Paper # 8022
Say Goodbye to Post Mortems, Say Hello to Effective Problem Management

Charles T. Foy
Siemens Medical Solutions USA, Inc.
51 Valley Stream Parkway, Mail Stop A08
Malvern, PA 19355
charles.foy@siemens.com

This paper describes the problem management process my company uses to investigate, classify, communicate and remediate the causes of service outages. Most outages have multiple addressable root causes; our process links these to the outage for analysis and assignment of multiple remediation actions. Root causes can also be analyzed independently, providing powerful trending metrics. The evolution of our problem management system is discussed, along with classification methods and items tracked. This process has proven to be very effective in eliminating repeat outages.

Introduction
I am a service level manager within Siemens Healthcare, Enterprise Hosting Services. Enterprise Hosting Services provides hosting services for several dozen applications running on all platforms from client/server to mainframe. Several years ago we recognized the need to have a single, consistent post-mortem process for all application service outages. That effort started with the goal of designing a new communication document to use after a service outage, and ended with a robust problem management system. This story describes the evolution of this process.

Driving forces behind creation of a Problem Management System
After Y2K passed, we recognized the need for a new and improved post-mortem process. There were several processes and forms in place dependent upon the application or system with format and content varying among them.

A team of two customer service managers (the author was one of them) was tasked with producing the new, consolidated post-mortem process. This small group was later expanded to include a process expert and a hosting service employee with 30 years experience. The team was assigned the task of creating a process that would be followed after every outage, using the same post-mortem report format and storing the post-mortems in a central repository for future reference.

First steps
We met as a team and discussed the requirements. A centralized location with all of the past post-mortems seemed like a good idea at first glance. Our new process could have been something as simple as a shared folder on the intranet, with a standardized document filled out for each outage. We could have stopped with just those two items established and we would have been done with our assignment. That was not the case, though. We went on to discuss possible uses of such a central repository. At minimum, if a customer called and asked for the reasons for an outage, the reasons would be there for any service representative to retrieve. This would provide improvement over the current process.

Over the course of several meetings we decided that while there was some value in a standard post-mortem document in a centralized folder, we could design a process and database that would drive down the amount of unscheduled downtime.

From a post-mortem document to a problem management system
As this idea developed, we realized it would take us considerably beyond the original request. The first thought we had was a document with standard text fields that described the outage, the root cause, and measures put in place to ensure the outage did not recur. Our thinking then evolved to another idea, to include in that document a field with a specific tag for the root cause, so that root causes could be searched for within the many documents in that folder. That idea of a 'tag' then morphed into the creation of a database that had specific fields for the root cause and for the preventive measures put in place, with the post-mortem documents that were sent to the customers attached to it.

The benefits of a database of post-mortems were numerous. When implemented, we would have a central repository with records of all outages, the customers affected, downtime incurred, hardware involved, root causes, and the preventive measures implemented.
This would provide:

- the ability to trend root causes over time and identify areas of improvement
- the ability to track preventive measures implemented, those pending implementation or deferred
- a resource to refer to after future outages to ensure we don't implement preventive measures already implemented and proven ineffective
- metrics for customers affected, applications affected, hardware involved, and unscheduled downtime for each of these

With a database such as this and a surrounding process, we could actually drive a documented reduction in unscheduled outages!

Our new goal then was to define a process and database that reduces unscheduled outages, increases availability, and communicates the root cause and preventive measures implemented to internal and external audiences.

Beware Acronyms!
Since this was a radical departure from the concept of "post mortem", a new name was needed for this process. At this time, we were just beginning our ITIL journey. It did not strike us that we were creating a problem management system. Our company has a tradition of naming everything with three-letter acronyms, so much so that the acronyms themselves were called 'TLAs' (three-letter-acronyms). Since this was a post-event analysis process, we settled on that as a working title with plans to change it in the future: thus the Post Event Analysis Process (PEAP) was born. The future rename did not happen and this is a learning that others may want to take into account when they are at this stage of a project.

Of course, the report produced by the Post Event Analysis Process must be called the Post Event Analysis Report, or the 'PEAR'. Our post-mortem document now had a name, and a four-letter acronym, which in our company gets you more credit than a three-letter-acronym.

