What Takes Us Down?

STUART KENDRICK

Stuart Kendrick works as a third-tier tech at the Fred Hutchinson Cancer Research Center (FHCRC) in Seattle, where he dabbles in troubleshooting, deep infrastructure design, and developing tools to monitor and manage devices. He started earning money as a geek in 1984, writing in FORTRAN on CRAY-1 for Science Applications International Corporation, worked in desktop support, server support, and network support at Cornell University, and reached FHCRC in 1993. He has a BA in English, contributes to BRIITE (http://www.briite.org), and spends free time on yoga and CrossFit.
skendric@fhcrc.org

What are the causes of IT service disruption? With access to an email archive recording both planned and unplanned events, I figured I could identify ways to reduce downtime. This turned out to be neither as easy nor as useful as I had hoped: the exercise raised questions but little that was actionable. Still, the path I took may help you analyze your own data.

Environment

I work at the Fred Hutchinson Cancer Research Center, a nonprofit biomedical research institute specializing in cancer and infectious diseases. I pay attention to deep infrastructure: power, cooling, cabling, transport (Ethernet, IP, WiFi, Fibre Channel), interstitial services (DNS, DHCP, authentication, directory services), email, storage, and file services. These days, I spend my time managing our Problem Management process and leading Root Cause Analysis efforts. (Yes, the ITIL borg is extending its filaments into our brains.)

In the mid-90s, the network team started posting service-affecting incidents, both planned and unplanned, to an email list. Over time, more and more departments followed suit, and more and more techs subscribed to the list. We've refreshed that list server over the years, trashing its archives each time. The current archive starts in October 2000.

In our culture, planned downtime goes over well—we negotiate service level agreements (SLAs) with our divisions specifying when we can take down applications; we notify users; they modify their work flows to dodge the windows during which we are disrupting service; everyone is happy (for some value of happiness). But unplanned downtime is another matter—no one enjoys that, and we invest effort to avoid it.

Today, the Center employs 2500 staff with an annual budget of $450 million (85% from federal grants and contracts) and has 8000 active Ethernet ports, 11,000 active IP addresses, 450 KW datacenter cooling, 1 PB+ mass storage, and national and international collaborations. Roughly 30% of the end-stations run Windows, another 10% Linux, 5% OS X, and the remainder fall into the miscellaneous category (IP phones, printers, etc . . . well most of those run Linux, but I want the Linux figure to reflect desktops/laptops/servers only).

The Outages List

Techs send email to this list using a standardized format. In our lingo, all service-affecting events are called "outages".

```
Subject:        Exchange 2003 Cluster Issues
Severity:       Critical (Unplanned)
Start:          Monday, May 7, 2012, 11:58
End:            Monday, May 7, 2012, 12:38
Duration:       40 minutes
Scope:          Exchange 2003
Description:    The HTTPS service on the Exchange cluster crashed, triggering
                a cluster failover.
User Impact:    During this period, all Exchange users were unable to access
                email.
                Zimbra users were unaffected.
Technician:     [xxx]
```

Figure 1: Example of an unplanned outage

```
Subject:        H Building Switch Upgrades
Severity:       Major (Planned)
Start:          Saturday, June 16, 2012, 06:00
End:            Saturday, June 16, 2012, 16:00
Duration:       10 hours
Scope:          H2 Transport
Description:    Currently, Catalyst 4006s provide 10/100 Ethernet to end-
                stations. We will replace these with newer Catalyst 4510s.
User Impact:    All users on H2 will be isolated from the network during this
                work.
                Afterward, they will have gigabit connectivity.
Technician:     [xxx]
```

Figure 2: Example of a planned outage

You can't see this in the template above, but we also categorize outages by window:

```
Prime       Monday – Friday 7am – 6pm
SLA         Sunday 8pm – Monday 4am or Wednesday midnight – 4am
Shoulder    Any other time
```

Figure 3: Windows

Yes, that service level agreement (SLA) window looks pretty darn generous . . . we can take down services every single week for eight hours straight?! Conceptually, yes, but in practice, research institutes live and die by grant applications. Those grant applications have submission deadlines, and those deadlines pop up almost every week. Thus, most of those SLA windows get blocked.

