“One Man” Development Process Model

Đinh Quang Trung
29 November 2015
Barcamp Saigon 2015

“One Man” Development Process Model

Đinh Quang Trung
Barcamp Saigon 2015

...or “Best way to work on a project by your own”

About me

Đinh Quang Trung
Code Writer

- Writing code since 2009
- Graduated FPT University 3 months ago
- Web Developer at Silicon Straits Saigon

Fpt University

SILICON STRAITS SAIGON

fb.com/trungdq88

@trungdq88

github.com/trungdq88

Overview

1. How did I manage my projects when I was “young”
2. Problems with my process
3. I learned I know how to manage my projects correctly
4. Tools and techniques I used for the process
5. Tips

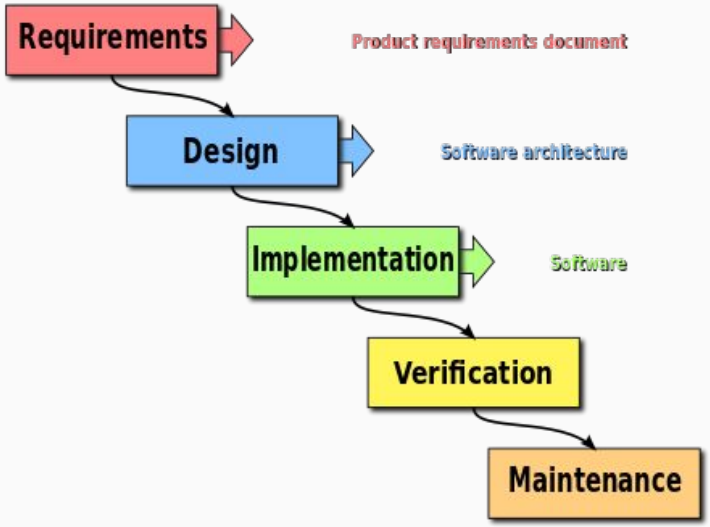
When I check an old code of mine

This presentation targets

People who

- Want to be a developer
- Want to be a better developer
- Want to understand developers

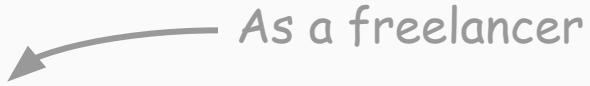
Software Development Process Models



But…
what if you only have one?

Project types

Project types



- You do a project for a customer

Project types

As a freelancer

- You do a project for a customer

lasts 1 - 2 weeks

- You do a small project for your own

Project types

- You do a project for a customer

As a freelancer

- You do a small project for your own

lasts 1 - 2 weeks

- You do a **BIG** project for your own

lasts longer than 1 month
and could potentially
become your startup!

