

# A Framework for Tech. Debt

# Technical Debt

# Introduction

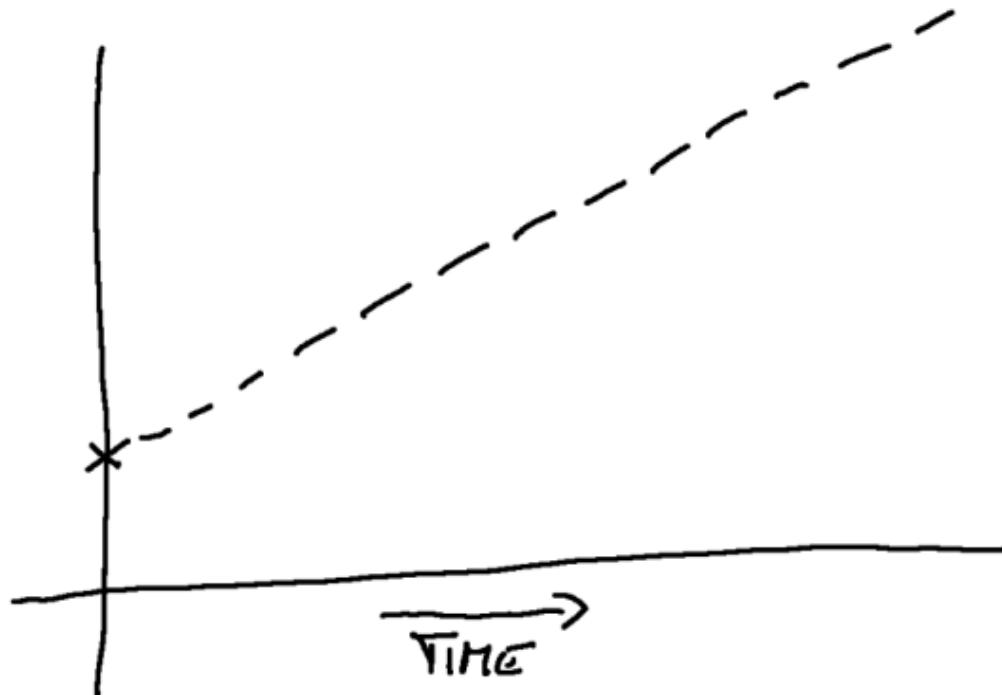
The diagram is a word cloud centered around the concept of code quality. It features several large, bold words in green, pink, and purple, each representing a different issue or pattern. Smaller, semi-transparent words are scattered around these larger ones, often appearing near their respective larger counterparts.

- Missing\_Tests**: Located at the top center, with smaller words like **Tight\_Coupling**, **Deployment\_Issues**, **Slow\_Builds**, **Workarounds**, **Magic\_Numbers**, **Readability\_Issues**, **Scalability\_Issues**, and **Brittle\_Code** surrounding it.
- Duplicated\_Code**: Located below **Missing\_Tests**, with smaller words like **Platform\_Debt**, **Performance\_Bottlenecks**, **Maintainability\_Problems**, and **Knowledge\_Silos** nearby.
- Code\_Complexity**: Located in the center, with smaller words like **Technical\_Bugs**, **Outdated\_Dependencies**, **Insufficient\_Logging**, **Over\_Engineering**, **Security\_Vulnerabilities**, and **Design\_Debt** scattered around it.
- Legacy\_Code**: Located at the bottom right, with smaller words like **Lack\_of\_Documentation**, **TODO\_Comments**, **N+1\_Qualities**, **Hardcoded\_Values**, **Test\_Debt**, **Infrastructure\_Debt**, and **Dead\_Code** surrounding it.

# Debt

# Introduction

---



# Software Rot

# Business Risk

# Me

---

👤 2+ Decades in Software Development

💻 Complete Stack

🛠 Staff @ Meister



## The Siblings of Meister



# Practical Example

---

You join a project as  
senior developer...

