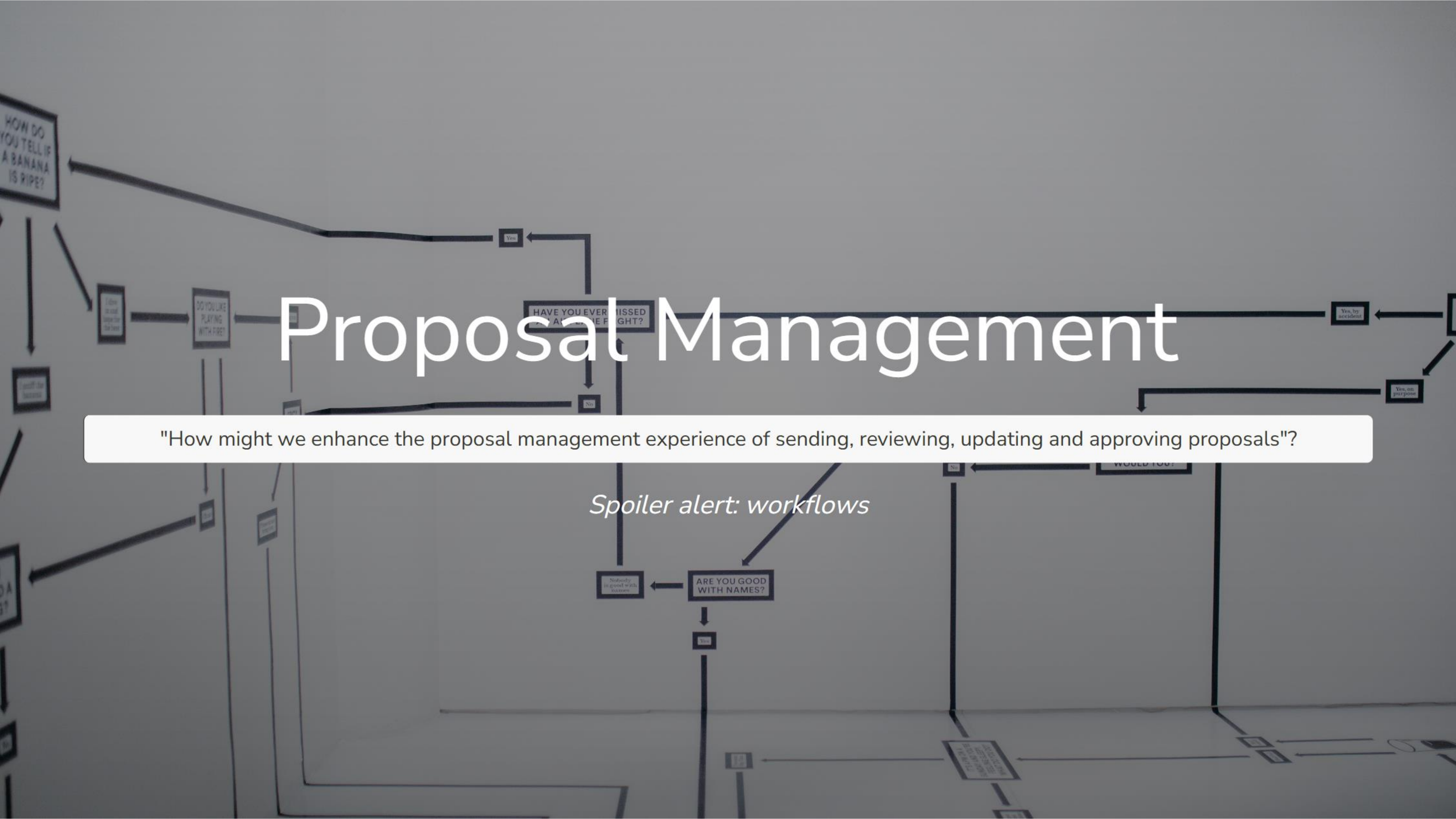

HOW DO YOU TELL IF A BANANA IS RIPE?
Yes
DO YOU LIKE PLAYING WITH FIRE?
HAVE YOU EVER MISSED AN AIRPLANE FLIGHT?
No
Yes, by accident
Yes, on purpose
ARE YOU GOOD WITH NAMES?
Nobody is good with names
Yes

Proposal Management

"How might we enhance the proposal management experience of sending, reviewing, updating and approving proposals"?

Spoiler alert: workflows

Background

My team is working on Proposal Management

Proposals

> A business proposal is sent from a supplier to a potential client for the purpose of winning a specific project. It is a written (paper or electronic) document and it can either be requested by the client or sent unsolicited.

