Google

Postmortem Culture at Google

How do we learn from failures?

2023-09-26 / mhoe@google.com / SRE MUC Meetup

Site Reliability Engineering

Who has heard about Postmortems?

Who has written Postmortems in their team(s)?

Who am I?

Martin Hoefling

Technical Program Manager

- since 3 ½ y @Google
- SWE and manager before Google
- background in Physics (which I still keep in my ❤)

Topics and Teams:

- Resource Management, Planning and Allocation
  - interface between teams: Borg & UFO, SRE and Dev
- Postmortems @Google: Tooling, process, governance, awards, analysis …

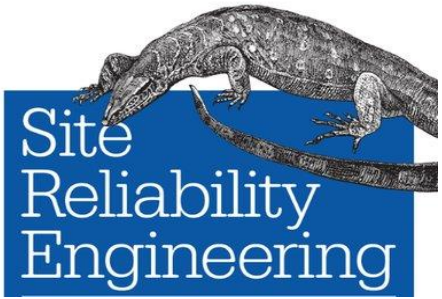
… ask me later if you would like to know more 😄

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

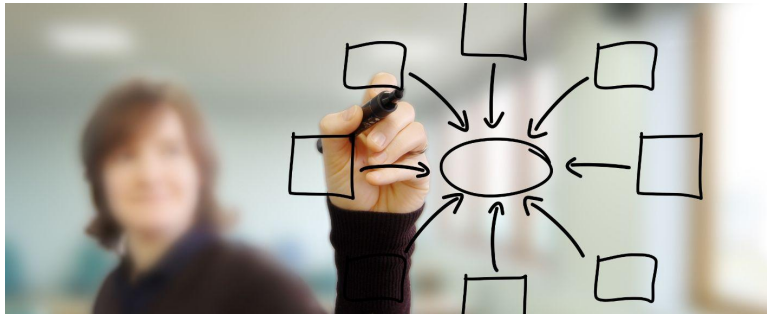
Postmortems / Incident reports are a common tool outside IT industry— e.g., in aviation or medical engineering.

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 pattern 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?

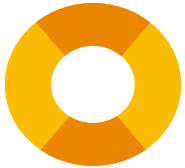
Impact criteria is met?

- 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!

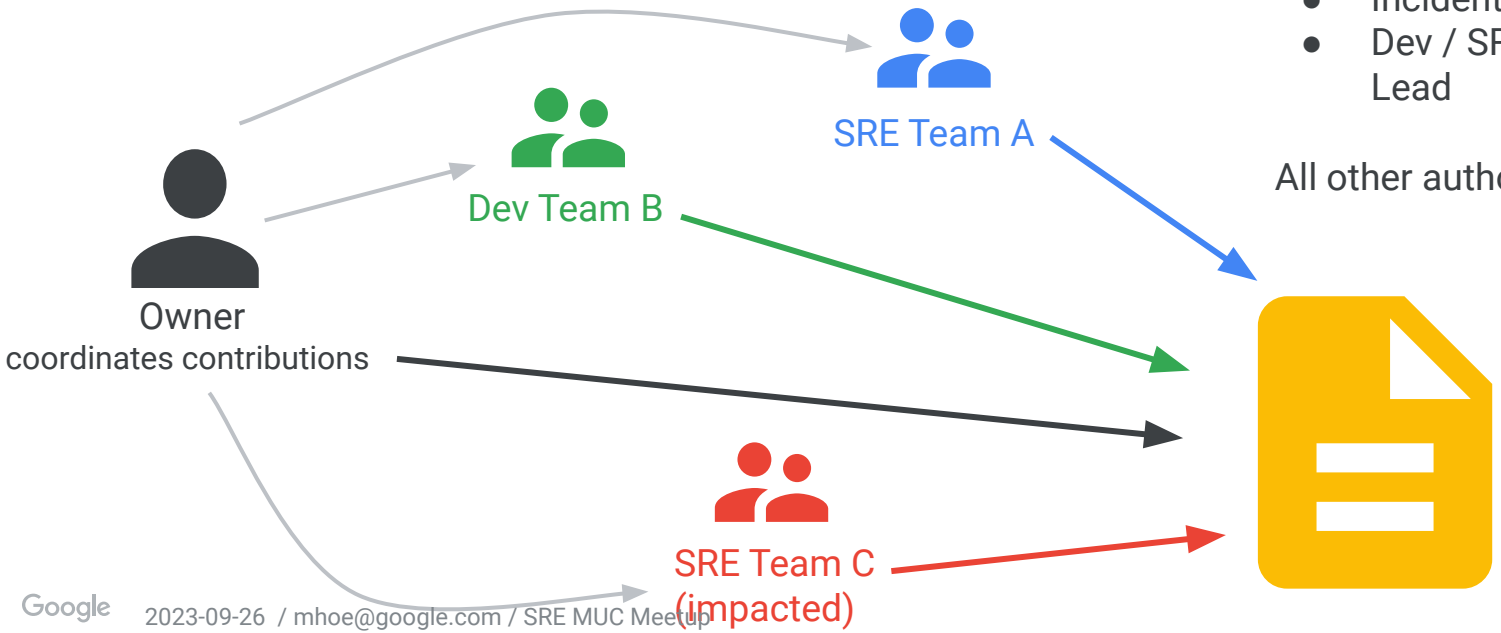
Interesting learning opportunity?