Database definition ahead of process definition, not a good idea
Years ago, when all this was going on, our process implementation was still incomplete and our sequencing actually placed some of the technical decisions before the process decisions. As we were thinking about the database requirements and design, we calculated the requirements would consist of fields for the root cause, the hardware involved, the software involved, customers affected, total downtime, preventive measures implemented, and applications affected. Along with that, we had the need to be able to generate a post-mortem report (PEAR) from the outage record.

We learned from the mistakes of others
As we were meeting discussing this new database and process, we learned of an unscheduled downtime at a neighboring company and we decided to use it to ensure our database would be adequate; it would serve as a preliminary proof-of-concept. This outage provided to us a great example of a 'typical' outage as it illustrates several of the aspects of this paper as well as our thought processes at the time.

There was a slowdown affecting almost all applications in this neighboring data center. Response time dropped to zero within 5 minutes for almost all applications. The staff there started looking for commonalities, and the network was a prime suspect. A Configuration Management Database (CMDB) would have helped, but they had none implemented at the time. They dug further into the symptoms and it started to look as if it was storage. The SAN was believed to be the probable culprit as it connected to all the affected applications. And then the problem stopped, forty-five minutes into the outage.

As response times rapidly improved, the staff working on the issue looked up from their consoles and saw the SAN vendor's engineer on the floor with his head in the SAN box. They called to him and thanked him for fixing the slowdown. The engineer replied with, "What slowdown? I just finished swapping out your battery and making sure it was running OK." "When did you start that process?" they asked. "Oh, about forty-five minutes ago" was the reply.

Root cause easily identified, or so they thought
The staff then did a lot of digging to find out what actually happened. The SAN controller is very smart, so smart in fact that it noticed a potential problem with its battery and sent an alert to the SAN vendor several days before this incident. The SAN vendor dispatched the engineer to swap out the battery. Since the engineer was very customer-focused, and had been in the data center every day for the past several months installing this SAN, he ran right over and was let in by the guard. But the data center manager there had no idea this was happening. The engineer swapped out the battery with a brand new one. The SAN controller, with all that built-in intelligence, detected a new, uncharged battery and decided this was a risk. The battery is there to ensure items cached in memory are written to disk in the event of a power failure. So, the SAN controller stopped caching transactions and all reads and writes went directly to disk. This way, if there was a power failure, nothing would be lost. But the result was, without caching, all I/O slowed down and eventually was almost at a standstill. Forty-five minutes later, when the battery was fully charged, the SAN controller started to cache transactions again and the problem cleared up in minutes.

It's a great story and we were especially glad it wasn't ours. The root cause for this incident is easy – it was the battery going bad. Or was it? As our neighbor analyzed this outage, these facts came to light:

- There was no maintenance schedule for swapping batteries out periodically to avoid this situation
- No one had any idea that swapping a battery would cause this to happen (the SAN engineer was surprised as well)
- The SAN controller e-mailed the SAN vendor with an issue, and the data center staff had no idea this happened even though their network firewall had a hole punched for this traffic
- The SAN engineer did not tell data center manager's staff that he was coming to do this work
- Security let the SAN engineer into the datacenter without an escort
- The datacenter did not have a CMDB that would have shown them relationships between affected applications
- The battery was not actually bad, it was merely suspect. It might even have been the case that a good battery was swapped out.
- There was no policy in place to swap batteries during non-peak periods, so if the engineer did alert them to this swap before it was done, they would have agreed to have the engineer proceed anyway.

We had a lot of discussion around this outage and then took our discussion internal to explore outages we had experienced. What had struck us about the original issue was that they could not settle on one root cause and we saw this same effect in the outages we discussed internally. What would we put as the root cause? It appeared that for most of the outages there was more than one root cause, and we definitely wanted to track and trend all the root causes and the corrective actions implemented for each.

The light bulb goes on!
As this discussion ensued, we quickly realized three things. First, most outages have more than one root cause and deciding which root cause was the primary root cause was nearly impossible. Second, we will need to track multiple root causes, multiple corrective actions, multiple corrective action owners, corrective action effectiveness, and internal and external communications. Lastly, we realized that what we were designing would be much more complex than we originally thought.