# Practical Example

---

- ✓ check README.md
- ✓ check Gemfile
- ✓ explore some tests
- ✓ explore some code
- ✓ check testsuite passes (in CI  )
- ✗ check RuboCop

# Practical Example



# RuboCop

# Practical Example

---

 check RuboCop

# Practical Example

```
$ cat .rubocop_todo.yml
# Offense count: 1
Lint/DuplicateBranch:
  Exclude:
    - app/model/team.rb
# Offense count: 8
Metrics/MethodLength:
  Exclude:
    - app/controllers/teams_controller.rb
    - app/model/team.rb
```

# Practical Example



check RuboCop

# Practical Example

```
$ wc -l .rubocop_todo.yml
1145 .rubocop_todo.yml # 1145 lines in file
```

# Practical Example

---

How can we get rid of the  
TODOs? 

## Four Steps Framework

# A Framework for Tech. Debt

---

## 1. Get the Numbers

# A Framework for Tech. Debt

---

```
# total number of lines
$ wc -l .rubocop_todo.yml
```

# A Framework for Tech. Debt

```
$ cat .rubocop_todo.yml
# Offense count: 1
Lint/DuplicateBranch:
  Exclude:
    - app/model/team.rb
```

# A Framework for Tech. Debt

```
$ grep 'Offense count:' .rubocop_todo.yml
Offense count: 1
Offense count: 4
Offense count: 2
Offense count: 28
# [...]

# extract number of rules violated
$ grep 'Offense count:' .rubocop_todo.yml | wc -l
```

# A Framework for Tech. Debt

```
# extract number of violations
$ grep 'Offense count:' .rubocop_todo.yml | \
    ruby -e 'puts $stdin.readlines.sum { it.split.last.to_i }'
```

# A Framework for Tech. Debt

---

## 1. Get the Numbers

...and store them away

# A Framework for Tech. Debt

---

Store in Database 😕

Create a File per Day in a  
Bucket



# A Frameowk for Tech. Debt

```
# lib/tasks/tech_debt.rake
namespace :tech_debt do
  desc 'Record development metrics'
  task metrics: :environment do
    total_lines = `wc -l .rubocop_todo.yml`.strip.split.first.to_i
    File.write("tmp/#{Date.today}.json", { total_lines: }.to_json)
  end
end
```

# A Framework for Tech. Debt

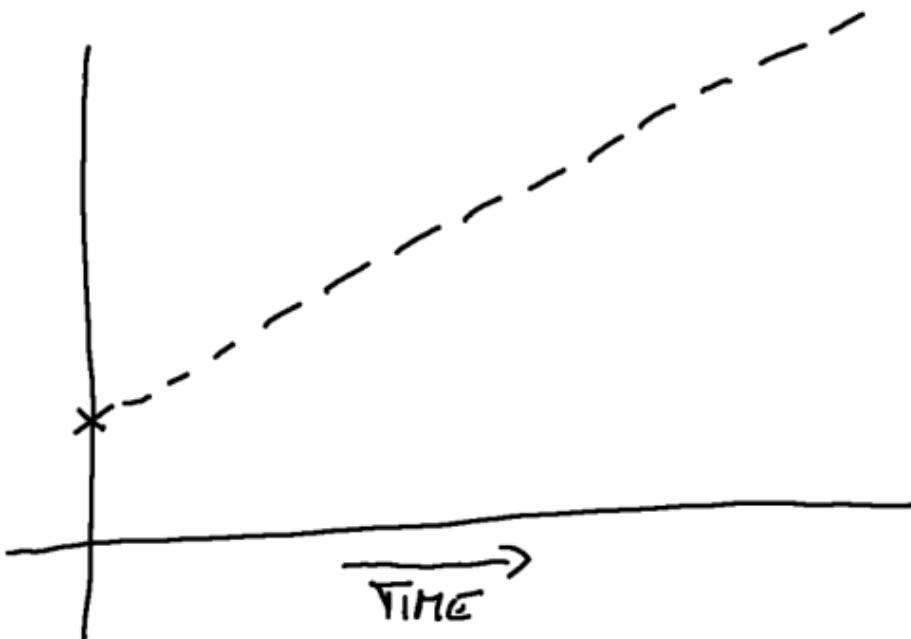
```
$ cat data/2025-06-23.json
{
  "rubocop": {
    "total_lines_of_todo_file": 1145,
    "number_of_rules_violated": 74,
    "number_of_violations": 2128
  }
}
```