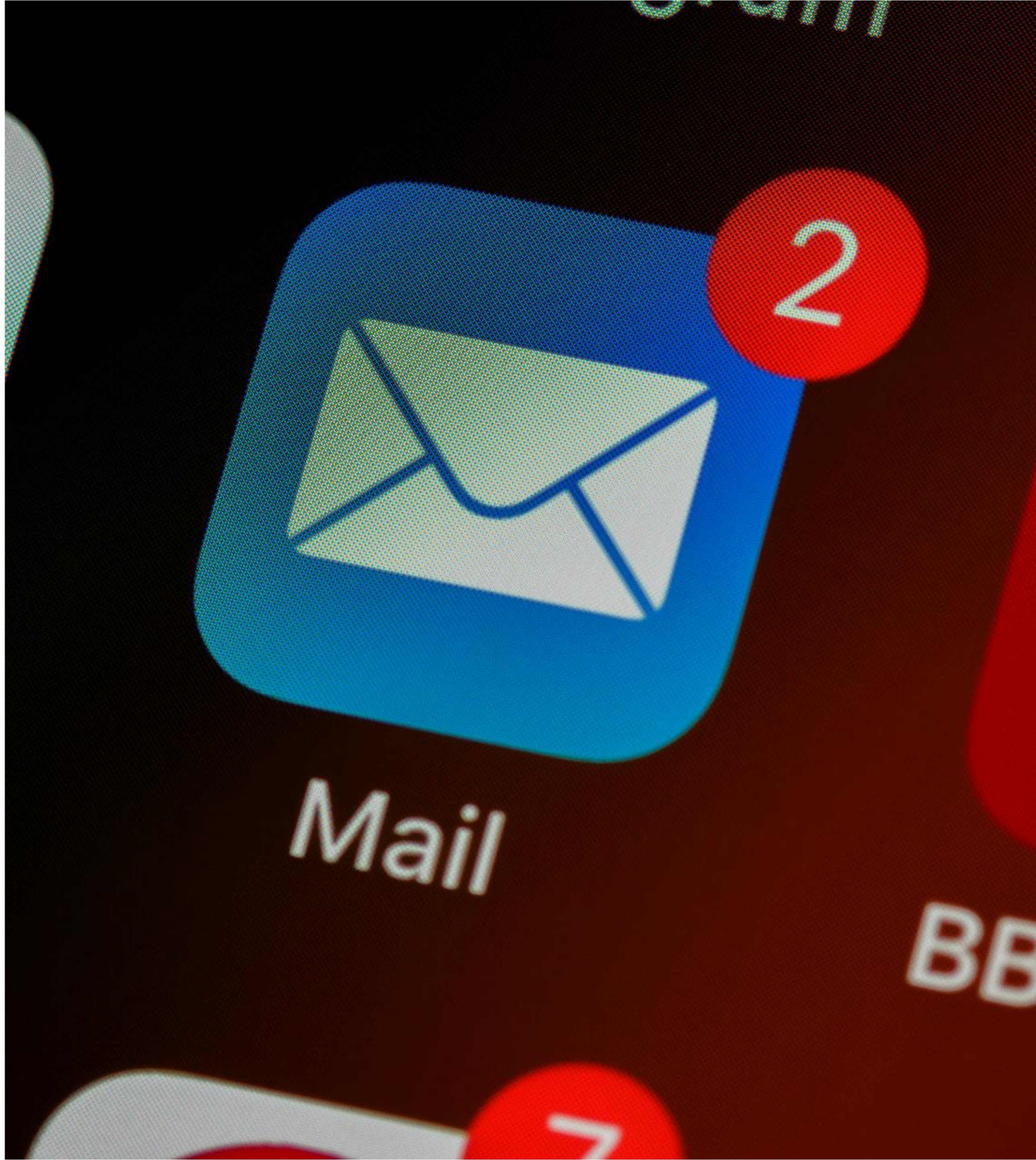
The keyword here being *document*.

Informing stakeholders of proposal changes

Stakeholders may wish to update a proposal, anything from terms to the cost of provided services or dates of deliverables.

Notifications

It’s important to be able to notify stakeholders when that proposal document is updated.



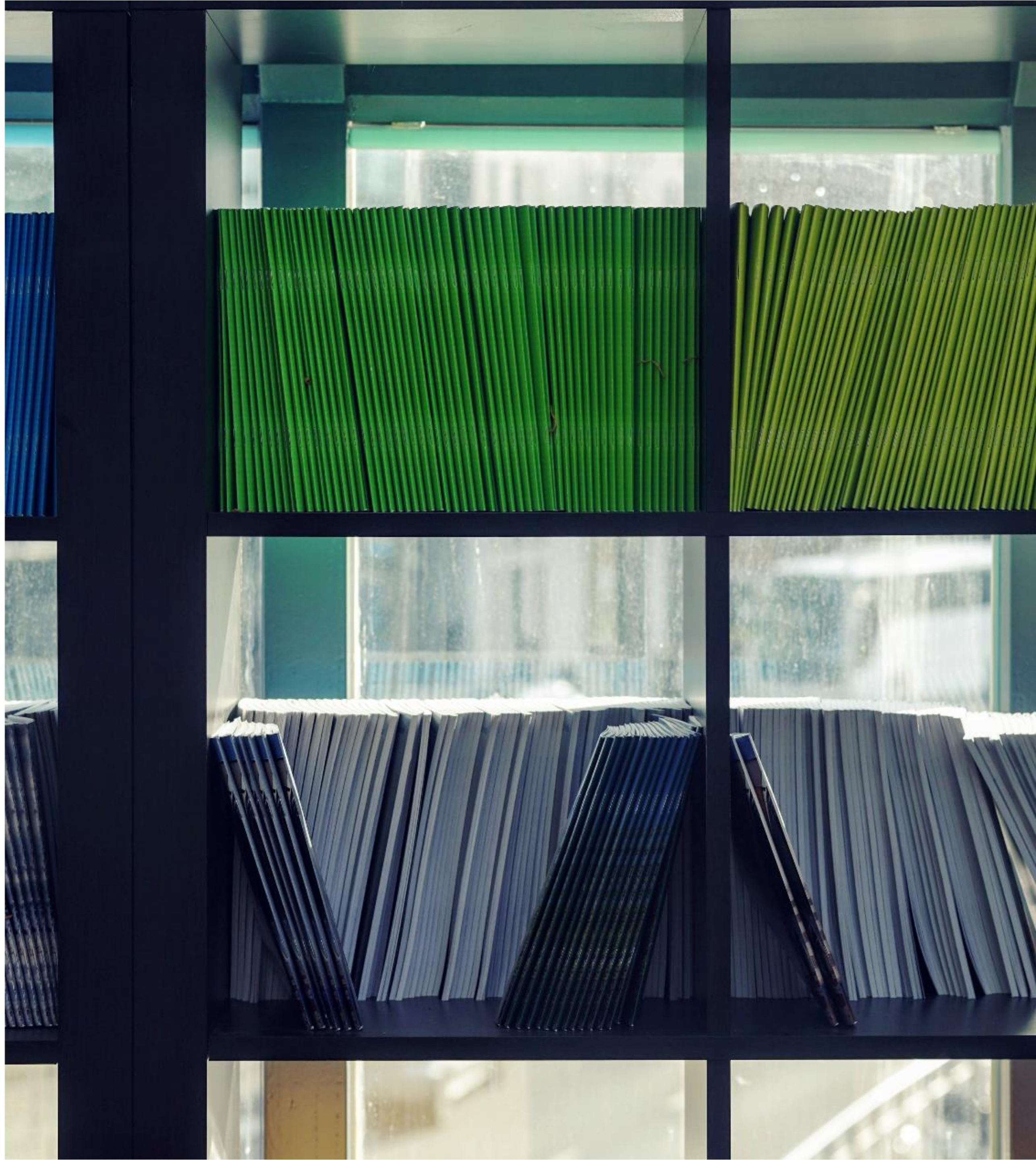
Ensuring that updates to a proposal document follow a defined procedure