Our next step was to decide if we should build a database and front end for this. Armed with our new requirements, we worked with internal resources and developed an estimate for the cost of coding this database and front end. As part of our ‘buy versus build’ process, we spoke to our Tools Group to discover if we had something like this already on the shelf. We were quite sure it was not. We were wrong again.

The light bulb goes on. Again!
We described the database and as much of the process as we had developed up to that point to the head of the tools group. She said, “OK, you want to track service outages. These are normally caused by hardware or software failure or even human error, right?” We agreed. “OK, so essentially you are tracking hardware and software *defects*, right?”, she asked. Another light bulb went on in our heads. “Well, yes, if you put it that way, we *are* tracking hardware, software and even human defects,” we admitted. “OK, then use this application we have, it is designed to track hardware and software defects and their resolution” was her recommendation.

The discovery process worked! First, we found out we were tracking defects. Then we found out that there was an existing application that tracks defects and their resolution, the same concept as root cause and corrective actions. In fact, many vendors offer this type of defect tracking system, often in conjunction with software change control and version control modules. It was not configured to track human errors, but we could change that.

This application did 90% of what we needed it to, and we were able to get it up and running very quickly. A longer time was needed for the configuration of the application’s fields. How would we classify these root causes? What values did we want to use? Most importantly, how detailed did we need to be with these classifications?

Root cause classifications
If a particular outage is caused by a server going down, we can easily classify that root cause as ’server failure’. Over a period of time, we would have metrics showing a trend of ‘server failure’ root causes, without a doubt. That is interesting information, but not very useful. In fact, we knew that ‘server failure’ was a root cause of many outages without having a problem management system to provide that answer. We needed information we could use to take action, to implement change and drive availability upward. If we listed the root cause as ’server power supply’, that might be more useful but would that show up as a trend? If we listed the root cause as ‘server power supply model XG236’ that would be very useful information, but it is doubtful that it would ever show as a trend over time. We needed a way to classify the outage in a fashion that made sense for both trending purposes as well as developing action plans to reduce unscheduled downtime.

We needed to solidify what would be classified. We could have worked within our small group of four but decided to ask our peers for input. Our peers would be the ones using the process and we needed their buy-in for the new process to be successful. We held several meetings for this purpose. The specific type of hardware and specific type of software that caused the outage were needed, as well as a means to track human error. Our peers needed that information for staffing and education purposes.

We were fortunate in this effort since the in-house defect-tracking application had several levels of ‘root cause’ fields called keywords, and every keyword was searchable. When the database was created, we were required to establish primary keyword groups which would have sub-keywords associated with them. Our three divisions of Hardware, Software and Human Error were perfect fits for this. Each database record would have the primary keyword, such as Hardware. It also would have two more keywords, called ‘keyword 1’ and ’keyword 2’. We decided that keyword 1 would specify ‘the item that failed’ and keyword 2 would specify ‘what specifically in <keyword 1> failed?’ And there were other fields available for the vendor of the ‘item that failed’, customers affected, downtime, applications affected, model number, and corrective actions implemented.

For an outage caused by a power supply on a server, the classification fields would look like this:

Primary classification:	Hardware
Keyword 1:	Server
Keyword 2:	Power Supply
Model:	XG236
Vendor:	Servers-R-Us, Inc.
Customers affected:	12355, 529404, 186332, 12298
Applications affected:	TimeTraker, Resource_Orders, VOIP

Figure 1
Each of these fields is searchable for trending purposes. And the database record also has notes fields for root cause description and corrective actions implemented.

Human-oriented defects?
As we discussed this configuration we knew we needed a primary classification for human error. We just were not comfortable with the term 'human error'. What does it mean to make an error?

At the most basic level, a human error means 'you did not follow a process'. A typographic error could be interpreted as 'failure to follow the process to spell commands correctly'. The more we talked about 'process errors', the more it made sense. In our attempt to develop a term for 'human error' we actually stumbled upon the truth – these really were process errors. This was another 'light bulb' moment for us.