Common approaches

1

Requirement

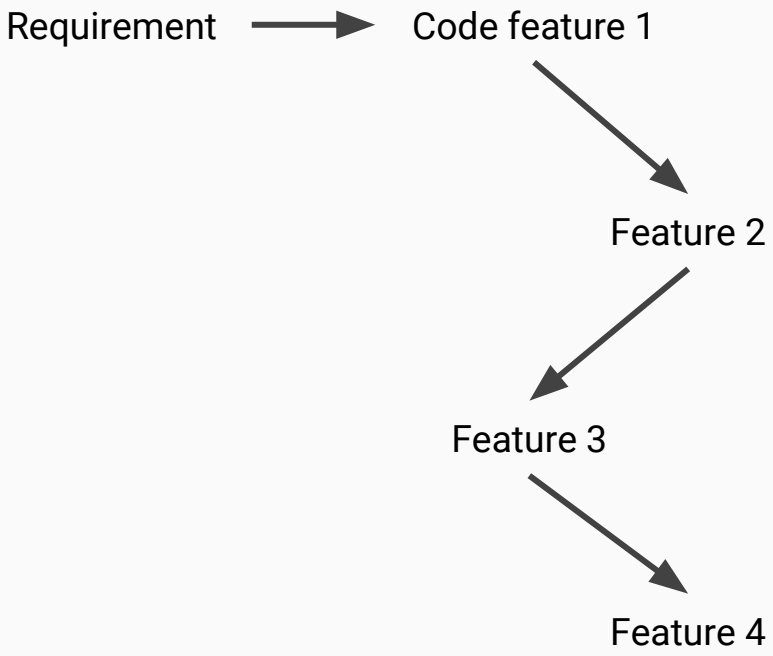
Common approaches

1

Requirement → Code feature 1

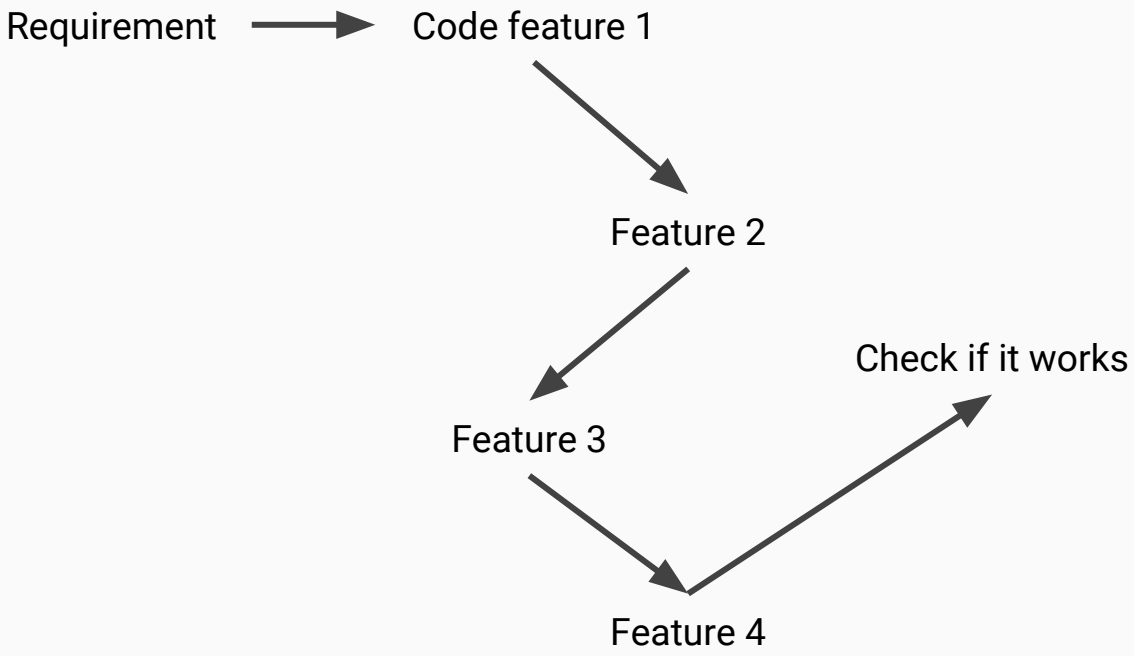
Common approaches

1



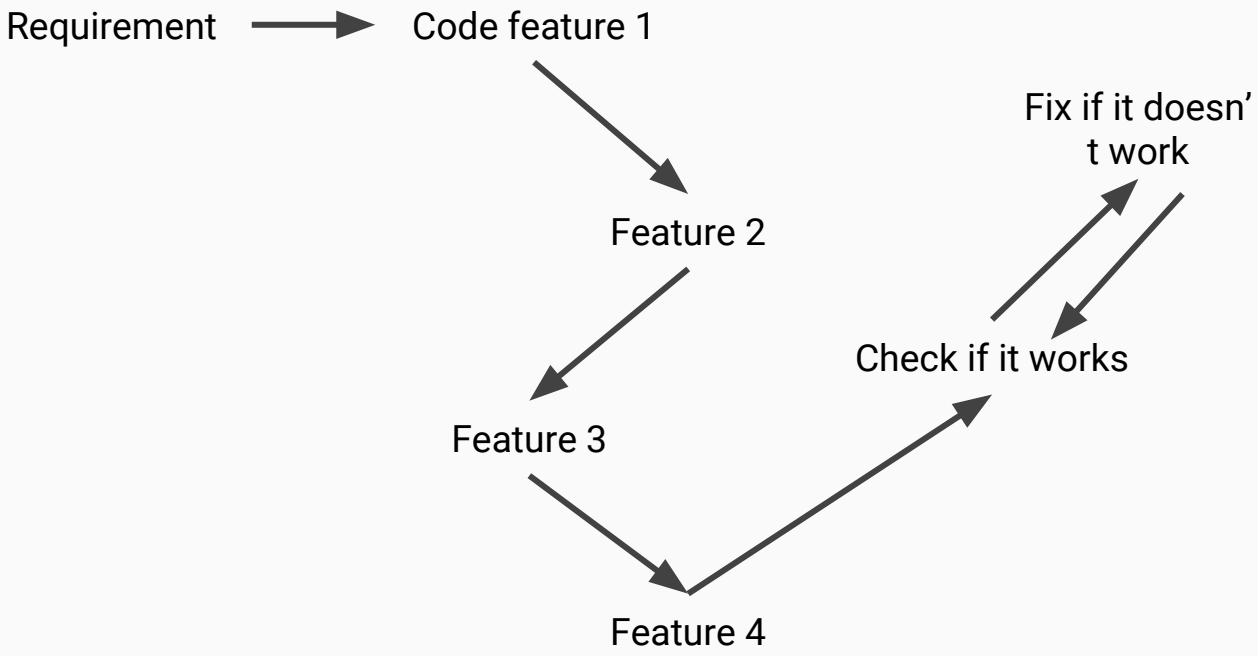
Common approaches

1



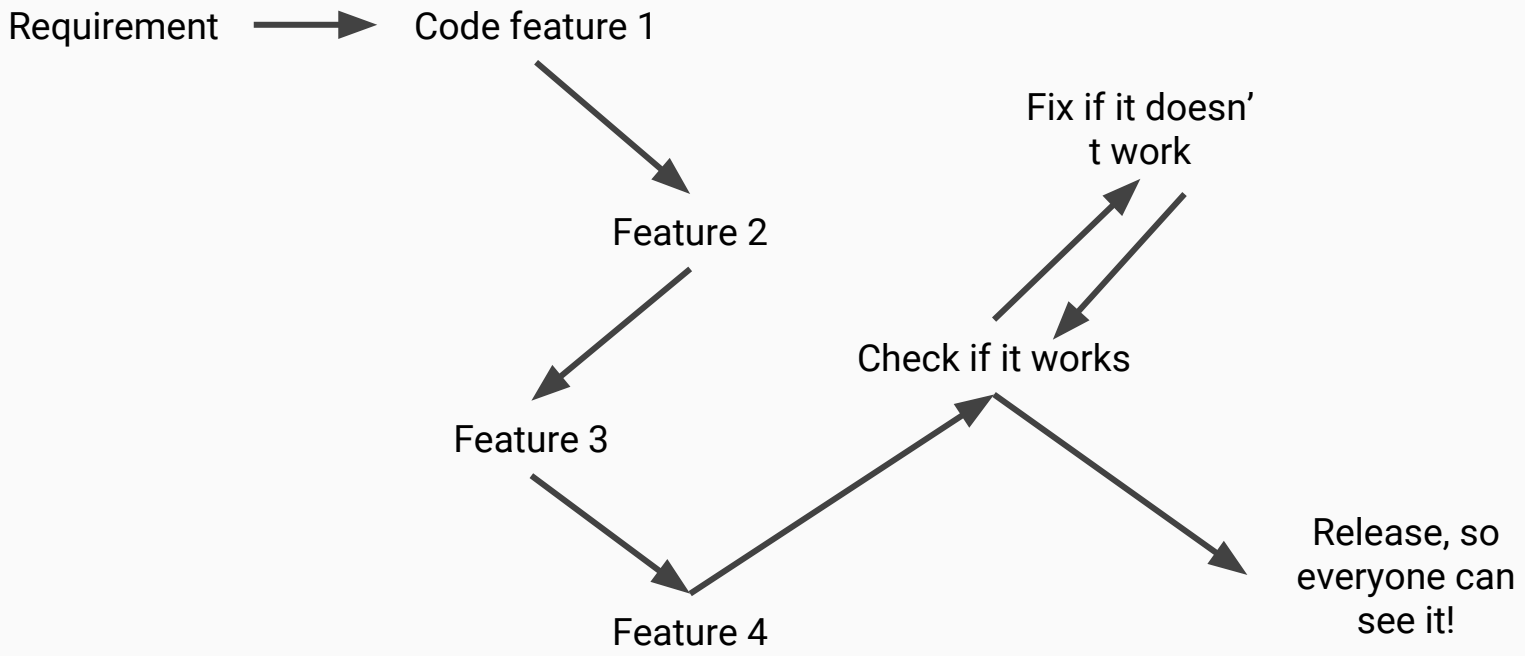
Common approaches

1



Common approaches

1



Common approaches

1

Problems?

Common approaches

1

- Codebase is crap

Common approaches

1

- Codebase is crap
- Fear of change

Common approaches

1

- Codebase is crap
- Fear of change
- It becomes worse when project have more features

Common approaches

1

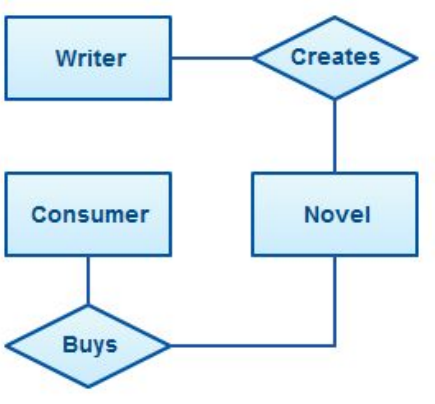
- Codebase is crap
- Fear of change
- It becomes worse when project have more features
- No one else can understand my code

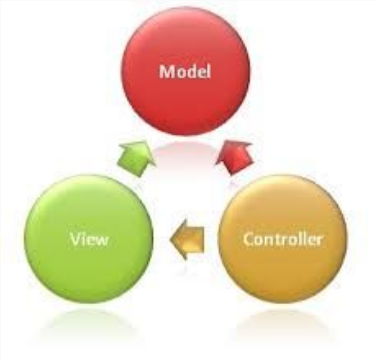
Common approaches

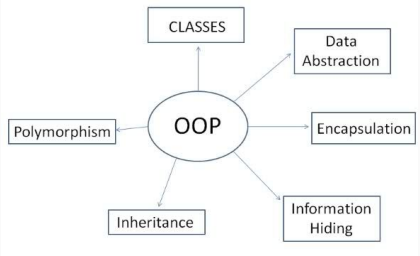
1

- Codebase is crap
- Fear of change
- It becomes worse when project have more features
- No one else can understand my code
- I don’t understand my code

So. I learned


Writer
Creates
Consumer
Novel
Buys


Model
View
Controller


CLASSES
Data Abstraction
Polymorphism
OOP
Encapsulation
Inheritance
Information Hiding


Abstract
Behavioral
Singleton
Template
Interpreter
Class
Command
Chain
Composite
Proxy
Structural
Decorator
Bridge
Design
Diagram
Factory
Visitor
Object
agile
UNIX
Facade
Strategy
Creational
Windows
Memento
Iterator
Builder
Flyweight
Adapter
Patterns
Interaction
Method
State
Prototype

