



Postmortem Culture at Google

How do we learn from failures? How can you, too?

Ramón Medrano Llamas <niobium@google.com> / Conf42 SRE



Site Reliability Engineering

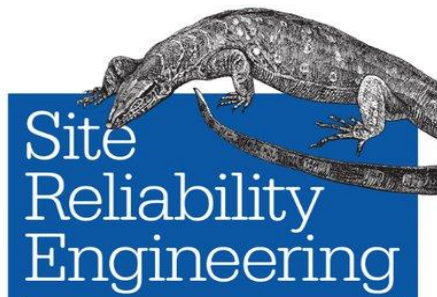
Introduction

- What are postmortems?
- Why should we write postmortems?
- How should we write postmortems?

Error culture



"100% is the wrong reliability target for basically everything."
(Benjamin Treynor Sloss, Vice President of 24x7 Engineering, Google)



*"We constantly enhance our services with **new features and add new systems**. Incidents and **outages are inevitable** given our scale and velocity of change."*
(Sue Lueder, John Lunney; Site Reliability Engineering)



Accept failure as normal!

What are postmortems?

Written record of an incident:

- Documentation of the incident (oncall actions, user impact, ...)
- Detailed summary of root causes and triggers
- Effective and preventive actions to reduce likelihood or impact of recurrence

Postmortems / Incident reports are a common tool outside IT industry— e.g., in aviation or medical engineering.



Why should we write postmortems?

Learning from failure...

Unless we have some formalized process of **learning** from these incidents in place, they may **recur ad infinitum**.

(Sue Lueder, John Lunney; Site Reliability Engineering)

... to prevent future outages.

- Postmortems are a great tool to learn about the reliability (problems) of complex systems.
- Postmortems enable qualitative and quantitative analysis of reliability engineering impact and help in prioritizing and decision making:

We've seen most common patterns in reviewing postmortems and are writing infrastructure to eliminate them
(Benjamin Treynor Sloss)

- Blameless Postmortems are core to Google's SRE values



But outages still happen?

- Infrastructure and best practices need to be applied across all our use cases.

Continuous process: New patterns emerge as others are rendered less common.

How should we write postmortems?

Blameless!

- Fixing systems and processes,
... not people.
- Psychological safety enables interpersonal
risk: ask questions; admit failures.
- Learn!

... but do not celebrate heroism!



Writing Postmortems

- When should you write a postmortem?
- Who should write the postmortem? Who's the audience?
- What information should be captured?
- What might a sample postmortem process look like?

When should you write a postmortem?

Impact criteria is met?

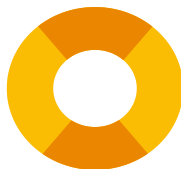


- X users affected?
- \$\$ revenue loss?
- **Potential** impact of X
- Severity: medium, major, huge

Define your own criteria when a postmortem **MUST** be written.

Define the criteria beforehand!

Near miss?



Defined criteria would have been met without luck?

→ **SHOULD** write a postmortem!


Interesting learning opportunity?

