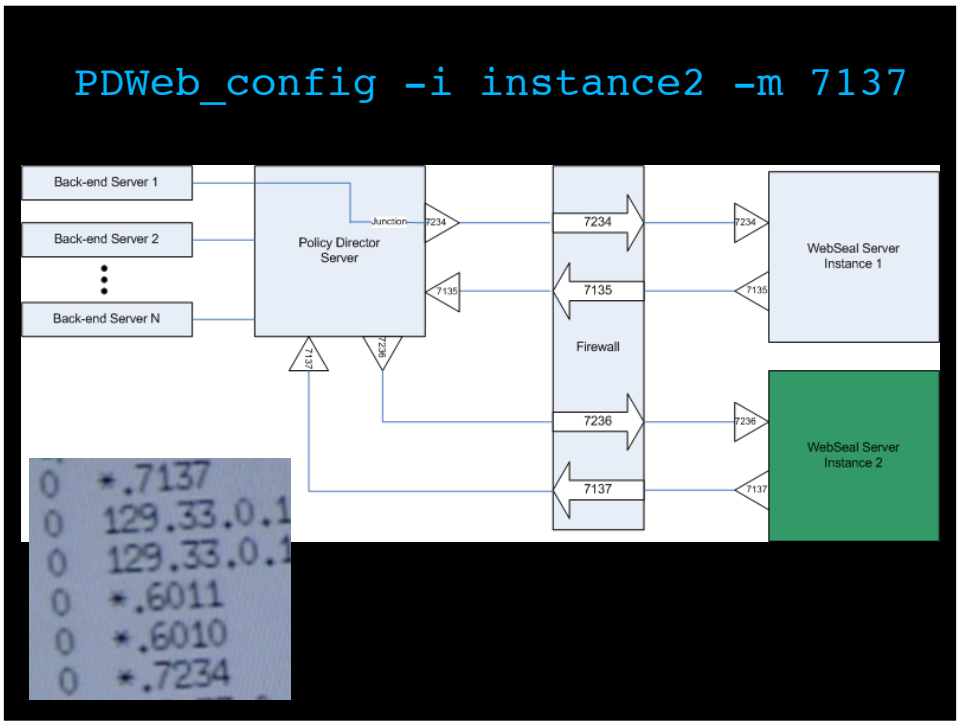
Tech support.



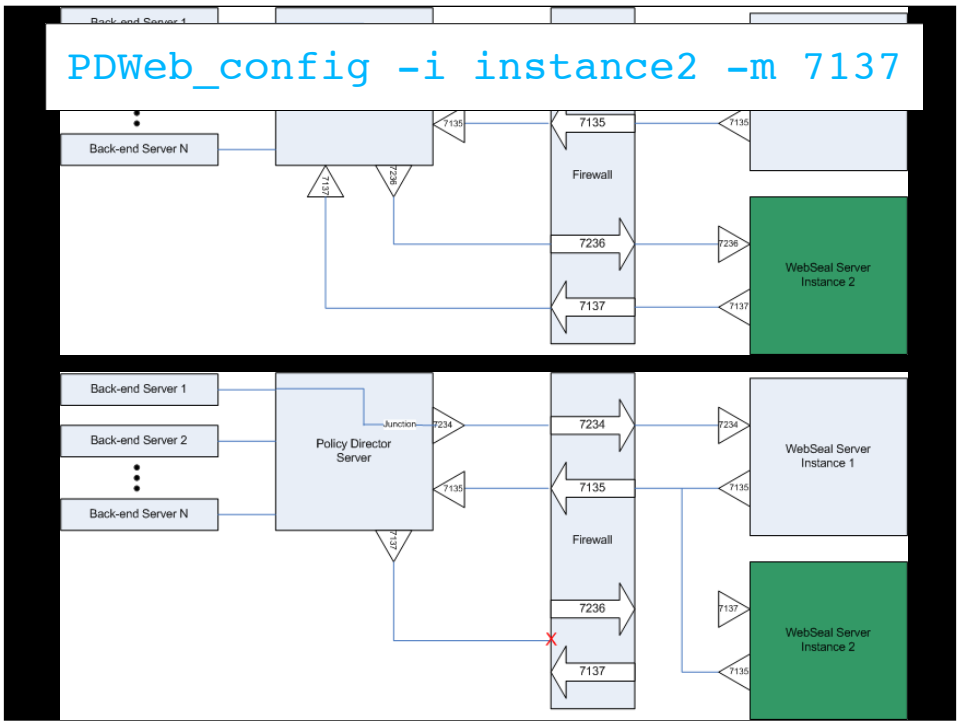
Tech support eventually called back (on the second line in the office), though George quickly moved the discussion to IM so he could continue talking to the architect while working with support.