The 'Process' category got a lot of attention at that point and our discussion took it far beyond typographical errors. For many outages we reviewed, there were some root causes that did not fit hardware, software or even human error. For example, if an issue was caused by someone not following a documented process, the keyword could be 'process not followed' as opposed to human error. But we had a lot of outages that were caused by something we were just not aware of at the time of the outage. Well, with our new category, those could be called 'process incomplete' or 'process incorrect'. We also added in 'Documentation' under the 'Process' category for those times when documentation is incorrect or inadequate. While incorrect documentation really is a form of 'incorrect process' (i.e., 'incorrect process to create and test documentation'), we decided it would be beneficial to distinguish documentation issues apart from the procedural issues.

Hardware, Software and Process
But, did these classifications work? Consider the SAN outage root causes and their corresponding classifications:

- No maintenance schedule for swapping batteries out periodically to avoid this situation
  - ⇒ Classification: Process Incomplete - in this case, the SAN maintenance process is incomplete
- No idea that swapping a battery would cause a problem
  - ⇒ Classification: Process Incomplete - in this case, the 'education process'
- The SAN controller e-mailed the SAN vendor with an issue but the data center was unaware of it
  - ⇒ Classification: Process Incorrect – the monitoring process needs to capture this alert and send it internally and not externally
- The SAN vendor did not tell the data center staff that he was coming to do this work
  - ⇒ Classification: Process Incorrect
- Security let the SAN engineer into the data center without an escort
  - ⇒ Classification: Process not followed
- Lack of a CMDB
  - ⇒ Classification: Process Incomplete - the incomplete process being 'documented configuration diagrams' or something similar
- There was no policy in place to swap batteries during non-peak periods
  - ⇒ Classification: Process, Incomplete – the SAN 'swap the batteries process' is incomplete
- If the battery was deemed to have gone bad
  - ⇒ Classification: Hardware, SAN Controller, Battery

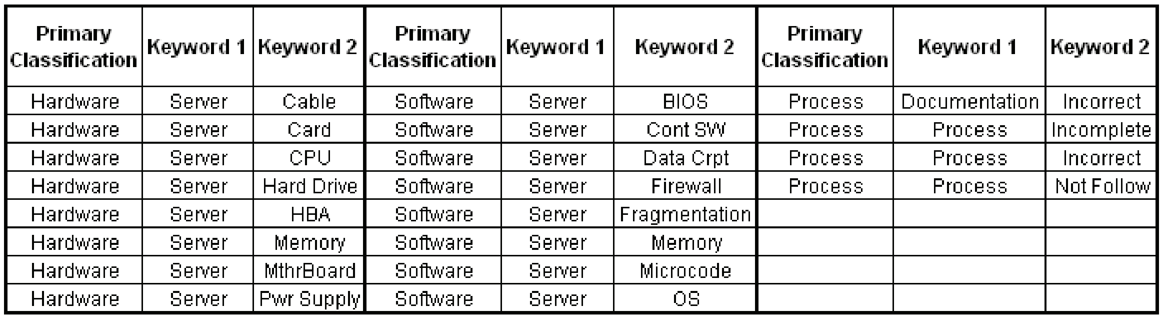
Multiple Root Causes
In the SAN example, there were at least eight root causes and, hopefully, eight corresponding corrective actions to be implemented. The defect-tracking database we use allows for records to be associated with each other in a parent-child relationship or peer-to-peer relationship. To track these multiple root causes, corrective actions, and corrective action owners was as easy as creating a database record for each root cause and assigning a corrective action owner. All the records would exist as children, under one parent or main database record so the outage could be viewed as a whole. The records also existed independent of one another, so trending of root causes would include all root causes and not just the parent. This relational aspect meant there would be no need to decide which root cause was the main root cause, we could just pick one and associate the others with it.

Using this database structure, analysis of outages to find trends and identify areas for improvement is the most accurate method as this method queries all root causes and not just a main root cause. If only main root causes were trended, analysis would produce some trends that do exist but certainly would miss many others. Consider the example where, during a one year period, there were ten outages that had a hardware failure and a process failure for each. If the trending was only for main root causes and all those root causes were hardware, then the trend of process issues would not show up. With this database structure, our reports would show two trends: one for hardware issues and another trend for process issues.