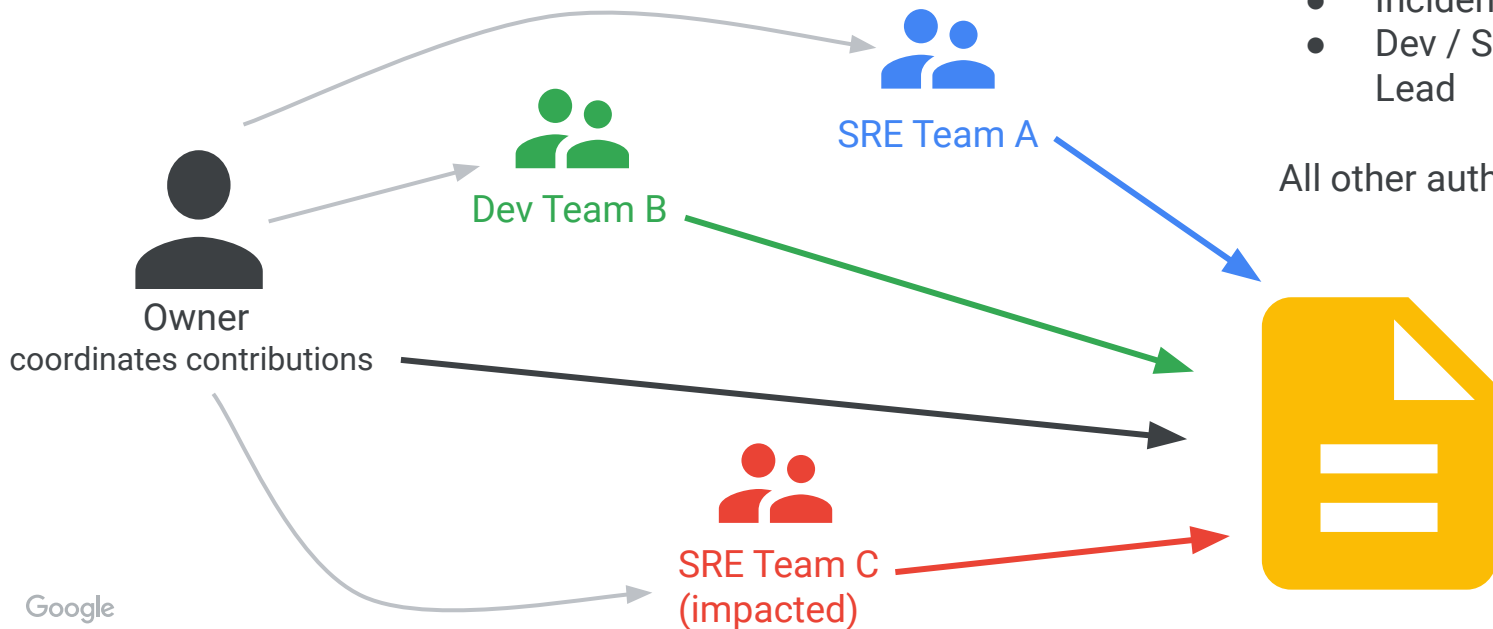


- Effort vs. learnings?
- Are there interested parties?
- Do a quick write up!
(relaxed reviews, action items)

→ **MAY** write a postmortem

Who should write a postmortem?

Writing a postmortem is a **collaborative** effort! 
Use real time / asynchronous collaboration tools whenever possible.



Who should own a postmortem?

→ Ideally a single person!

Popular choices:

- Incident Commander
- Dev / SRE Manager / Tech Lead

All other authors: *collaborators*

Who will read your postmortem?



Team

Who?

Engineers, managers, and others
in the team

- Only little background is needed, people are **familiar** with the context.
- Main interest: root cause analysis balanced action item plan.

→ **All postmortems**



Company

Who?

Directors, architects,
affected teams, ...

- Detailed background required, people **not very familiar** with the context
- Main interest: organizational learning, risk management
- **High impact, near miss, cross team postmortems**



Customers / Public

Who?

Customer engineers, legal, ...

- People **not familiar** with the company internals and context!
- Main interest: (re-)gain trust in company action plan and execution. Learning from failures of others.

Typically requires a differently structured document (high level)

What incident highlights / lowlights should be captured?

Essentials

Root Cause, Trigger, and Impact

- *Canary metrics didn't detect a bug in a previously unused feature. (Root Cause)*
- *Feature was enabled by accident and rolled out. (Trigger)*
- *Enabled feature put all our worker threads in an error state.*
- *Product ordering was unavailable for 4h, resulting in a revenue loss of \$ (Impact)*

Action Item plan

- *Implement canary analysis to **detect** failures (error state) before rollout to all workers.*
- ***Prevent** accidental enabling of features by additional rollout checklist item.*

Lessons Learned

- *What went well / poorly? Where we got lucky?*

Supporting Material

- *Chat logs, Metrics, Timeline, (Design) Documentation, Links to Commits / Code*

