

# 7 techniques to tame a Legacy Codebase

> Nicolas Carlo

IT'S DANGEROUS TO GO  
ALONE! TAKE THIS.



MenderCon 2021

**Legacy Code is profitable code  
you are afraid to change**



Legacy Code is **profitable** code  
you are afraid to change

Legacy Code is **profitable** code  
you are **afraid** to change







YOU

No test

Outdated docs

Short deadline



# Nicolas Carlo

 @nicoespeon

Tech Lead @ Busbud

 [understandlegacycode.com](https://understandlegacycode.com)





1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

# Architecture Decision Records

Never gets outdated!

- Log decision taken now to revisit in the future



- Detail the context around the decision

- You can't be wrong 





## 3. Never read on master DB

---

Date: 2020-05-06

### Status

---

Accepted



### Context

---

We noticed that the master database was acquiring locks due to read requests, using [heroku pg:locks](#) and [heroku pg:outliers](#). It prevented some migration to run because the level of lock it required was conflicting with the locks those other queries were acquiring.

This PR for instance needed to drop a constraint (which is supposedly a fast operation) but couldn't due to this:

<https://github.com/busbud/napi/pull/6774>



### Decision

---

The decision taken is to never use master DB for reads, and make this an "opt-in" solution rather than "opt-out", by removing the `useMaster: true` from Sequelize models' default scopes.

See <https://github.com/busbud/napi/pull/6787>



### Consequences

---

- We only use master for write operations
- We have to choose to use master explicitly and back up this decision













# Start now!

## There's a CLI to iron everything

- Use adr-tools CLI

- Great additions to PRs!

- Keep it simple

	0001-record-architecture-decisions.md	Add ADR TOC generator	2 years ago
	0002-secure-JSON-output.md	Accept JSON security ADR ( <a href="#">#30</a> )	2 years ago
	0003-files-in-JSON-output-diagram.p...	Image and markup change to display image	16 months ago
	0003-files-in-JSON-output.md	Update 0003-files-in-JSON-output.md	16 months ago
	0004-microservice-authentication-me...	add 0004-microservice-authentication-method ( <a href="#">#49</a> )	15 months ago
	0005-integration-with-github.md	Add ADR for Github integration	11 months ago
	0006-reduce-number-of-emails-per-...	ADR Reduce number of emails per form submission	10 months ago
	0007-formatron3000.jpg	Added new ADR 7. Changing the storage for form metadata.	4 months ago
	0007-replacing-the-storage.md	Added new ADR 7. Changing the storage for form metadata.	4 months ago
	0008-the-runner.jpg	New ADR for the new runner.	2 months ago
	0008-the-runner.md	New ADR for the new runner.	2 months ago
	README.md	New ADR for the new runner.	2 months ago

Example: UK Ministry of Justice



1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

1. ADRs

 2. Brain Dump

3. Over-committing

4. Mikado Method

5. Hotspots Analysis

6. Approval Testing

7. Coding Katas

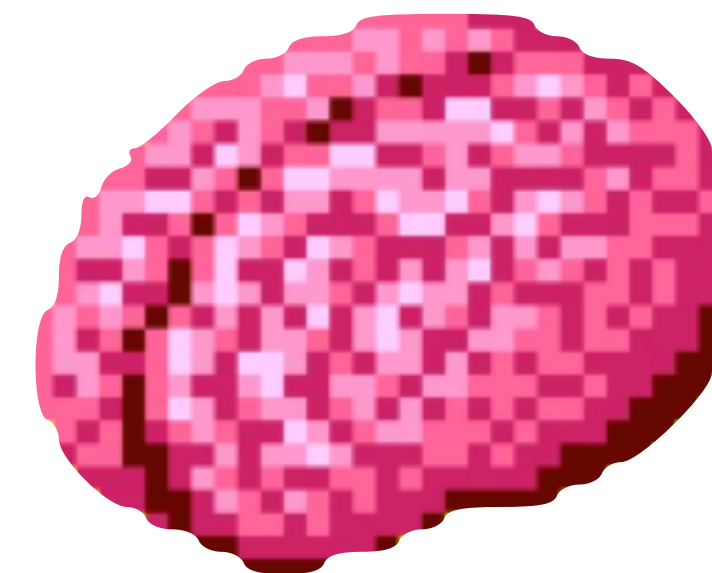
# Your brain can't hold much



Short-term memory capacity: 7 items

What if you had a second brain?

- It holds information
- You tap into at any time
- You can focus on **thinking**





such flexible

wow.



very low tech

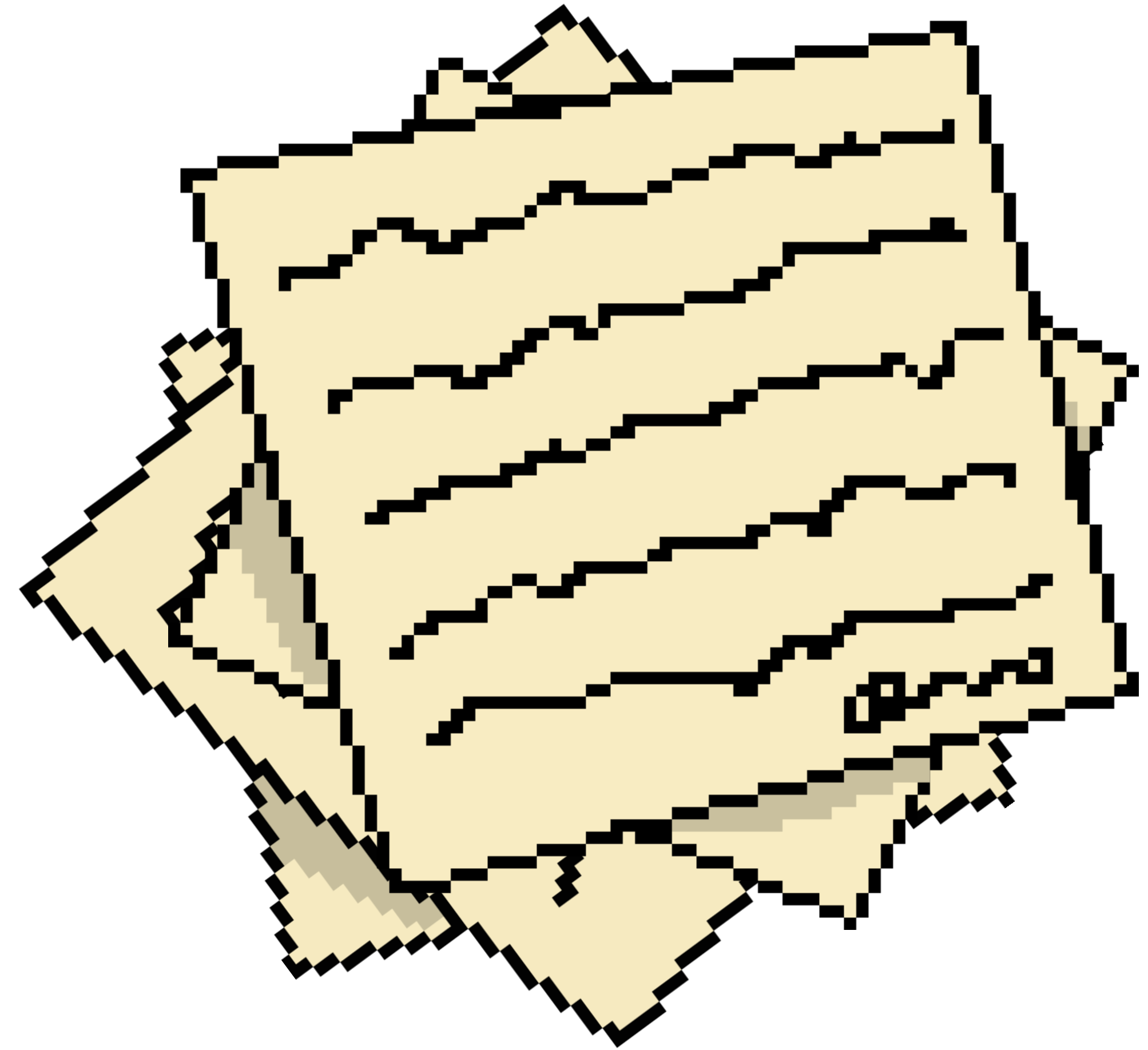
i haz one!