The severity field has intention glued onto it via parentheses: planned (we intended the outage) or unplanned (we were surprised). Severity itself can contain the following values:

Drama	Most of the Center for 60+ minutes during prime time
Critical	One or more buildings or divisions
Major	Multiple floors or multiple departments
Minor	One floor or one department
None	No end-user effect

Figure 4: Severities

You might ask why we bother to have a Severity of None—doesn't sound like an outage if the end-user impact was None, right? Well, the motivation is two-fold. First, most of us want to be kept informed of changes to the environment (because what you change might in fact interfere with something I do), and the outages list serves as that forum. Second, we're trying to flush out errors in our understanding of the environment; if I claim that an event did not cause a service disruption and you know differently, you'll tell me (and then I'll send a correction to Outages).

Complicated and subjective? Yup.

The Outages Database

So I figured I'd write code to grab the list archives and crawl through them, creating a database entry for each outage. How hard could this be? After all, we use this structured template . . . Ahh, the naiveté and eternal optimism of youth: two weeks and 2500 lines of Perl later (the grossest code I've ever written), I ran it, took the partially processed results, imported into my database, and started scrubbing manually. Turns out we don't follow the template exactly: I never knew there were so many ways to write a date/timestamp, techs twink with the spelling of key words regularly, techs tend to send multiple messages describing each outage (the first announcing the event, the last announcing the completion of the event, and often others in between correcting errors or adding new information) . . . I'll quit whining here.

Furthermore, I wanted a feel for Service and Cause, neither of which is specified in the template. I added those during my manual passes.

Service	**Description**
Application	End-user facing apps (minus email, MIS, and printing)
Email	Exchange, Zimbra, mail relay, spam/malware scrubbers
HPC	High Performance Computing
Interstitial	DHCP, DNS, NTP, authentication, directory services
MIS	Financial Management / Human Resources systems
Power	Electricity
Print	Print servers
Storage	Anything providing file or block services
Transport	Ethernet, WiFi, Remote Access, Fibre Channel
Virtualization	VMware, Xen
Voice	Telephones and pagers

Figure 5: Services

This list reflects the focus of the groups who post to outages. Most of the application support groups do not post; I lump their contributions into a single category.

More interestingly, I wanted to identify the proximate cause of each outage—again, not something defined in the template, so I added this during my manual passes, interpreting the description field, dusting off memories, and making a judgment call.

Cause	Description
Cockpit Error	Techie mistake (fat finger, config error, bump the power cord)
Design Failure	Service didn't behave as the designer intended
External Services	Service provider issue (electric utility, ISP, telecom carrier)
Hardware Failure	The magic smoke escaped
Maintenance	Patching, database compression, shuffling data, minor fixes
Malware	Virus or worm infection
Overload	Too much of a good thing
Software Bug	Memory leak, unhandled exception, Blue Screen of Death
Testing	Validating expected behavior, typically involving high-availability
Unknown	Never figured it out
Upgrade	Adding major functionality (new gear, major software update)

Figure 6: Causes

There's a lot of fuzz here. When we popped a ceiling tile trying to trace cables and knocked an unsecured electrical wire loose, it triggered an emergency power off (EPO) event in our largest datacenter. I categorized the service as Power and the cause as Design Failure. I could have categorized the service as Application (every application in that datacenter went down) and the cause as Cockpit Error (the electrician who installed the EPO circuits intended to screw that wire into place but forgot). I didn't because I try to push cause down the OSI stack (Power sits a whole lot lower than Application), and I try to pick the proximate cause as opposed to the root cause: at the moment we popped the ceiling tile, Power did not behave according to the datacenter designer's intent.

Or, to take another example, if the server stayed up, but became too slow to be usable on account of too many users, I categorized that as Overload. If the server crashed because it ran out of RAM (on account of too many users), I categorized that as Software Bug.

Yup, pretty darn subjective.

NOTES

- External services isn't really a cause, but since the service provider space is opaque to us (did the WAN circuit go down due to human error, an unannounced upgrade, or a power event?), we lump them together.
- Hardware Failure lumps together both unplanned events, during which the magic smoke escaped, and planned events, during which we replace, say, a disk controller which is failing diagnostics but hasn't actually fried yet.
- Maintenance is driven by patching, mostly Windows patching.
- Testing is driven by the network team, which reboots redundant switches and routers monthly.

Results

I ended up with ~2300 outages [1] spanning the last 11+ years. Is that an under-count? Definitely. Departments vary in how frequently they report: some use Outages rigorously, others not at all. And of the entries in Outages, I threw away hundreds which my code didn't parse or which were too terse for me to categorize manually.

Q: How often are we surprised?
A: We're surprised half the time.

Planned: 55% Unplanned: 45%

Q: What takes us down?
A: Software bugs take us down.