Useful metadata to capture

Ownership / Authors

- ❑ Owner / Collaborators / Owning teams

Review Status

- ❑ Draft / In Review / Published
- ❑ Reviewers pending / with LGTM

Executive summary (1-2 sentences) of

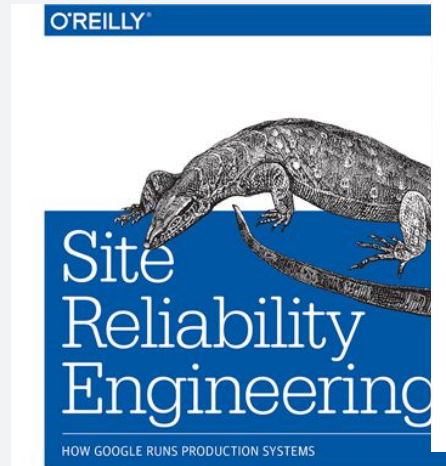
- ❑ Impact / Root Cause

Impact quantification

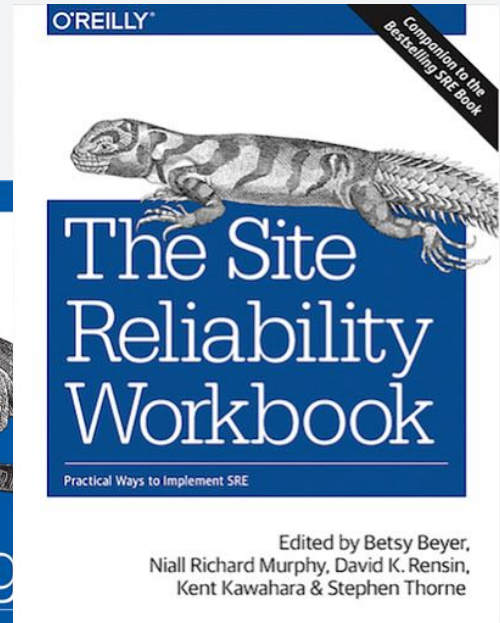
- ❑ SLO violation duration
- ❑ Number of affected users, regions, or customers
- ❑ Revenue loss

Timeline for:

- ❑ Root cause introduced and triggered
- ❑ Start of impact
- ❑ Incident detected, escalated, mitigated, and resolved
- ✓ enable quantification of MTTx (mean time to detection / escalation ...) from multiple outages



Edited by Betsy Beyer, Chris Jones,
Jennifer Petoff & Niall Murphy



Edited by Betsy Beyer,
Niall Richard Murphy, David K. Rensin,
Kent Kawahara & Stephen Thorne

contain [detailed example](#), [case study](#), and [checklist](#)

Example Postmortem - Metadata

Shakespeare Sonnet++ Postmortem (incident #465)

Date: 2015-10-21

Authors: jennifer, martym, agoogler

Status: Complete, action items in progress

Summary: Shakespeare Search down for 66 minutes during period of very high interest in Shakespeare due to discovery of a new sonnet.

Impact:¹⁶³ Estimated 1.21B queries lost, no revenue impact.

Root Causes:¹⁶⁴ Cascading failure due to combination of exceptionally high load and a resource leak when searches failed due to terms not being in the Shakespeare corpus. The newly discovered sonnet used a word that had never before appeared in one of Shakespeare's works, which happened to be the term users searched for. Under normal circumstances, the rate of task failures due to resource leaks is low enough to be unnoticed.

Trigger: Latent bug triggered by sudden increase in traffic.

Resolution: Directed traffic to sacrificial cluster and added 10x capacity to mitigate cascading failure. Updated index deployed, resolving interaction with latent bug. Maintaining extra capacity until surge in public interest in new sonnet passes. Resource leak identified and fix deployed.

Detection: Borgmon detected high level of HTTP 500s and paged on-call.

Most postmortems require some narrative, details of what happened, background etc. This is omitted in this simple example.

Example Postmortem - Action Item Plan

Action Item	Type	Owner	Bug
Update playbook with instructions for responding to cascading failure	mitigate	jennifer	n/a DONE
Use flux capacitor to balance load between clusters	prevent	martym	Bug 5554823 TODO
Schedule cascading failure test during next DiRT	process	docbrown	n/a TODO
Investigate running index MR/fusion continuously	prevent	jennifer	Bug 5554824 DONE
Plug file descriptor leak in search ranking subsystem	prevent	agoogler	Bug 5554825 DONE

Example Postmortem - Learnings & Timeline

Lessons Learned

What went well

- Monitoring quickly alerted us to high rate (reaching ~100%) of HTTP 500s
- Rapidly distributed updated Shakespeare corpus to all clusters

What went wrong

- We're out of practice in responding to cascading failure
- We exceeded our availability error budget (by several orders of magnitude) due to the exceptional surge of traffic that essentially all resulted in failures

