Lowering the Stakes of Failure with Pre-mortems and Post-mortems

Liz Sander

Senior Data Scientist, Civis Analytics

GitHub: elsander

@sander_liz

Agenda

- Introduction to Failure
- Blameless Post-mortems
- Pre-mortems

Failure is scary.

What does failure look like?

It depends on your job!

- System downtime
- Security vulnerability
- Shipping a critical bug
- Model is very wrong or unfair
- Consulting engagement with negative ROI
- Missing a critical deadline

Why is failure so scary?

Failure isn’t (just) about you.

- Mistakes happen within a context!
- Team
  - Time pressures
  - Incentives
  - Norms
  - Training
  - Expertise

Failure isn’t (just) about you.

- Mistakes happen within a context!
- Systems
  - Testing (automated or manual)
  - Documentation
  - Time/issue tracking
  - Code/methods review

Think as teams, not just individuals.

- Individuals will make mistakes no matter what.
- We need to work as teams to establish systems to catch and address failures
- Failure is an opportunity to learn and improve these systems
- This is where blameless post-mortems come in!

Failures will happen.

Addressing them as a team lowers the emotional stakes and lets us learn from our mistakes.

What's a blameless post-mortem?

- Process for **documenting** incidents, identifying **root cause(s)** of the incident, and determining **action items** to prevent/mitigate impact of future incidents
- "Incident" traditionally means system downtime
- Post-mortems aren't just for software reliability engineers!
  - It's a really effective way to learn from failure
- Core process: meeting to discuss the incident
- Core deliverable: post-mortem document and action items

Why a “blameless” post-mortem?

- Encourage people to report incidents and talk about them!
- Focus on understanding and improving, rather than assigning blame
- *“Accountability, not responsibility”*

OK, but what if one person is directly responsible?

This really isn't true most of the time.

"A blamelessly written postmortem assumes that everyone involved in an incident had good intentions and did the right thing with the information they had."

- Site Reliability Engineering: How Google Runs Production Systems

It's great to reflect on how you individually can improve. But that's a personal development issue, and **not the point of a post-mortem**.

If someone has performance issues, their manager should address it, but **not as part of a post-mortem**.

My First Post-mortem

v2.0.2
528174b
Verified

2.0.2 - 2017-12-05

elsander released this on Dec 5, 2017 · 2 commits to v2.0.2 since this release

Fixed

- Fixed bug preventing parallel training/validation with custom dependencies ([PARO-533], #414)
- Use StandardScaler instead of Normalizer with stacking classifier meta-estimator ([PARO-542], #418)

Changed

- Install civis 1.7.1 in the Dockerfile ([PARO-534], #417)

Bug hidden in Docker image :(

v2.0.3
0e282c3
Verified

2.0.3 - 2017-12-05

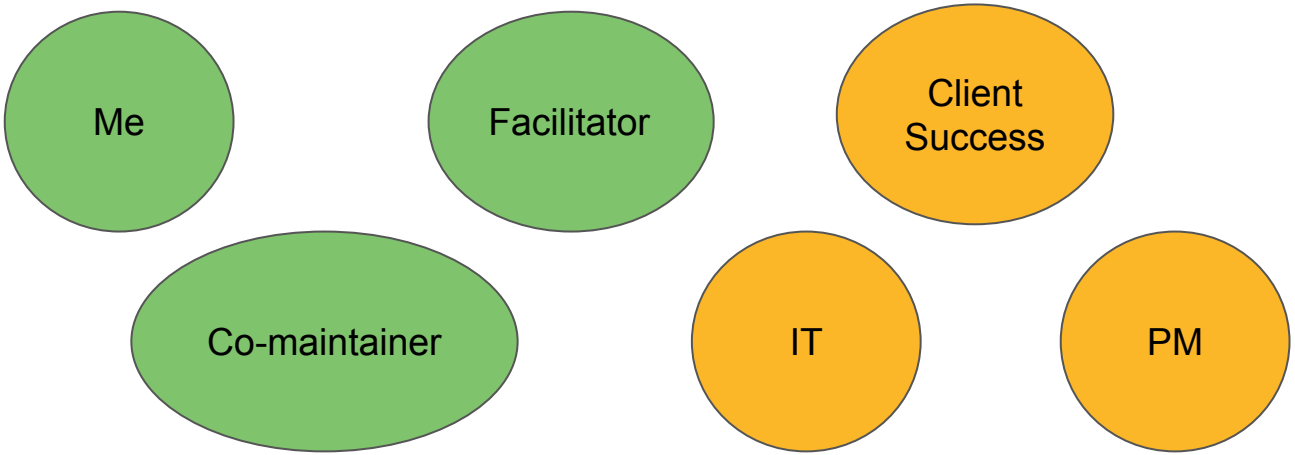
elsander released this on Dec 5, 2017

Fixed

- Reverted installation of civis 1.7.1, fixing a bug where files were not gzipped before being read into CivisML (#421)

Before the post-mortem: who and what to bring?