The model was still the same, George interacted with the system while the architect and tech support asked questions and suggested actions. At some point George's colleague Thad started working on the problem as well, mostly working with George but also taking advantage of his own access to the system.

The interactions with tech support started out o.k.



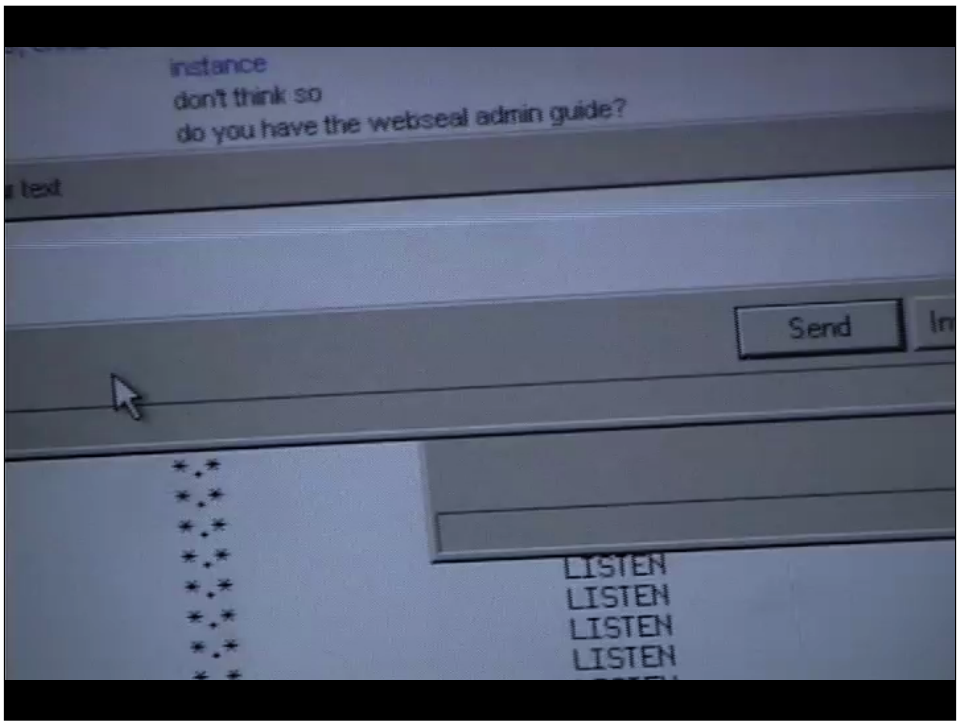
George just saw, then dismissed, the cause of the problem. The machine running instances 1 & 2 is listening on ports 7137 and 7234. But from what George says elsewhere, the incoming ports should be 7234 and 7236. George misunderstood the meaning of the original command-line argument when creating the instance, and his misunderstanding is preventing tech support from helping him.



Through analysis of the video and the manuals, we eventually were able to make these pictures. George's view was the top picture, with two ports for each instance. Reality was the bottom picture, with communication on the 7137 port going the other way, and being blocked by the firewall.