Where we got lucky

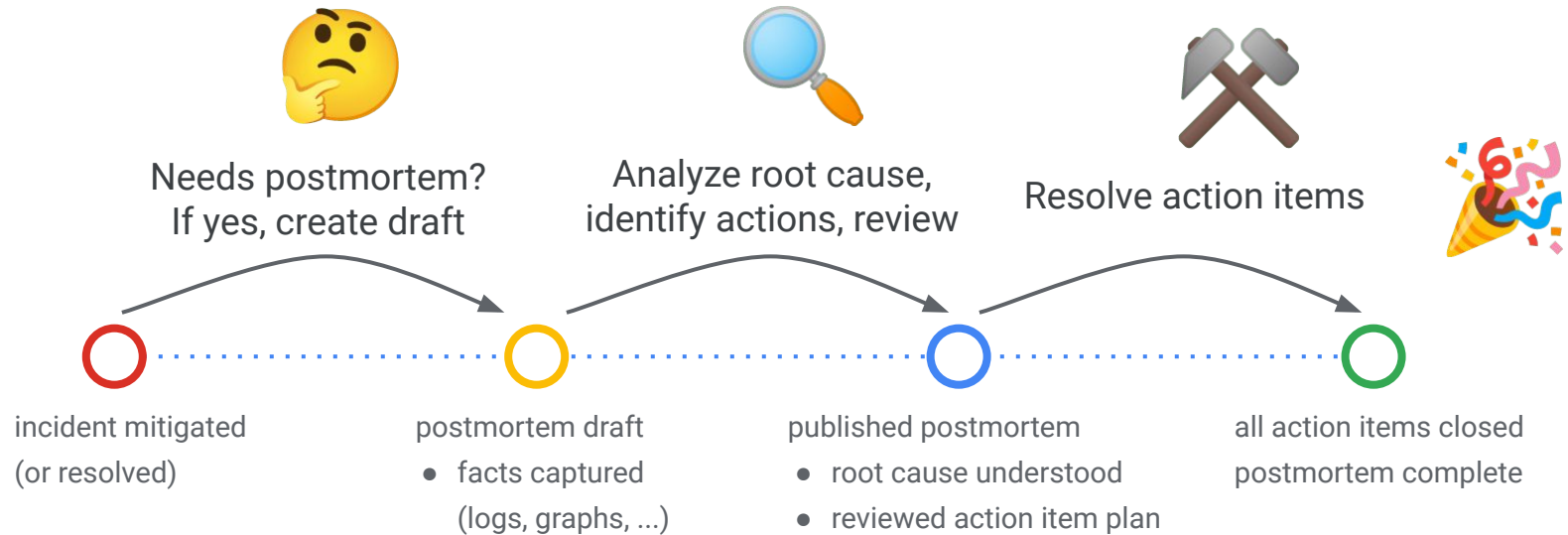
- Mailing list of Shakespeare aficionados had a copy of new sonnet available
- Server logs had stack traces pointing to file descriptor exhaustion as cause for crash
- Query-of-death was resolved by pushing new index containing popular search term

Timeline

2015-10-21 (all times UTC)

- 14:51 News reports that a new Shakespearean sonnet has been discovered in a DeLorean's glove compartment
- ...
- 14:54 **OUTAGE BEGINS** — Search backends start melting down under load
- ...
- 14:58 docbrown starts investigating, finds backend crash rate very high
- 15:01 **INCIDENT BEGINS** docbrown declares incident #465 due to cascading failure, coordination on [#shakespeare](#), names jennifer incident commander
- ...
- 15:36 jennifer reverts load balancing to resacrifice USA-2 cluster in preparation for additional global 5x instance count increase (to a total of 10x initial capacity)
- 15:36 **OUTAGE MITIGATED**, updated index replicated to all clusters
- ...
- 15:47 nonsacrificial clusters' HTTP 500 rates remain within SLO, no task failures observed
- 15:50 30% of traffic balanced across nonsacrificial clusters
- 15:55 50% of traffic balanced across nonsacrificial clusters
- 16:00 **OUTAGE ENDS**, all traffic balanced across all clusters
- 16:30 **INCIDENT ENDS**, reached exit criterion of 30 minutes' nominal performance

Postmortem Process



Action Items

- Prerequisites
- Balanced Action Item Plans
- Execution

“To our users, a postmortem without subsequent action is indistinguishable from no postmortem.”

Benjamin Treynor Sloss

Vice President of 24x7 Engineering, Google

Prerequisite: understand the root causes

one simple approach to get started:
Five Whys

Key idea:
Ask "Why" until the root cause(s) are **understood and actionable**. Start from the impact.

