Getting Started with Incident Retrospectives

Moshe Zadka – https://cobordism.com

Acknowledgement of Country

Belmont (in San Francisco Bay Area Peninsula)

Ancestral homeland of the Ramaytush Ohlone people

I live in Belmont, in the San Francisco Bay Area Peninsula. I wish to acknowledge it as the ancestral homeland of the Ramaytush Ohlone people.

0.1 The Basics

Basic Terms Defined

Getting on the same page

Before talking about how to do incident retrospectives, we need to make sure we are all on the same page. Let's start by definining some terms.

0.1.1 Incident

Incident

Something happened that was not meant to happen

and it would be bad "enough" if it happened again

An incident is when something happens that should not have happened. To be an incident it should be the case that it would be *bad* if it happened again.

0.1.2 Incident Retrospective

Incident Retrospective

Analysis

Joint review

Agreement on next steps

An incident retrospective is an analysis of the incident. The analysis should be *consensual*: the team must agree the analysis is correct. The team must also agree on what the next steps are.

0.1.3 Goals of Incident Retrospective

Goals of Incident Retrospective

Understanding

Doing better

An incident retrospective is a lot of effort. Someone has to prepare a draft write-up, and the team needs to comment on it.

For the retrospective to be worthwhile, it has to give commensurate benefit. The benefits of a retrospective include *understanding* the system that failed better, and how it can fail.

The most important benefit is that retrospectives are a guide to *doing better*. What is doing better?

One way of doing better is to avoid such incidents altogether. This is not the only way!

Better incidents include faster remediation, less impact on end users, less effort for the team to remediate, and other aspects. When discussing how to do better, it is important to understand in what sense would it be better.

0.1.4 Non-Goals of Incident Retrospective

Non-Goals of Incident Retrospective

Deciding who's responsible

Punishing

It is equally important to note what are the non-goals. Incident retrospectives are a process of improvement, not a process of discipline.

They are not used for deciding which person is responsible. They are not used for deciding on a punishment.

0.1.5 Timeline

Timeline

Sequence of events

Minute/second resolution

Consistent time zone

Increasing

Thorough

Part of a retrospective is a timeline. Timelines, in general, are in "human resolution": minutes, or at best seconds.

A timeline should be in a consistent time zone even, or especially, in the face of a globally distributed team. It should be in the order of time – the stamps should always increase. A timeline should also be *thorough*: include all things which were involved in the incident.

0.1.6 Action Items

Action Items

Concrete

Actionable

Part of a retrospective are action items, or reocmmendations. Such recommendations should be *concrete* and *actionable*.

Being concrete, without being actionable, does not make something a recommendation. Being actionable, without being concrete, makes it impossible to know if the recommendation has been implemented.

0.2 Process

Incident Management Process

Step by step

Incident retrospective is a *process*. It is not only about the end-goal: the journey is important as well.

0.2.1 Incident Management Notes

Incident Management Notes

Rough

Immediate

While analyzing and remediating an incident, it is important to keep notes. This is should not delay the actual remediation process.

One way of doing it is to give the team a heads-up on any actions needed, or analysis results, on a chat channel. This is important, in any case, for allowing the team to help and understand what is being done.

0.2.2 Assigning Responsibility

Assigning Responsibility

Prepare the retrospective

Not: "At fault"

Not: "Handled the incident"

Not: "Knows best"

There should be one person assigned to the retrospective. This assignment should be treated as "who best to write the retrospective", and not implied to be a "punishment".

This means it is not based on who "caused" the incident, or who was assigned to remediate it. It is also not necessarily the person who knows the system best.

The assignment should be based on a combination of ability to do it in terms of availability and skills as well as who would learn from it the most. It is important that everyone in the team will have experience in writing retrospectives.

0.2.3 Research

Research

Prepare timeline(with references)

Understand the system

Writing a retrospective requires doing research. The research includes both gathering information on exactly what happened as well as how the system functioned, and malfunctioned.

0.2.4 Write-up

Write-up

...More details later

The details of the write-up are worthy of their own section. For now, we will skip them.