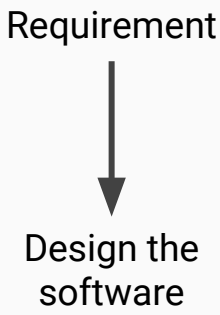
Common approaches

2

Requirement

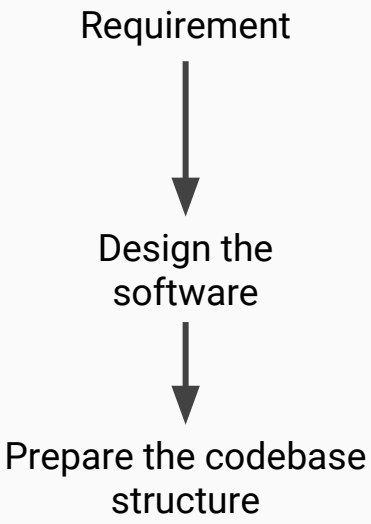
Common approaches

2



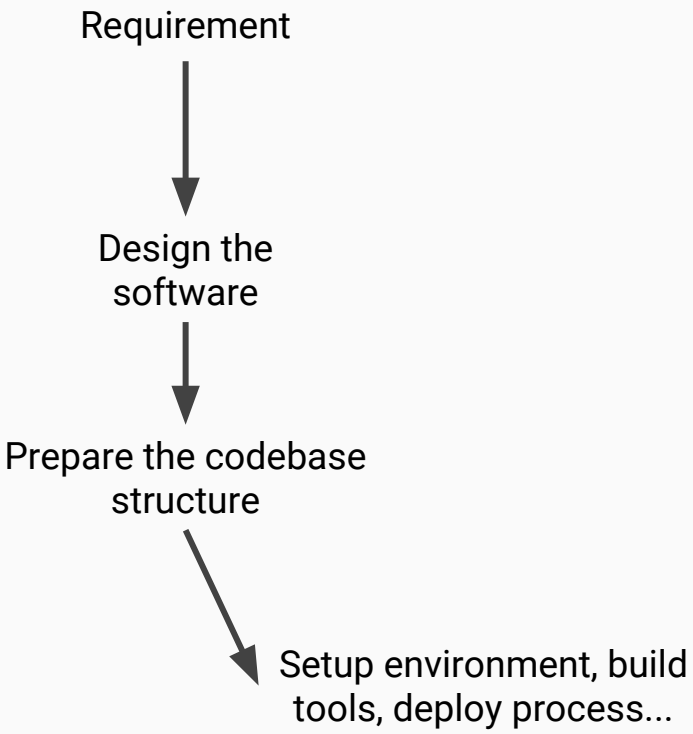
Common approaches

2



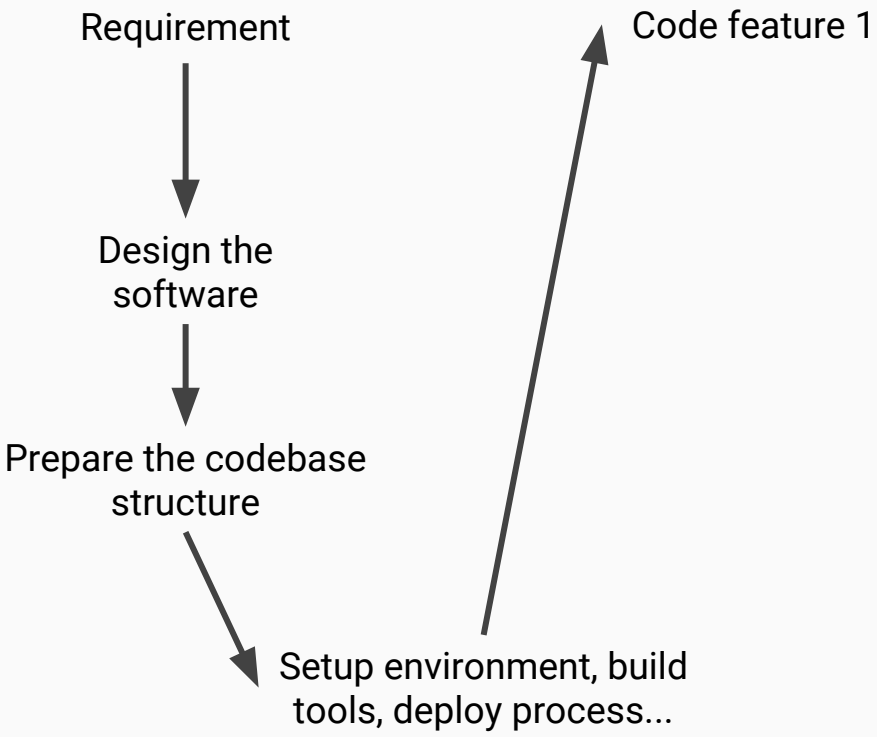
Common approaches

2



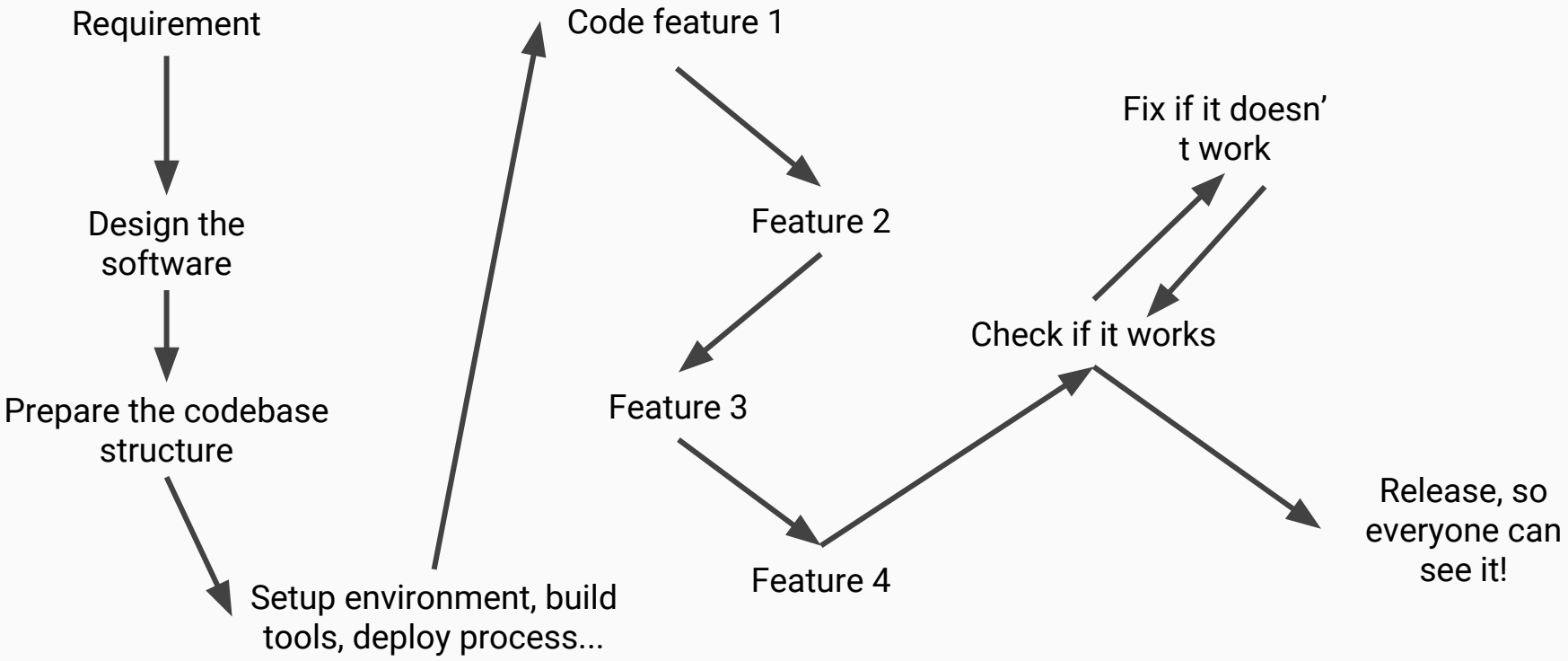
Common approaches

2



Common approaches

2



Common approaches

2

Still has problems

Common approaches

2

- Codebase is better

Common approaches

2

- Codebase is better
- Less fear of change

Common approaches

2

- Codebase is better
- Less fear of change
- Slow release

Common approaches

2

- Codebase is better
- Less fear of change
- Slow release
- Sometimes I prepare things that I never use

Common approaches

2

- Codebase is better
- Less fear of change
- Slow release
- Sometimes I prepare things that I never use
- Take too much time to test features

