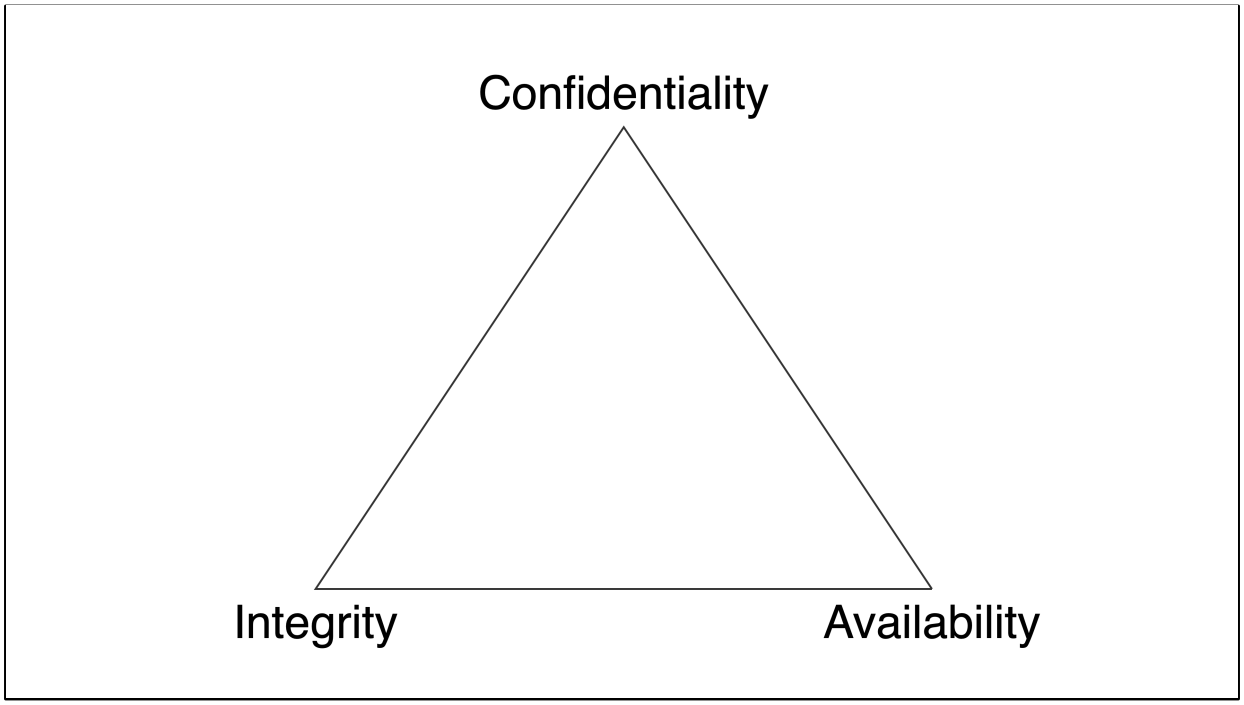
Measuring and Communicating Availability Risk

John Benninghoff

About 3 years ago I left Application Security and returned to my roots in infrastructure, joining and then leading the team establishing a Site Reliability Engineering practice at Cigna.

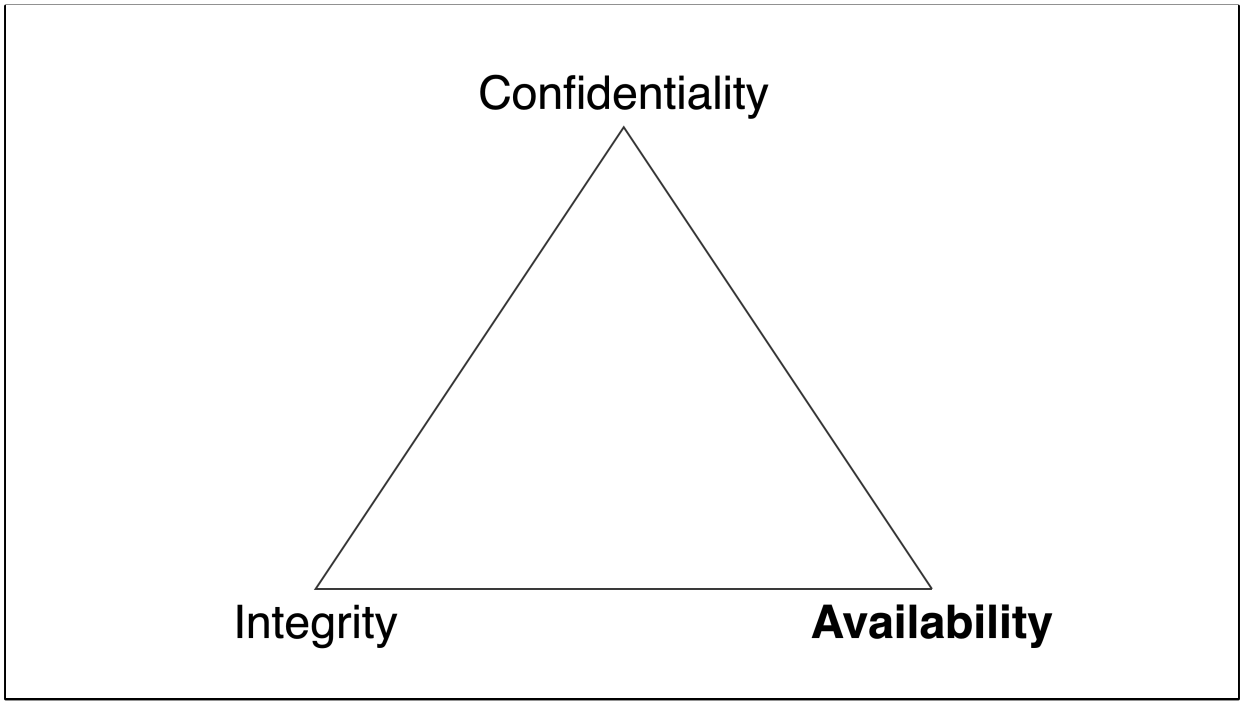
https://www.information-safety.org/2021/04/06/what-is-resilience-engineering/