# A Framework for Tech. Debt

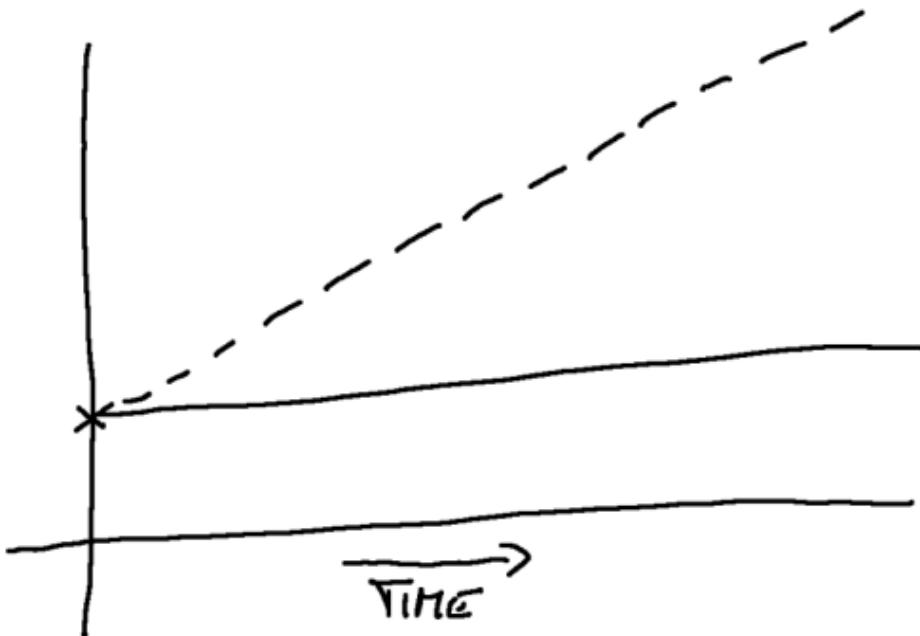
---

1. Get the Numbers 
2. Visualize Numbers 

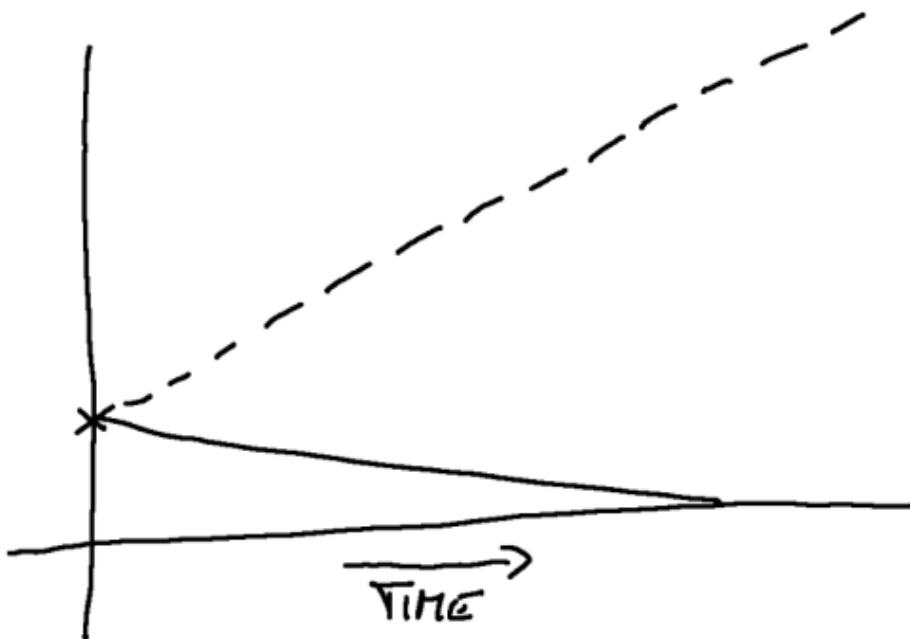
# A Framework for Tech. Debt



# A Framework for Tech. Debt



# A Framework for Tech. Debt



# A Framework for Tech. Debt

---

1. Get the Numbers 
2. Visualize Numbers 
3. Create a Vision 

## Vision for RuboCop



Good Practice

# A Framework for Tech. Debt

Benefits 💪

- \* Ruby Practice & Learning
- \* Readability
- \* Consistency
- \* Collaboration
- \* Maintainability
- \* Code Complexity
- \* Performance
- \* Security