(organisation requirements, legal requirements, etc.)

Any document generally goes through *phases* such as:

- Being drafted
- Being reviewed
- Submitted to client

In some circumstances a document should not be updated once it's been submitted to a client until they have reviewed it.



Notifications, enforcement of document requirements, preventing edits outside of key phases...

That could soon become complex and implementing these sorts of processes in software can often lead to "accidental complexity" and "technical debt".

A solution - workflows

> "the sequence of industrial, administrative, or other processes through which a piece of work passes from initiation to completion."

Let's break it down a little, we need:

- To send out notifications when a proposal document is updated
  - Do we want every type of edit to sent a notification or just key phases?
- Enforce a particular set of document guidelines we might be following
- Prevent invalid edits and updates occurring outside of specific phases in alignment with the guidelines.

How to implement a workflow?

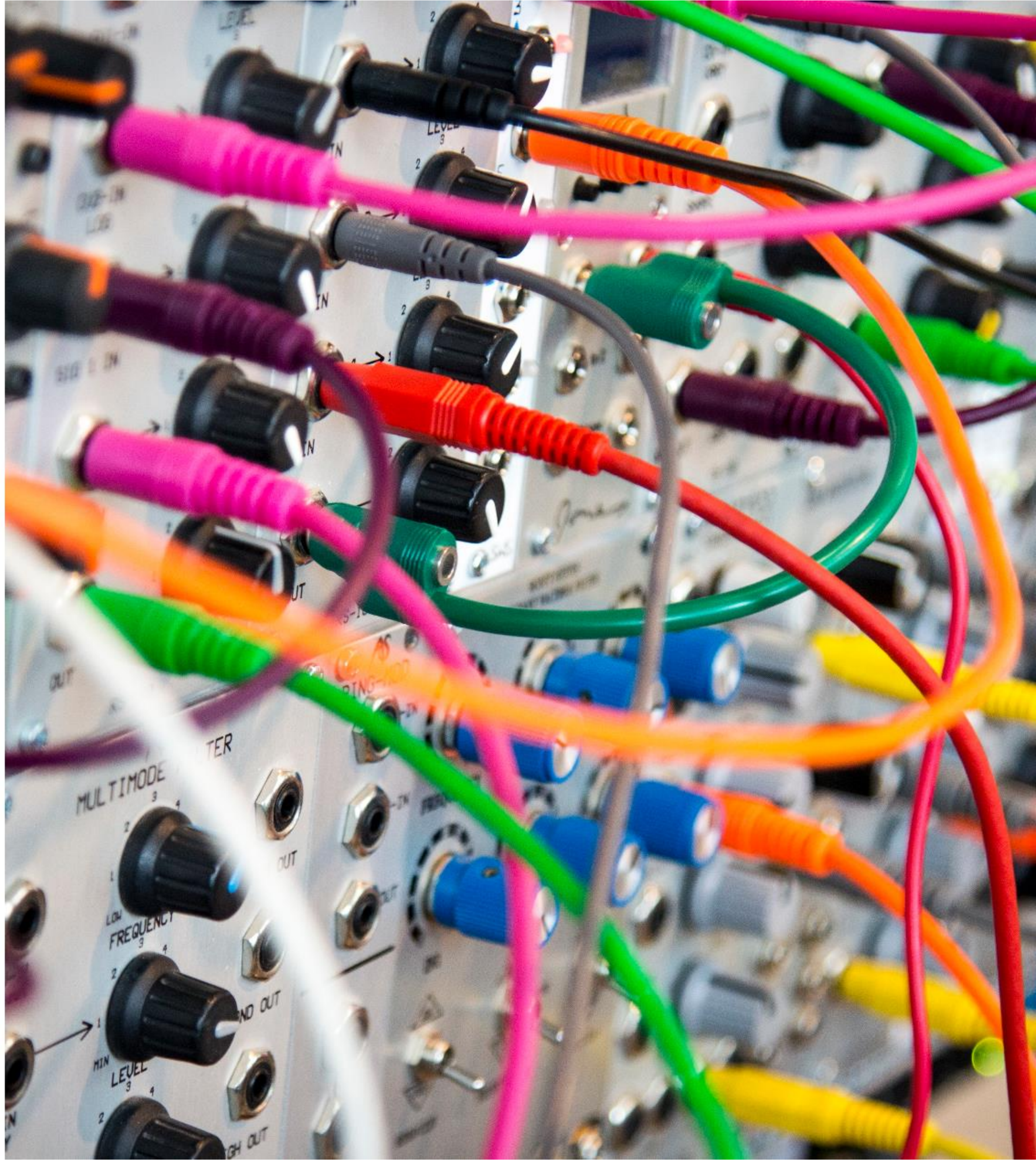
Answer: Finite State Machines

> A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition

FSM’s are a "universal" concept that applies across multiple disciplines.

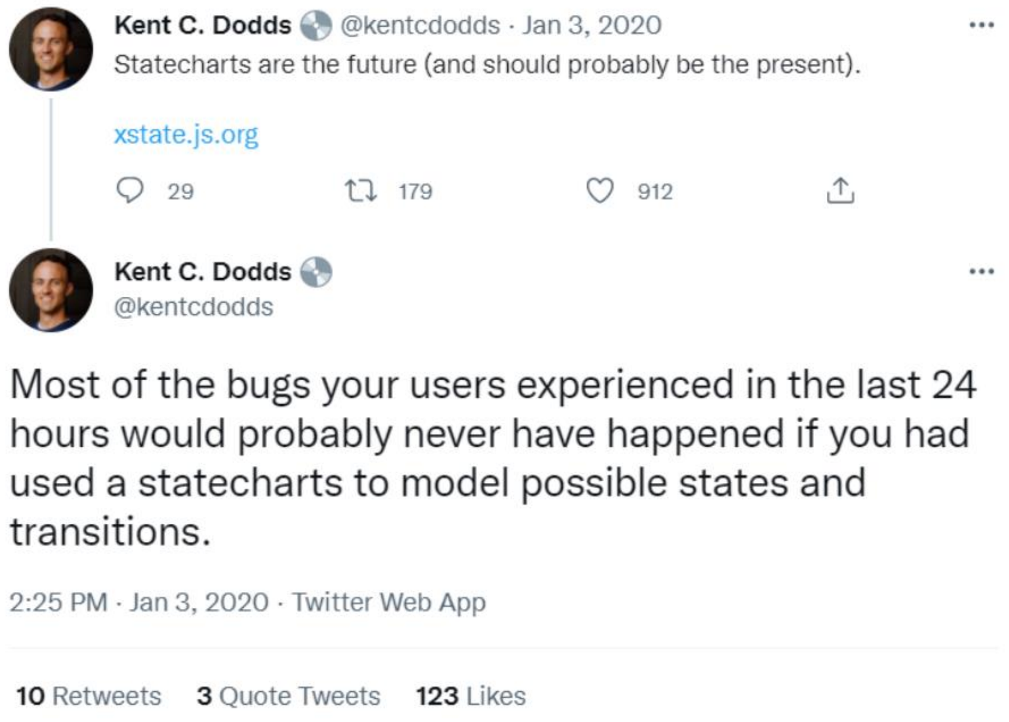
That sounds pretty scary. 😱

Let’s explain it in simple terms...



Simply explained… designing robust software

One of the many huge benefits of FSM's



Kent C. Dodds @kentcdodds · Jan 3, 2020

Statecharts are the future (and should probably be the present).