Dog (& Gertie) Quantification: +1



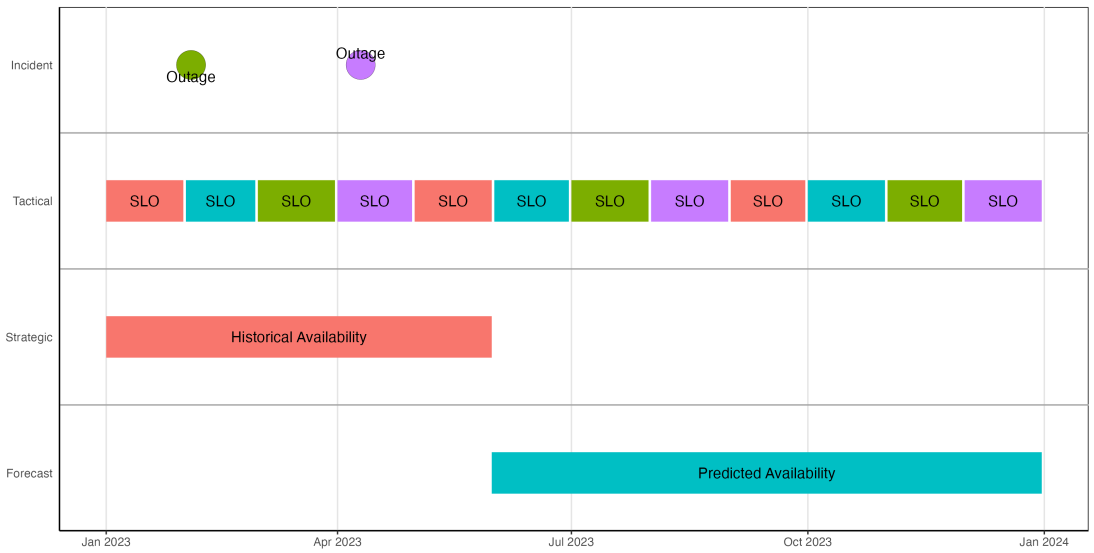
Security CIA triad; as I have to sometimes remind cybersecurity people, Availability is a key part of Information Security and managing information risk.

https://en.wikipedia.org/wiki/Information_security#Key_concepts



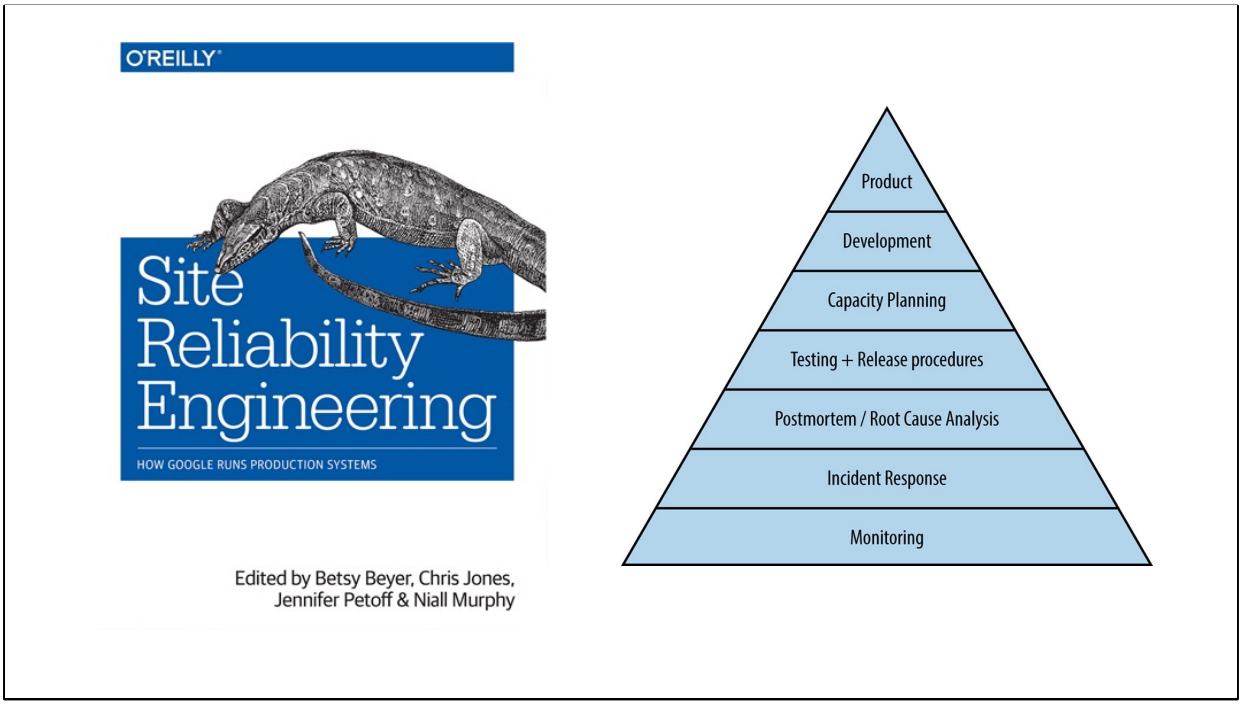
The focus of this talk is on availability, which can be easier to measure than confidentiality or integrity.