0.2.5 Review

Review

Team: On-call rotation

Offer corrections/amendments

After the draft write-up is ready, is should be put up for review by the team. The definition of the "team" here includes, at least, anyone in an on-call rotation for the system.

It can also include adjacent teams as well as people who contribute code but are not on the rotation.

0.2.6 Follow-up

Follow-up

Recommendation to implementation

The final piece of the retrospective is to make specific recommendations. The incident is not properly "done" until the recommendations have been implemented.

0.3 System Safety and Control Theory (3)

System Safety and Control Theory

Applications to writing incident retrospectives

Before talking about incident retrospective write-ups, it is important to understand the theory of systems failure. This theory guides as to how to analyze the system, what belongs in the timeline, and how to move from analysis to recommendation.

0.3.1 System

System

Parts we control

The "system" is the part under the team's control. The definition of the "system" depends on the definition of the "team".

0.3.2 Environment

Environment

Parts we don't control...

...that effect the system

The "environment" is anything that interacts with the system that is not under the control of the team.

This includes infrastructure the system depends on, as well as any systems that depend on it. The environment is also any ambient factors: for example, other systems using the same infrastructure.

0.3.3 Control

Control

System responding to environment

A "control" is a part of the system that takes input from the environment and changes the behavior of the system. In general, controls are designed to keep a system functioning inside given parameters.

0.3.4 Safety Control

Safety Control

Part of system

designed to avoid failure

A *safety control* is a control that is designed to prevent, or mitigate, a system failure. It responds to a potential issue and changes the system to make the failure less likely.

0.3.5 Safety Control Failure

Safety Control Failure

Safety control...

...environment...

...failure

A safety control failure happens when a safety control encounters a problem it was designed to mitigte and does not mitigate it appropriately.

0.3.6 Further Resources

Further Resources

Professor Nancy Levenson

CAST Handbook

This was a speedrun of the concepts involved in CAST, Causal Analysis based on Systems Theory. Professor Nancy Levenson of MIT has published the CAST Handbook, based on her research.

The book is available for free on the internet. Reading it is highly useful.

0.4 Anatomy of a Write-up: High-level (4)

Anatomy of a Write-up

Table of Contents

Now that the basics of systems theory has been established, it is time to talk on how to apply it to writing incident retrospectives. The first step is to design the "table of contents" for the write-up.

0.4.1 Summary

Summary

One paragraph

Appears first....

...written last

Though the summary appears first, it is written last. Nothing should be in the summary that is not a result of the rest of the write-up.

0.4.2 Timeline

Timeline

Bullet list

Time in timezone

What happened

Optional: Link to references

The next section is the timeline. It consists of a bullet list of timestamps and a short description of what happened.

The timeline, ideally, includes links to references and context for what happened. Alarms, pull requests, and more are all useful.

0.4.3 What Went Well

What Went Well

A bad time...

...great as a morale boost

It is always useful to congratulate the team on what went well. While not strictly useful to doing better, it works well as a morale boost.

The rest of the incident retrospective is about problems. It is often preceived as a criticism.

Putting this front and center allows the reader to have a little moment of relief.

0.4.4 What Could Have Been Done Better

What Could Have Been Done Better

Appears before recommendations...

...written after

This is not the recommendations yet. This is not necessarily actionable – some of the "what could have been done better" things could only be clear in retrospect.

This is more of a "fantasy": how would things go in the best of all scenarios. This is useful for inspiration later on.

0.4.5 Analysis

Analysis

Based on CAST

Details later

The analysis is based on CAST: Causal Analysis based on Systems Theory. The details of how to write an analysis using CAST will be clarified later. For now, let's skip to the next part.

0.4.6 Recommendations

Recommendations

Concrete...

...actionable

Open tickets (issues/bugs/...)

As explained before, the recommendations must be concrete and actionable. If you take nothing else from this talk, this is the most important part.

The goal of the retrospective is to make things better. The way to make things better is to have specific things that can be done.

0.5 Anatomy of a Write-up: Details (5)

Anatomy of a Write Up: Analysis

...Finally, details!

After all the build-up, let's dive in: how does CAST analysis works? Here is the step-by-step guide.