xstate.js.org

29 179 912

Kent C. Dodds
@kentcdodds

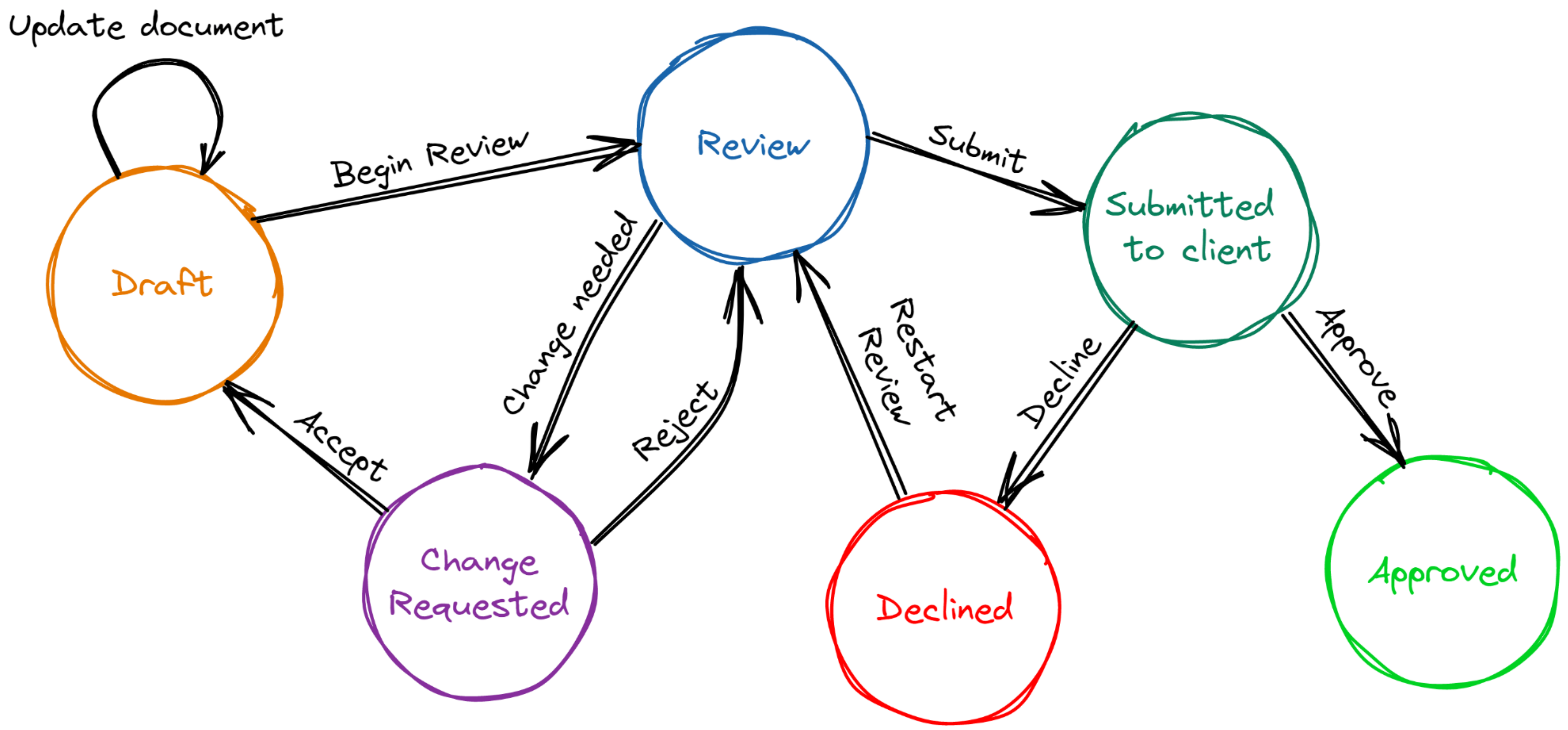
Most of the bugs your users experienced in the last 24 hours would probably never have happened if you had used a statecharts to model possible states and transitions.

2:25 PM · Jan 3, 2020 · Twitter Web App

10 Retweets **3** Quote Tweets **123** Likes

A hypothetical proposal document workflow

Time for a demo!

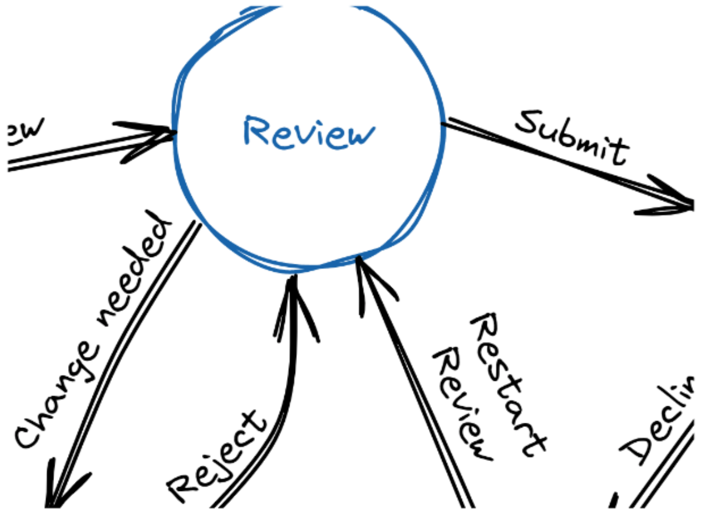


An overview of the code

How this the diagram is implemented in code

```
machine.Configure(State.Review)
    .Permit(Triggers.ChangedNeeded, State.ChangesRequested)
    .Permit(Triggers.Submit, State.SubmittedToClient)
    .OnEntryAsync(OnReviewEntryAsync)
    .OnExitAsync(OnReviewExitAsync);

// other states and transitions continued...
```

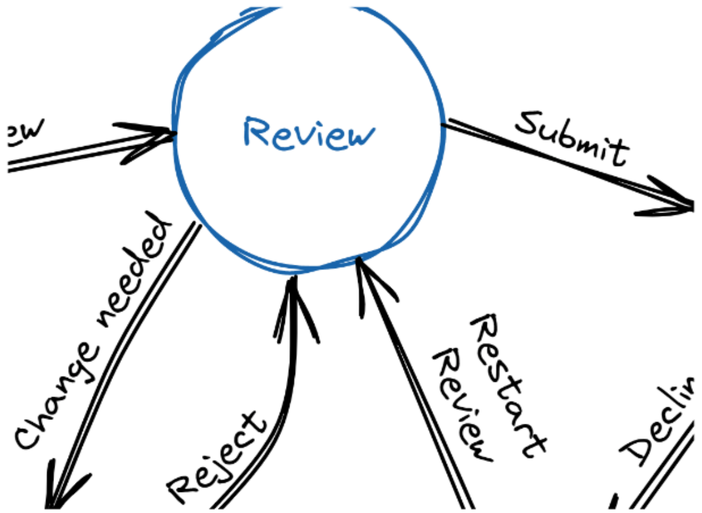


An overview of the code

How this the diagram is implemented in code

```
machine.Configure(State.Review)
    .Permit(Triggers.ChangedNeeded, State.ChangesRequested)
    .Permit(Triggers.Submit, State.SubmittedToClient)
    .OnEntryAsync(OnReviewEntryAsync)
    .OnExitAsync(OnReviewExitAsync);

// other states and transitions continued...
```