Why Measure Availability?



Why Measure Availability? *Decision support* across different time frames: Incident (minutes), Tactical (month), Strategic (year), Forecast (future)

Thanks to: https://www.random.org/calendar-dates/ (incident dates), https://shosaco.github.io/vistime/articles/gg_vistime-vignette.html (timeline plot)

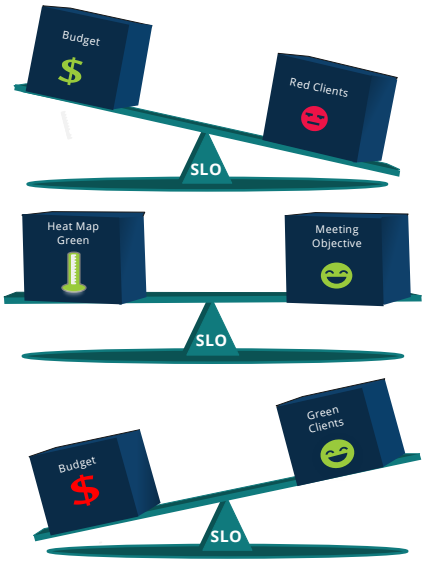


Intro to SRE. SRE started at Google, now being adopted more broadly; annual conference (SREcon). Google's SRE book discusses Reliability Risk and managing the risk of unplanned downtime (availability risk). 100% availability is not achievable and more reliability than needed.

SRE Chapter 3: https://sre.google/sre-book/embracing-risk/.
Book Cover: https://sre.google/books/, Pyramid: https://sre.google/sre-book/part-III-practices/

SRE at Cigna

- Motivation: move from emotional reactions to outages to data-driven decisions about investments in availability
- Things our SRE team learned (lessons), things that resonated with decision makers (hits)



Technology leadership's goal: to change from emotional conversations about the latest outage to data-driven decisions about the level of availability needed ("how good is good enough"?) and the resources needed to achieve that level of availability, to balance the budget and happy clients/stakeholders using SLOs. SLO = Service Level Objective, a defined and measurable performance/availability. Not meeting your SLO is a signal to prioritize work to improve reliability; very much like what Ira talked about yesterday, optimizing investments in risk.

The talk will cover things we learned (lessons) and things that resonated with decision makers (hits)

Incidents

"How bad is it?"

Supporting decisions in real-time

Service Level Indicators

Our Experience

- Latency
- Errors

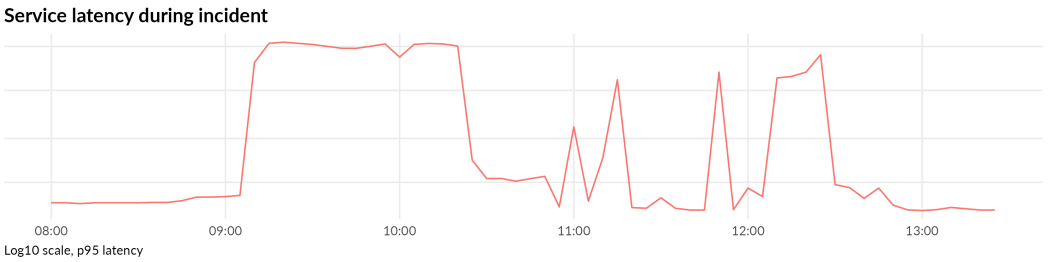
Industry (Four Golden Signals)

- Latency
- Traffic
- Errors
- Saturation

Industry uses the “Four Golden Signals”. We found that Latency and Errors (Error Rate) were easier to start with, as have other organizations, and are working on implementing ’default’ SLOs using a combination of Latency and Error rate.

SRE Chapter 6: https://sre.google/sre-book/monitoring-distributed-systems/

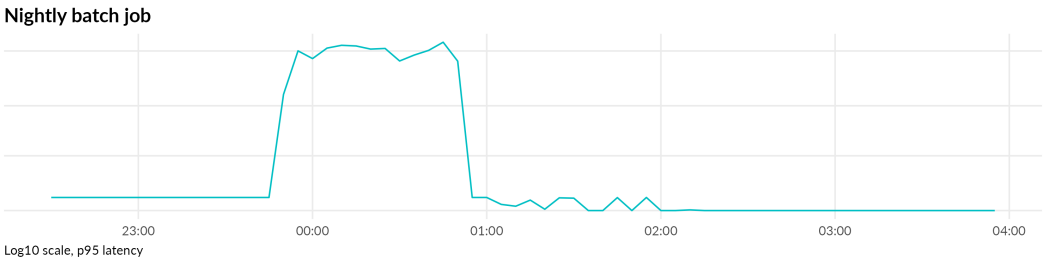
Lesson: Infrastructure Service



First SLI: created on the fly during an incident. Discovered that teams declared an incident when the 95th percentile of processing time exceeded a specific threshold. Measuring latency during the incident identified not only when service recovered, but also which cluster nodes were heathy and unhealthy.

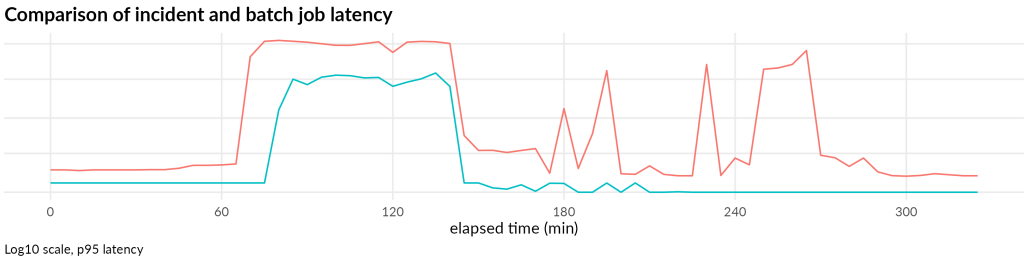
Tools: R/tidyverse/ggplot2

Lesson: Infrastructure Service



Reviewing latency over time, we discovered what looked like an incident happening when a nightly job ran.