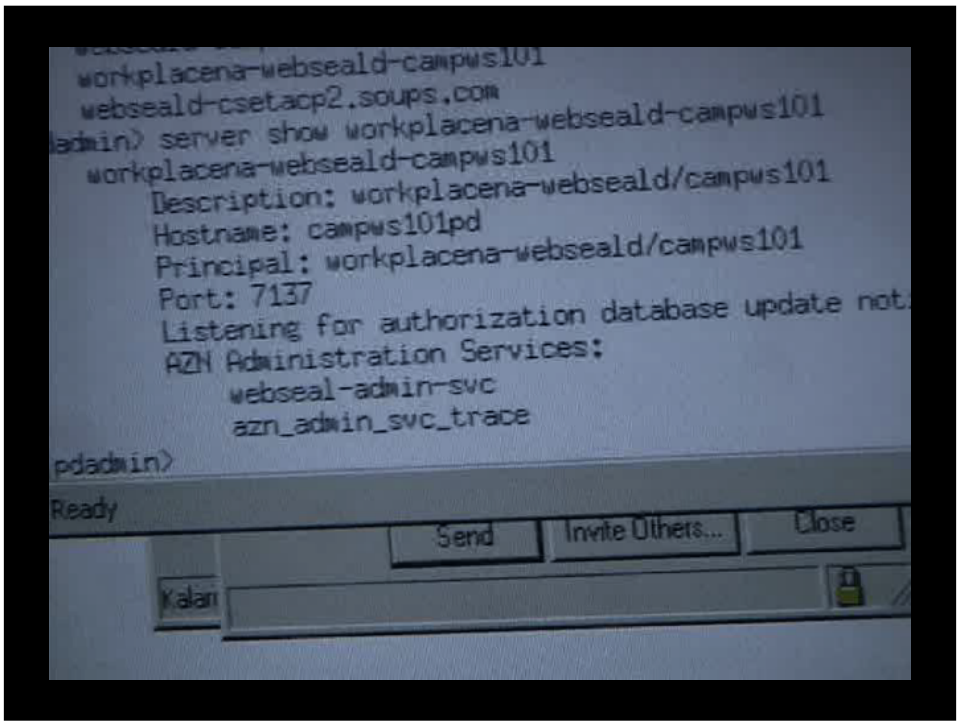
Maybe if George had a chart like this, he might have realized what was wrong, but he was keeping it all in his head.

Now the problem to be solved is not debugging the system, but debugging the system administrator, and that won't be easy. George's reply to tech support confuses things further, and tech support doesn't answer in a way that help things. It's a tragedy all around.



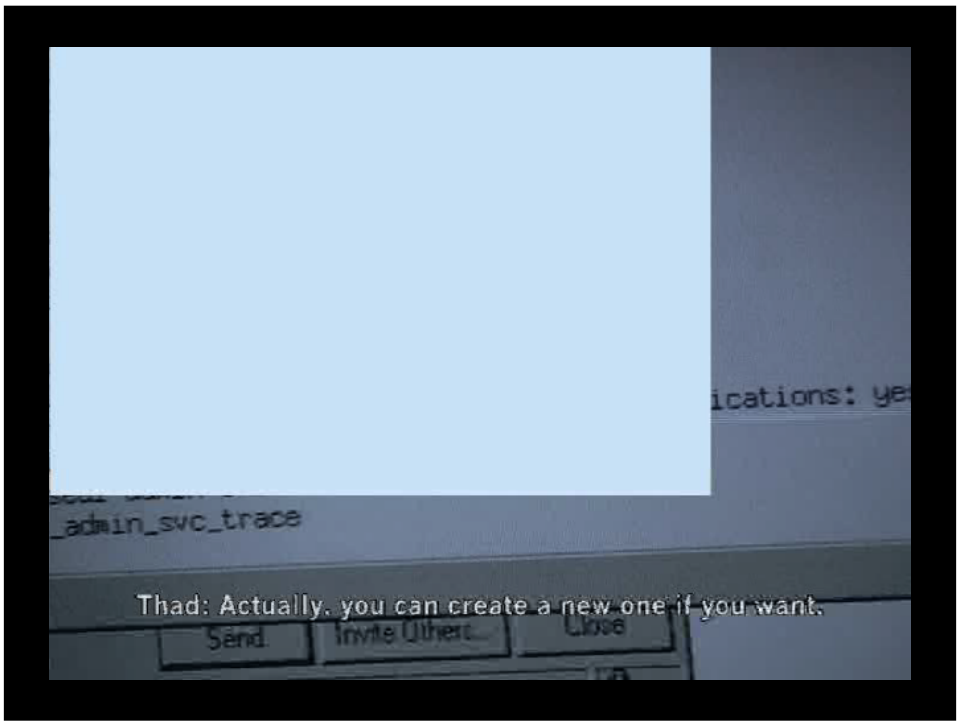
I've often wondered if this conversation would have gone the same way over the phone. With the give and take of a real conversation, would this misunderstanding have reached this point? Or would each person have demanded clarification and gotten to a solution?