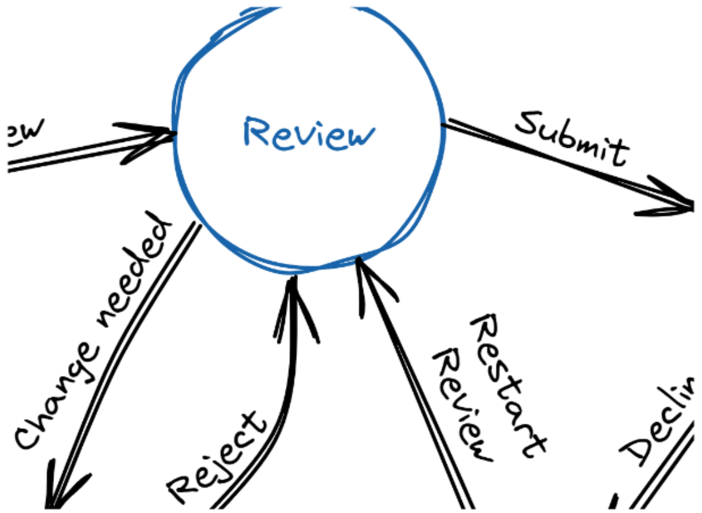


An overview of the code

How this the diagram is implemented in code

```
machine.Configure(State.Review)
    .Permit(Triggers.ChangedNeeded, State.ChangesRequested)
    .Permit(Triggers.Submit, State.SubmittedToClient)
    .OnEntryAsync(OnReviewEntryAsync)
    .OnExitAsync(OnReviewExitAsync);

// other states and transitions continued...
```

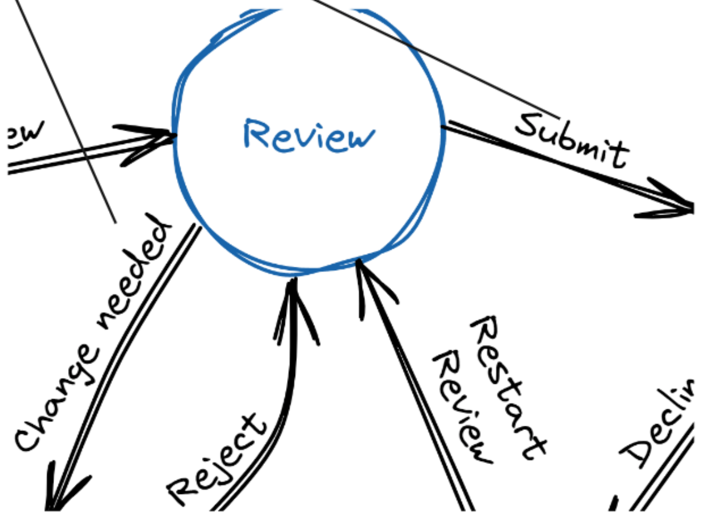


An overview of the code

How this the diagram is implemented in code

```
machine.Configure(State.Review)
    .Permit(Triggers.ChangedNeeded, State.ChangesRequested)
    .Permit(Triggers.Submit, State.SubmittedToClient)
    .OnEntryAsync(OnReviewEntryAsync)
    .OnExitAsync(OnReviewExitAsync);

// other states and transitions continued...
```