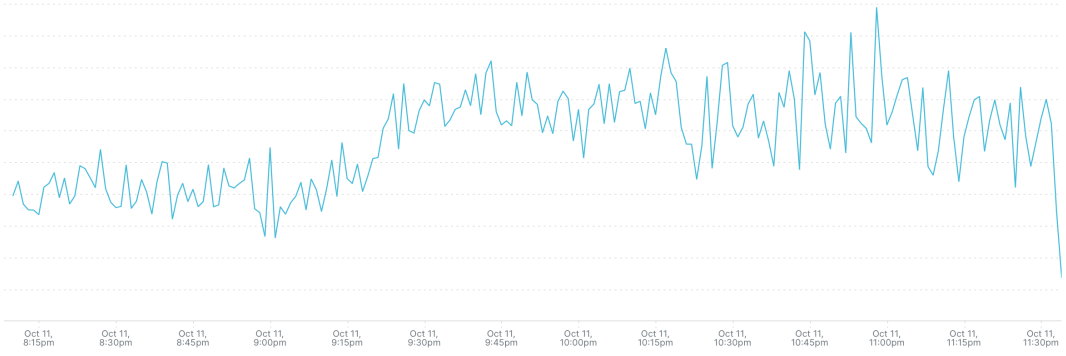
Lesson: Infrastructure Service



Comparison of incident and nightly batch job; the latency spike was similar, although the impact was less noticeable as it happened overnight. Unfortunately, this wasn't a hit: we built it, the team supporting the service liked it, but it never caught on. Building this talk, this was a theme: we needed to build a data story that resonated with the decisionmaker. (Kudos to Sara for her excellent talk!)

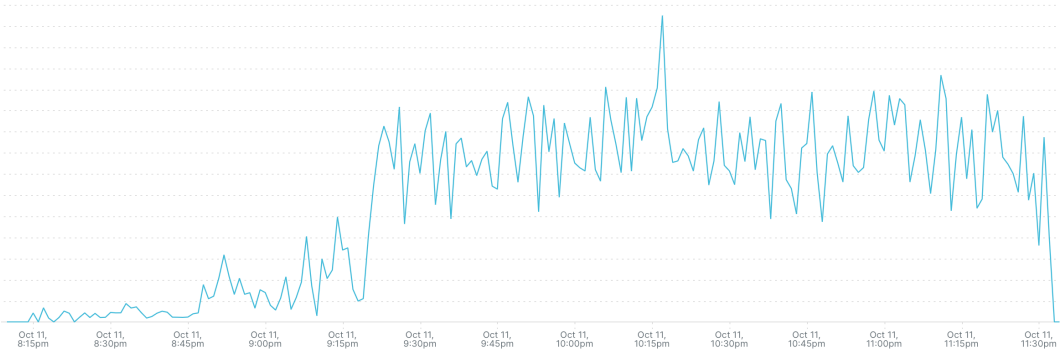
While this SLI/SLO wasn't a "hit" (the service is being replaced and retired), it did help us identify a problematic batch job which was shut off after an investigation found it was no longer needed.

Hit: Claims Failure Rate



(Foreshadowing) while working on the Error Budget Dashboard, we discovered that we'd found a better way of measuring availability real time. Background on Claims Processing (real-time at the pharmacy). Historically, we looked at the claims reject rate and took action when claims were rejecting above normal. This is a view from our monitoring tool of an incident on Oct 11, which started around 9:30 pm. The problem with using rejects is that it generates false alarms: we can and should reject some claims (refill too soon, drug utilization review).

Hit: Claims Failure Rate



In the improved view, we measured a failure/error rate: the number of claims that weren't processed due to a technology failure. This is a view from our monitoring tool of the same Oct 11 incident. Because claims are normally failing at or near 0%, there is a higher signal-to-noise ratio with this visualization and we can see there was impact earlier in the evening. In hindsight, this new view would have allowed us to detect the issue sooner and avoid the larger impact at 9:30. This resonated with both the teams who support this service and our incident responders.

Tactical

"Is it safe to do maintenance, and how much time do we have?"

Supporting decisions on scheduling maintenance and changes

Meaningful Availability

$$\text{availability} = \frac{\text{good service}}{\text{total demanded service}}$$

Early on, I discovered a Google presentation from the 2020 USENIX Networked Systems Design and Implementation conference, which describes the history and evolution of SLOs at Google; I use it as a roadmap. The formula here describes what we're trying to do when we measure availability.