- 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.*

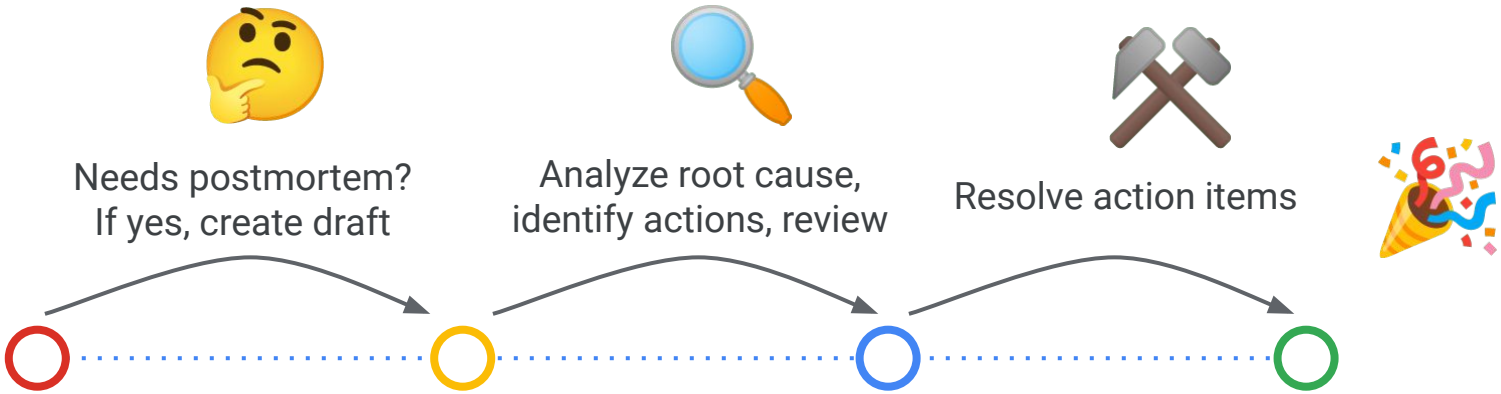
Lessons Learned

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

Supporting Material

- chat logs, metrics, timeline, (design) documentation, links to commits / code

Postmortem Process (Make it repeatable!)