# Use Pen & Paper

But use it properly

1. Stay Focus
2. Avoid Tunnel Effect
3. Do One Thing





1. ADRs
- ▶ 2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

1. ADRs
2. Brain Dump
- ▶ 3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

# Commit more often

Then, commit even more!

- Every 5 minutes
- Every 2 minutes
- At every change



Add response to error details

 nicoespeon committed 2021-02-02T15:17:30Z

Extract variable

 nicoespeon committed 2021-02-02T15:17:42Z

Add response to error details

 nicoespeon committed 2021-02-02T15:17:55Z

Create ImpossibleResponse class

 nicoespeon committed 2021-02-02T15:18:49Z

Use ImpossibleResponse class directly

 nicoespeon committed 2021-02-02T15:19:19Z

Remove redundant mutation

 nicoespeon committed 2021-02-02T15:19:32Z

Inline variables

 nicoespeon committed 2021-02-02T15:19:44Z

Remove TODO ...

 nicoespeon committed 2021-02-02T15:22:33Z

Extract function

 nicoespeon committed 2021-02-02T15:26:22Z

Re-use extracted function

 nicoespeon committed 2021-02-02T15:26:49Z

Rename variables for consistency

 nicoespeon committed 2021-02-02T15:27:07Z



# Try it for 1h

Can you feel the benefits?

- It's faster to retreat and try something different
- Prevents "while we're at it" moments
- Easier to review



**vanessa** 13:10

reviewed! no worries about the size of the PR. your changes usually aren't hard to follow because your commits are really well-organized.

1. ADRs
2. Brain Dump
- ▶ 3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

1. ADRs
2. Brain Dump
3. Over-committing
- ▶ 4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

# Brain Dump, with a process

1. Write down your goal

UPGRADE ORM

2. Start a 5min timer



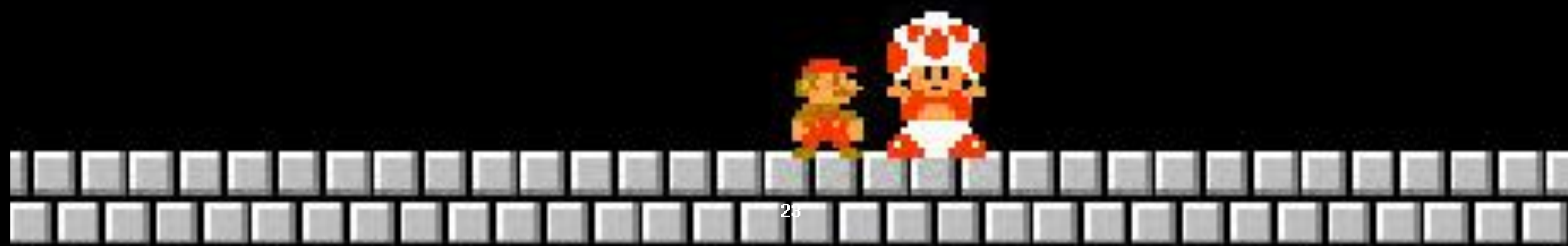
3. Try to complete your goal





THANK YOU MARIO!

BUT OUR GOAL IS IN  
ANOTHER CASTLE!

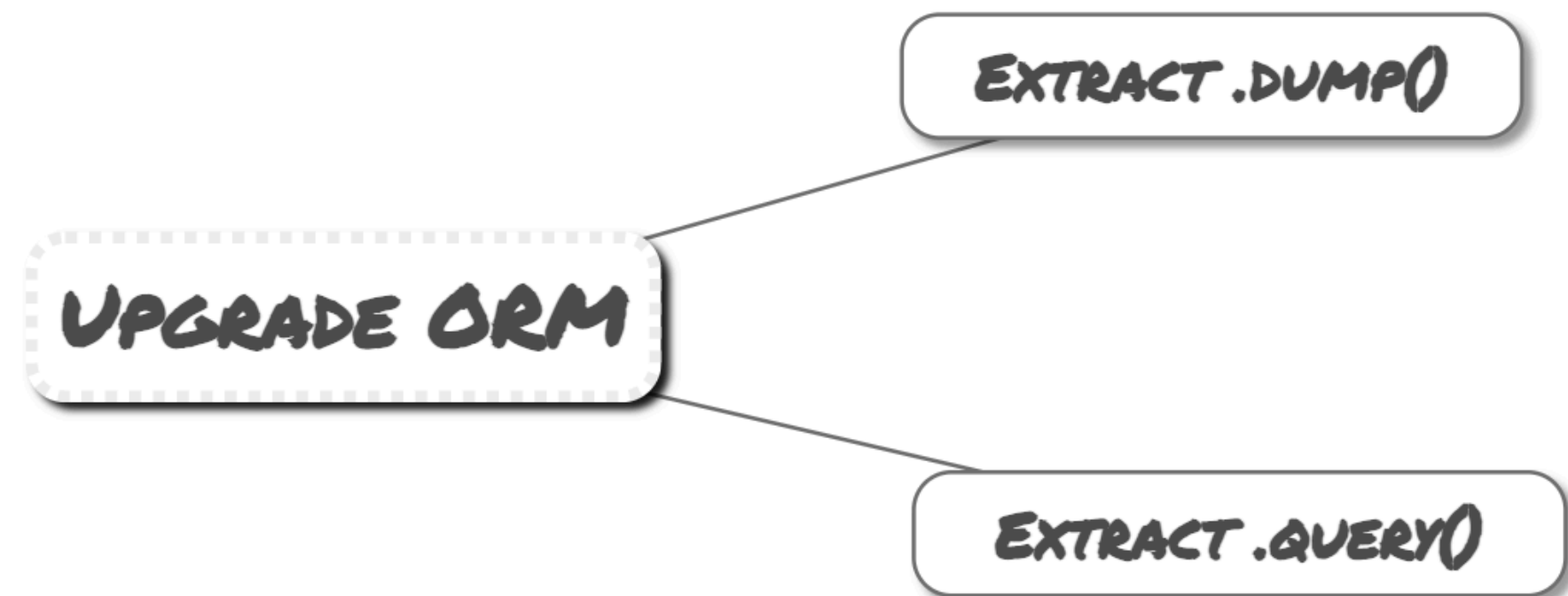




# Timer rings, that's fine!

Stop, revert, think

1. Note what's blocking you = sub-goals



2. **Revert** (seriously, do it)

3. Pick a sub-goal, start over



THANK YOU MARIO!