Common approaches

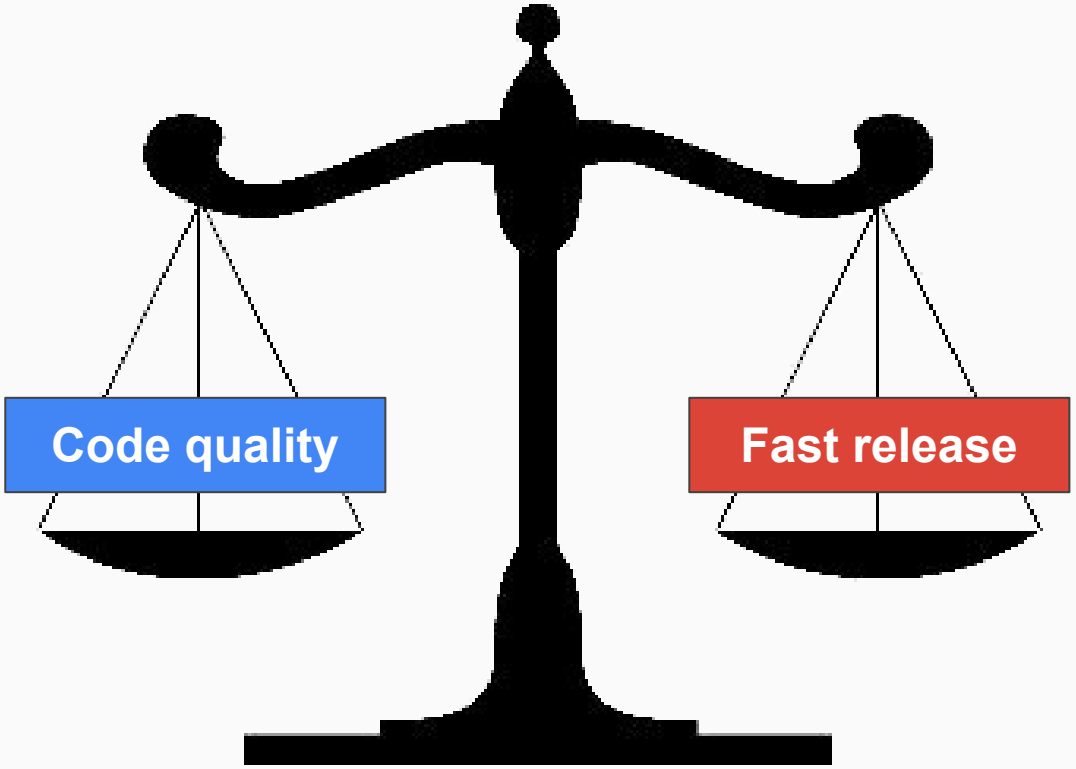
2

- Codebase is better
- Less fear of change
- Slow release
- Sometimes I prepare things that I never use
- Take too much time to test features
- Still no one understand my code

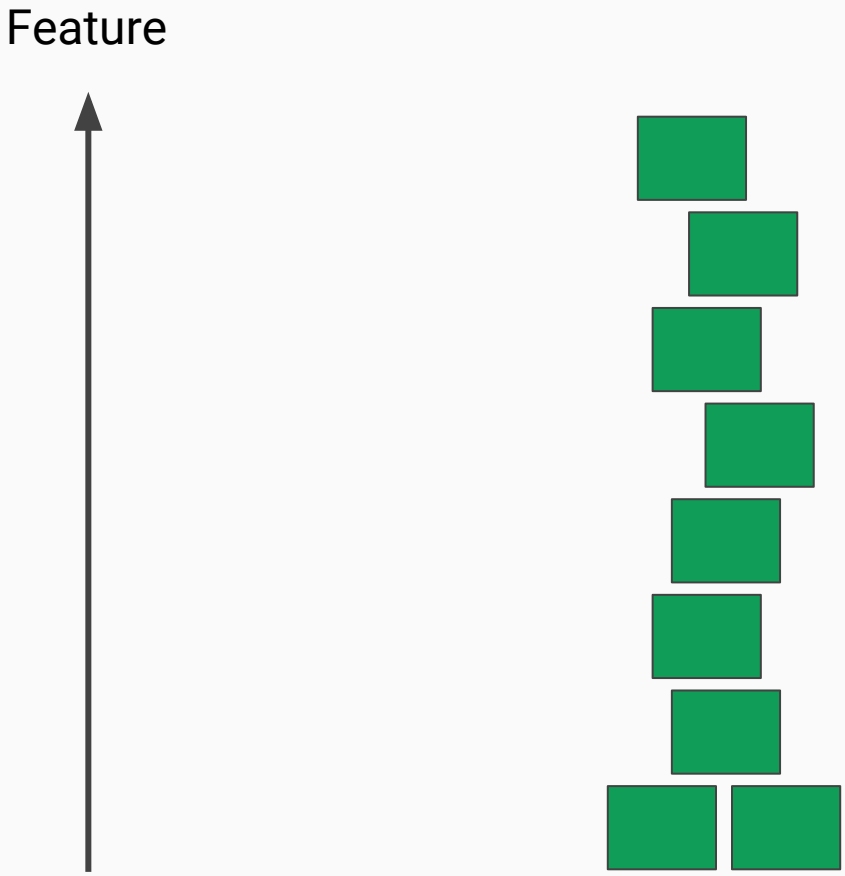
:-(

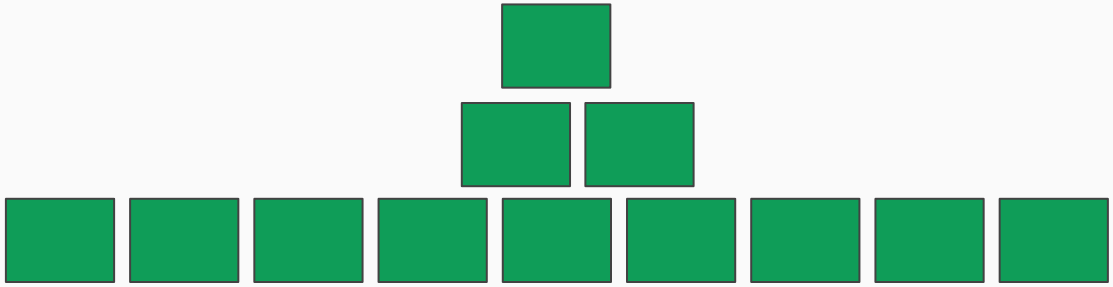
What we want?

- A beautiful codebase that you proud of and **want to work on it** in the future.
- Well documented for anyone who want to **join the project**
- Quality control for your project so that you can continue to develop without the fear of **breaking something accidently**
- **Stable release** schedule, well defined **features** and **versions**