Since each of the root causes has an associated database record, and that record can be assigned to anyone, the person performing the post event analysis does not have to own each of the follow-up items or corrective action plans. In the SAN example above, many of the issues belong to the manager of the SAN group. But the ‘security process not followed issue’ belongs to the manager for building security. The ‘vendor not notifying the data center about planned work issue’ (Process, Process Incorrect) belongs to the vendor’s management. The remainder of the root causes can be assigned in the same fashion.

Assigning the various root cause records to the appropriate department makes sense as it is easier for the manager of that department to implement change internally, and it correctly assigns the work away from the person assigned as the primary Post Event Analysis Process owner for that outage.

Sample Keywords



Primary Classification	Keyword 1	Keyword 2	Primary Classification	Keyword 1	Keyword 2	Primary Classification	Keyword 1	Keyword 2
Hardware	Server	Cable	Software	Server	BIOS	Process	Documentation	Incorrect
Hardware	Server	Card	Software	Server	Cont SW	Process	Process	Incomplete
Hardware	Server	CPU	Software	Server	Data Crpt	Process	Process	Incorrect
Hardware	Server	Hard Drive	Software	Server	Firewall	Process	Process	Not Follow
Hardware	Server	HBA	Software	Server	Fragmentation			
Hardware	Server	Memory	Software	Server	Memory			
Hardware	Server	MthrBoard	Software	Server	Microcode			
Hardware	Server	Pwr Supply	Software	Server	OS			

Figure 2

Post Event Analysis Process defined
Armed with an excellent database, we then returned to three questions we documented at the beginning of our effort. We started with “What is an outage?”, “How do we kick off this process?”, and “Who is going to drive this process each time?”

By now, ITIL was being used much more widely across the company, so we looked there for the answers. In our division, there is a customer-facing group called the Customer Service Center. One of their duties is to manage a service outage (ITIL: Incident) until it is solved. They have a process called the Outage Management Process (OMP) that is led by an Outage Manager who coordinates programmers and technicians, and communications to the customer. The OMP is implemented whenever there is an outage, which means of course, <u>they have already defined what an outage is</u> and we didn’t have to do that. One item down, two to go.

How do we kick off this Post Event Analysis Process? How do we know when an outage is over? Reason dictates that the Outage Manager knows these facts, since he or she will notify the customer when their application is available again. Why not use the Outage Manager? Since they know when the outage (incident) is over, they can transfer it to the Post Event Analysis Process (ITIL: problem management process) and initiate that process. In ITIL terms, we would be transferring the issue from incident management to problem management. Two issues down, one to go.

Who will drive this process? What do we call that role? We wanted to emphasize that the person driving this Post Event Analysis Process (PEAP) or problem management process owned it completely, and also had to drive others to complete their tasks. We settled on the person being a manager, someone who could effect change within their department as well as influence change in other departments. The role needed a title. What would you call the owner-driver of the PEAP process? The PEAP Owner-Driver or POD would be the name, of course.

We still needed to define who the POD would be. It seemed logical that the manager of the group responsible for the main or parent root cause should be the POD. However, if you identify three root causes for an outage, related to three separate groups, you will have three convincing arguments from each of those managers about how their group’s issue was not the main root cause. We looked then to the Outage Management Process.

The Outage Manager is very familiar with the outage and what group was primarily responsible for getting the application back up and running. Therefore, it made sense that the Outage Manager would decide which specific group would receive the PEAP assignment. The manager of that group is assigned automatically as the PEAP Owner-Driver (POD). That manager is notified by e-mail that they are the POD and they can decide if they wish to continue in that role or transfer it to another manager, but they would need a good reason for such a transfer.

Integration
Integration was created between our break/fix (incident) management system and the new problem management (PEAP) database. Using one command, the incident is copied to the PEAP database along with all details of the incident, and the PEAP record is assigned to a manager who is now called the PEAP Owner-Driver, or POD. The PEAP database sends an e-mail to that manager alerting them to the PEAP database record in their name, and notifies an administrative assistant who sends another e-mail to ensure that manager is aware that they are the POD.