Talk/Paper: https://www.usenix.org/conference/nsdi20/presentation/hauer

Meaningful Availability Metrics

- Time-based metrics (uptime)
- Count-based metrics (ratio)
- Probes (synthetics)
- Advanced metrics

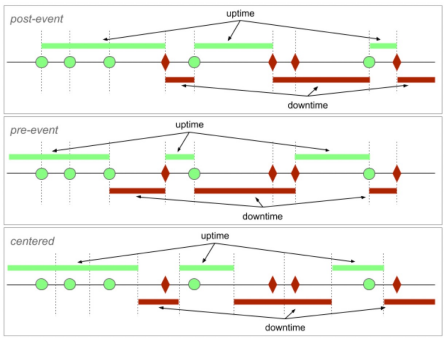
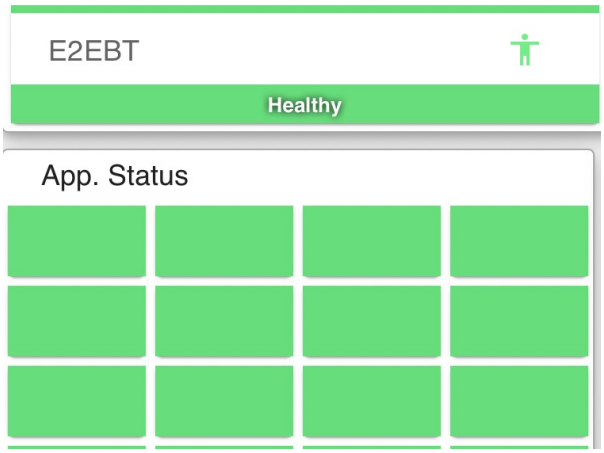


Figure 4: Three choices to extrapolate uptime or downtime from neighboring events

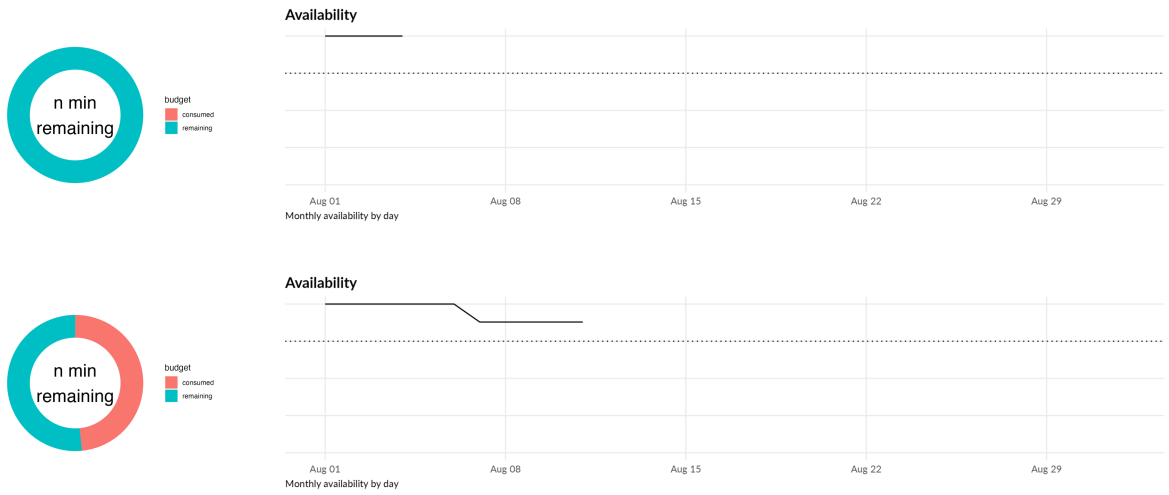
The paper describes different ways of measuring availability in a way that we've discovered lines up with a natural progression: uptime measures are the easiest to understand, and where we started. Fairly quickly people discover that all minutes are not created equal, which leads to ratio-based measures and synthetics. The paper describes more advanced user-based metrics that we haven't had a need for yet.

Lesson: E2EBT Dashboard Light



(redacted) This is a zoomed in view of the dashboard light we built for our enterprise monitoring status dashboard. Below you can see where individual applications will show up when there's an issue (yellow, red). Above the "E2EBT" shows healthy. For the pilot, we picked a specific customer interaction with our call center. We identified 2 redundant ways the call center could answer the customer question; this light turns yellow when one has failed and red when both have failed. While the demo was a success, the approach didn't catch on – in retrospect, the 'customer journey' we picked was too narrow and not meaningful/important enough to our customers and internal stakeholders. However, it did demonstrate what was possible and led to development of the Error Budget Dashboard.

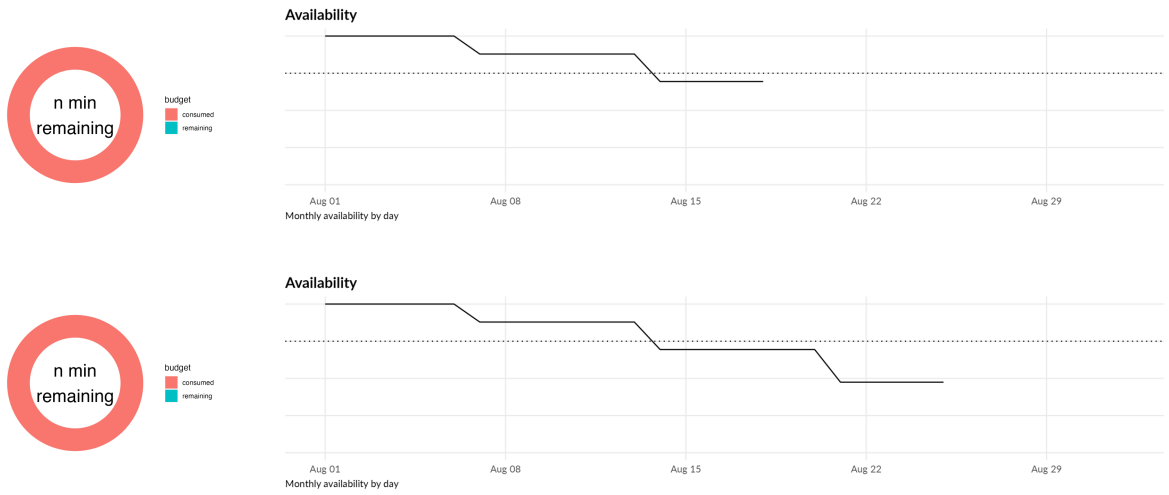
Hit: Error Budget Dashboard



(Big Hit! Emotional connection) Mock-up of the Error Budget Dashboard showing a "what if" scenario: what if we had to count planned downtime against our claims availability? A client made this change to our contract (performance guarantees) so we needed a way to measure it. This shows how the error budget burns down over time. The "n min remaining" shows how many minutes left we have to spend before we fall below our availability commitment. The production dashboard uses automation to populate a similar dashboard with monitoring data.