YOUR QUEST IS OVER.

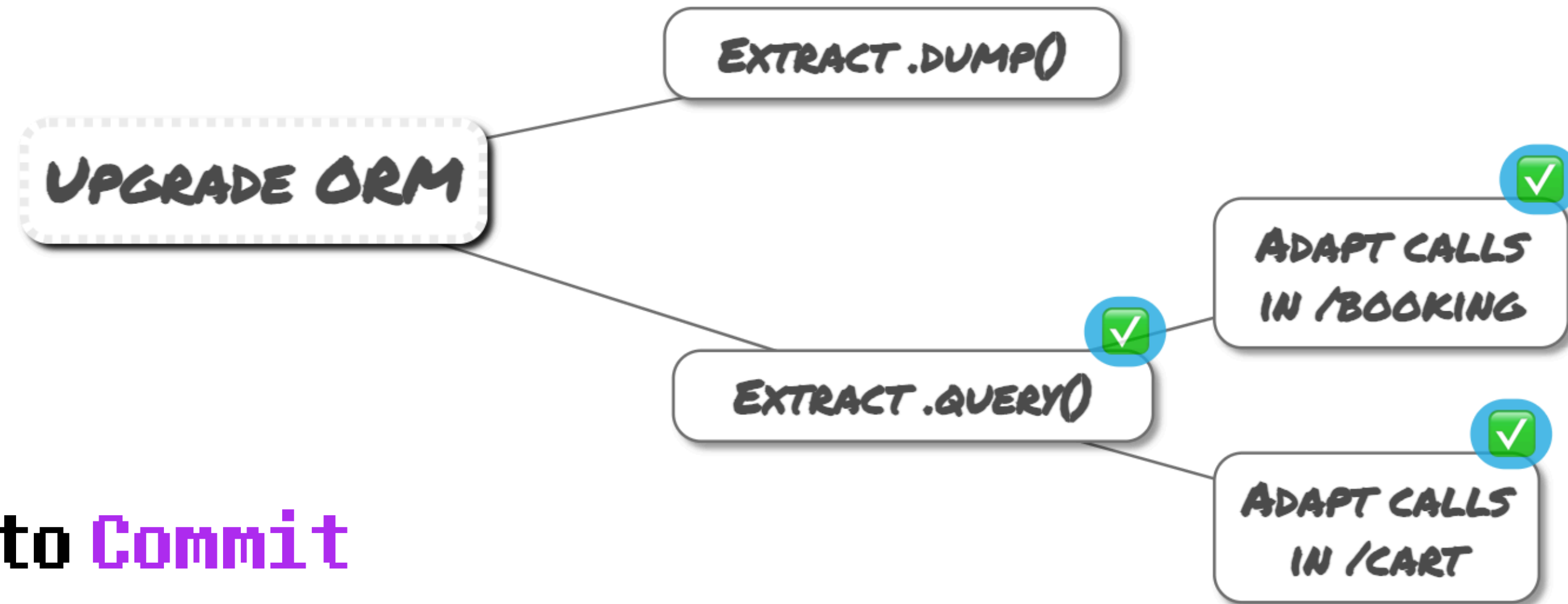
WE PRESENT YOU A NEW QUEST.