The Post Event Analysis Process (problem management system) in a thumbnail

- An incident takes place where an application is not available
- The Outage Manager facilitates that incident until it is resolved
- The Outage Manager transfers that incident record to the Post Event Analysis Process (PEAP) database, which initiates the PEAP process and assigns an owner-driver (POD)
- The POD is responsible for ensuring all steps of the Post Event Analysis Process are followed
- Notification is made to legal and marketing communications that they will have a customer communication to review and approve shortly
- Analysis of the outage, including the 5 why's, is performed
- Root causes are each given a database record and assigned to a manager to implement a corrective action
- An internal report is sent listing the root causes and follow up items (the Post Event Analysis Report or PEAR)
- A customer communication is created, approved by legal and marketing communications, and sent to impacted customers
- All action items are implemented, PEAR and customer communications are attached to the database record, and the database record fields are accurately valued.
- The Post Event Analysis Report is reviewed at a weekly PEAR review meeting, where the owner-driver (POD) explains the outage and all corrective actions to senior directors of the organization.

Rollout of the process – lessons learned
Once our new process was defined, education was created for the Outage Managers (incident managers) and any manager who may be assigned as a PEAP owner-driver (POD), and all were required to attend this class. We monitored compliance with the PEAP process, and gathered feedback from managers as they used the PEAP process to analyze outages and produce reports. This monitoring uncovered some challenges and we created solutions for each, these are listed below.

Challenge 1
The process was well defined in our quality system, but that definition spanned numerous pages of reading material. A manager may be a POD once every three months and have to be re-educated regarding the process.
Solution:

> We created a checklist that had thumbnail descriptions of each step, along with hotlinks to a complete explanation of it in the full documentation. This checklist created an additional benefit of being able to change the process quickly by simply changing the checklist (as well as the full documentation) and sending the updated checklist to the PODs.

Challenge 2
There were too many steps in the original process, making it burdensome and lengthy.
Solution:

> We streamlined the process, dropping the number of steps from twenty down to nine. This did require a second round of education, which also helped us with challenge 3 below.

Challenge 3
The PEAP was not always being followed completely, reports were not always produced and action items were not always being implemented. We recognized this was a culture change, and were accepting of gradual adoption of this process over a period of time.
Solutions:

First, we phased the roll-out of the PEAP to have the process include a small segment of outages at first, then a larger segment, and then to include all outages. For example: originally, only outages more than fifteen minutes affecting multiple customers qualified for the PEAP. Then after a few months this was expanded to all multi-customer outages of any length. Finally, it was expanded to any outage in the data center, with a single affected customer or multiple affected customers.

Second, the change in the PEAP process mentioned above and the resultant second round of education sessions, afforded another opportunity to drive the culture change and the value of the process.

Third, a directive was issued from senior management that all such external communications are sent within the time parameters of the PEAP process. We track compliance with that target by generating metrics by manager and by director, and we publish those metrics monthly. This transparency worked very well. Lastly, we enlisted an administrative assistant to notify managers that they have been assigned a PEAP, and to remind them of outstanding artifacts due (e.g., internal or external communications).

Challenge 4
Managers assigned to drive the process (PODs) were not identifying all of the root causes.
Solutions:

First, each outage is reviewed at a weekly review meeting staffed by senior directors in the organization. Every POD presents their Post Event Analysis Report (PEAR) including the details of the outage, their root cause analysis and the corrective actions identified and implemented. The challenge is to attend this meeting and not have any root causes identified by the larger group that you have not already identified and addressed.

Second, the internal communication template (the PEAR template) has a section with 'five why's' listed, the POD is expected to fill in as many as possible so that as many root causes as possible are identified. 'Five why's' is a standard method to determine all of the root causes and perhaps a main root cause. It starts with a statement like "The application was down for 30 minutes" and you ask, "Why?" The answer may be, "The server crashed." Then ask, "Why did the server crash?" and the answer may be, "The server crashed because it ran out of disk space on the system disk." The next 'why' would ask, "Why did the system disk run out of space?" and then "Why didn't monitoring detect this before the server crashed?" and so forth.

Problem management system value added
The implementation of the Post Event Analysis Process has resulted in several benefits, most of which were anticipated.