Tools: R/tidyverse/ggplot2

Hit: Error Budget Dashboard



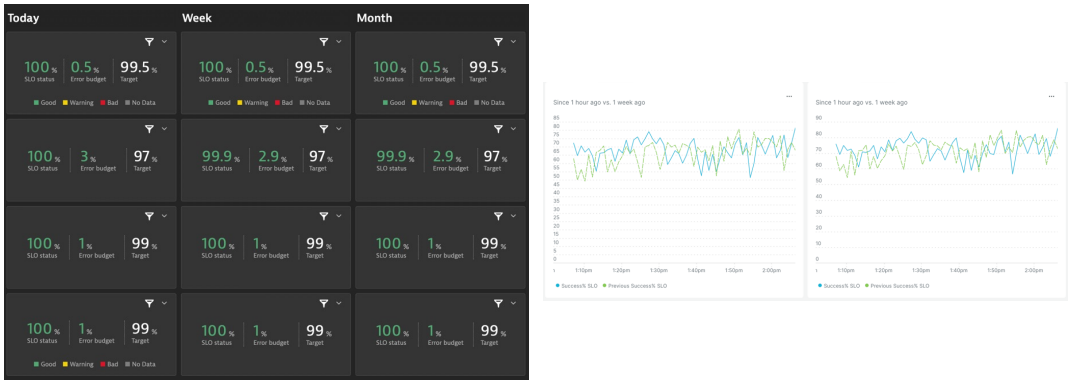
As you can see, our planned maintenance would have quickly brought us below our availability threshold. We knew this and planned for this to change our maintenance schedule, but needed a way to track our budget after changing to the new contract which counted both planned and unplanned downtime.

Error Budget: Results

- Measuring availability with monitoring increased accuracy
- Consistently meeting new availability target
- Team actively consults the dashboard when planning maintenance to make a risk-informed decision
  - "Do we have enough minutes?"
- I get emails right away when the dashboard isn't accurate

The dashboard is actively informing decisions on maintenance, and over time as it becomes more widely known and understood should have a larger impact by informing decisions on other changes, including deployment of new features that could affect this service's availability. In this case, the real work was done by our technology team, we were the scorekeepers, which turned out to be a valuable service.

Hit: Digital SLO Dashboards



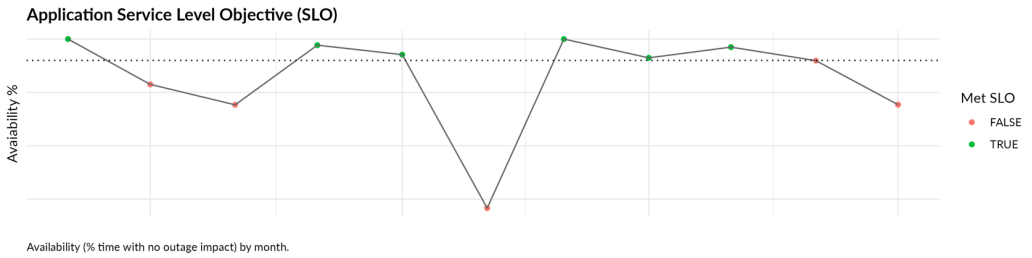
(redacted) The early work we did on infrastructure SLI/SLO influenced our Digital (Web and Mobile) teams to build their own SLO dashboards! They built a couple of different versions, one showing the SLO and Error Budget based on synthetic success/failure (left) and based on application metrics (right, a combination of error rate and latency). In practice, the dashboard on the right is more useful in real time (incidents) and the dashboard on the left reflects the historical impact of incidents.

Strategic

"Which applications are underperforming?"

Supporting executive decisions on prioritization

Lesson: measuring availability using outages

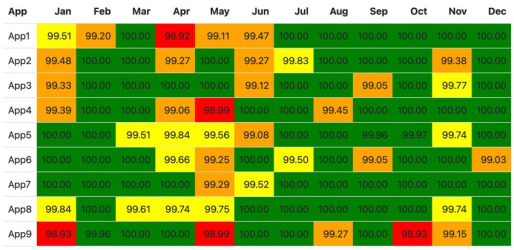


When we were tasked with creating an SLO for a key application, we had good data on a) outages that affected the app b) outage severity and duration and c) customer and internal stakeholder sentiment (red/yellow/green). Using this, I was able to create a SLO based on outages that aligned to sentiment. The SLO for this application was defensible, but never took hold.