# You completed the goal!

Yay, commit, repeat

1. Celebrate!



2. Now it's a good moment to Commit

3. Pick another sub-goal, start over



MARIO  
022600

×16

WORLD  
1-2

TIME  
371

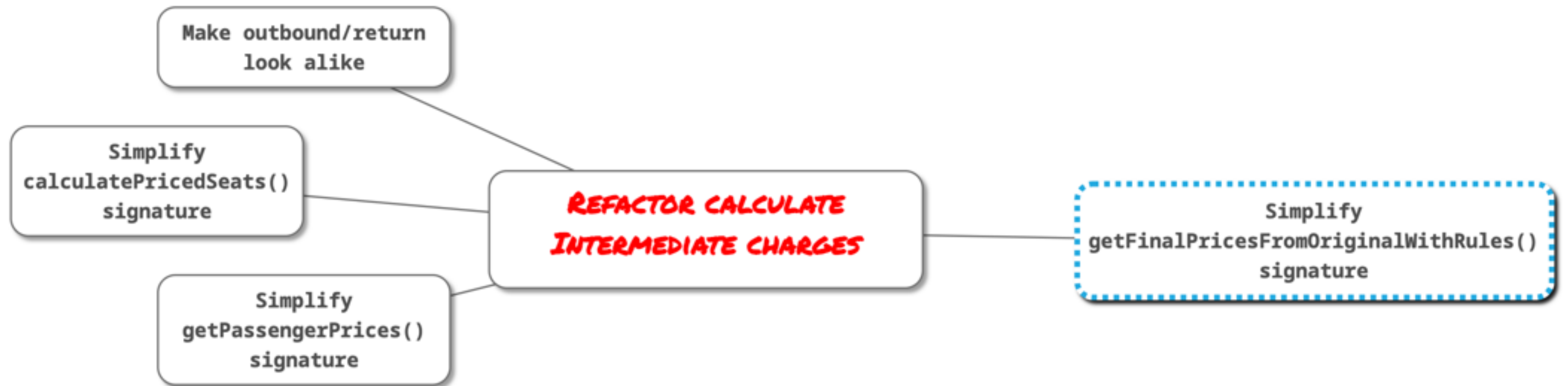


# Phase 1: map the terrain

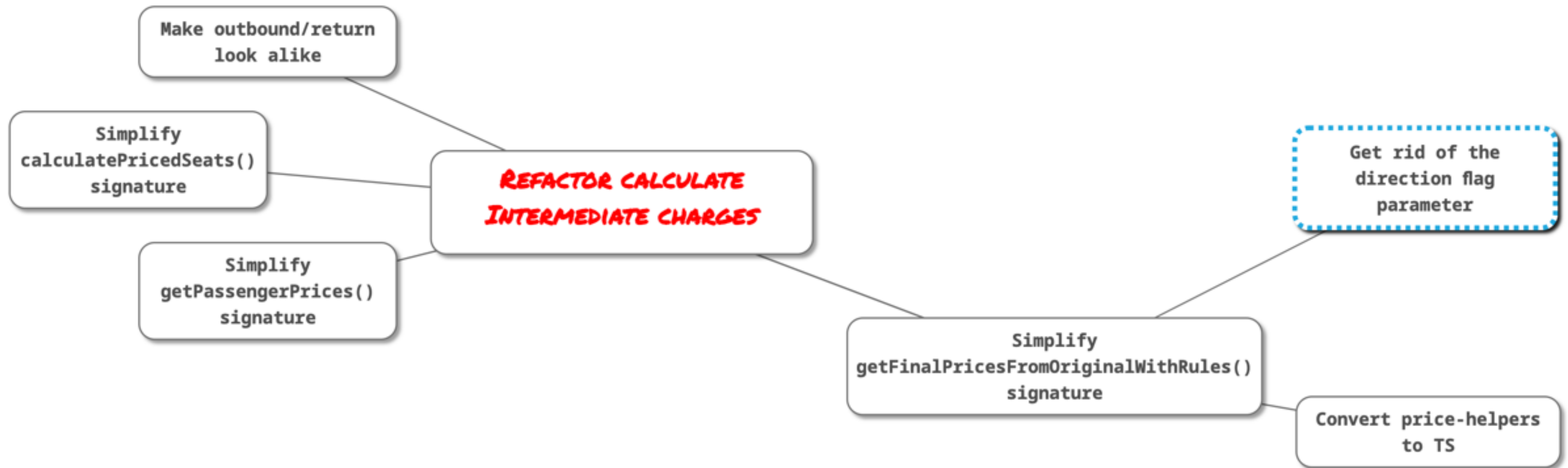
**REFACTOR CALCULATE  
INTERMEDIATE CHARGES**