0.5.1 Safety Controls

Safety Controls

List safety controls

Short explanation

Hopefully, the system had some safety controls. Though they failed to stop the incident, they are still important.

List each one, and note what it does.

0.5.2 Safety Control Failures

Safety Controls Failures

Every safety control failed

This is what "incident" means

The relevant safety controls all failed. It is important to clearly analyse the failure.

This often requires the most research. You will need to understand exactly how the control was supposed to work.

0.5.3 Systemic Problems

Systemic Problems

Beyond an individual safety control

Systemic problems leading to the incident

Are there any systemic problems?

For example, what is the process of deciding on safety controls? How are they tested?

0.5.4 Missing Safety Controls

Missing Safety Controls

Aspects that lack any safety controls

Are there any aspects that are completely missing safety contorls? Add a list of safety controls that could have stopped or mitigated the incident.

0.5.5 Why not "root causes"?

Why not "root causes"?

What is the "root"?

What is special about the "root"?

"Root" often in environment

Often, the analysis is called "root cause analysis" or "contributing causes analysis". Root cause analysis, properly speaking, is one which identifies the one "cause" leading to the issues, and then *fixing it*.

This is a fundamentally broken way of working. The problem is not the "root cause": the problem is that the system did not have appropriate safety controls.

The "cause", or "contributing causes", might be outside of our locus on control. Increased customer load or underlying infrastructure failure can bring a system down.

We do not have control over the customers or the infrastructure. We can make sure that we shed load appropriately or fail over to alternative infrastructure.

0.5.6 Actionable Recommendations

Actionable Recommendations

Recommendation must be related to issues identified

Additional safety control

Improvement to existing safety control

The recommendations should be related to the analysis. Each recommendation should be tracable to one of the specific problems written in the analysis.

Specifically, avoid recommendations which would *not* have helped in those particular circumstances.

What Are Recommendations?

Software changes

Configuration changes

Process changes

Clear definition "done"

Recommendations can include anything. Often, they'll include changes to software or configuration.

Recommendations can, however, also include "process" changes. For example, are there are things that someone needs to read before joining an on-call rotation? Do we need to develop more training materials?

A recommendation does need to communicate what would “done” look like.

0.6 Incident Retrospective Review (6)

Incident Retrospective Review

”Post-Mortem Meeting”?

De-emphasize the meeting!

When we talk about incident retrospectives, or “post-mortems”, the usual word that follows is “meeting”. This should, in general, be the least important part of the process.

This is not when the review starts, or ends.

0.6.1 Async Feedback

Async Feedback

Comments from the team

Give enough time

The review starts when there is a draft of the written analysis ready. At that point, ask for feedback from the team.

Allow enough time, and have some method, for the team to collaborate on feedback. For example, this can be done using pull request or merge request comments, comments in a shared document writing system, or anything else that works.

This allows people enough time to properly consider the analysis, read it, and think about ways it could be done better. Comments need to be clear about what kind of change they are asking for, and why.

0.6.2 Addressing Async Feedback

Addressing Async Feedback

Change

Clarify

Disagree

For each comment, you might want to make the change it suggests, ask a question if it is not clear what change to make, or disagree that this change should be done better. All of these options are relevant and reasonable.

For some comments, it will turn out that the clarification is harder to achieve in an asynchronous way. For others, there might be fundamental disagreement.

This is the point where it is reasonable to schedule a meeting to discuss those.

0.6.3 Incident Retrospective Meeting

Incident Retrospective Meeting

Limit back-and-forth

Invite on an as-needed basis

In general, avoid more than one back-and-forth on a specific comment. Any comment where the thread is at least three bits long should be resolved during the meeting.

When setting up the meeting, only the original author and the people involved in the discussion are mandatory participants. Others in the team can be "optional".

0.6.4 Roles in Incident Retrospecive Meeting

Roles in Incident Retrospective Meeting

Presenter

Moderator

Note-taker

Meetings are expensive. They spend people's time, they need to be set up in advance, and there are limited slots.

Make the most out of the retrospective meeting by assigning roles in advance. The presenter is usually the person who wrote the report.