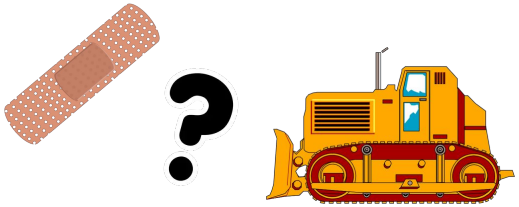
Action Items

- 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

Best practices for a balanced action item plan



Band aid fix vs. solving root cause of the problem

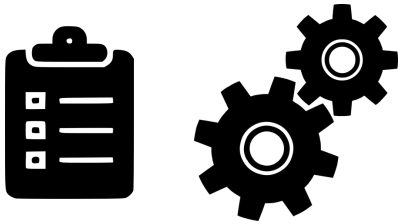
Fixing the root cause is often expensive:

➔ **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

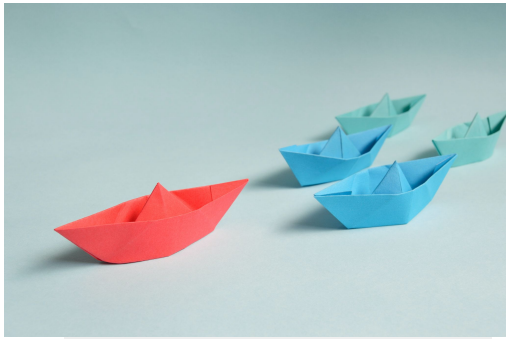
- What's the progress?
- Are the priorities still aligned? Postmortems aren't static!
- Is a postmortem complete (all AIs resolved)?

Executive Focus: Ensure that execs...

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

How do you get started?

What do you need to get started?



Get **leadership buy-in** for

- effort of writing postmortems
- *blameless* postmortems
- prioritizing action items

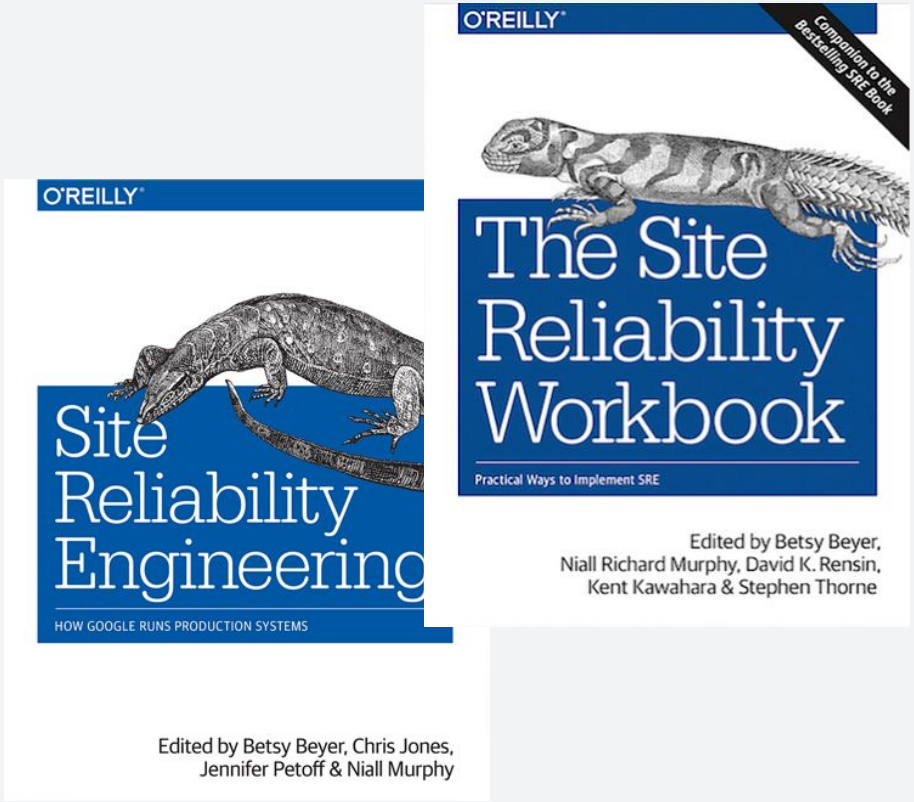
Define (and iterate) on **criteria** for writing postmortems

- build up some *muscle memory* on low impact outages

Have a simple **process** with clear ownership and responsibilities.

Start simple— perfection can wait!

Site Reliability Engineering



https://sre.google/books/ contain detailed example, case study, and checklist

Questions?