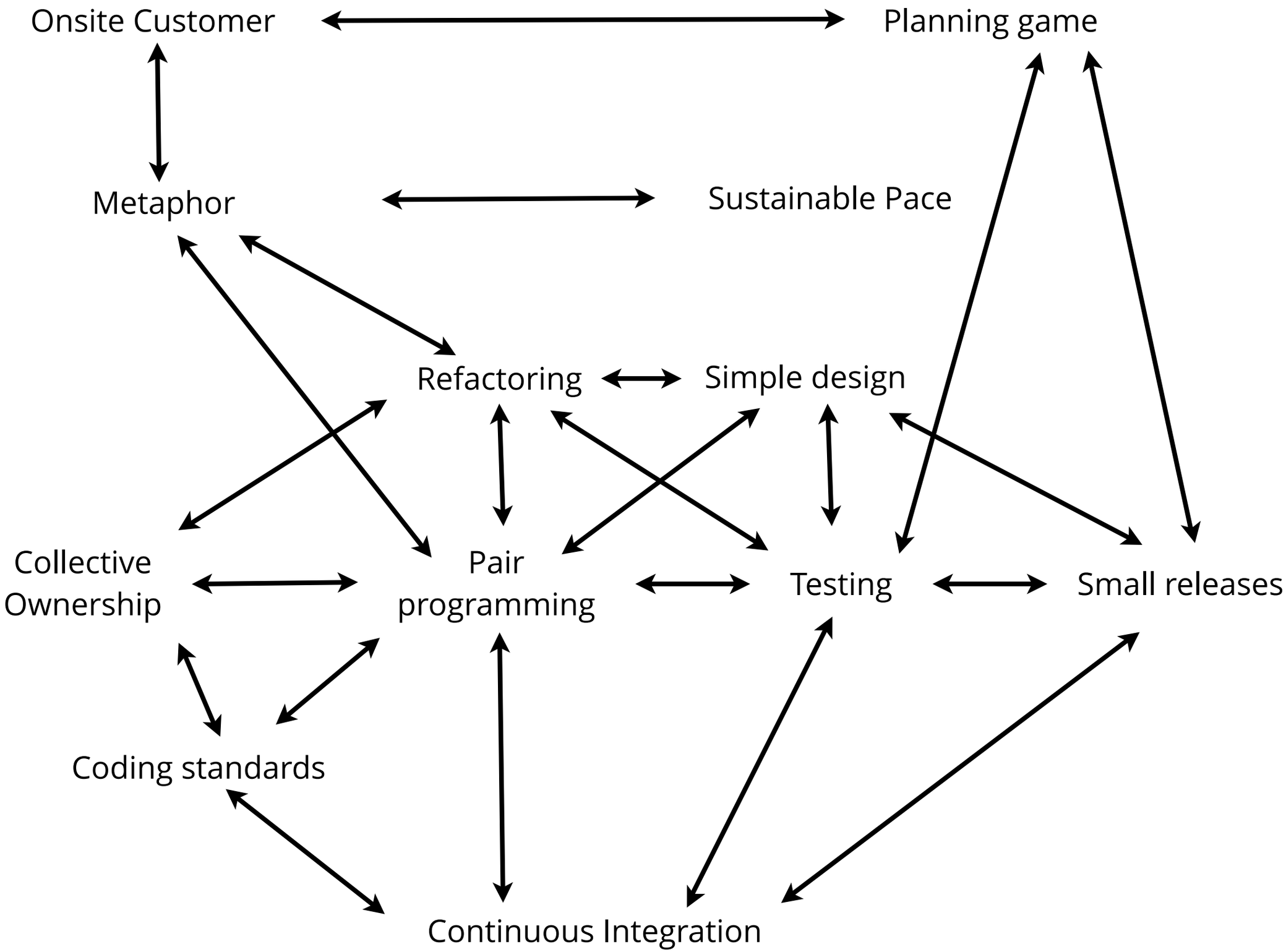