The moderator is most often the line manager involved, but can be anyone. Another person needs to take notes, or the meeting will have been a waste of time.

While the role is named "presenter", the goal is not to read the report aloud. The presenter presents each point that needs agreement, and opens the floor for questions.

The moderator makes sure everyone's voice is heard, and that a clear decision on what to do is reached. The note taker summarizes points made, and the final decision.

0.6.5 Follow-up on Incident Retrospective Meeting

Follow-up on Incident Retrospective Meeting

Attach notes

Make agreed-on changes

Announce

The notes are best attached to the retrospective. They give context for why things were analysed, or decided, a certain way.

Then whatever changes the decisions pointed to should be made. Finally, the draft is ready for "final approval". Announce to the team that this is so.

0.6.6 Finalizing Write-up

Finalize Write-Up

Formally note "write-up is complete"

List of "done" write-ups

There should be a clear point where the retrospective is "done". The final approval is "mechanical": it checks that the decisions that have been reached (in the notes) have been applied.

There should be a list of retrospective write-ups that are "done", and the new retrospective added to the list.

0.7 Implementing Recommendations (7)

Implementing Recommendations

The main point!

The final approval of the retrospective is still not the end of the incident! Assuming some recommendations have been made, and documented as tickets or issues in the tracking system used, these need to be implemented.

Again, this is the main point of the retrospective: it is only worthwhile if the recommendations are implemented.

0.7.1 Prioritizing

Prioritizing

Clearly decide

The recommendations, as tickets, need to be prioritized with all other tickets assigned to the team. This should be done using whatever process is already used for prioritization.

It should be *clearly noted* what is the relative priority, and when it can be expected to be tackled.

0.7.2 Tracking

Tracking

Labels/links

Searchable

Each recommendation should be linked to the retrospective it was made in. This can be done by labels or links, as long as it is searchable *both ways*: by retrospective (show all recommendations in the retrospective) or by issue (show the retrospective that made this recommendation).

This allows tracking the issues and seeing how the team is making progress on implementing those.

0.7.3 Commitments

Commitments

When?

Document

It is worthwhile to make specific commitments about when these things will be tackled. Otherwise, it is easy to push a recommendation to "next sprint".

0.7.4 Revisiting Recommendations

Revisiting Recommendations

Not written in stone

It is ok to decide a recommendation is not worthwhile. Maybe things have changed, maybe implementing it turns out to be harder than it was thought to be.

In either case, changing or removing the recommendation is a decision that can be made. Document it in the recommendation and in the retrospective analysis.

0.7.5 Common Recommendations

Common Recommendations

Some issues cause multiple incidents

It will often be the case that some recommendations come up often. This is the case with issues that are hard to solve and persistent source of problems.

In that case, properly model this in the ticket system. For example, you can use "tracking tickets" or multiple links, as the case may be.

0.7.6 Communicating Status of Recommendations

Communicating Status of Recommendations

Planned/in-progress/dropped/done

It should be easy to see if a recommendation is still planned, in progress, dropped, or done. Ideally, the state or labels on the ticket reflect that.

0.8 Onboarding (8)

Onboarding

From here to there

Doing retrospectives is a complicated process. If it is new to the team, do not expect to be an expert in it overnight.

The team needs time to learn, and so do the individuals in the team. Mistakes will be made.

0.8.1 Documenting the Process

Document the Process

Step-by-step

Template

Start by clearly documenting the process. Have a clear step-by-step manual.

This should include *when* to do a retrospective (who decides what's an incident? does every incident get one?) and who does it. It should also include the exact systems used.

What system is used to share the draft? To annotate it with comments? To open recommendations? What labels or directories need to be used?

The most important part is a template. The template should include all parts that need to be filled out.

If some fields are optional, note it. If there are writing guidelines, note them in the template.

0.8.2 See One, Do One, Teach One

See one, Do one, Teach one

Medical school system

Watch someone else doing it

Do it with someone else watching

Reverse roles in step 1

See one, do one, teach one is how doctors learn. The first step is to see someone doing it: have a "shadow" who follows the retrospective writer through the steps: research, write-up, and finalization.