In any case, at this point, George gives up on tech support and works with the architect and Thad. About 15 minutes later, Thad has an insight, but he needs to convince George.

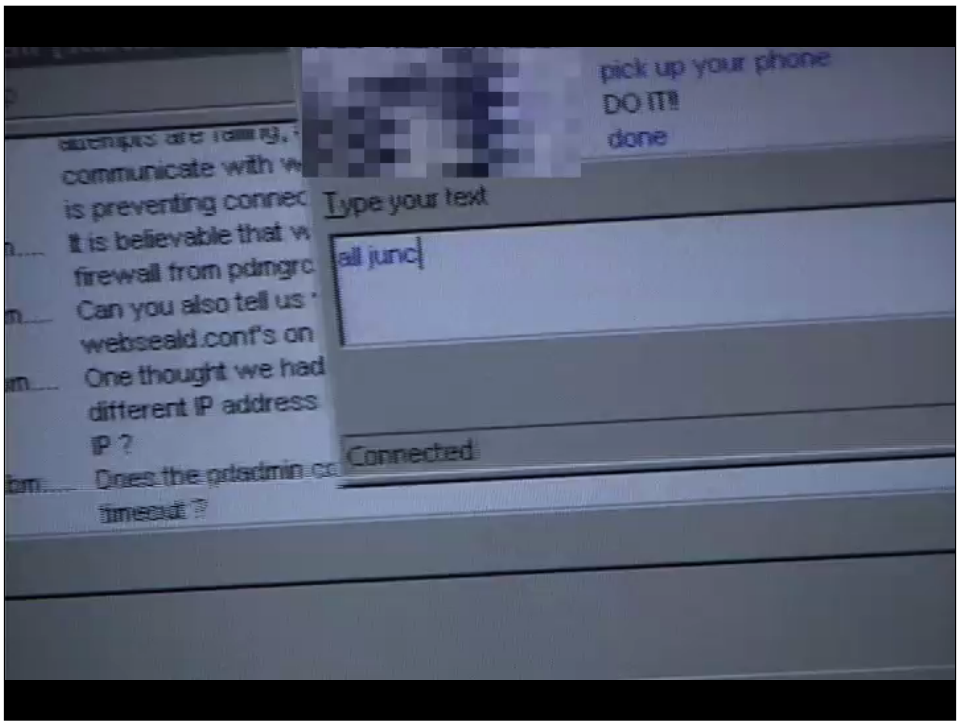


George and Thad are exchanging instant messages.

At this point George is mostly convinced, but Thad wants to make sure that George understands.



Eventually, however, George gets the changes made, and makes up with Thad.



And that's where we'll end this story. And this was just one story from one chapter of our book. But it highlights many of the important aspects we saw when studying your life and work.

- Collaboration and Communication between people is critical. Different collaboration tools are good for different things, but sharing system state is always hard.

- Misunderstandings are a fact of life, and are the source of many bugs. How can your reduce misunderstandings? Collaborate, communicate, share system state, even draw pictures.