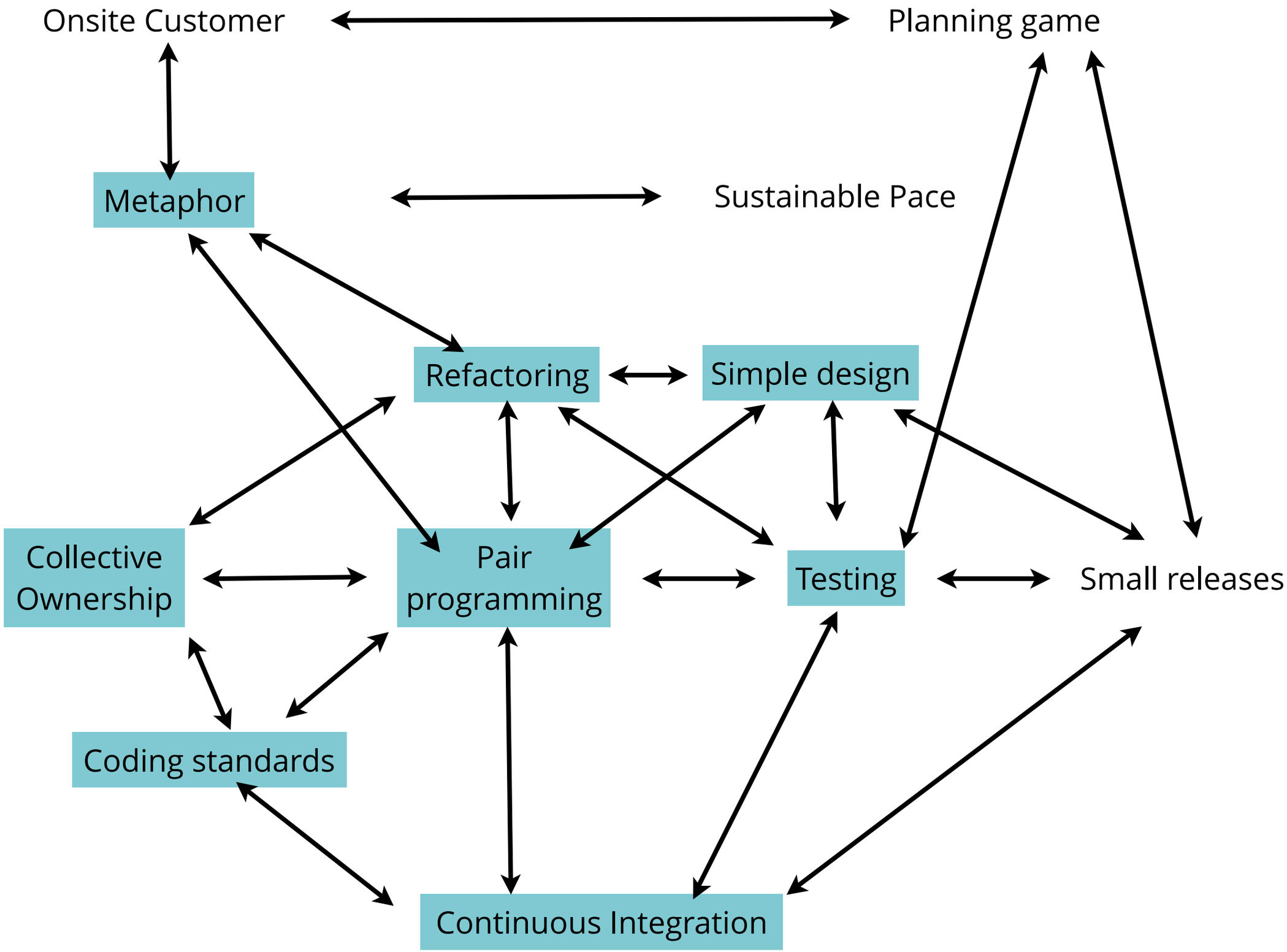
ThoughtWorks®