# Phase 1: map the terrain



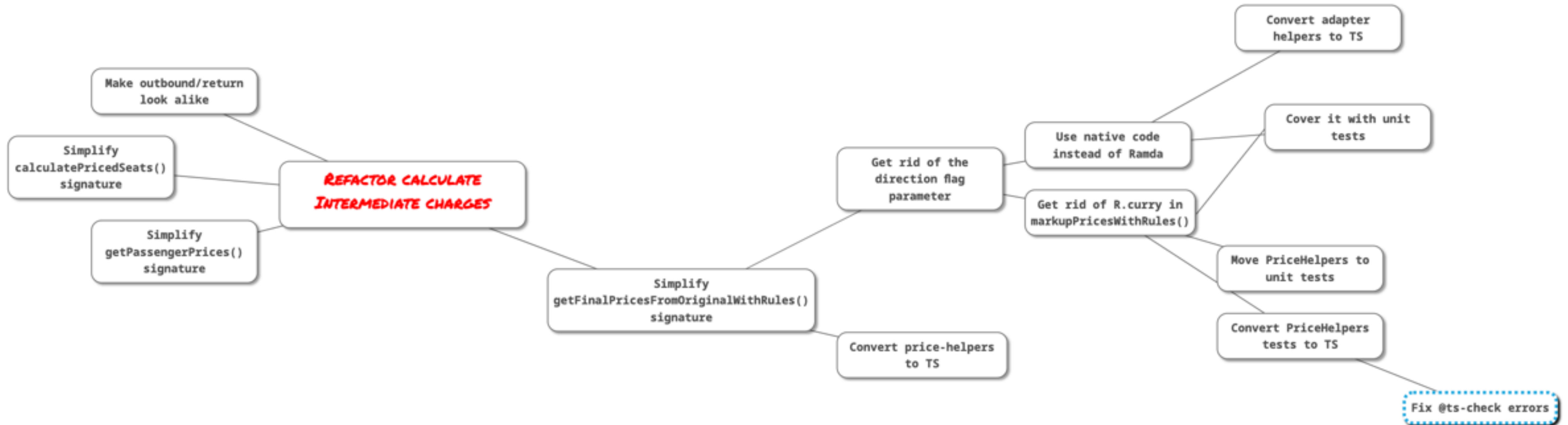
# Phase 1: map the terrain



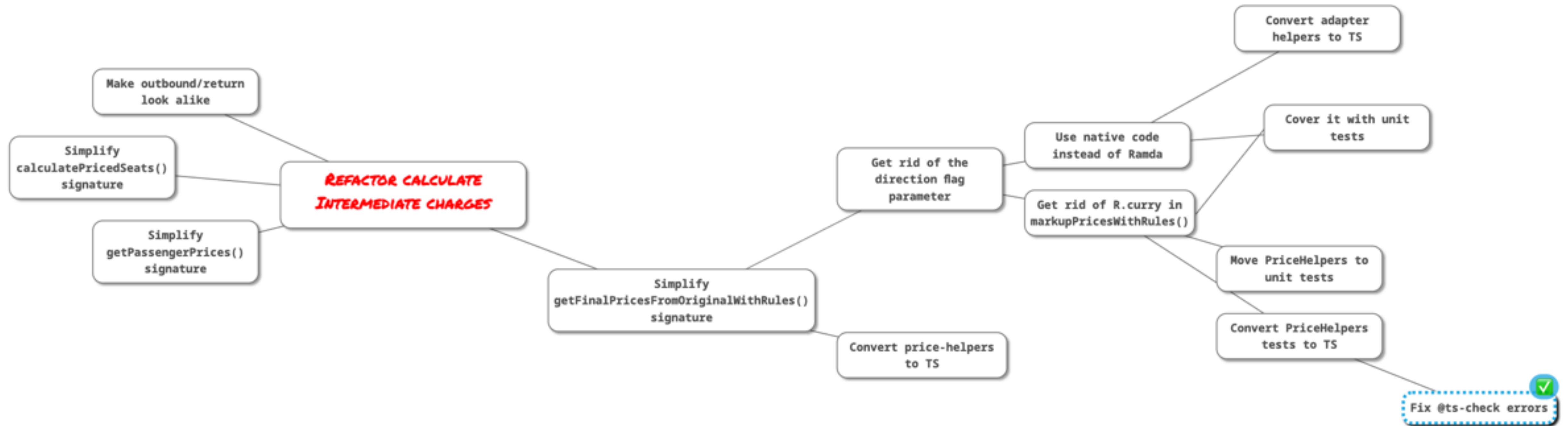
# Phase 1: map the terrain



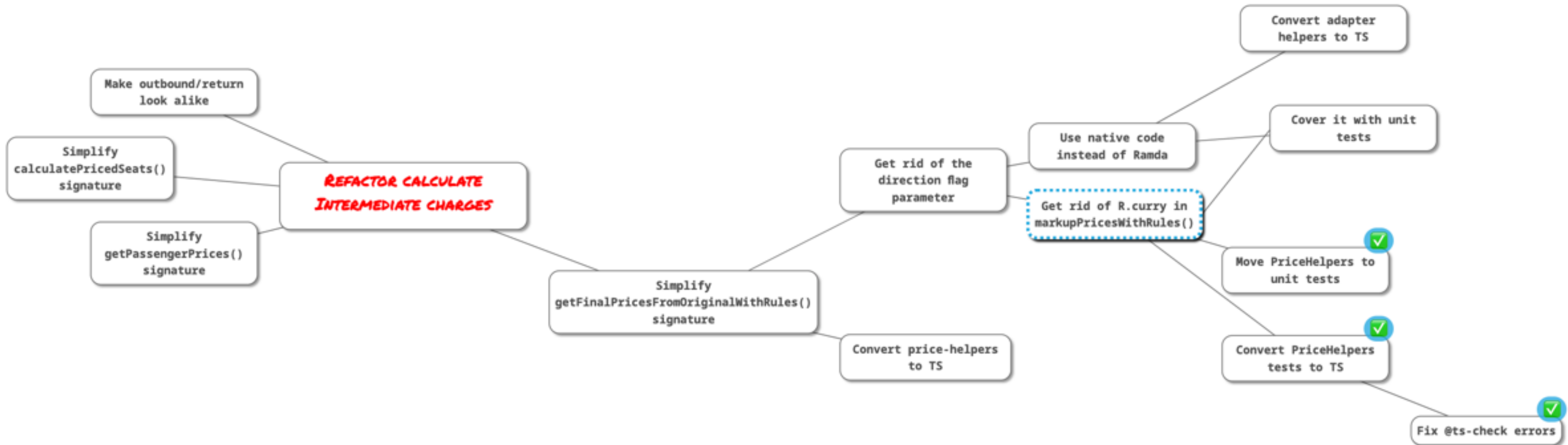
# Phase 1: map the terrain



# Phase 2: Productivity strikes

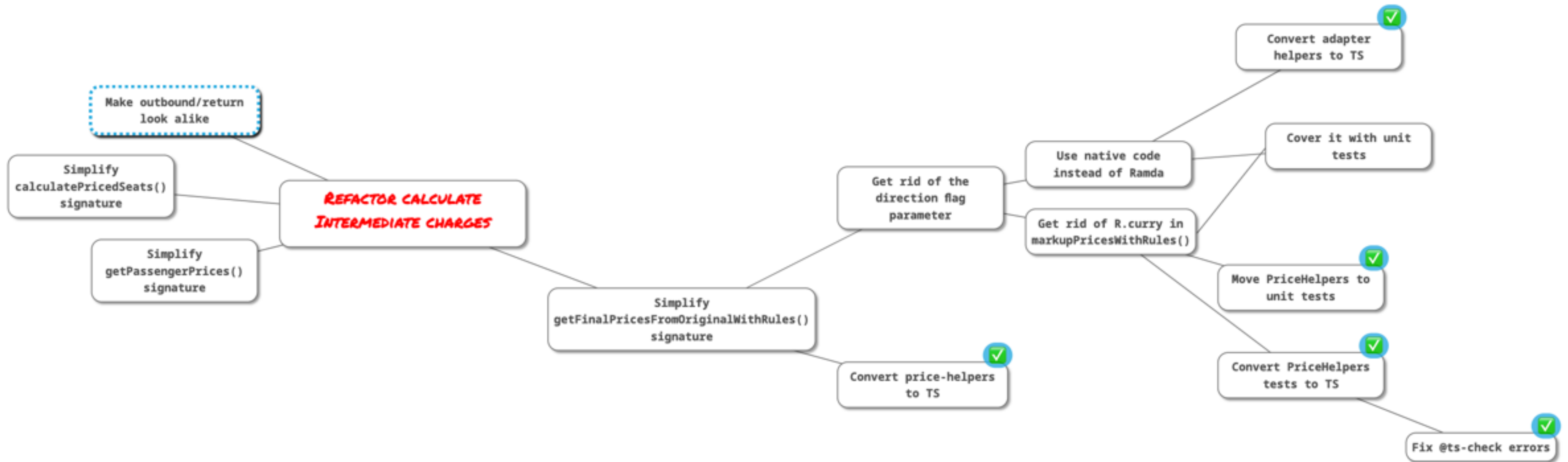