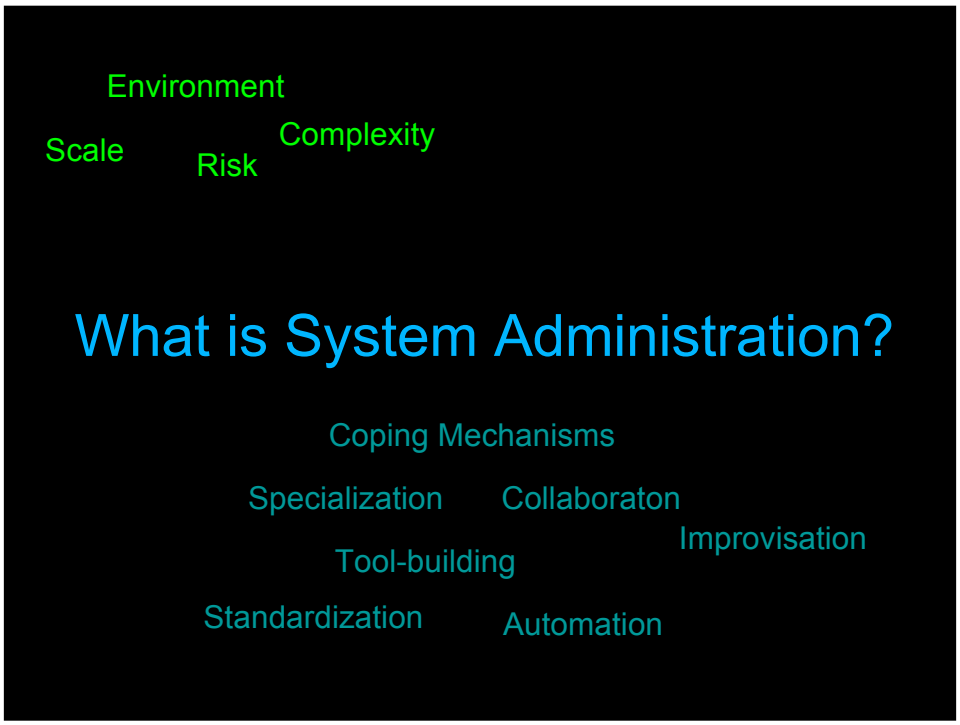
- Technology is way too complicated, and only getting worse.

- Administrators over time do develop methods and tools for handling their environment.

IT is not just the job of system administrators, it takes many closely meshed organizations to make things happen.

Standardization of tools and practices can help, but is difficult due to the ideosyncracies of each site.

And much of what goes on is community work, with people working together beyond a single site or company.



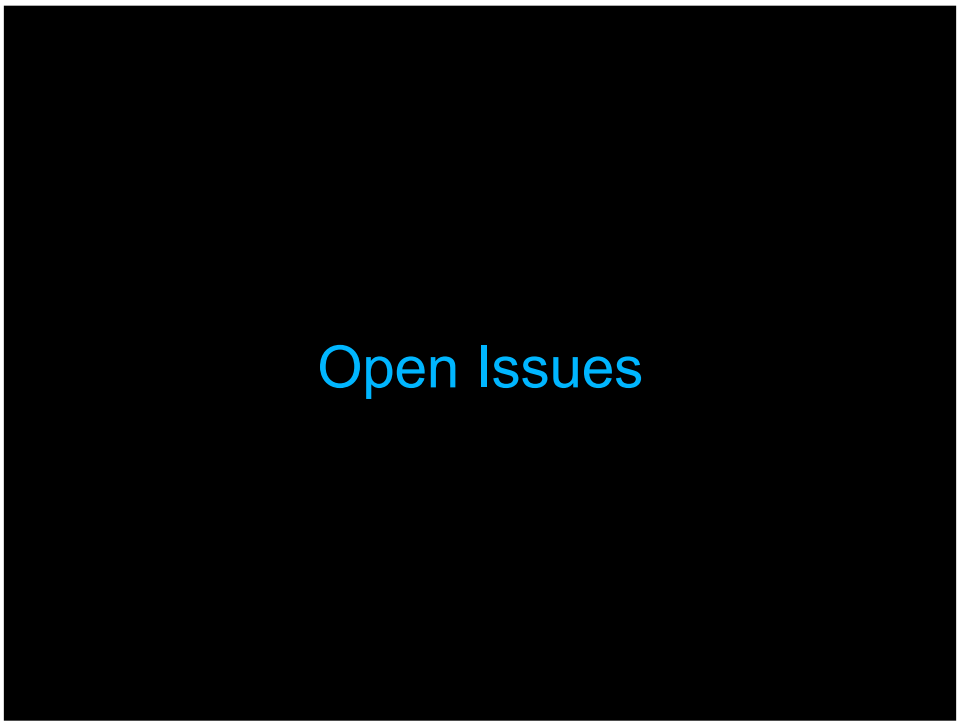
So, to the point of this talk, what is the work of system administration, and who are the system administrators?