Tools: R/tidyverse/ggplot2

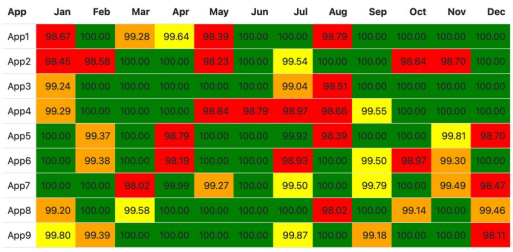
Hit: measuring availability using outages

Sev1 Outage Time



App	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
App1	99.51	99.20	100.00	98.92	99.11	99.47	100.00	100.00	100.00	100.00	100.00	100.00
App2	99.48	100.00	100.00	99.27	100.00	99.27	99.83	100.00	100.00	100.00	99.38	100.00
App3	99.33	100.00	100.00	100.00	100.00	99.12	100.00	100.00	99.05	100.00	99.77	100.00
App4	99.39	100.00	100.00	99.06	98.99	100.00	100.00	99.45	100.00	100.00	100.00	100.00
App5	100.00	100.00	99.51	99.84	99.56	99.08	100.00	100.00	99.96	99.97	99.74	100.00
App6	100.00	100.00	100.00	99.66	99.25	100.00	99.50	100.00	99.05	100.00	100.00	99.03
App7	100.00	100.00	100.00	100.00	99.29	99.52	100.00	100.00	100.00	100.00	100.00	100.00
App8	99.84	100.00	99.61	99.74	99.75	100.00	100.00	100.00	100.00	100.00	99.74	100.00
App9	98.93	99.96	100.00	100.00	98.99	100.00	100.00	99.27	100.00	98.93	99.15	100.00

Sev1 + Sev2 Outage Time



App	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
App1	98.67	100.00	99.28	99.64	98.39	100.00	100.00	98.79	100.00	100.00	100.00	100.00
App2	98.45	98.58	100.00	100.00	98.23	100.00	99.54	100.00	100.00	98.84	98.70	100.00
App3	99.24	100.00	100.00	100.00	100.00	100.00	99.04	98.51	100.00	100.00	100.00	100.00
App4	99.29	100.00	100.00	100.00	98.84	98.79	98.97	98.66	99.55	100.00	100.00	100.00
App5	100.00	99.37	100.00	98.79	100.00	100.00	99.92	98.39	100.00	100.00	99.81	98.70
App6	100.00	99.38	100.00	98.19	100.00	100.00	98.93	100.00	99.50	98.97	99.30	100.00
App7	100.00	100.00	98.02	99.99	99.27	100.00	99.50	100.00	99.79	100.00	99.49	98.42
App8	99.20	100.00	99.58	100.00	100.00	100.00	100.00	98.02	100.00	99.14	100.00	99.46
App9	99.80	99.39	100.00	100.00	100.00	100.00	99.87	100.00	99.18	100.00	100.00	98.11

The lesson from the single application influenced the creation of our application “heatmap”. This is a mock-up of the heatmap using randomly generated data. The heatmap showed availability and performance of our most important applications month-to-month, and has been used in our weekly CIO reliability update for ~1 year. Prior to the heatmap, we reported only on outages and overall availability; introduction of this view has helped facilitate conversations with our internal stakeholders outside of technology about the reliability of specific applications.

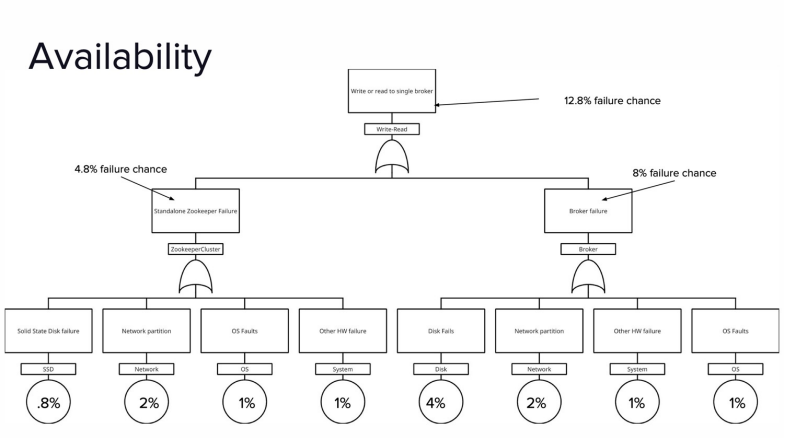
Tools: R/tidyverse/kableExtra

Forecast

"Should we invest in improving reliability?"

Supporting decisions on future investment

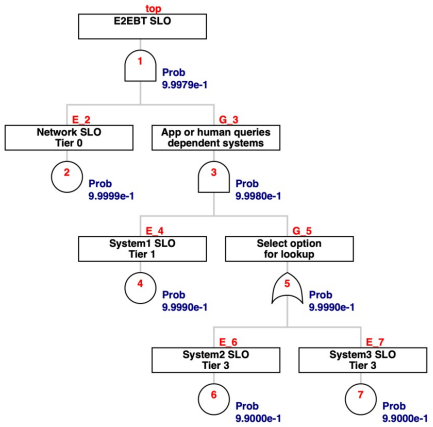
Fault Tree Analysis



Fault Tree Analysis Applied to Apache Kafka, Andrey Falko, Lyft. In this talk, Andrey uses a combination of FTA and SLOs to forecast the availability of Apache Kafka in different design configurations. FTA is a reliability/failure analysis method originally developed by Bell Labs in the 1960s to evaluate the Minuteman launch control system using logical AND and OR gates.

Reference:
https://www.usenix.org/conference/srecon19americas/presentation/falko,
https://en.wikipedia.org/wiki/Fault_tree_analysis

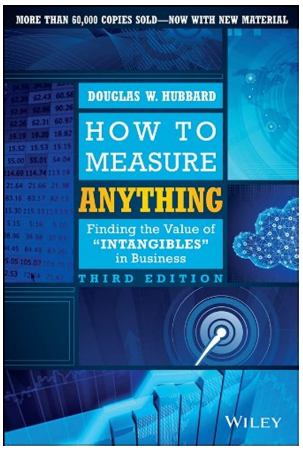
Lesson: Fault Tree Analysis



This fault tree shows some of the analysis behind the E2EBT (For the pilot, we picked a specific customer interaction with our call center. We identified 2 redundant ways the call center could answer the customer question). This shows how we translated the Journey Map into a reliability model, using placeholder SLOs based on ”Tier”. This model helped us forecast the reliability of this particular transaction. Additionally, we were able to forecast the availability after the planned retirement of System2: it turned out that System3 was highly reliable, so the retirement of System2 would have minimal impact on availability. While I found these insights useful, they didn’t resonate as well with the stakeholders, who found this too ‘academic’ or ‘theoretical’. [Technically, this is backwards – the FTA should show the risk of failure, not success, but the math still works]