# Phase 2: Productivity strikes

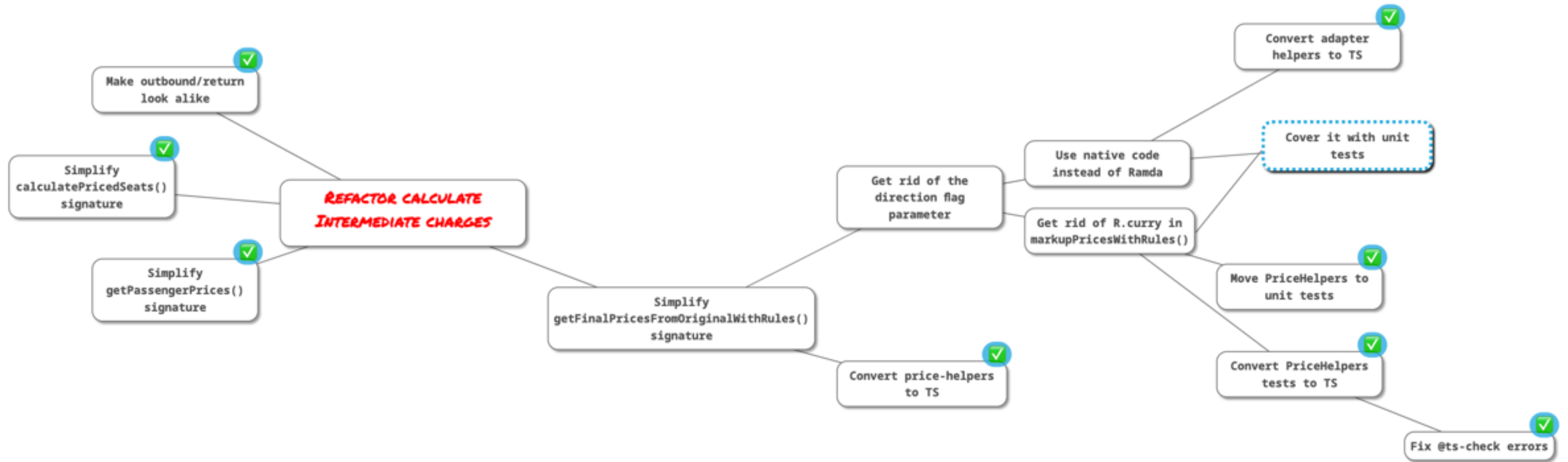




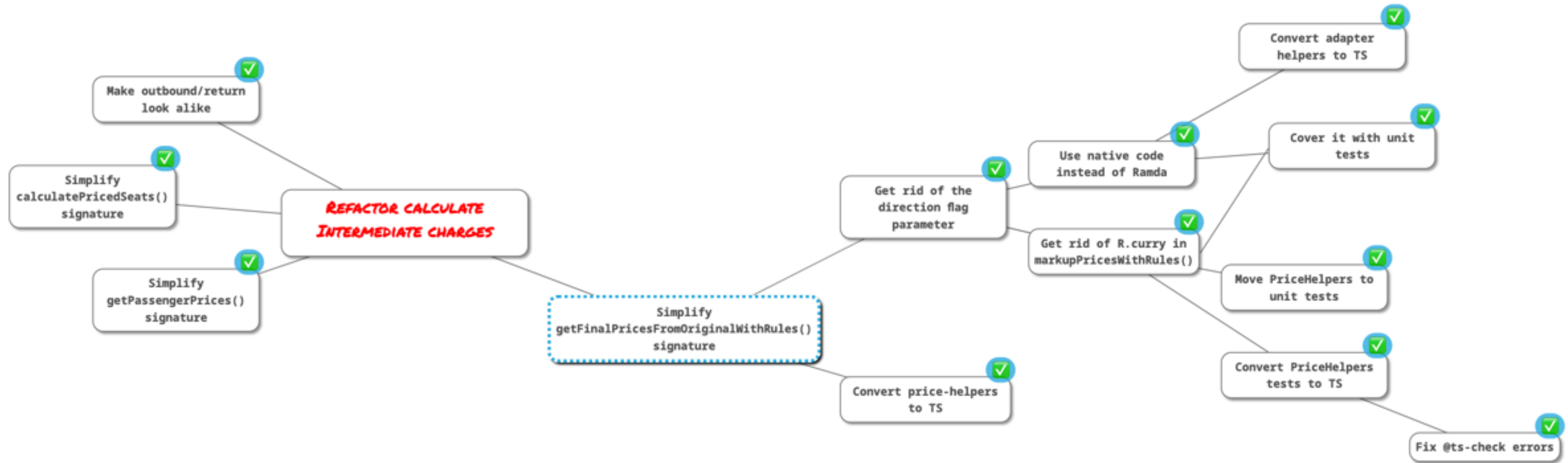
# Phase 2: Productivity strikes



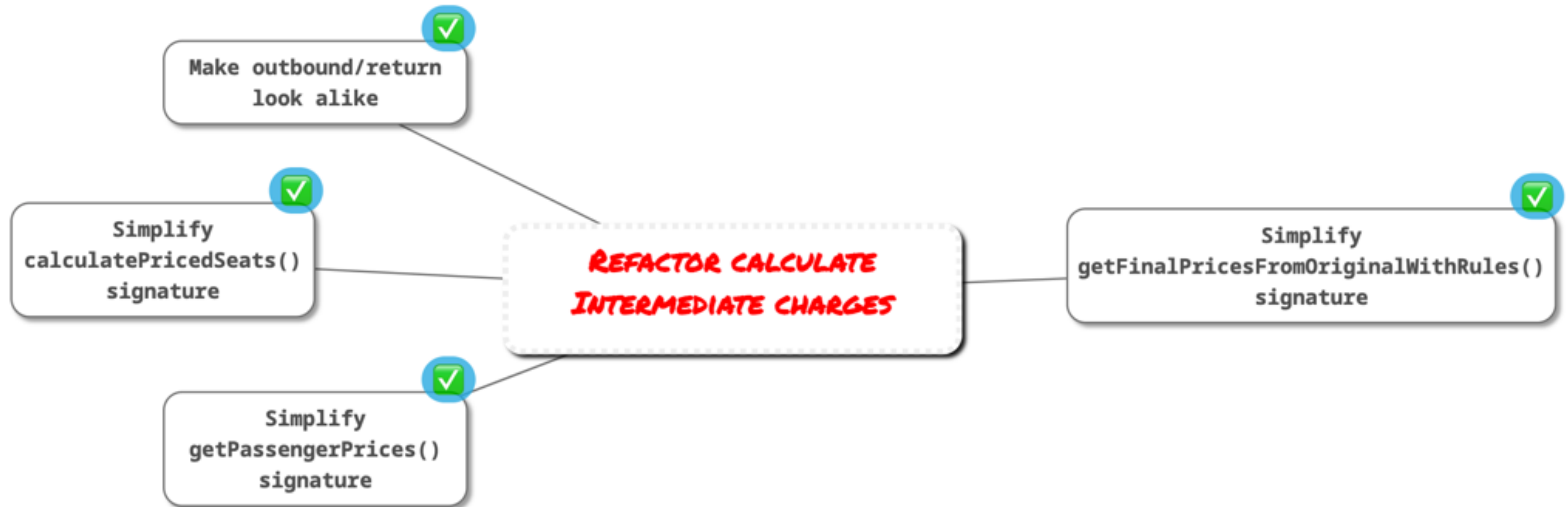
# Phase 2: Productivity strikes



# Phase 2: Productivity strikes



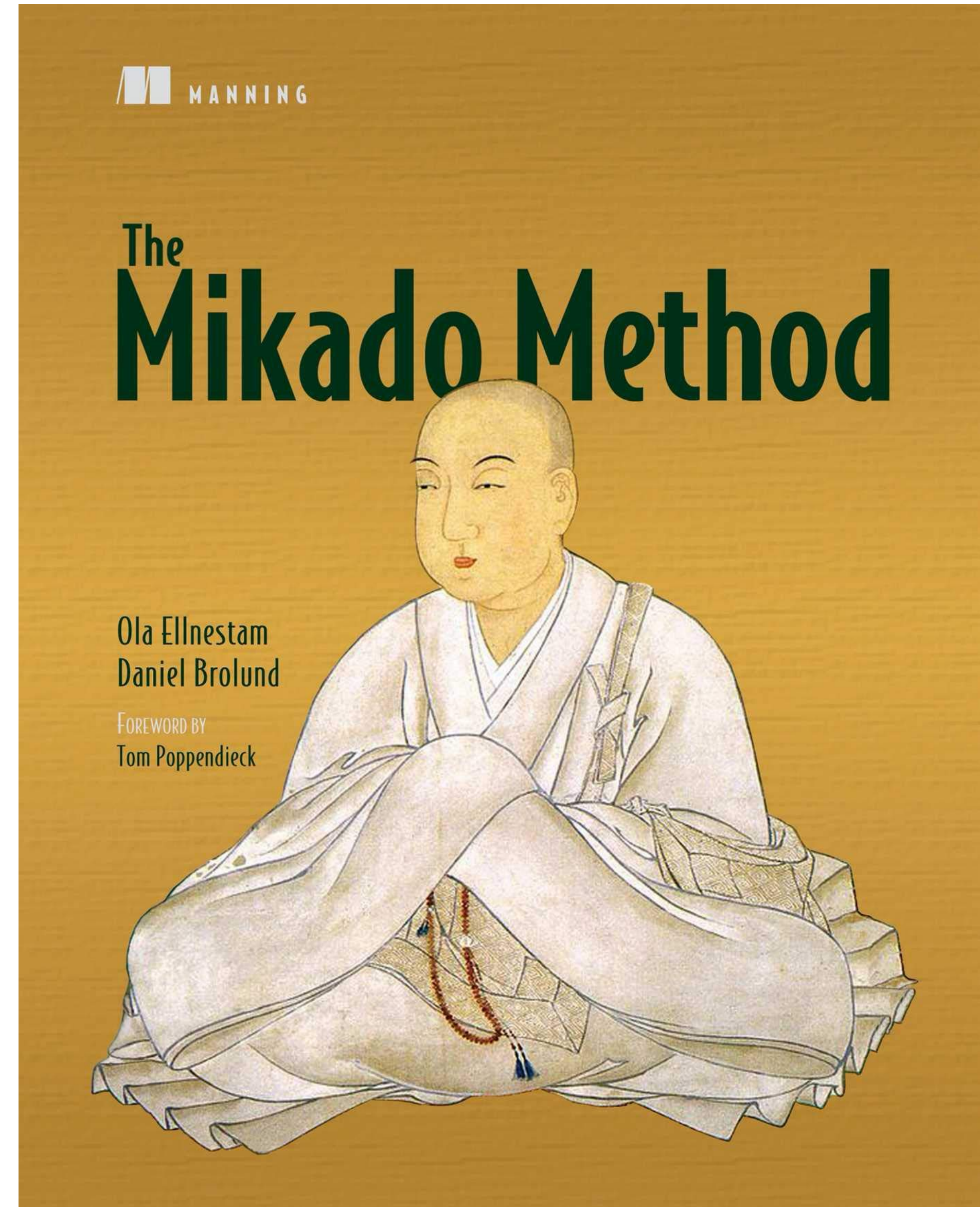
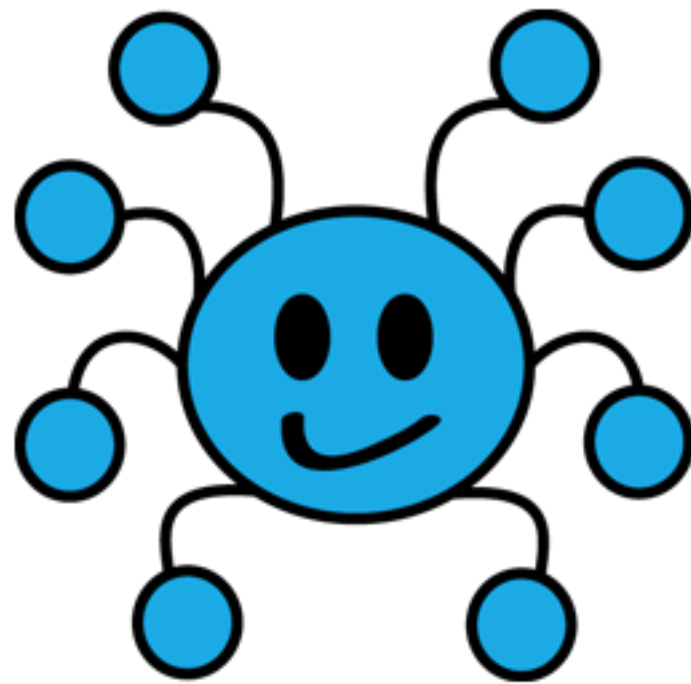
# Phase 2: Productivity strikes