- Invite people who were directly involved
- For major incidents, affected stakeholders



Before the post-mortem

Fill in:

- **Incident period** (including initial trigger and resolution)
- **Status** (usually resolved, but mention ongoing issues if any)
- **Summary**
- **Impact** from the perspective of users
- **Trigger** (specific event(s) that caused the incident)
- **Detection**
- **Resolution** (what actions were taken, including unsuccessful ones)
- **Action items** (items to be addressed)

During the post-mortem

- Facilitator (probably not the person most directly involved)
- Read through timeline

Time	Event
12/5/17, 11AM	Tagged 2.0.2 release, Liz discovered bug
12/5/17, 11-1	Liz worked with IT to try and revert the docker images
12/5/17, 1PM	Liz brought co-maintainer in to start debugging
12/5/17, 6PM	Liz and co-maintainer tagged 2.0.3 release, fixing the bug

During the post-mortem

- Agree on trigger, impact, and root causes
  - **Trigger:** what is the immediate cause?
    - Latent bug in release appeared only in production environment
  - **Impact:** what was affected
    - ~1 day of CivisML downtime for internal users; client-facing data scientists were affected, but no deadlines were missed
  - **Root Causes:** What are the underlying systems that resulted in the problem?
    - We didn't have a Docker image for our release branch

During the post-mortem

- **What went well?**
  - What parts of our process should we keep/replicate elsewhere?
    - We caught and reported the issue immediately!
- **What went badly?**
  - Identify areas that need attention
    - No established process for testing on release containers
- **Where did we get lucky?**
  - Identify areas that didn’t break this time, but need attention to avoid future problems
    - Internal release only

During the post-mortem

- **Agree on action items and owners**
  - Updates to release checklist
    - Instructions for running tests in an environment that exactly matches prod
    - Never make a release without two maintainers available
- **Lessons Learned**
  - Test in the prod environment before release
  - Triage a critically buggy release by cutting a new version that reverts to the latest working version

What happens next?

- Follow up on action items
- Make the document available to the company
  - Keep postmortems together for future reference/learnings
  - Today’s “action items” may be tomorrow’s “what went well”s!

Post-mortems are a great way to iteratively improve and learn from past failure.

What if you’re starting a new project and want to avoid pitfalls in the first place?

Can we post-mortem a project *before* it fails?

Enter the pre-mortem!

Pre-mortem Structure

Our project has failed.

What happened?

My First Pre-mortem

- Building a Flask app for building and exploring “audiences” within customer base
- Big project with lots of moving parts
  - Cross-functional team of data scientists and engineers
  - Data & models
  - Different clients want different data, models, visualizations
- Big project + uncertainty + hard deadlines = anxiety
- Henry: instead of being anxious, let’s talk about it!
  - Stakeholders across departments: engineers, data scientists, product, sales, client success

Pre-mortem Structure

Our project has failed.

What happened?

Pre-mortem Structure: Brainstorm

Our project has failed.

What happened?

Security breach

ETL problems

Models are bad

It's hard to deploy

It's too slow

No one wants to buy it

It doesn't actually solve the user problem

It takes way too long to build

Users don't understand how to use the tool

Pre-mortem Structure: Organize

Our project has failed.

What happened?

Risk Category
Performance
Security
Timeline
Feature Gap
Non-Use

Pre-mortem Structure: Estimate importance & Discuss

Our project has failed.

What happened?

Risk Category	Probability	Impact	P*I
Performance	2	1.5	3
Security	1	3	3
Timeline	2.5	1.5	3.75
Feature Gap	2.5	2.5	6.25
Non-Use	1.5	3	4.5

Pre-mortem Structure: Estimate importance & Discuss

Our project has failed.

What happened?

What could we have done to avoid or mitigate the failure?

Risk Category	Probability	Impact	P*I
Performance	2	1.5	3
Security	1	3	3
Timeline	2.5	1.5	3.75
Feature Gap	2.5	2.5	6.25
Non-Use	1.5	3	4.5

Pre-mortem Structure: After the meeting

- Assign someone to send out notes
- Check in on risks and action items regularly

Why bother?

- Team members (especially non-managers) can be reluctant to bring up concerns
- Turns those concerns into a valuable asset
- Reveals domain-specific issues to the whole team
  - Important to bring in both technical and non-technical stakeholders
- Reflect on the project and processes before something fails
- Helps get everyone on board

What did we learn?

- Ongoing user research is important to make sure we’re building the right thing
- Coordinate directly with client team to make sure our data was formatted correctly
- Next time, check in on pre-mortem doc as part of sprint retrospective

Closing Thoughts

Conclusion

We can only learn from failure by bringing it into the open.

But to do that, we need to lower the emotional stakes, both of failing and talking about failure.

Pre-mortems and post-mortems are tools to do this, both before a project and after an incident.

The most important thing is to focus on systems and processes, rather than blaming individuals.

Resources

Site Reliability Engineering: How Google Runs Production Systems (especially c. 15 on Postmortem Culture)

- https://landing.google.com/sre/sre-book/chapters/postmortem-culture/

Pagerduty's Post-mortem process (lots of links to example post-mortems)

- https://response.pagerduty.com/after/post_mortem_process/

"The Pre-Mortem: A Simple Technique to Save Any Project from Failure"

- https://www.riskology.co/pre-mortem-technique/

Atlassian "Team Playbook" on pre-mortems

- https://www.atlassian.com/team-playbook/plays/pre-mortem

Thank you!

What if I don’t have a team?

- You can still do pre-mortems and post-mortems
- Do them on your own, bring in other stakeholders for high-priority issues
- Increased number/severity of incidents is a risk of single person teams! It’s a tough situation to be in

How do I bring these to my workplace?

- Talk to your team!
  - A department/team meeting is a good place
- Buy-in from leads is really important
- These strategies are fundamentally about evaluating failure points in systems, not maintaining server uptime