First, this work is done in an environment that is large scale, highly complex, and risky.

Actors include individuals, organizations, and broader communities of practice, as we have here at LISA.

How do sysadmins cope? Through specialization, collaboration, customization, standarization, automation, and improvisation.

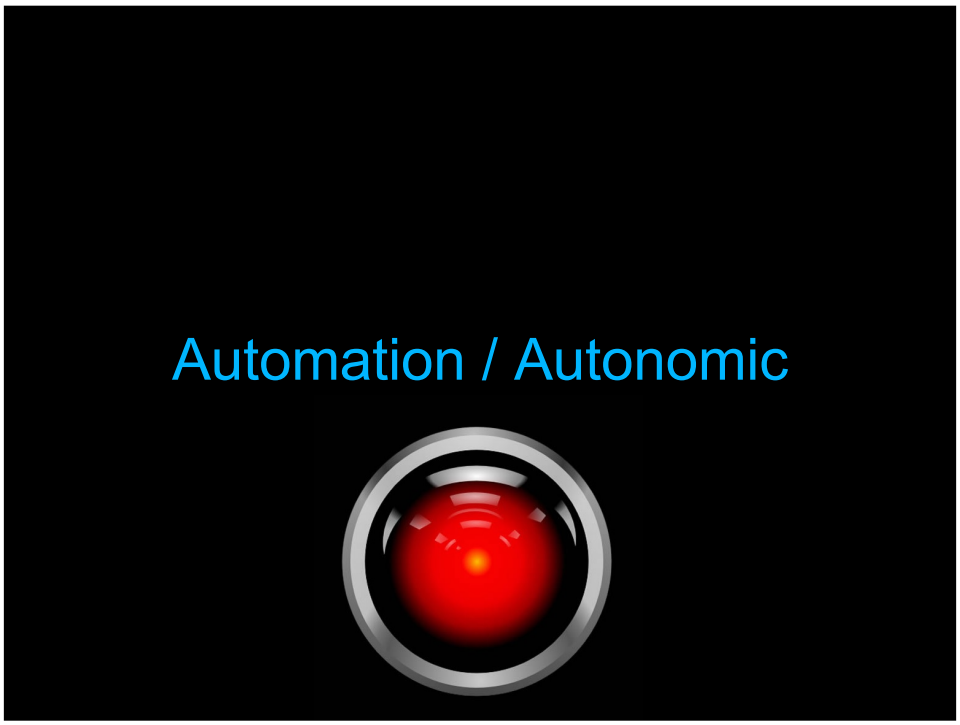
To sum it up, our observations describe highly complex, large-scale -- and often idiosyncratic -- environments in which people perform lengthy and risky operations given dynamic requirements and dynamic configurations. IT workers cope with these situations through specialization, innovation in tools and practices, and standardization. But these coping strategies interact. The picture that emerges suggests: (1) technical complexity leads to specialization which demands communication and coordination between individuals and teams, as workers and organizations spend considerable time establishing common ground with each other and with their systems; and (2) in a complicated dance, technological complexity and change demands innovations from a variety of actors, with system



Of course, as I just said, there are a number of factors pulling things in different directions. For example, Local Creativity and Accepted Standards are in tension,how to find the balance?