- Decreased downtime has resulted from the implementation of the Post Event Analysis Process. Outages that repeat at great intervals or perhaps spanning different platforms were not easily identified previously as having any relationship. Now they no longer are assumed to be one of a kind. This information can be analyzed and preventive actions designed to reduce the identified root cause's frequency and duration.
- Increased customer satisfaction resulted from timely, accurate and honest communication about the causes of unscheduled downtime. Many of the direct contacts in our customer base are the IT staff at the business, and they have end-users putting a lot of pressure on them whenever there is unscheduled downtime. With a customer communication from our company in hand, the IT staff can educate the end user as to the cause, demonstrate that actions are being done to prevent recurrence, and reinforce their own contributions and value to their constituents.
- A shift in focus from platform and operating system monitoring to application and end-user experience monitoring happened as a result of root cause analysis. Operating system, disk and network monitoring are still very important and some outages do crop up where operating system level monitoring may need to be adjusted. However, a sizable number of outages relate to application and/or configuration issues that need monitoring at that higher level.
- Fewer outages resulting from maintenance running late. This was an unexpected benefit. If system or network maintenance is scheduled for a specific period of time and that time is exceeded, the result is documented as unscheduled downtime and the Outage Management Process is invoked as it would be for a hardware outage. A PEAP is generated and assigned to

the manager of the group who performed the maintenance. The Post Event Analysis Process clearly documented these instances and managers adjusted processes to create more accurate predictions of downtime. Managers also, in some cases, streamlined the tasks performed during a scheduled outage to avoid exceeding the scheduled outage downtime estimate.

Sample of root cause analysis
Here are breakdowns of root causes we developed from our problem management database.

The data does not represent actual numbers.