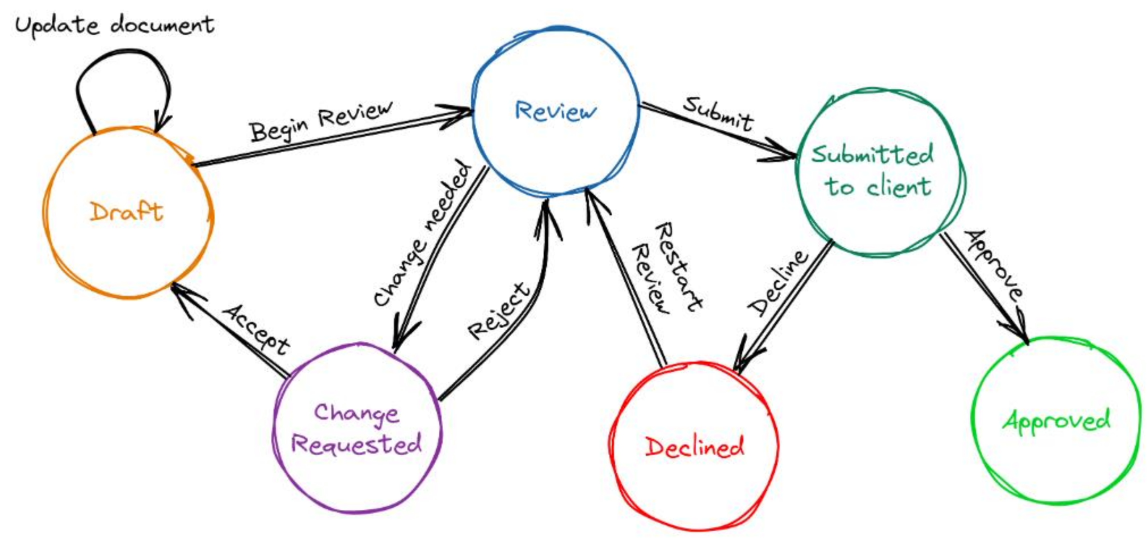


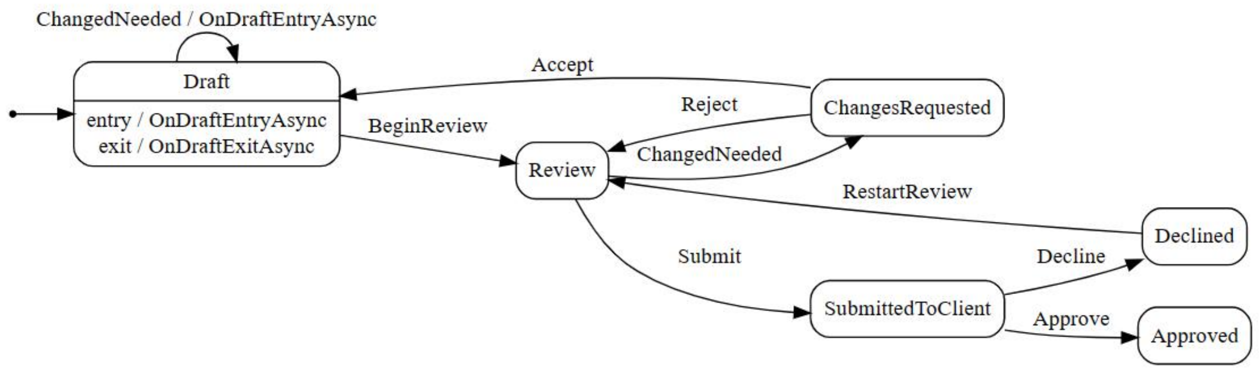
Keeping the domain design and code in sync

Keeping the domain experts/business analysts and the developers in sync via the power of finite state machines

Developers can make changes in accordance with new requirements and then assert that the code matches

On the left is the diagram created by hand and on the right is the output of the finite state machine library