Also, given the importance of communication and collaboration, it seems that we need improved communication, collaboration tools? How can we better share system information? We all know how important social networking is, but can we make social information a first class item in system management tools?

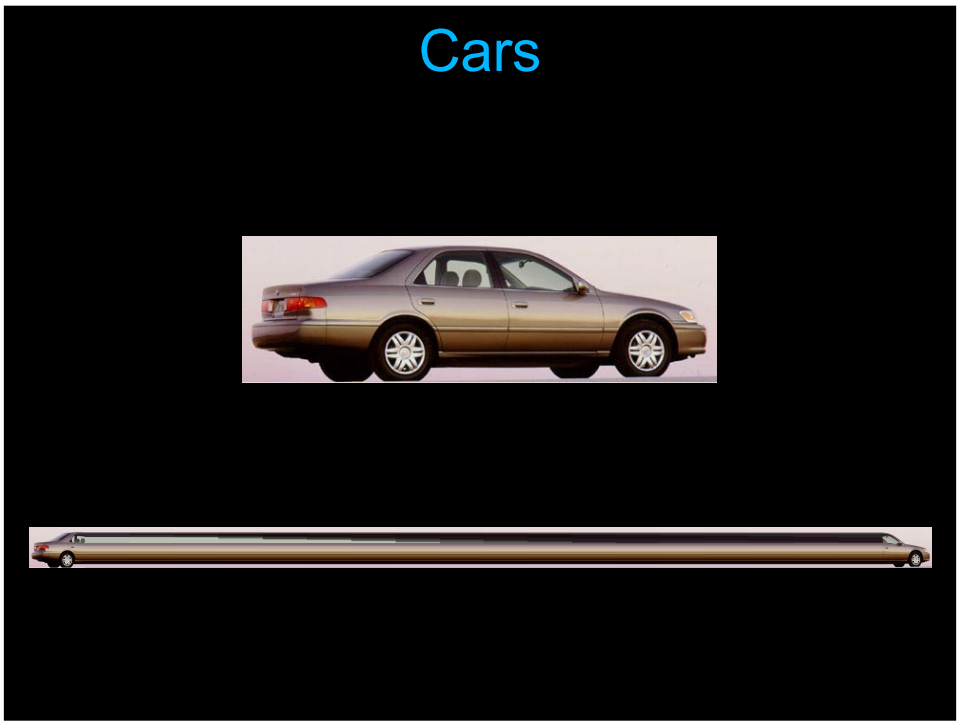
Where is System Administration Going?



I started this talk by mentioning automation and autonomic computing, the dream of an executive who wanted to contain the spiraling human costs of system administration. But is this really feasible? 9 years have passed since 2001, and IBM is not yet selling any “autonomic” systems, though many aspects of middleware have been improved and automated. Just walk through the expo and you’ll see plenty of improved administration automation tools. Are there models for something closer to autonomic in IT?