# There's a book!

- Avoid Tunnel Effect
- Stop & ship any time
- [mindmup.com](http://mindmup.com) to collaborate





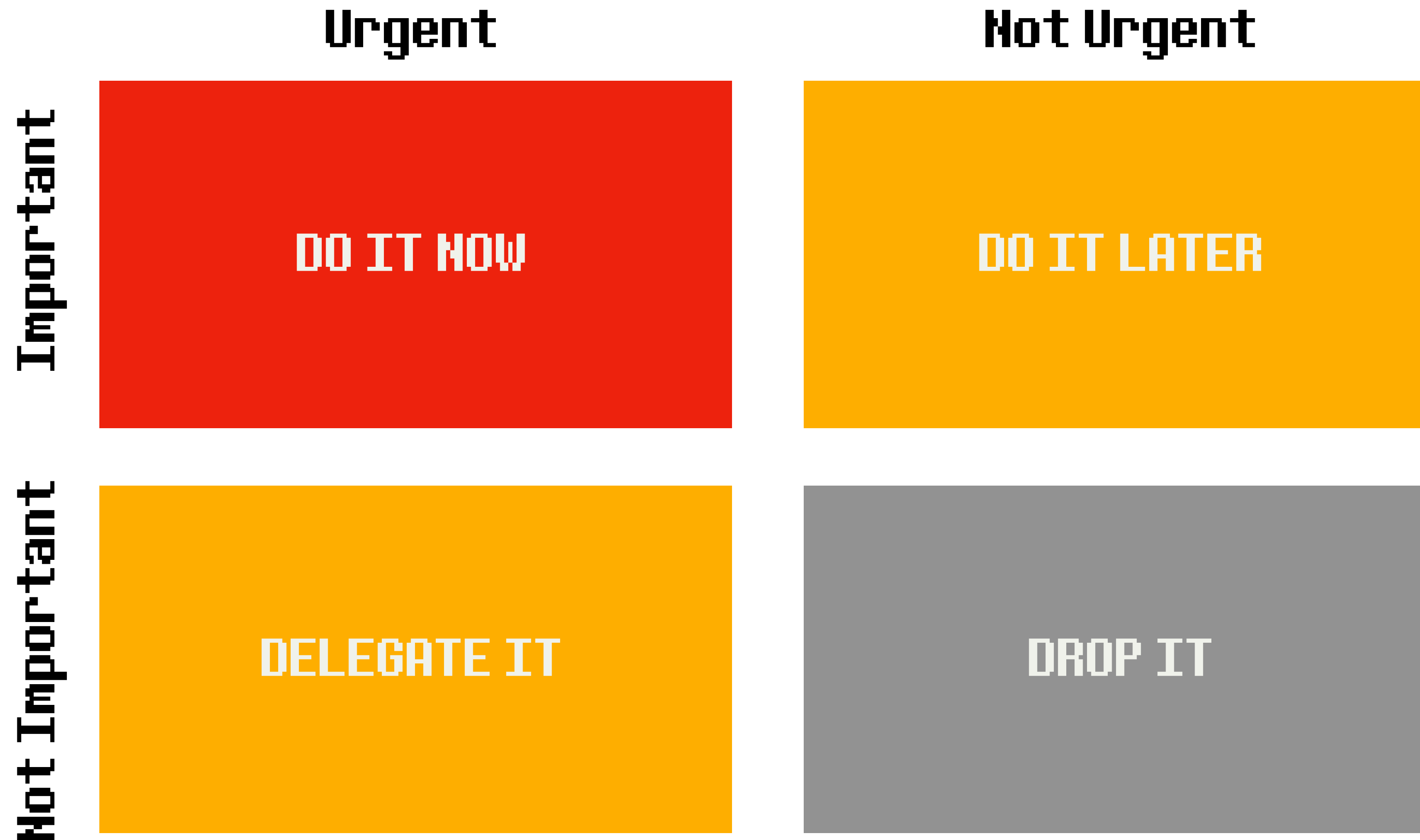
1. ADRs
2. Brain Dump
3. Over-committing
- ▶ 4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
- ▶ 5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

**When everything is urgent,  
nothing is**

# Eisenhower Matrix

A tool to prioritize stuff



# Static Analysis tools

They can't save you

- They give you the **Importance**

Codebase summary

MAINTAINABILITY



8 yrs

TEST COVERAGE



Can we stop delivery for 8 years  
and fix these 2,219 critical issues?

- But where to find the **Urgency**?