For unplanned outages, software bugs are the dominant contributor. And for planned outages, a third arise from maintenance, which is driven by patching, i.e., fixing software bugs.

Planned		Unplanned	
Cause	**Proportion**	**Cause**	**Proportion**
External Services	2%	Cockpit Error	13%
Maintenance	32%	External Services	7%
Other	14%	Hardware Failure	12%
Testing	11%	Other	7%
Upgrade	41%	Software Bug	61%

Figure 7: Planned vs. unplanned by cause

Q: When do we go down?
A: In the middle of the day.

If unplanned outages were to occur at random, regardless of time of day, we would predict that some unplanned outages would land during the Prime window (55 hours/week), most during the Shoulder window (101 hours/week), and a few during the SLA window (12 hours/week). But, in fact, we see that far more land during Prime time than we would expect based purely on chance—perhaps because our users are exercising the systems and uncovering bugs in the process.

Window	Predicted	Measured
Prime	33%	67%
Shoulder	60%	31%
SLA	7%	2%

Figure 8: Unplanned outages by window: predicted vs. measured

Q: What causes induce the most pain?
A: The same causes which induce major and minor pain.

I tried slicing and dicing in other ways but did not uncover new information. For example, when focusing just on the most painful events (Drama and Critical), causes break down pretty much the same as they did when considering all severities:

Planned		Unplanned	
Cause	**Proportion**	**Cause**	**Proportion**
External Services	4%	Cockpit Error	17%
Maintenance	30%	External Services	9%
Other	12%	Hardware Failure	8%
Testing	16%	Other	11%
Upgrade	38%	Software Bug	55%

Figure 9: Planned vs. unplanned by severity (Drama + Critical only)

Q: How often does it hurt a lot?
A: Severity shows a normal distribution.

We experience a few of the really painful Drama outages and a few outages with no end-user effect: most land in the middle.

	Planned	**Unplanned**
Drama	0%	5%
Critical	16%	19%
Major	46%	35%
Minor	34%	39%
None	4%	2%

Figure 10: Planned vs. unplanned by severity

Q: What breaks most often?
A: Transport and email are weak spots.

But see caveats below.

	Planned	**Unplanned**
Application	18%	16%
Email	20%	21%
Other	16%	20%
Storage	14%	8%
Transport	32%	35%

Figure 11: Planned vs. unplanned by service

Reality Check

Fuzzy Data

Those cute tables with numbers in them look good . . . but as I apply cultural knowledge, I lose confidence. For example, the network team founded the outages list; the email team jumped onto the bandwagon shortly thereafter: these two groups have been posting the longest and have become ruthless about reporting every event, no matter how embarrassing. Furthermore, they have the most mature monitoring systems, reporting even minor hiccups. Are Transport and Email our most fragile services? Or do they top the list because of cultural factors: habit, conscientiousness, visibility?

Cockpit Error

Reading thousands of descriptions of outages gave me a chance to smile—I remember many of these events from personal involvement and know each of the techs posting to the list. Senior techs tend to acknowledge their errors directly, using language like "I fumbled the configuration," "I accidentally typed rm –rf * from root," "I broke the Internet connection," whereas junior techs tend to slide into passive voice and circuitous language when they describe their errors: "The service went down during trouble-shooting," "It was discovered that the configuration file contained an error," "On investigation and after analysis, the power cord was found to be detached." Where possible, I flagged the cause as Cockpit Error, but I'm confident that I missed plenty. For that matter, I suspect that Cockpit Error leads to unreported outages, as techs try paddling up the Nile in their efforts to dodge embarrassment. We have a remarkably shame/blame-free environment—as far as I know, no tech has ever been fired for making a mistake and causing an outage. In fact, management likes to stress that making mistakes is how we learn (yeah, OK, sometimes they look a little nervous when they make this point, but still, the sentiment is there). How can we boost the Cockpit Error reporting rate?

Who Else Quantifies This Stuff?

A casual search turned up a handful of studies in this space, with variously sized data sets (typically 100–1000 incidents spanning 1–5 years).

