Turning Problems into the Known

JEANNE SCHOCK

Jeanne Schock has a background in Linux/FreeBSD/Windows system administration that includes working at a regional ISP, a large video hosting company, and a DNS and top-level domain registry services provider. She is a certified Expert in the IT Infrastructure Library (ITIL) process framework with in-the-trenches experience in change, incident, and problem management. Jeanne also has a pre-IT academic and teaching career and is an experienced trainer and public presenter. jeanneschock@gmail.com

System administrators rightly associate problem management with identifying and removing the underlying root causes of incidents. But finding and resolving root causes requires resources and interdepartmental political will that are not always available. Living with a documented yet unresolved problem is not a failure of your team. Nor is it a failure of process. The real failure would be to overlook the smaller, incremental improvements that can be gained by addressing the factors and conditions that contribute to incidents. Make sure that your problem process is focused on outcomes beyond technical solutions to root causes: knowledge gain, effective decision-making, elimination of negative activities such as finger-pointing, and the only metric that really matters in IT—customer satisfaction.

The Problem Management Process

We define an incident as a disruption to normal IT service, and a problem as the unknown cause of an incident. It is unknown because we don't know either what caused the incident or how to prevent it from happening again. Incidents don't become problems; rather, problems cause incidents. A problem management process manages the life cycle of problems: identification/categorization/prioritization, establishing workarounds, tracking activities, changes in status and decisions, investigating and determining causes, finding and implementing permanent solutions. One purpose of the process is to prevent recurrence of incidents through permanent solutions. A second, often-overlooked, purpose is to minimize the impact of unavoidable incidents. Reasons why future incidents might be unavoidable include resource limitations, technical limitations, or practical decisions to live with the problem based on a cost to benefit analysis.

Establish Good Workarounds

Where does this leave a technical team? With a good workaround, we can both mitigate the impact of an unresolved problem and buy more time for implementing a permanent solution. Our objectives are to quickly detect the conditions indicative of a recurrence of the incident, to reduce the mean time to resolve, and to continually improve the workaround as we learn more about the problem. The workaround may be good enough to complete a project that entirely eliminates the software or application or network device that underpins all other causes of the incident. We speak disparagingly of workarounds as "duct tape," because we know from experience that duct-tape solutions usually become permanent. That's why it is critical to build into your problem management process routine reviews of all workarounds. Define the review schedule based on the priority of the problem: e.g., high-weekly, low-monthly. Then define the behavior that you want and write a policy to match: all workarounds must be continually reassessed for effectiveness in mitigating the impact of the incident, cost in staff time to maintain, and level of confidence that it will continue to work.

Document and Track Known Errors

If you have recorded the problem, and you either have a workaround or preliminary investigation hints at the nature of the problem, you can declare a known error. This may sound like an attempt to mask the problem. But turning an unknown cause of incidents into a known error has value. Which situation would you prefer: a published list of known errors, linked to incidents they are suspected to have caused, perhaps even with a confirmed workaround, or undocumented problems that may re-manifest at any time in the form of another service interruption? Turning the unknown into the known is time well spent. You already use the known error concept when you are waiting for in-house developers or vendors to patch software bugs and vulnerabilities. In these examples, the known error is well-understood, and you are just waiting for a solution. But you can also say that you have a known error when you have even a vague understanding of the correlation between causes of the incident. The objective isn't necessarily to solve every single root cause, but to make good decisions and to track and continually reassess those decisions.

This may require you to shift your thinking about your role as problem solver to provider of strategies, information, and tools that can help your company manage problems. The known error database (which could simply be a list maintained in a Google doc or wiki page) informs better risk analysis and decisions around authorization of change requests. It is a useful reference for on-call shift changes. As a list of "improvement opportunities," it helps with departmental planning and goal setting. Tracking a known error includes linking new incidents, or even older incidents that you come to realize were likely caused by the same problem. This helps build a case for developers to prioritize a bug fix, or convince management to re-prioritize resources in ways usually reserved for root cause investigations after large or embarrassing incidents. If you find your team is tracking an increasing number of known errors that are beyond your control to repair, try building relationships outside your team that you can leverage to resolve some of those problems.

Root Cause

Let's talk about root cause. System administrators know well that complex technologies most likely have multiple correlated causes. At a purely semantic level, you should be comfortable replacing "the root cause" with "multiple root causes." Don't assume that you must always delve into an investigation for underlying causes. Think of the problem process as one of the many tools in your toolbox that you can use for improving the services for which you are responsible. Improvement can come in the form of large-scale leaps, but it is more likely to result incrementally from small, iterative steps forward. Ask yourself, what is within your control to improve:

- What tools could you build that would enable you to detect the conditions that were present during past manifestations of the problem?
- Can you automate the workaround?
- What tools or knowledge would reduce the time required to resolve the incident?
- Can you reduce the conditions that contribute to slow troubleshooting by asking the DBAs to train your team over lunch on replication errors?
- Can you be prepared to collect more data during the next occurrence of the incident that would enhance your understanding of the problem?
- Can you revise alerts and escalation procedures to get the right people looking at the incident faster?
- Can you reduce dysfunctions by improving your team's relationship with teams that are both ahead and behind you in the value chain?

These are all actions that can be taken that do not require root cause investigation or permanent solutions. And they should be acceptable outcomes for teams with limited resources.

Human Error

Can you mitigate the risk of human error? Human error is a common trigger or contributing factor to incidents. Human error is an inevitability, just like hardware failure. Systems that are not designed and built for resilience in the face of inevitabilities are incidents waiting to happen. There is no value in assigning the root cause of an incident to human error, as there will never be a permanent fix. We can't eliminate human error, but we should anticipate it and work to mitigate its impact.

Conclusion

How we respond to human error and to problems says a lot about our culture. We should expect both and have effective, established, and well-understood processes in place that enable good decisions and positive outcomes. The ideal outcome of any problem investigation or postmortem is finding and solving root causes. But don't let the perfect become the enemy of the good. Improving the experience of your users and customers and providing cost-effective solutions that benefit your company should be the drivers of any IT process, including problem management.