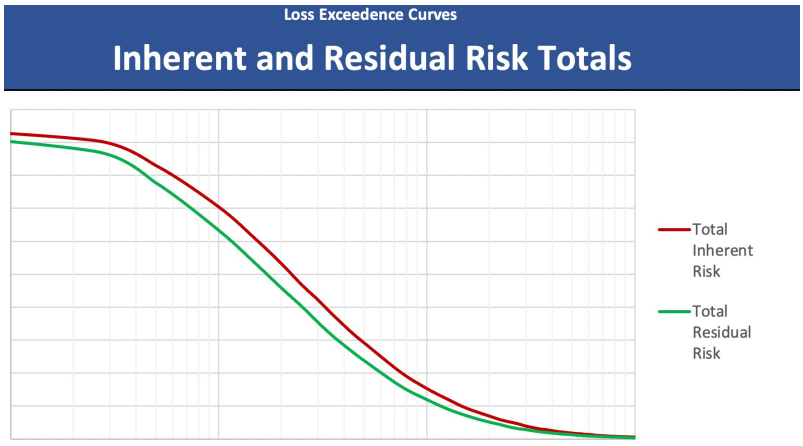
Tools: R, FaultTree, FaultTree.widget (http://www.openreliability.org)

Risk Quantification



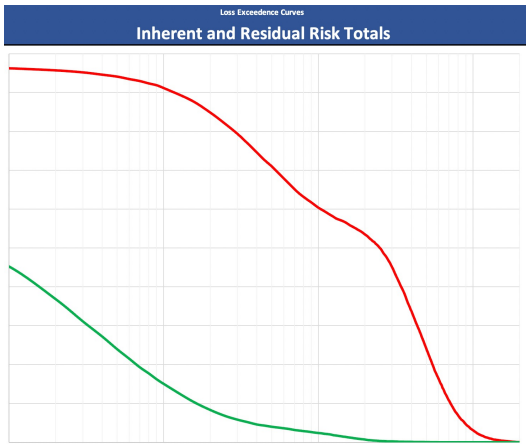
Our availability risk quantification pretty much follows the Hubbard playbook. Not following a specific framework, like FAIR, turned out to be a good decision.

Lesson: Risk Quantification



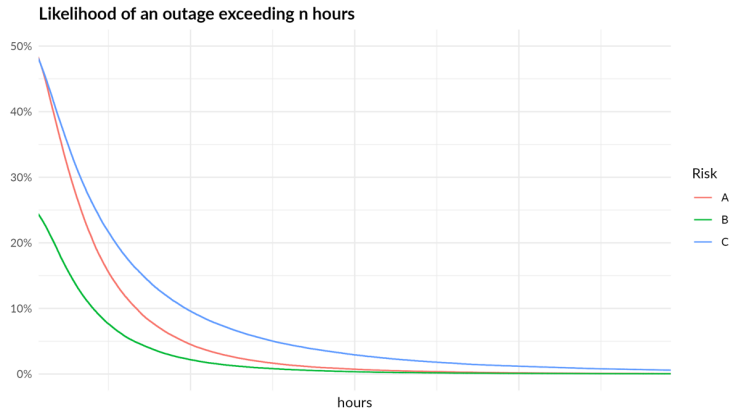
We did a RQ as part of the same E2EBT pilot, measuring availability risk before and after a planned system upgrade. We discovered that there wasn't a significant reduction in risk, which was a bit of a surprise, but in reflection makes sense (the benefits of the upgrade go beyond just availability improvements, and the inherent risk is tolerable).

Hit: Risk Quantification



A separate RQ was a much bigger success. This RE plot told a different story: our technology team asked for our RQ team's help in estimating the availability risk of a legacy system, compared to a (hypothetical) replacement system. Following a previously proven methodology, the RQ analyst (Dave Grimmer, here virtually) scheduled interviews with both technology staff and operational/business staff, and most importantly, didn't limit his questions to just the risk due to an unplanned outage, instead asking all stakeholders "What else should we be worried about?" This identified additional risks that contributed more to the overall inherent risk than availability. Results: our technology team used this report to present the risks identified by their own team to business leadership, who funded a project to replace the legacy system (scheduled to complete this year). As we've seen with prior presentations, quantifying risks in monetary values resulted in a level of engagement with our business partners that the technology team hadn't seen before, and led to a better investment decision.

Future: Scalable Risk Quantification



What's next? I'm working on scalable risk quantification. The challenge with a full FAIR or Hubbard analysis is that it requires a fair amount of time and work. This demo which shows 3 arbitrary risk shows how we could use a consensus SME estimate of likelihood and a fitted lognormal distribution derived from historical incident duration to create a risk exceedance curve that can be used to compare different risks. Using this approach, we should be able to get an estimate in 30 minutes or less.

Tools: R/tidyverse/ggplot2/rlnorm/runif

Lessons Learned

- To be effective, measurements must be meaningful to customers and stakeholders
  - SLOs based on SLAs with real penalties are meaningful!
- Use the data you have
  - Incidents by application provides a reasonable measure of availability
- Don't limit the scope of your risk quant!
  - "What else should we be worried about?"

Questions?

https://www.information-safety.org
https://www.linkedin.com/in/jbenninghoff