	Gray [2]	Kuhn [3]	Enriquez [4]	Oppenheimer [5]			Offord [6]	Kendrick
Published	1990	1997	2002	2003			2011	2012
				SP1	SP2	SP3		
Cockpit Error	13%	25%	38%	33%	36%	19%	42%	17%
Software	58%	14%	7%	27%	25%	24%	38%	55%
Hardware	18%	19%	30%	25%	4%	10%	—	8%
Other	11%	42%	25%	10%	31%	33%	20%	20%

SP = Service Provider

Figure 12: Similar surveys

I'm skeptical that I'm comparing apples to apples here—both environments and methodologies vary widely. For example, Gray, Kuhn, and Enriquez were all analyzing data sets taken from homogeneous systems (Tandem Computers and the Public Switched Telephone Network), while Oppenheimer, Offord, and I are analyzing heterogeneous environments (Windows/Linux-based systems running on IP/Fibre Channel networks). Or, to take another example, Offord extracts his data set from the log of Root Cause Analysis jobs his company has performed for customers—not exactly outages but rather long-running problems. In 42% of their cases, the problem was fixed by making a configuration change, which I recategorized as Cockpit Error, in order to fit his data into my taxonomy—probably not a precise match.

Tentatively, I see all these data sets directing our attention toward software flaws and operator fumbles as places for improvement.

What to Do?

Software Bugs

For us, our unplanned downtime is driven by Software Bugs (~60%). We know that we lag on patching. When a service fails repeatedly, we'll investigate and often find a patch addressing the issue which the vendor shipped months or years prior. I would like to think that if we patched more regularly, we would convert unplanned outages into planned outages. Still, this is a tricky area—most of our teams don't have test environments (we are nonprofit after all)—so we test patches by running them in production, and as we all know, patches can fix issues we weren't having while introducing new issues. How many unplanned outages would we dodge by patching more aggressively?

Testing

Until recently, the network group tested their redundant routers and switches monthly, rebooting them in series, analyzing failure, fixing the issues they uncovered (typically Cockpit Errors, e.g., misconfigurations), working with sysadmins to fix misconfigured servers (servers which weren't configured to take advantage of the dual Ethernet switches in datacenters), and helping the security groups buff up highly available firewalls. Of our really painful planned outages, Testing contributed 16%. I would like to think this approach saved us a similar number of really painful unplanned outages and thus was a win. On the other hand, testing requires substantial staff time. How to quantify the costs and benefits?

Insights

I have been struck by the number of axes on which one can measure an incident. Each of the authors I cite developed their own taxonomy. To recap, here's mine:

Function	Our Term	Description
Pain Level	Severity	Drama, Critical, Major, Minor, None
Intention	Planned	Planned or Unplanned
Time Frame	Window	Prime, Shoulder, SLA
End-User Impact	Service	The thing that went down
Proximate Cause	Cause	What caused the downtime

Figure 13: Taxonomy

I am troubled by how subjective my categorization process is—I made multiple passes through the database, recategorizing as I became more familiar with my data; nevertheless, I expect that I made inconsistent choices. Also, many outages don't fit the taxonomy cleanly: what to do with a planned outage which incurred unplanned consequences? Or an outage which knocked out multiple services? And cause remains tricky—an outage has so many causes, how to pick just one?

Still, at the end of the day, I'm headed back to problem management meetings to suggest patching and testing as ways to convert unplanned events into planned ones.

Doubt is uncomfortable; certainty is absurd. —Voltaire

References

[1] See http://www.skendric.com/problem/incident-analysis for the summarized data.

[2] Jim Gray, "A Census of Tandem System Availability Between 1985 and 1990," IEEE Transactions on Reliability, vol. 39, no. 4, pp. 409–418, Oct. 1990.On p.6, I used the All Faults column and categorized maintenance +operations + process as Cockpit Error.

[3] Richard Kuhn, "Sources of Failure in the Public Switched Telephone Network," IEEE Computer, vol. 30, no. 4, April 1997, pp. 31–36. I counted only errors from telco staff as Cockpit Error, allocating "Human error—external" to Other.

[4] P. Enriquez, A.B. Brown, and D. Patterson, "Lessons from the PSTN for Dependable Computing," Proceedings of the 2002 Workshop Self-Healing, Adaptive, and Self-MANaged Systems (Shaman), 2002, pp. 1–7. Again, I allocated "Human error—external" to Other.

[5] David L. Oppenheimer, A. Ganapathi, D. Patterson, "Why Do Internet Services Fail, and What Can Be Done About It?" USENIX Symposium on Internet Technologies and Systems, 2003.

[6] Paul Offord, "RPR Statistics," Advance7, October 2011. I map the sum of "bug fix" and "programming" into my Software Bug. Programming is Advance7's term for a bug fixable by internal resources, e.g., a bug found in an in-house application, while bug fix is Advance7's term for a bug found in software acquired externally, e.g., commercial or open source.