Incident Retrospectives as Code

Moshe Zadka – https://cobordism.com

Acknowledgement of Country

Belmont (in San Francisco Bay Area Peninsula)
Ancestral homeland of the Ramaytush Ohlone people

Outlines

What are retrospectives?

Outlines

What are retrospectives?
What is code?

Outlines

What are retrospectives?
What is code?
How?

Outlines

What are retrospectives?
What is code?
How?

Incidents: An Introduction

Align on terms

Incident

Incident

Something bad...

Incident

Something bad...
...we didn’t want...

Incident

Something bad...
...we didn't want...
...and we would like to prevent.

Retrospective

(AKA "post-mortem")

Retrospective

(AKA "post-mortem")
Analysis...

Retrospective

(AKA "post-mortem")
Analysis...
...and an improvement plan.

Retrospective Review

Retrospective Review

Part of "signing off" on a retrospective

Retrospective Review

Part of "signing off" on a retrospective
Feedback

Retrospective Review

Part of "signing off" on a retrospective
Feedback
Feedback addressed

Retrospective Review

Part of "signing off" on a retrospective
Feedback
Feedback addressed
Sync/Async

Why Retrospectives?

Why Retrospectives?

"Make new mistakes"...

Why Retrospectives?

"Make new mistakes"...
...for sufficiently large values of "new".

Why Retrospective Reviews?

Why Retrospective Reviews?

Verify analysis...

Why Retrospective Reviews?

Verify analysis...
...verify recommendations...

Why Retrospective Reviews?

Verify analysis...
...verify recommendations...
...teach.

Code

What do I mean by ”code”?

Code: Format

Source code is:

Code: Format

Source code is:
Text

Code: Format

Source code is:
Text
In a computer language

Code: Source Control

Examples:

Code: Source Control

Examples:
Git, Mercurial, Fossil, ...

Code: Source Control

Examples:
Git, Mercurial, Fossil, ...
Keeps history

Code: Source Control

Examples:
Git, Mercurial, Fossil, ...
Keeps history
Directory organization

Code: Source Collaboration

Examples:

Code: Source Collaboration

Examples:
GitHub, GitLab, ReviewBoard

Code: Source Collaboration

Examples:
GitHub, GitLab, ReviewBoard
Feedback

Code: Source Collaboration

Examples:
GitHub, GitLab, ReviewBoard
Feedback
Changes

Code: Source Collaboration

Examples:
GitHub, GitLab, ReviewBoard
Feedback
Changes
Approval

Code: Source Collaboration

Basic unit:

Code: Source Collaboration

Basic unit:
"Draft Patch"

Code: Source Collaboration

Basic unit:
"Draft Patch"
AKA

Code: Source Collaboration

Basic unit:
"Draft Patch"
AKA
Pull request,

Code: Source Collaboration

Basic unit:
"Draft Patch"
AKA
Pull request,
Merge request,

Code: Source Collaboration

Basic unit:
"Draft Patch"
AKA
Pull request,
Merge request,
Review request

"As": An Analogy

Infrastructure as Code:

"As": An Analogy

Infrastructure as Code:
Infrastructure defined by code

"As": An Analogy

Infrastructure as Code:
Infrastructure defined by code
Approval is approval

"As": An Analogy

Infrastructure as Code:
Infrastructure defined by code
Approval is approval
Merge is "finalize"

How?

Easier than you think!

Source Control

Step 0: Source control

Format

Format

Your favorite ”lightweight markup” language

Format

Your favorite "lightweight markup" language
Default: Markdown

Format

Your favorite "lightweight markup" language
Default: Markdown
Other alternatives: ReStructured Text, Asciidoc

Format

Your favorite "lightweight markup" language
Default: Markdown
Other alternatives: ReStructured Text, Asciidoc
Choose and commit

Template

Template

Common ToC

Template

Common ToC
Clarify what belongs

Template

Common ToC
Clarify what belongs
Note any guidelines

Template

Common ToC
Clarify what belongs
Note any guidelines
Note mandatory/optional

Template

Common ToC
Clarify what belongs
Note any guidelines
Note mandatory/optional
NOT a replacement for process docs