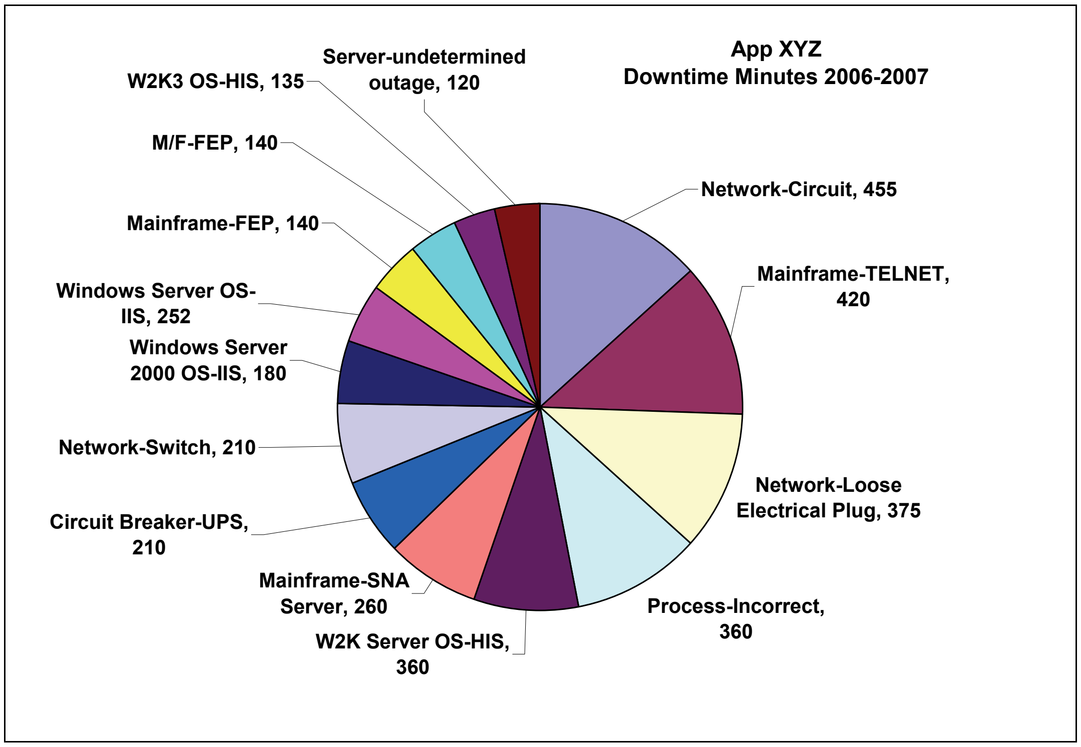


Figure 3

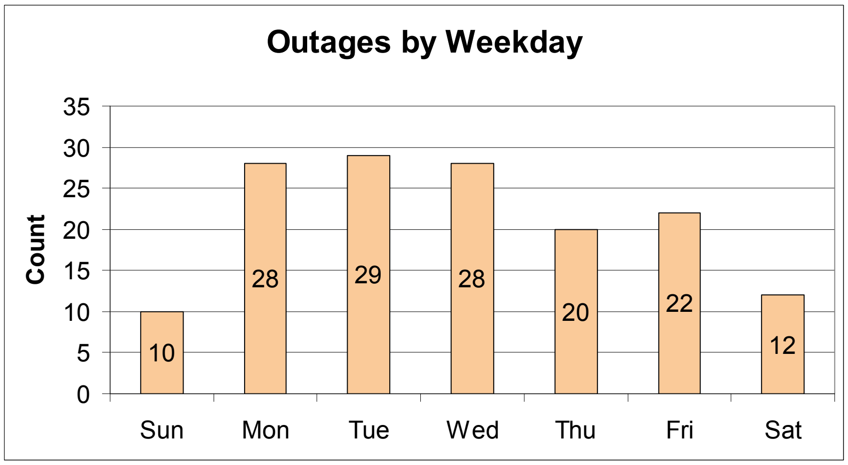


Figure 4

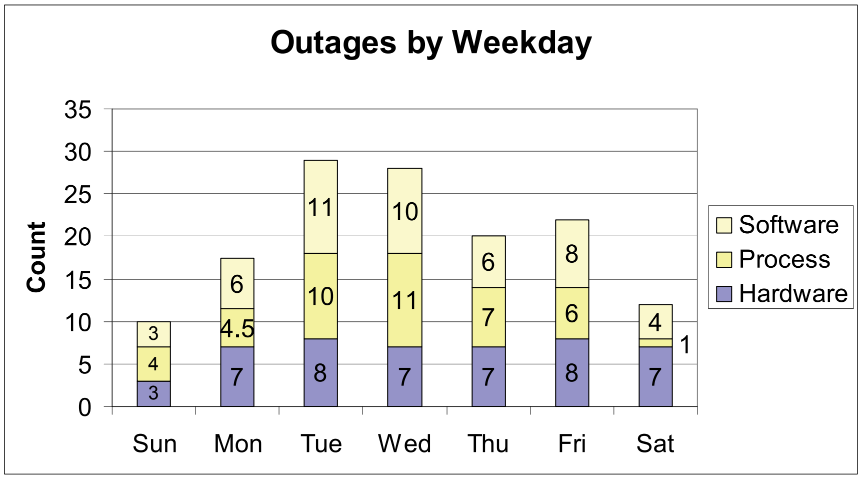


Figure 5

Once database analysis produces this type of information, the next step in the analysis is to select an area to focus on, pull more metrics with more detailed fields (such as vendor), perform in-depth analysis, and develop action plans to address those root causes. For example, in Figure 3 above for application XYZ, the root cause 'loose electric plug' resulted in 375 minutes of downtime over a two-year period. Further analysis may reveal that plugs were loosened when maintenance was being done nearby, and an action plan to secure plugs to hardware not related to nearby maintenance could be developed.

Figure 4 shows a chart of outages by day of the week for one year. Further analysis may reveal that hardware 'knows' it is Sunday (Figure 5) and does not fail as often as the other six days. The explanation may lie with human activity. Upgrades and hardware maintenance may be less frequent on Sundays than other days of the week. As you recall, each outage can have multiple root causes. Therefore an outage due to a human error (process not followed) that is complicated by a hardware failure would show as both a hardware issue and a process issue in this chart. Since more process failures happen during the other six days when there is more human activity, the chances of a hardware failure are reduced on Sundays when there is less human activity. Compare four (4) process issues on Sundays with eleven (11) on Wednesdays.

Conclusion

The methodology we developed and implemented including classifications, database design, and the over-all process; can be applied to a large IT department such as ours, or tailored to a smaller shop. An elaborate defect-tracking database is critical for us, but the same result could be achieved on a smaller scale with a standard document and a standard use of keywords. A smaller shop may not have a group dedicated to each platform or landscape, but someone is responsible for each of these areas. And be very careful with a 'working' acronym, it just might stick around for a while!