DEVELOPMENT PRACTICES

AGENDA

- Agile software development
- Test Driven Development
- Refactoring
- Evolutionary Architecture
- Continuous Integration
- Collective Ownership


Onsite Customer
Planning game
Metaphor
Sustainable Pace
Refactoring
Simple design
Collective Ownership
Pair programming
Testing
Small releases
Coding standards
Continuous Integration


Onsite Customer
Planning game
Metaphor
Sustainable Pace
Refactoring
Simple design
Collective Ownership
Pair programming
Testing
Small releases
Coding standards
Continuous Integration

TEST DRIVEN DEVELOPMENT

Write a failing test

Make it pass

Make it better

Red

Green

Refactor

TDD helps communicate intentions

TDD focuses on behaviour, not implementation

REFACTORING

Refactoring is a behaviour preserving transformation

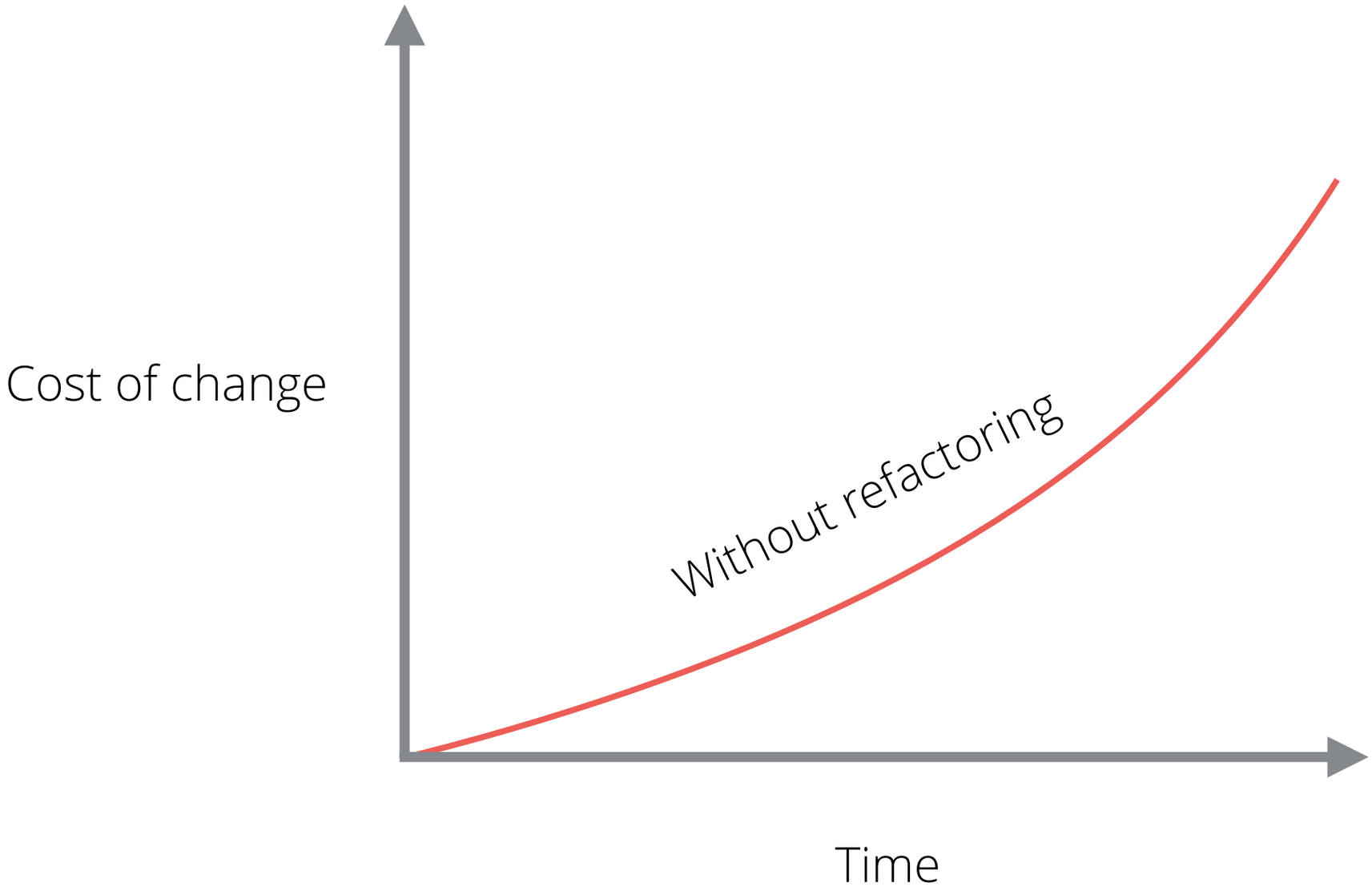
Functionally, the software is the same

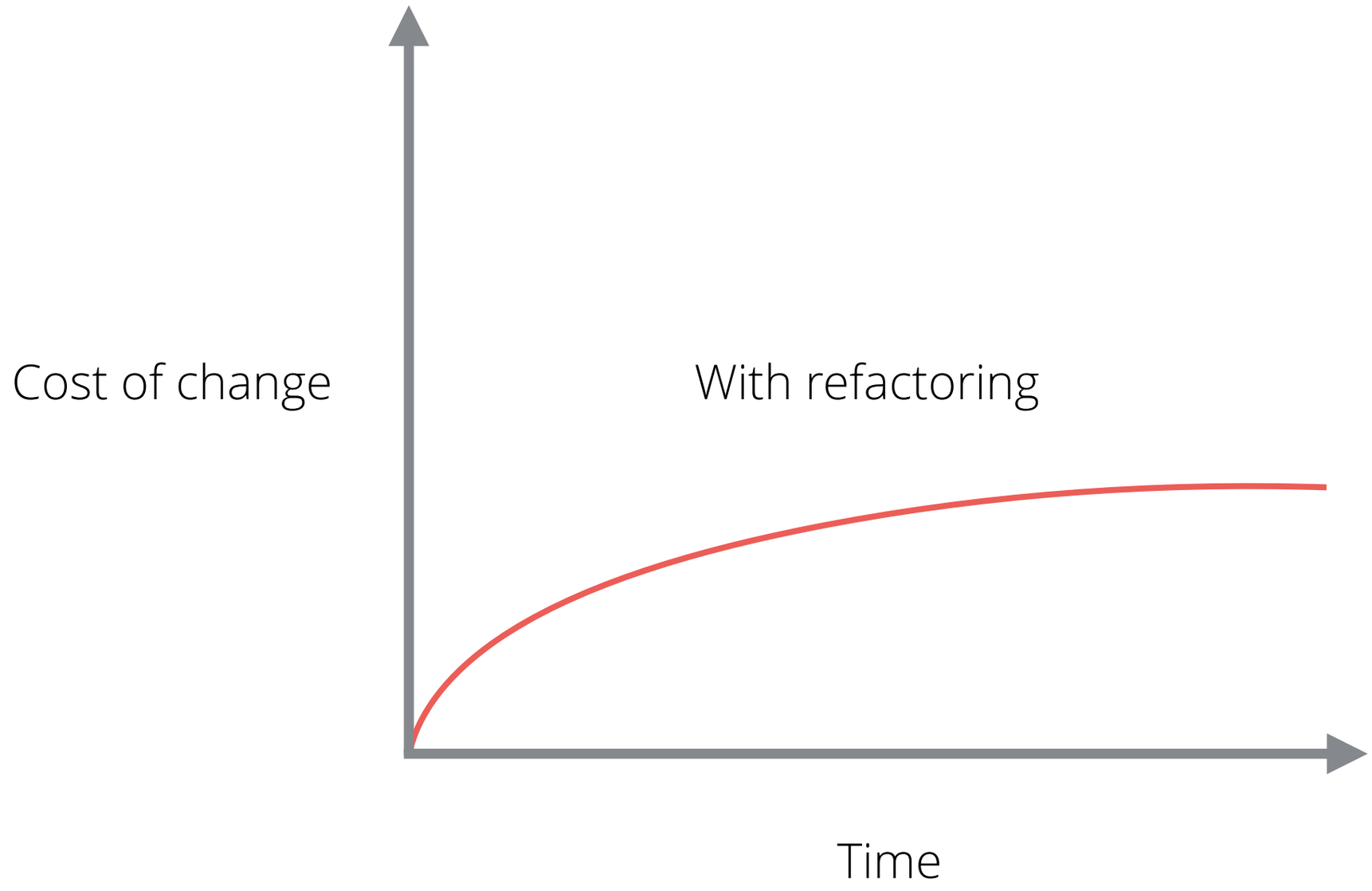
Refactoring improves

maintainability & extensibility

Refactoring vs redesign

WHY REFACTOR?