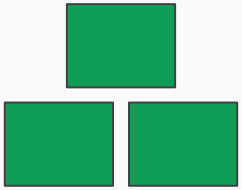

Code quality
Fast release

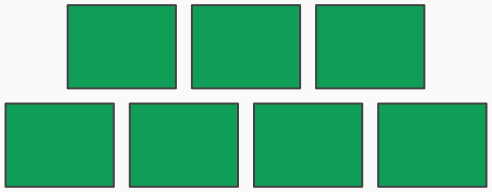
Project grows

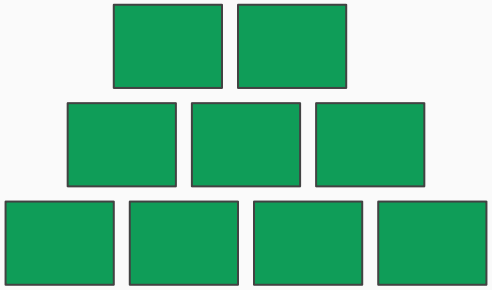

Feature

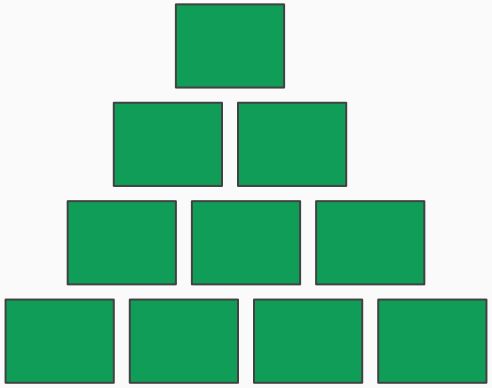


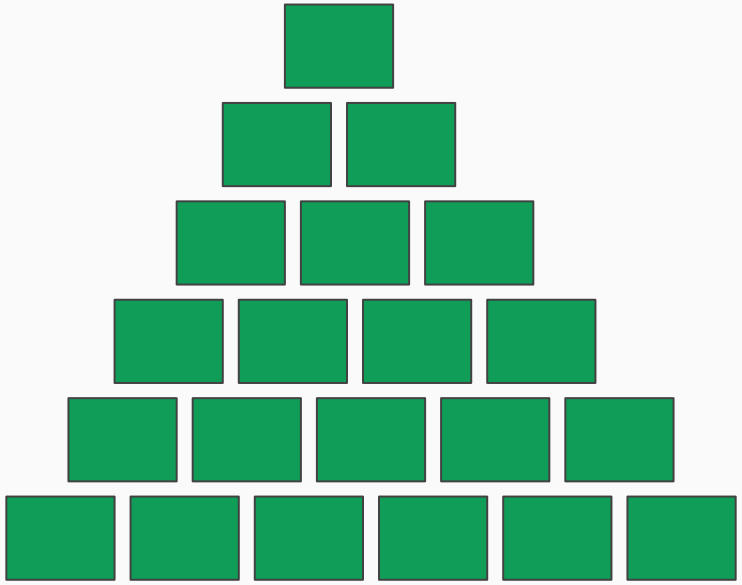
What we want

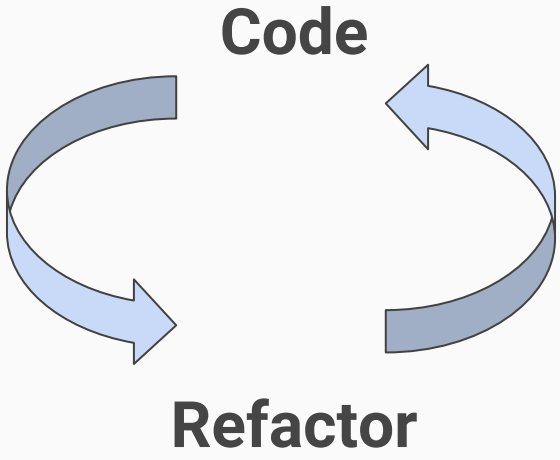











Code
Refactor

Slow release?

Break your project into versions

1.0.0

- Feature 1
- Feature 2
- Feature 3

1.1.0

- Feature 4
- Feature 5
- Feature 6

1.2.0

- Feature 7:
  - Smaller feature 1
  - Smaller feature 2
  - Smaller feature 3
  - Smaller feature 4
  - Smaller feature 5
- Feature 8
- Feature 9

No one understand my code?

Write documents

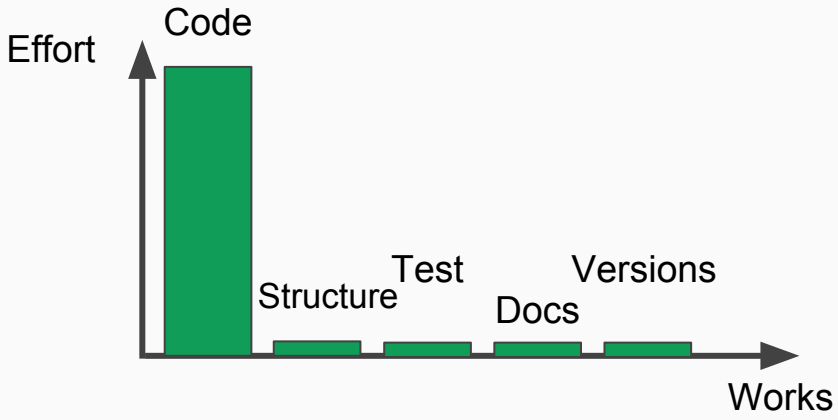
- Comment in your code
- Explain your code structure, architecture…
- Draw diagrams
- Write instruction for setup steps, deployment process, tools used…

Write a lot of tests

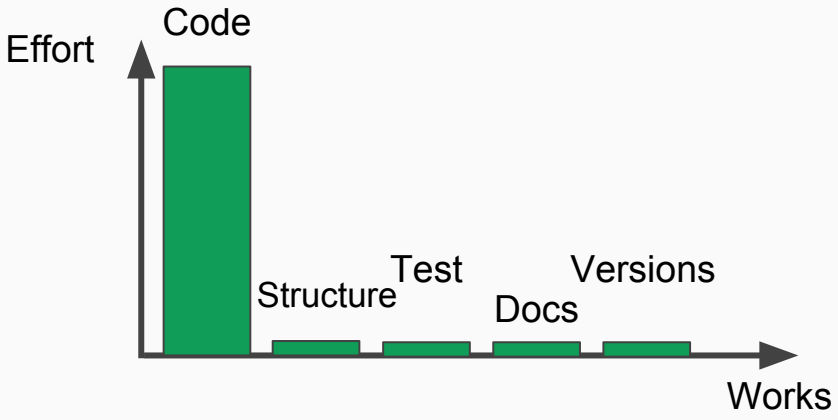
- **Tests** is the only thing can make sure that you don't **accidently break something** when changing/refactoring your source code

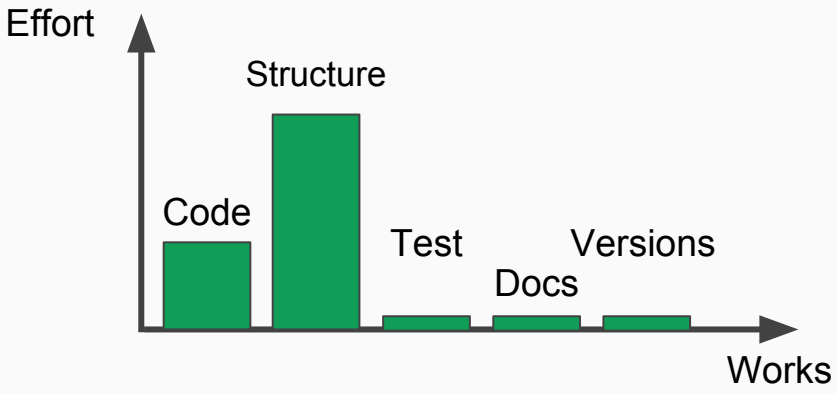
- Test techniques:
  - Unit tests
  - Automation tests