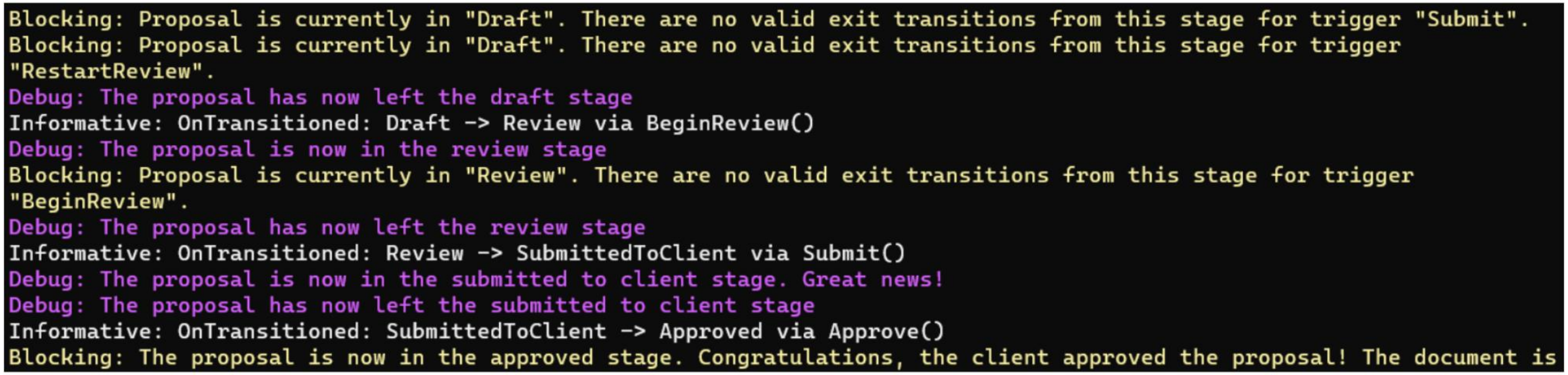




Auditing, event logs, integrations

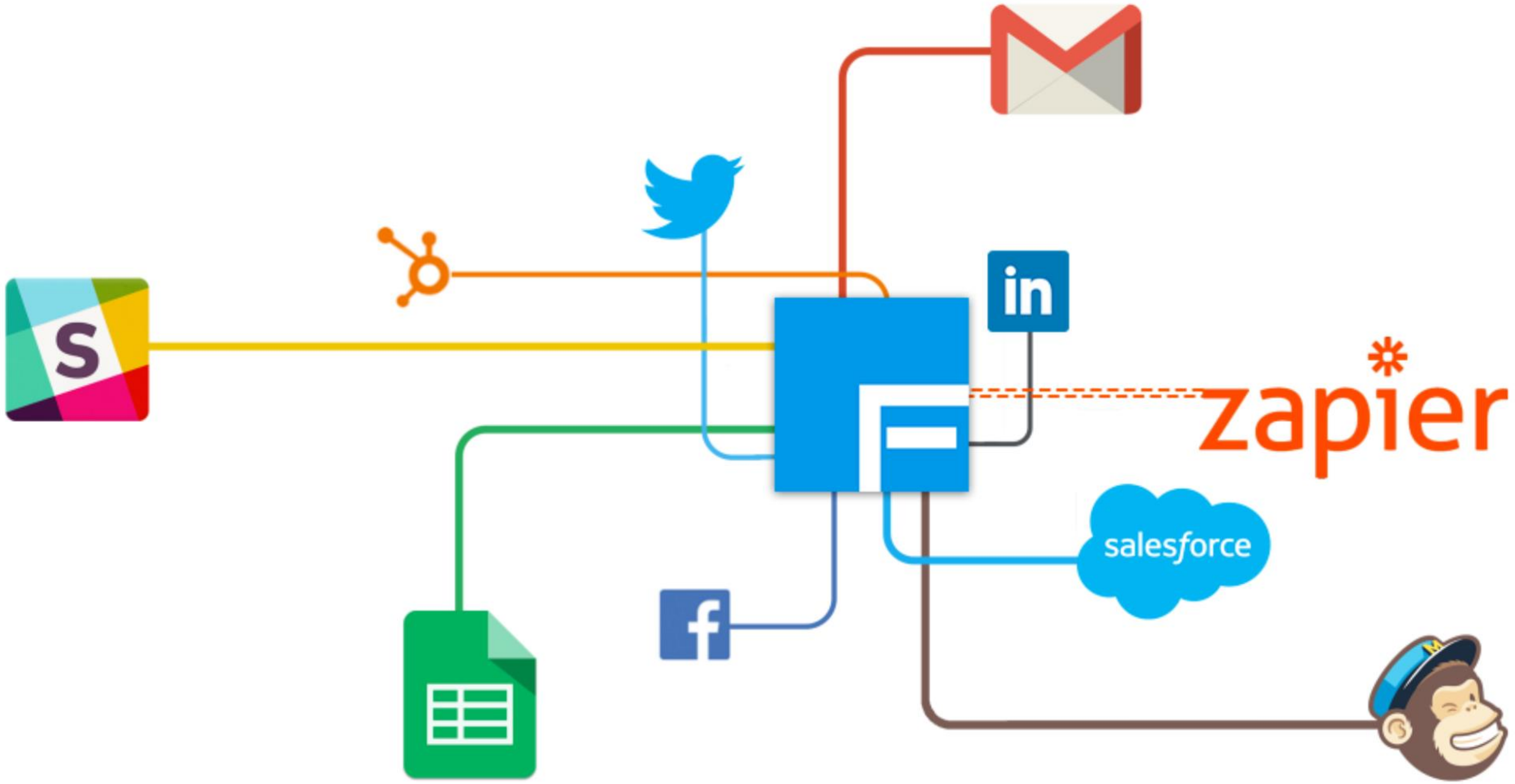
Now that we've established how a workflow is designed with an FSM we can cover all of the "extra" benefits

- Every line of text in the screenshot can be plugged into some integration
  - Right now as you've seen SMS is one such integration
- Everything can be logged, audited, and tracked over time
- Perfect for legal documents or simply finding out who updated the document and why
- Perfect for Event Driven Architecture - every update can be published to a queue ("event bus")



```
Blocking: Proposal is currently in "Draft". There are no valid exit transitions from this stage for trigger "Submit".
Blocking: Proposal is currently in "Draft". There are no valid exit transitions from this stage for trigger
"RestartReview".
Debug: The proposal has now left the draft stage
Informative: OnTransitioned: Draft -> Review via BeginReview()
Debug: The proposal is now in the review stage
Blocking: Proposal is currently in "Review". There are no valid exit transitions from this stage for trigger
"BeginReview".
Debug: The proposal has now left the review stage
Informative: OnTransitioned: Review -> SubmittedToClient via Submit()
Debug: The proposal is now in the submitted to client stage. Great news!
Debug: The proposal has now left the submitted to client stage
Informative: OnTransitioned: SubmittedToClient -> Approved via Approve()
Blocking: The proposal is now in the approved stage. Congratulations, the client approved the proposal! The document is
```

Integration examples



Improved maintainability and testing

We can go from code that looks like this that grows in complexity over time due to all the complex conditionals...

```
if (document.IsInDraft)
{
    if (!document.HasBeenReviewedByLegal)
    {
        ...
    }
    else if (document.IsSentToCustomer && document.Approved || document.Rejected)
    {
        ...
    }
     ...
}
```

...to code that looks like this:

```
  private async Task OnSubmittedToClientEnterAsync()
  {
      await notificationService.SendUpdateAsync(Priority.Verbose, "The proposal is now in the submitted to client stage.
  }
```

Conclusion

- Explained a part of what Proposal Management involves
- I devised a system that will allow us to model complex document workflows that can be changed easily when required
- Described how we can use integrations to inform stakeholders when changes have been made (SMS being one way)
- Gave examples of how we can use this across [company project name] via Event Driven Architecture

UX/UI

The existing Proposal Management software is a work in progress and based on outsourced development

- The UX and UI does not yet fully match [company project name]
- A requirement was to make it look *unique*
- I researched what other proposal software looks like in order to understand real world use cases...

[slides showing internal product removed]