Almost always, more than one root cause:
Try to assess the risk of each to later decide on action items efforts.



Best practices for a balanced action item plan



Band aid fix vs. solving root cause of the problem

Fixing the root cause is often expensive:

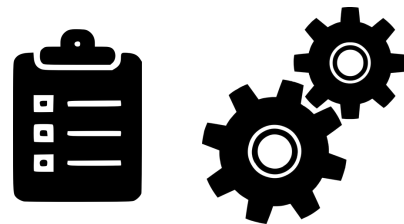
→ **Balance** short term and long term action items.



Think beyond *prevention!*

Early detection, improved diagnosis and triage capabilities can shorten and thus reduce the impact of future outages.

E.g., require a "Detect"-type action item if users detected the issue first.



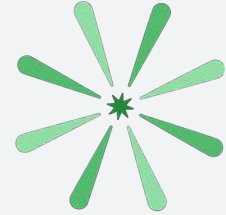
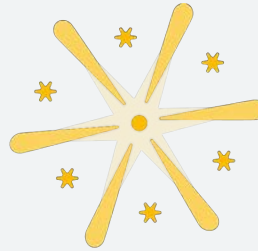
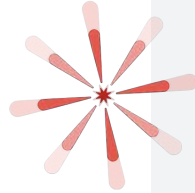
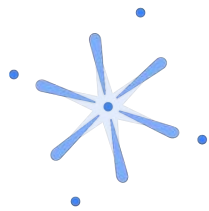
Humans as root cause?

Don't fix humans - reduce their chance to introduce errors!

These action items are often cheap:

- Add / clarify documentation and playbooks
- Protect dangerous flags
- Automate!

Postmortem published! What's next?



Celebrate Postmortems by

- Sharing in Review Clubs
- Postmortem of the month
- Replaying as "Wheel of Misfortune"



Executing Action Item Plans



Pick the right priorities!

- What has the biggest impact reduction potential for the lowest investment?

Antipattern: Low effort band-aid action items with a low priority.



Regularly review all open action items

- What's the progress?
- Are the priorities still aligned?
Postmortems aren't static!
- Is a postmortem complete (all AIs resolved)?

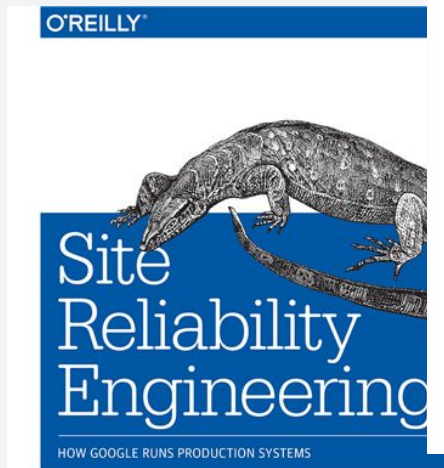


Executive Focus: Ensure that execs...

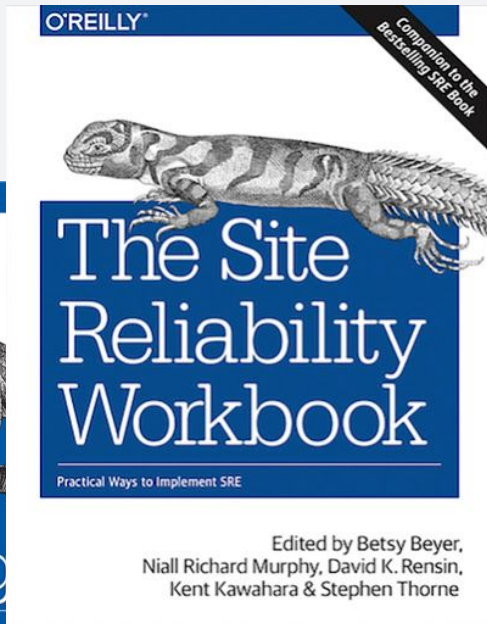
- have **high level** visibility in postmortem and action item progression
- **reward** and incentivize the resolution



Site Reliability Engineering



Edited by Betsy Beyer, Chris Jones,
Jennifer Petoff & Niall Murphy



Edited by Betsy Beyer,
Niall Richard Murphy, David K. Rensin,
Kent Kawahara & Stephen Thorne

<https://sre.google/books/> contain [detailed example, case study, and checklist](#)