Project Quality

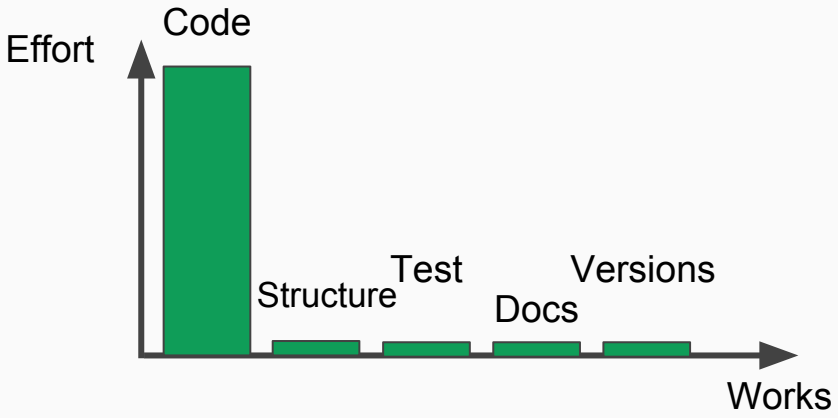


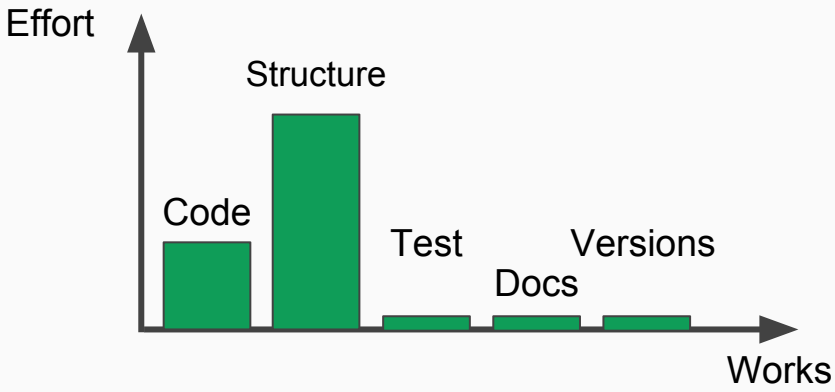
Project Quality

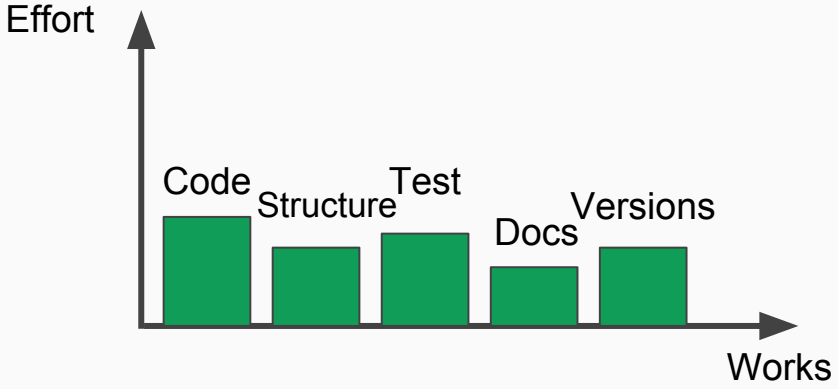




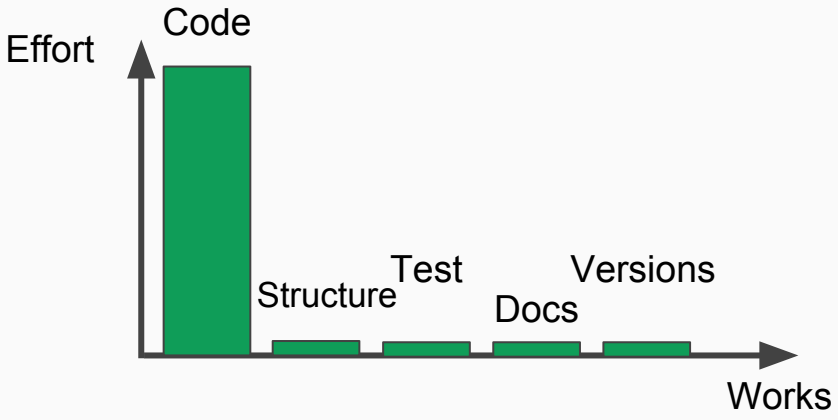
Project Quality

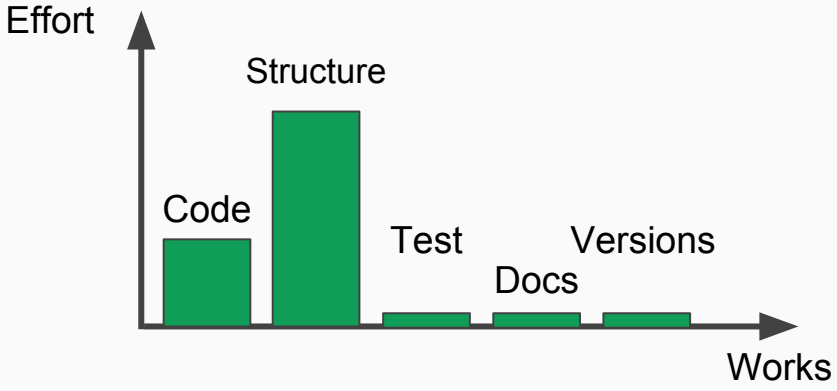


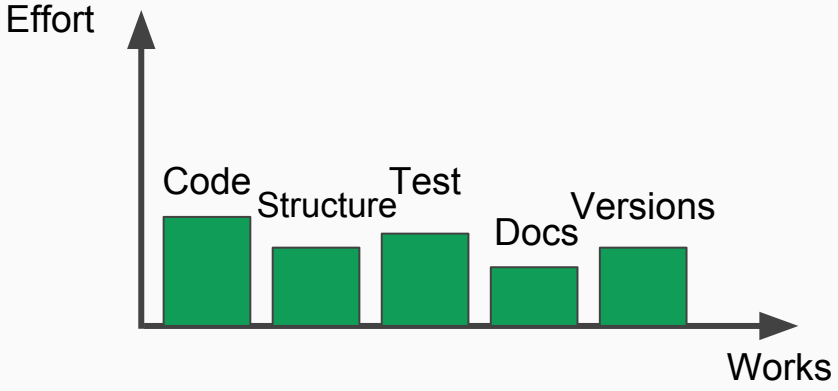


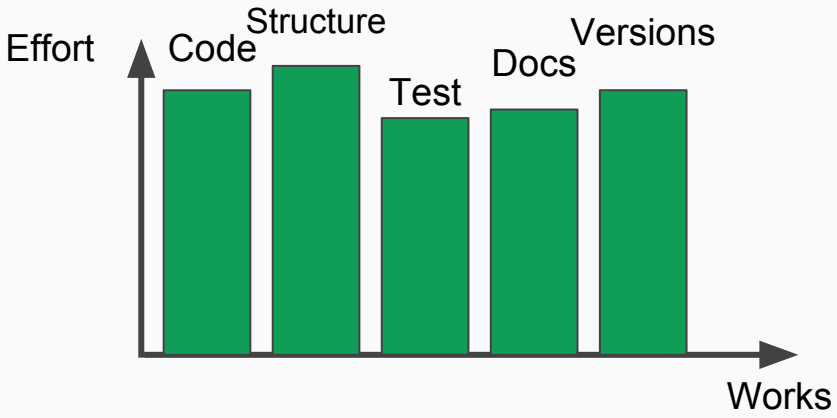


Project Quality







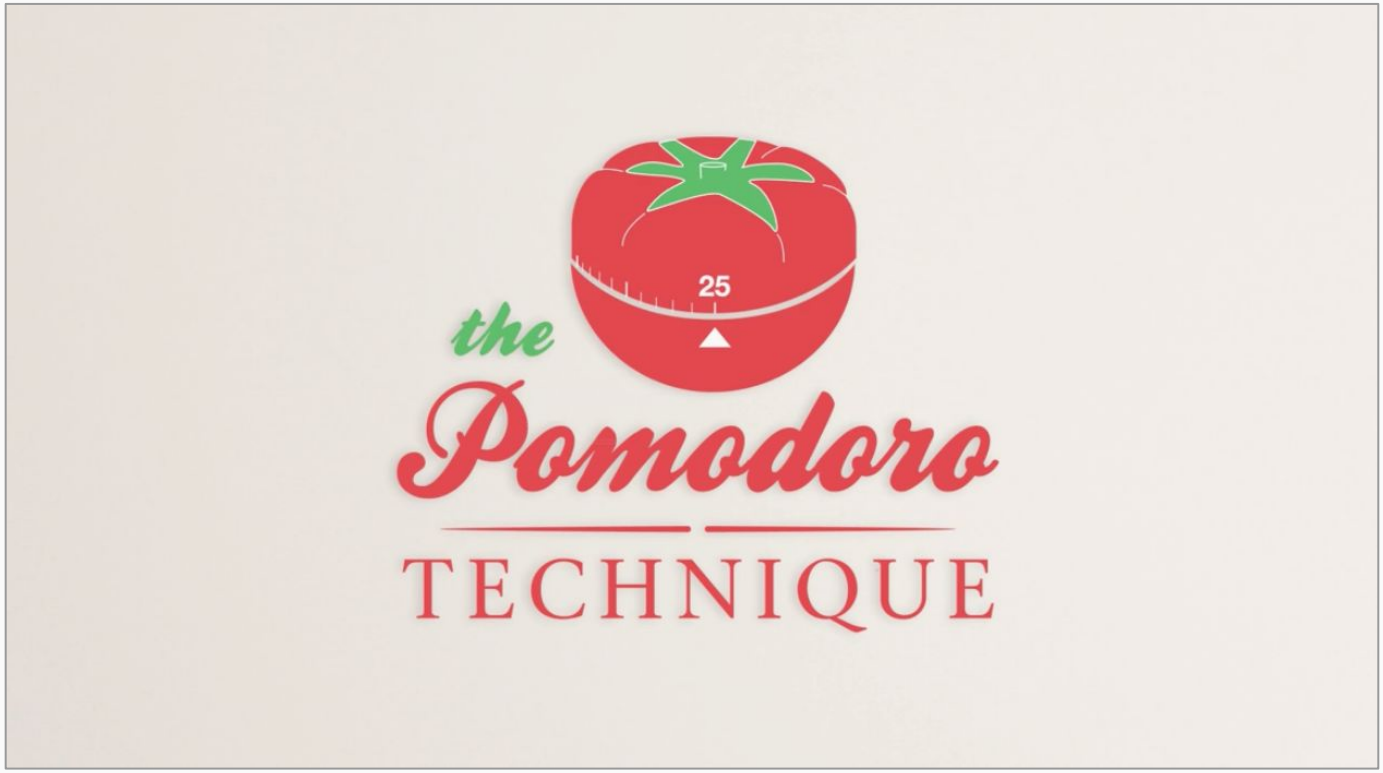


Use tools to help you

Use tools to help you

- **Source controls:** Git, SVN
- **Issue tracking:** Github, Bitbucket or even a todo list.
- **Documentation**: Google Docs, Github and Bitbucket also supports wiki for documents
- **Version management**: Milestone (in Github), Versions (Bitbucket)
- **Diagrams**: LucidChart
- **Test runner (or CI)**: Travis CI, Jenkins

The Pomodoro technique



http://pomodorotechnique.com/