Flight Engineer

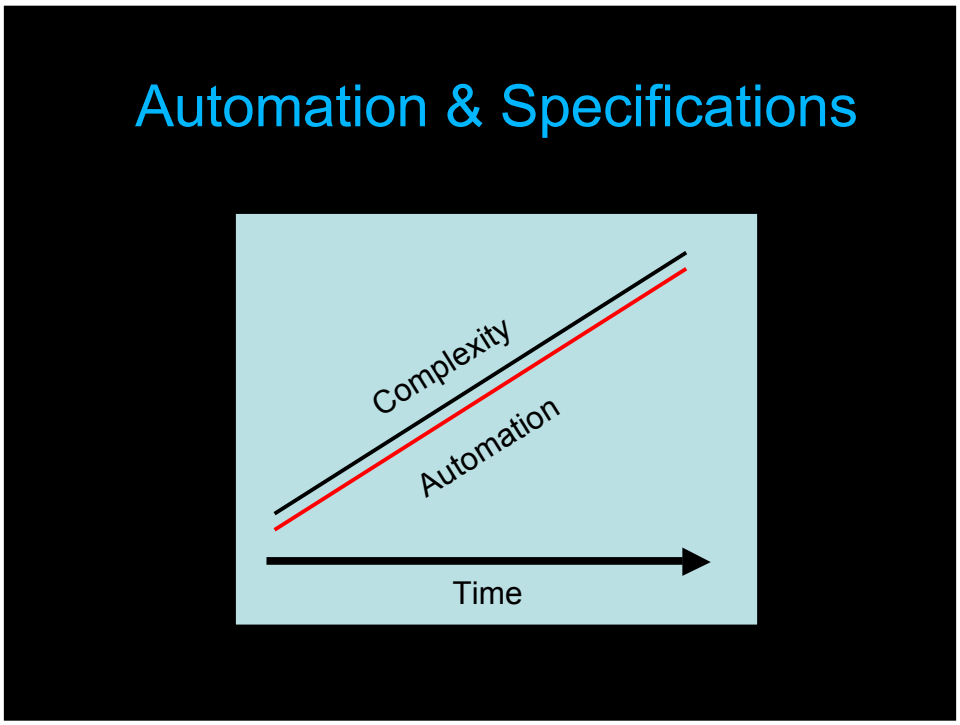
Other fields have shown dramatic examples of automation. Up until the 1980s, passenger jets were designed with a third person in the cockpit, the flight engineer, whose job it was to keep all the various systems on the aircraft functioning. With the advent of planes like the 767 and 747-400, those functions were automated, or minimized to the point where the pilot and co-pilot could take care of them. Is this a model for IT Automation? Maybe, maybe not, since each plane of a given model is fairly uniform, changes very slowly, remains in service for decades, and has very well defined load and operation parameters. From what I've seen, corporate IT systems are much more diverse in design, construction, and requirements, and change much more quickly.



How about Cars? Cars are an example of complex technology that is mostly tamed. You don't need to bring your mechanic with you when you drive places. Problems are still sometimes hard to debug, but they're infrequent.

Can IT commoditized like cars? Again, the requirements on cars are probably more restricted than IT systems. For example, My employer wants a Camry. Your employer wants a Camry, but tells you it needs to hold 100 passengers and go 500 miles/hour.

Also, cars seem to be going in the wrong direction. I was listening to a talk by an IT guy at BMW two weeks ago. He mentioned that current BMWs have more than 15GB of software on board, supporting 2100 customer functions, and more than 12,000 error messages. BMW receives 10GB/day of data from dealers, and in some cases even directly from car models that report their error codes over the cell phone networks. To me this seems to be going in the wrong direction! On the other hand, even if your job gets outsourced, there's a good chance that soon every BMW owner will need their own sysadmin.



Now automation is certainly increasing and improving - just look at the expo or the LISA proceedings over the years to see a stream of new and improved tools. Yet at the same time, systems are growing more complex and demands on IT management increase. It seems to me that automation has permitted systems to grow as complicated as they have, but automation has only just kept pace. Perhaps at some future time the complexity curve will flatten out, and automation will catch up and begin to bring management effort down.

I think, however, that there is a huge barrier to totally automated, autonomic systems, and that is the underlying specification language. IT systems exist to help businesses function, to help people accomplish human goals. One of your jobs is to take the business goals, the human goals of your organization, and transform them into working systems. Yet those goals are not specified in any machine-readable form. Even a well written set of IT policies usually includes terms and conditions that require human judgement to evaluate, arbitrate, and even negotiate with users. I believe that autonomic systems are an example of an AI-complete problem: only when we have systems that can truly understand and interpret human language, will we be able to have automation that fully meets human goals and needs.

Is autonomic computing possible? I certainly think so, but I don't expect it any time soon. And in the mean time we'll need to keep working on tools to help human beings keep up with the ever-growing complexity of

If there’s going to be a book out there about you, what do you want to be sure goes into it?

For other similar work, see
CHIMIT 2010, Fri/Sat, downstairs

Thank you!