## Vision for RuboCop



Good Practice



Less Problems

## Vision for RuboCop



Good Practice



Less Problems



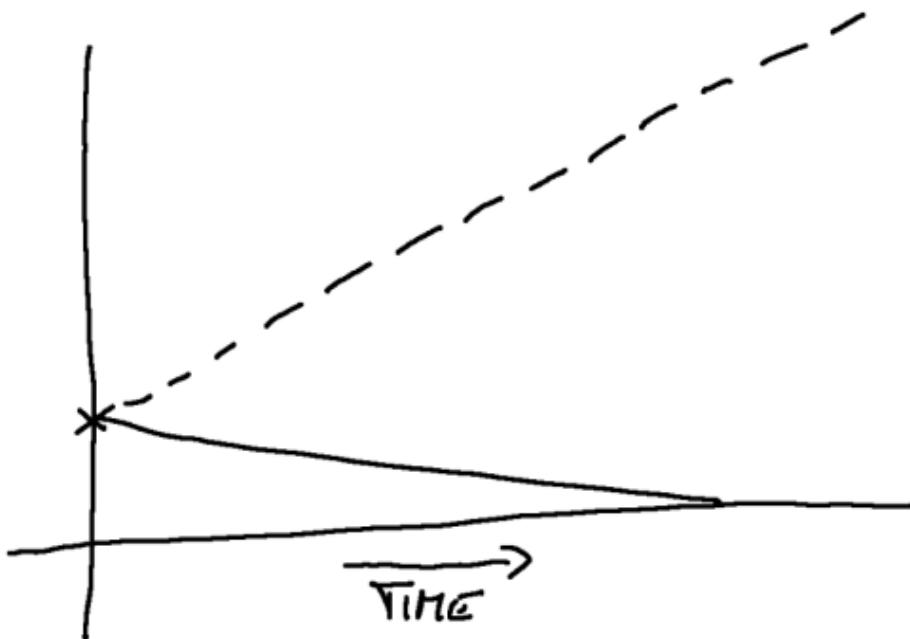
Easy Changes

# A Framework for Tech. Debt

---

1. Get the Numbers 
2. Visualize Numbers 
3. Create a Vision 
4. Make Steps Towards the Vision 

# A Framework for Tech. Debt



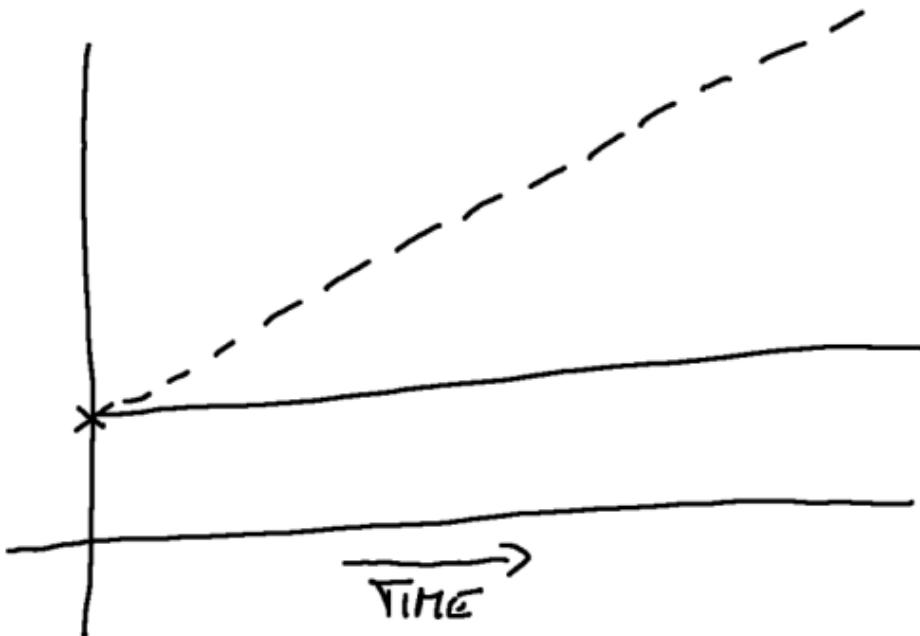
# A Framework for Tech. Debt

---

## Extend our Playbook:

- On Additions: Accept No Violations.
- On Changes: Refactor When Possible.