The Pomodoro technique

My workflow 2

Having an idea

My workflow 2

Having an idea

Define all features

My workflow 2

Having an idea

Define all features

Break features into versions

My workflow 2

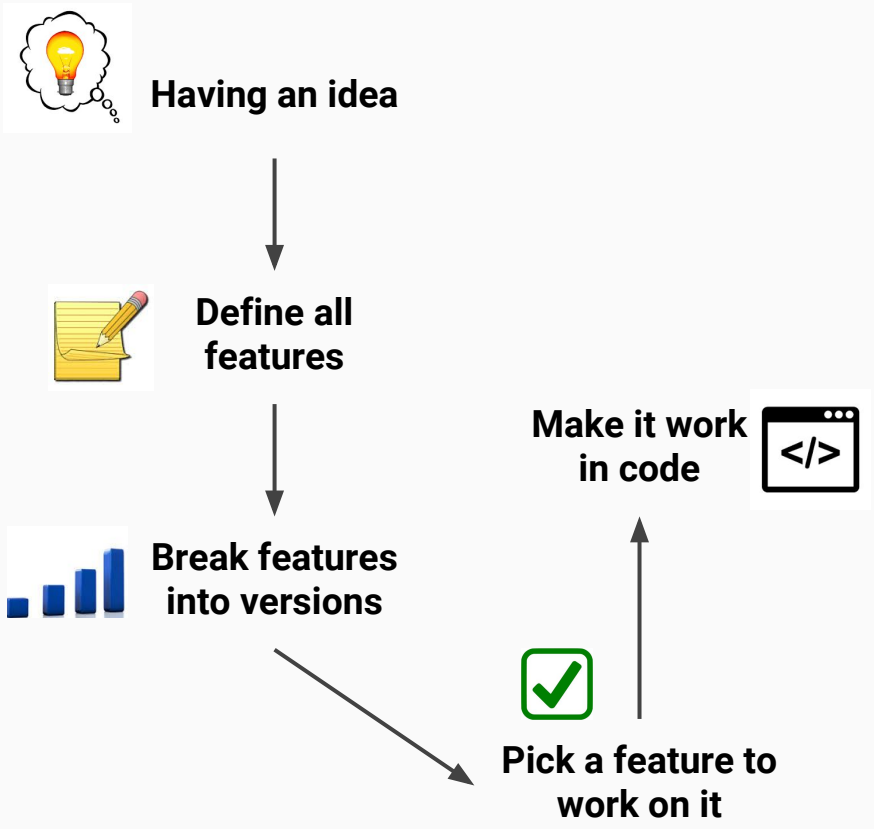
Having an idea

Define all features

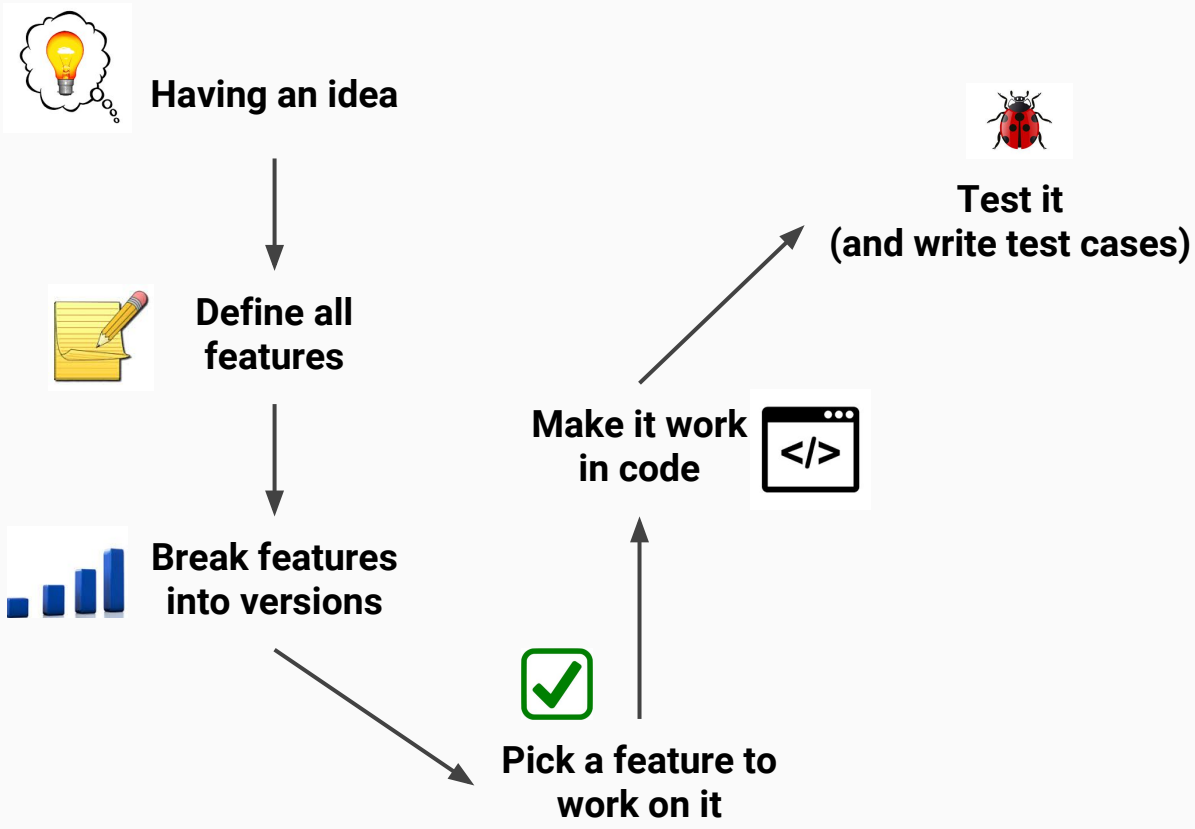
Break features into versions

Pick a feature to work on it

My workflow 2



My workflow 2



My workflow 2

Having an idea

Define all features

Break features into versions

Pick a feature to work on it

Make it work in code

Test it (and write test cases)

Refactor code (if needed)

My workflow 2

Having an idea

Define all features

Break features into versions

Pick a feature to work on it

Make it work in code

Test it (and write test cases)

Refactor code (if needed)

Write document (diagrams, setup guide, READMEs…)