Then, have a "reverse shadow": someone who watches the new person doing it and offers real-time feedback. Finally, have the student become the teacher the next time someone needs to be taught.

0.8.3 Allocating Time

Allocating time

A new job responsibility

Writing a retrospective is a new job responsibility. This means that it needs to be allocated for when planning.

This means both short-term planning, as replacing some of the work in the "sprint", and long term planning, as in taking some percentage of total team capacity.

0.8.4 Integrating Feedback

Integrating feedback

Professional and clear

Retrospectives, and incident process in general, tend to be stressful. Keep feedback professional and clear. Ask for what you want, without making allegations or accusations.

0.8.5 Clear Standards

Clear standards

What's "good enough"?

Have a clear standard: what is the level of retrospective we want? For example, what is the level of writing clarity you strive for?

0.8.6 Deadlines

Deadlines

When is it due?

Have a clear guideline on *when* the first draft should be done, when feedback is gathered, and when it is addressed.

0.9 Iterating (9)

Iterating

Getting better

Processes are not static. Maybe your template is missing some fields. Maybe the guidelines for how to use the ticket systems are unclear, or use it badly.

0.9.1 Updating the Process

Updating the Process

Process for update

There needs to be a clear process for updating the retrospective process. Who can suggest changes? Who needs to authorize them? Who needs to be consulted?

0.9.2 Communicating Changes

Communicating changes

Announce clearly

If the process changes, make a clear announcement. Note what changed, why, and any context needed.

0.9.3 Clearly Needed Changes

Clearly needed changes

Missing things

Unclear

Some changes are more clearly needed. For example, someone going through the process might notice that the part about setting up a meeting is completely missing details. Maybe the guidelines for how to phrase recommendations are unclear.

This is a good time to change the retrospective process, following the update process above.

0.9.4 Analyzing Retrospectives Over Time

Changes over time

Common problems

Make it easy to analyze

It is worthwhile to complemenet this with a more intentional iteration process. Take time to read through reviews, notes, and recommendations, and see if there are any issues that keep coming up. Maybe the process can be improved to prevent those?

For this to work, it needs to be easy to analyse the retrospectives. This is where the list comes in handy.

0.9.5 Standard vs. Team-led

Standard vs. Team-led

Can different teams have different processes?

What is the level of standardization you want? There is no one answer.

Some organizations tend towards more standards, while others tend to have teams which decide on their processes. Make it clear which teams use which process, and what kind of standards you do expect at different levels.

0.9.6 Intentional Learning

Intentional Learning

Research

Discuss

Avoid counting only on yourself and your team when improving the retrospective process. Research what other teams (internal and external) are doing. Discuss with people what you are doing.

Make an intentional effort to incorporate new information into the process.

0.10 Summary (10)

Summary

Highlights

This is a long talk because it is a complicated topic. Incident retrospectives are easy to get wrong!

That said, I want to highlight some of the things that are most important.

0.10.1 Back to basics: Blameless

Basics: Blameless

Avoid blame

"Humans are not the cause"

Blamelessness is an implicit part of CAST. Humans, and human nature, are part of the environment unless we know how to modify humans.

This means humans are never part of the cause. Make it painfully clear that this is so.

0.10.2 Consistency Matters

Consistency

Process

Consistency is important. Have people follow the process. If the process is bad, fix it, don't work around it!

0.10.3 Measure

Measure

Is it useful?

Retrospectives take a lot of time and effort. They better be worth it.

Are they? Check recommendations, see how they fared.

0.10.4 Iterate

Iterate

Remove what doesn't work

Add what's needed

The meta-process is important. Iterate and keep improving the retrospective process itself.

0.10.5 Keep the Goal in Mind

Goal

Better incidents!

There is one goal: "better incident". Not necessarily "fewer incidents", although this is one measure.

"Incidents that are easier to resolve", "incidents with less customer impact", and more are all within the purview of the retrospective process. Know what you are optimizing for!

0.10.6 Enjoy!

Do

Start doing retrospectives...

...and iterating on the process!

Whether you are doing retrospectives and wish you could do them better, or whether you haven't started, I hope you start on the path. Let me know how it works out for you!