# A Framework for Tech. Debt



“We don’t need to fix  
everything.”

# A Framework for Tech. Debt

---

1. Get the Numbers 
2. Visualize Numbers 
3. Create a Vision 
4. Make Steps Towards the Vision 

## Impact on Communication

# Technical Debt for Developers

# Software Rot for Technical and Non-Technical People

# **Business Risk for Non-Technical People**

# Applications of the Framework

---



... but there's so many  
kinds of Tech. Debt

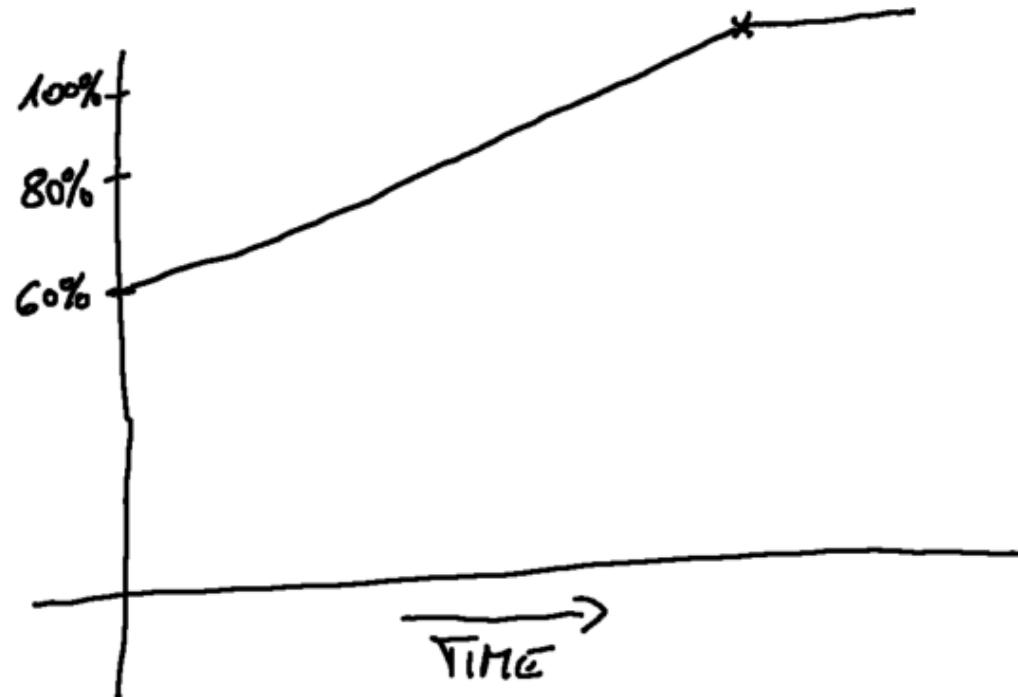
# Applications of the Framework

---

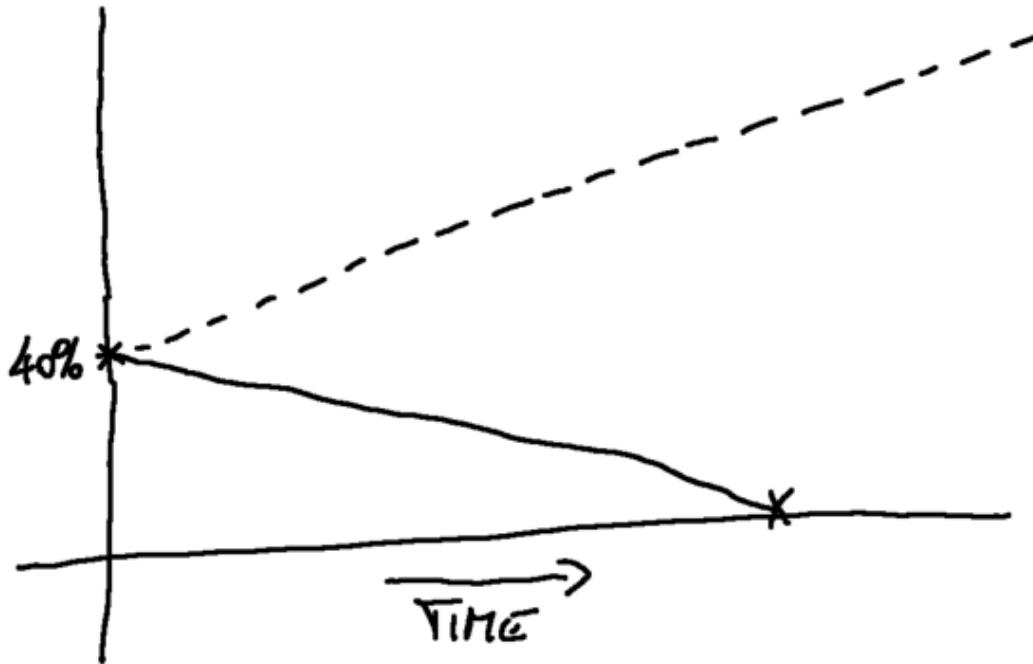


## Low Test Coverage

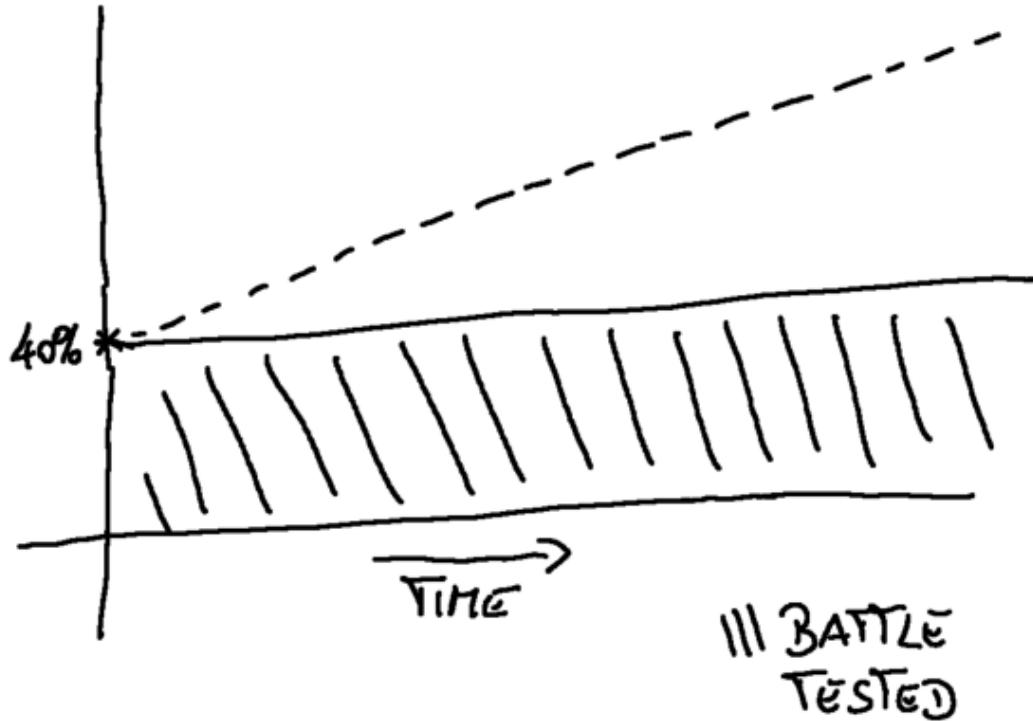
# Applications of the Framework



# Applications of the Framework



# Applications of the Framework



# Applications of the Framework

---



RubyGem  
Dependencies?

# Applications of the Framework

```
# List installed gems with newer versions available
$ bundle outdated --help
$ bundle outdated --filter-major --only-explicit | wc -l
```

# Applications of the Framework

---



Replace a Service?

# Applications of the Framework

---

```
$ grep LegacyService::Client | wc -l
```

Strong Coupling  
& Low Cohesion?

# Applications of the Framework

```
$ packwerk # check for violations
$ packwerk update-todo
```

# A Framework for Tech. Debt

1. Get the Numbers 
2. Visualize Numbers 
3. Create a Vision 
4. Make Steps Towards the Vision 

<https://github.com/unused/a-framework-for-tech-debt/>