# Version Control metadata

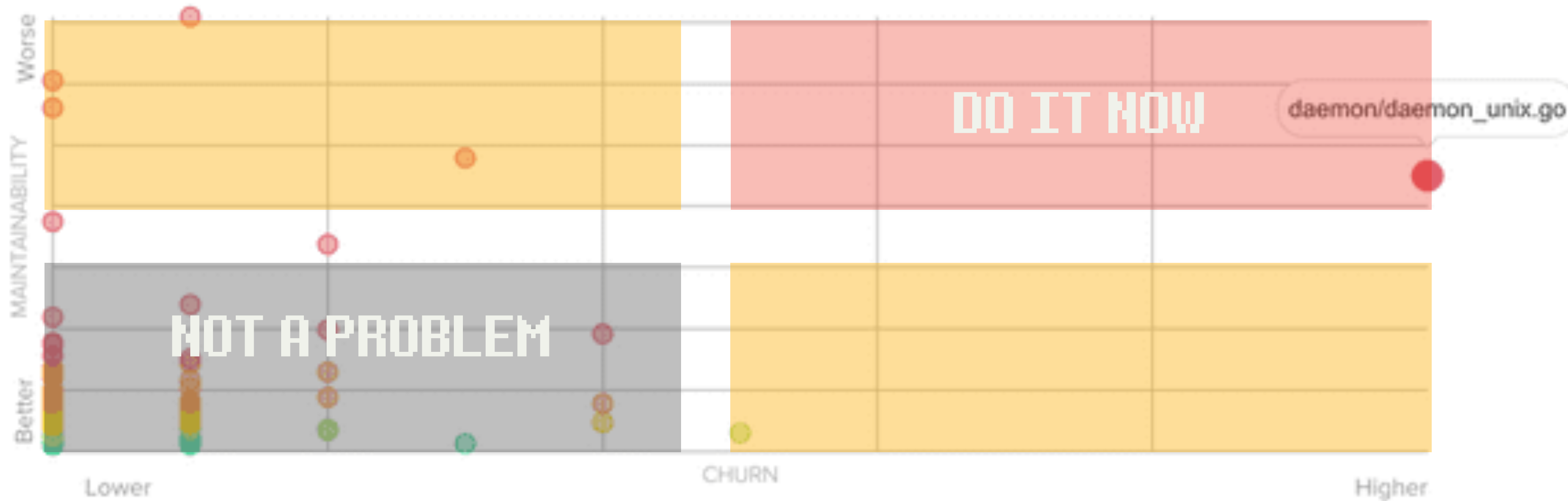
Git knows it all!

- How often you touch the code => **Code Churn**
- 50 files most touched in the past year (that's enough):

```
git log --name-only --since=12.month --format=format: \%n\%  
| egrep -v '^$' \  
| sort \  
| uniq -c \  
| sort -nr \  
| head -50
```

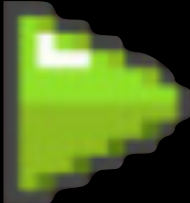
# Prioritize Tech Debt

Best ROI for all stakeholders



Analysis of the Docker engine

1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
- ▶ 5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
-  6. Approval Testing
7. Coding Katas

What the system **does** is more  
important than what it **should do**





YOU

No test

Outdated docs

Short deadline

# Fastest way to write missing tests

High-level recipe 📖

1. Generate an output you can put in a file

```
it("should update quality", () => {  
  expect(updateQuality("foo", 0, 0)).toMatchSnapshot()  
})
```



# Fastest way to write missing tests

## High-level recipe 📦

### 2. Use tests coverage to find missing inputs

```
14  updateQuality() {
15  1x    for (var i = 0; i < this.items.length; i++) {
16  1x      E if (
17          this.items[i].name != "Aged Brie" &&
18          this.items[i].name != "Backstage passes to a TAFKAL80ETC concert"
19      ) {
20  1x      I if (this.items[i].quality > 0) {
21          if (this.items[i].name != "Sulfuras, Hand of Ragnaros") {
22          this.items[i].quality = this.items[i].quality - 1;
23          }
24      }
25      } else {
26          if (this.items[i].quality < 50) {
27          this.items[i].quality = this.items[i].quality + 1;
28          if (
29          this.items[i].name == "Backstage passes to a TAFKAL80ETC concert"
```

# Fastest way to write missing tests

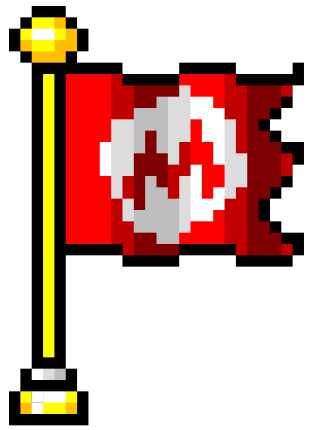
High-level recipe 🍳

## 3. Use mutations to verify snapshots

```
it("should update quality", () => {  
  expect(updateQuality).toVerifyAllCombinations(  
    ["foo", "Aged Brie", "Sulfuras"],  
    [-1, 0],  
    [0, 1, 2, 50]  
  )  
})
```

Testing 24 combinations

# Now you are safe!

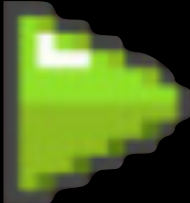



You can quickly:

- Refactor the code
- Write unit tests on the code you create
- Change behavior using these new tests
- Remove approval tests you don't need



Detailed version: [bit.ly/approval-testing](https://bit.ly/approval-testing)

1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
-  6. Approval Testing
7. Coding Katas

1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
-  7. Coding Katas





# Coding Katas

Exercises to practice your skills

5 stages to practice working with Legacy Code:



1. Gilded Rose kata
2. Tennis Refactoring kata



3. Trip Service kata
4. Trivia kata



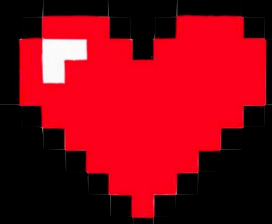
5. Baby Steps Timer kata

# When to practice?

- At home, if you're lucky
- Use company's training budget
- 1h "meeting" every Sprint, with other devs
- Code Retreat:
  - Full day practice, free
  - Friday and/or Saturday (Global Day, November) 🌍

1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

1. ADAs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

Bonus Stage  Going Further



# Legacy Code: **First Aid Kit**

> [understandlegacycode.com/first-aid-kit](https://understandlegacycode.com/first-aid-kit)

- ▶ 14 techniques to quickly and safely rescue a codebase
- ▶ ~200 pages
- ▶ PDF, EPUB, and MOBI
- ▶ Light & Dark themes
- ▶ 3 printable cheat sheets + 1 exercise sheet

30% discount with this link > [bit.ly/first-aid-menders](https://bit.ly/first-aid-menders)



1. ADRs
2. Brain Dump
3. Over-committing
4. Mikado Method
5. Hotspots Analysis
6. Approval Testing
7. Coding Katas

# End Credits

- <https://www.fontmirror.com/determination>
- <http://pixelartmaker.com/>
- <http://www.addletters.com/>
- <https://vectorpixelstar.itch.io/>
- <https://www.pinclipart.com/>
- <https://nostalgic-css.github.io/NES.css/>