My workflow 2

Having an idea

Define all features

Break features into versions

Pick a feature to work on it

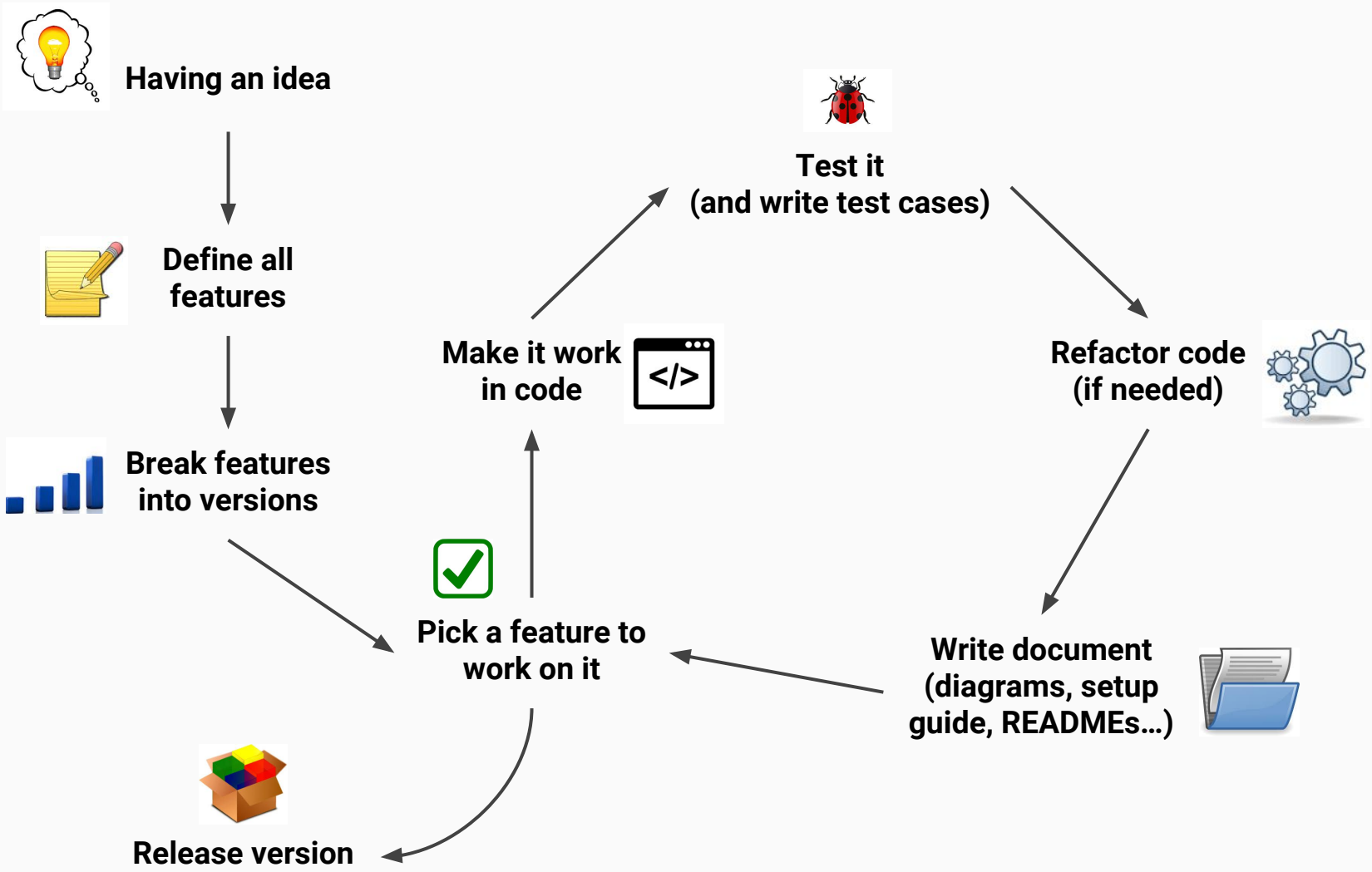
Make it work in code

Test it (and write test cases)

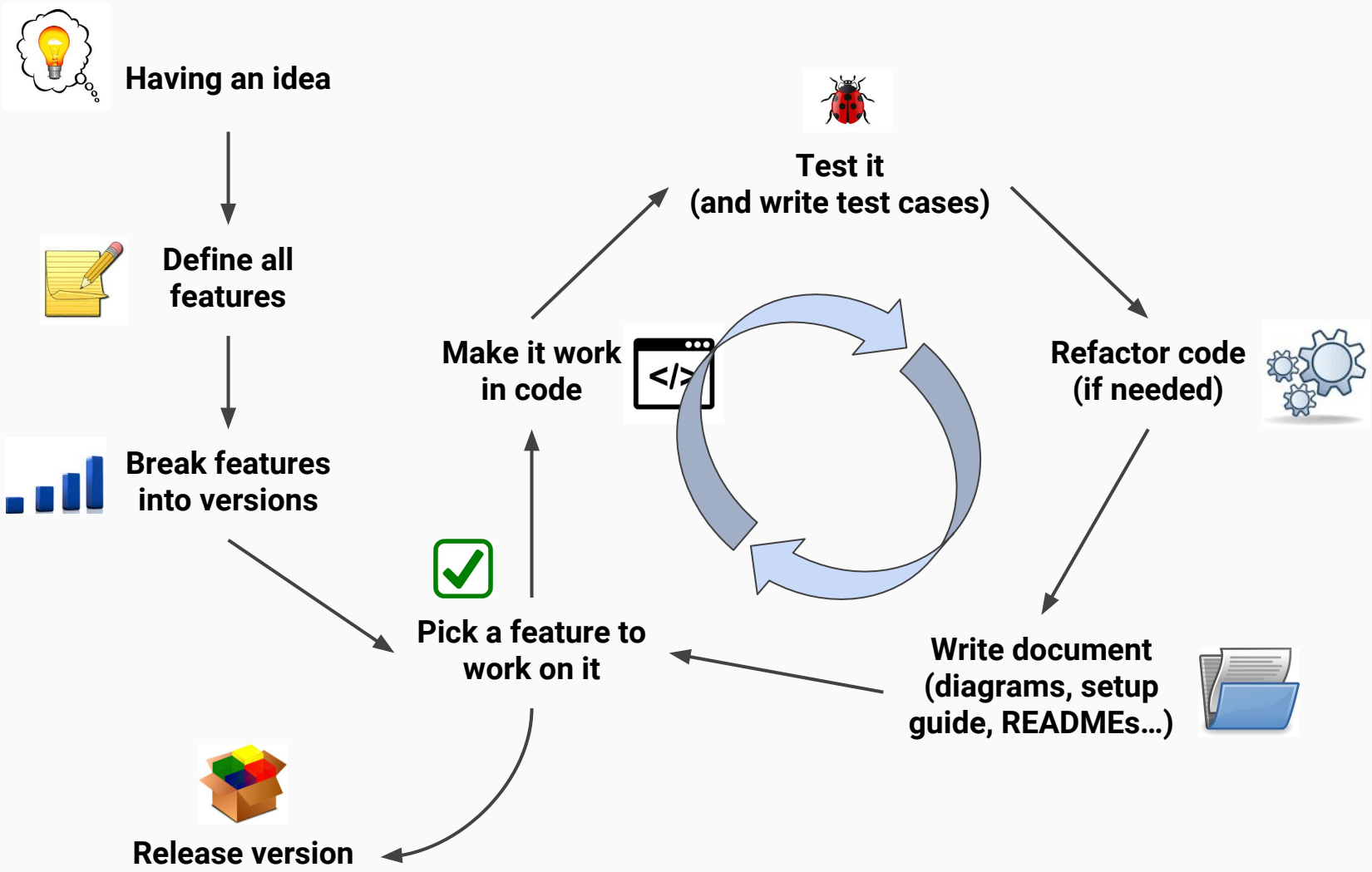
Refactor code (if needed)

Write document (diagrams, setup guide, READMEs…)

My workflow 2



My workflow 2



Tips to prepare your mind

Know your targets

Why are you doing this project?

YOU are the only one
who knows what’s going on in the code

Don’t let this happen

Make a project something that you proud of

- Executables (.apk, .war, .exe…)
- Source code
- Documents
- Diagrams
- Test cases
- …

Deliver whole project, not only the executables or the code

Be ready to welcome contributors!

Thank you


fb.com/trungdq88


@trungdq88


github.com/trungdq88