Organization

Organization

Directory structure (flat/hierarchical)

Organization

Directory structure (flat/hierarchical)
Directory structure (images)

Organization

Directory structure (flat/hierarchical)
Directory structure (images)
Naming

Collaboration for Retrospectives

Collaboration for Retrospectives

Step 1 of "as code"

Collaboration for Retrospectives

Step 1 of "as code"
...because you don't push straight to main in your other repositories

Draft Patch

”Asking for feedback”

Draft Patch Feedback

Draft Patch Feedback

Line-by-line comments

Draft Patch Feedback

Line-by-line comments
Overall comments

Draft Patch Feedback

Line-by-line comments
Overall comments
Ask for changes

Addressing Draft Patch Feedback

Addressing Draft Patch Feedback

Push new commits

Addressing Draft Patch Feedback

Push new commits
Reply to comments

Getting Approval

Getting Approval

Who approves?

Getting Approval

Who approves?
"Discussion has resolved on everything"

Merging

Merging

Merge patch

Merging

Merge patch
Close ticket

Why?

Why?

Tooling

Why?

Tooling
Workflow

Why?

Tooling
Workflow
Access controls

Why: Extra Credit

Why: Extra Credit

CI

Why: Extra Credit

CI
Analysis

Start!

That's all there is!

Continuous Integration for Retrospectives

Continuous Integration for Retrospectives

...just like for your other code repositories, right?

CI for Retrospectives: Wait What?

CI for Retrospectives: Wait What?

Never send a human to do a machine's job

CI for Retrospectives: Lint

CI for Retrospectives: Lint

Automatic checking of guidelines

CI for Retrospectives: Rendering

CI for Retrospectives: Rendering

Easier to read

CI for Retrospectives: Rendering

Easier to read
Easier to search

Example Lint

```
def get_timestamps(input_text):
    ...

def check_order(timestamps):
    timestamps = iter(timestamps)
    last = next(timestamps)
    for current in timestamps:
        if current <= last:
            raise ValueError(
                "got decreasing",
                last,
                current,
        )
        last = current
```

Example Lint Failure

```
try:
    check_order(get_timestamps(input))
except Exception as exc:
    for arg in exc.args:
        print(arg)

got decreasing
2022-01-02 19:12:33
2022-01-02 19:12:31
```

Analyzing Retrospectives' Data

Analyzing Retrospectives' Data

Input, not decisions

Analyzing Retrospectives' Data

Input, not decisions
Easier

Retrospective Process

Retrospective Process

Theory: Incident, Research, Draft, Review, Approve, Implement

Retrospective Process

Theory: Incident, Research, Draft, Review, Approve, Implement
Real life: Infinite variations

Iterating Retrospective Process

Iterating Retrospective Process

Is our variation good?

Number of Recommendations

```
def find_section(input_text, name):
    ...

def get_recommendations(input_text):
    bullets = find_section(
        input_text,
        "Recommendations",
    )
    recommendations = bullets.findall("list_item")
    return len(recommendations)

get_recommendations(input_text)
```

```
2
```

Recommendation Done

Recommendation Done

Difficult to fit on slide

Recommendation Done

Difficult to fit on slide
Similar ideas + API to ticket system

Share Analysis

Share Analysis

One off: Distribute Notebook

Share Analysis

One off: Distribute Notebook
Repeated: Dashboards

Why not?

So what's wrong with other options?

Why not Wiki?

Why not Wiki?

Edit with usual editor!

Why not Wiki?

Edit with usual editor!
Local backup

Why not Shared Docs?

Why not Shared Docs?

E.g., GDoc, Dropbox Paper, MS Sharepoint...

Why not Shared Docs?

E.g., GDoc, Dropbox Paper, MS Sharepoint...
Non-proprietary format

Why not Shared Docs?

E.g., GDoc, Dropbox Paper, MS Sharepoint...
Non-proprietary format
Common tooling