Cost of change
Without refactoring
Time


Cost of change
With refactoring
Time

WHEN TO REFACTOR

- Adding functionality
- Fixing a bug
- Doing a code review

WHEN NOT TO REFACTOR

- Code is too messy
- Near a deadline

DEMO

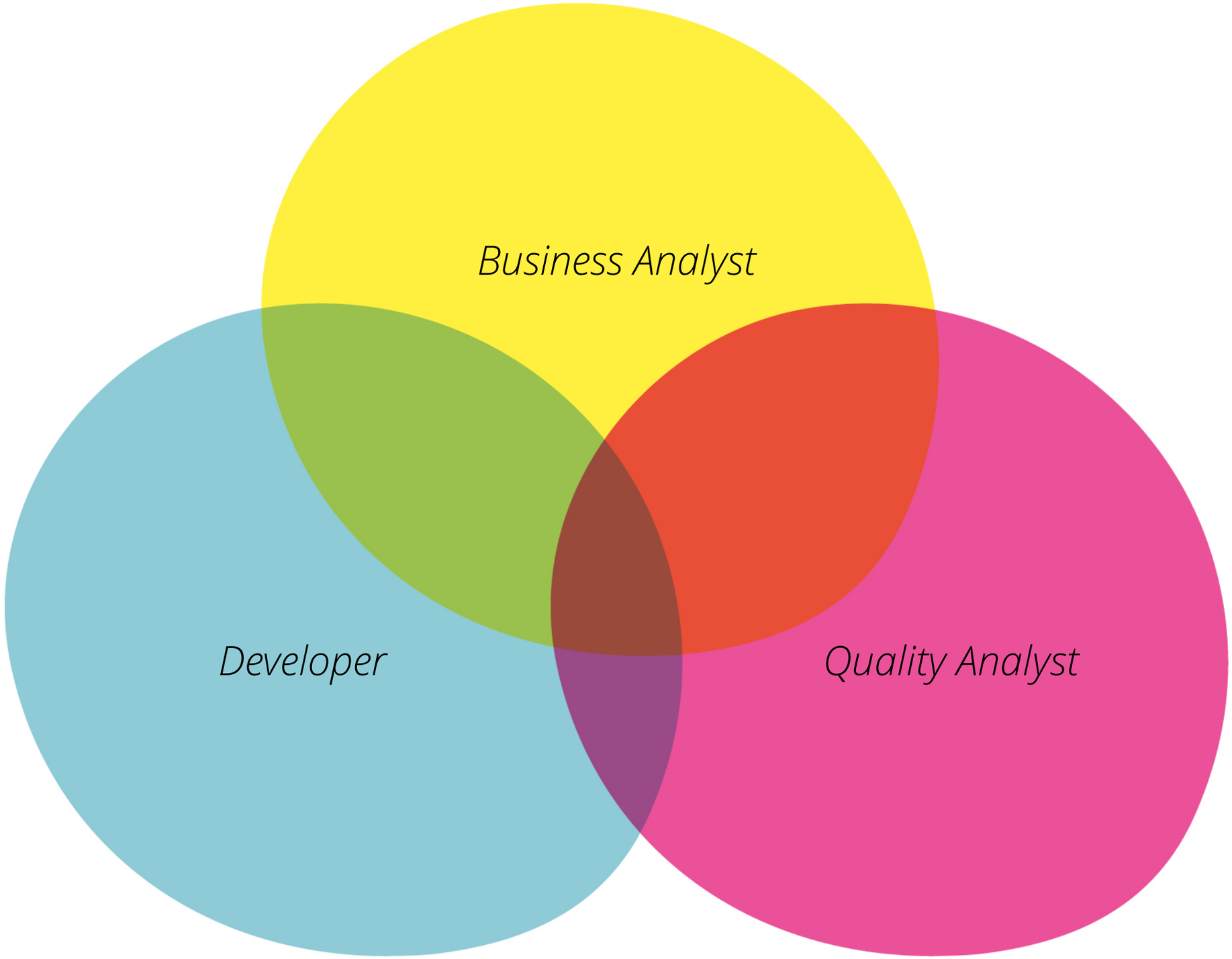
PAIR PROGRAMMING

WHY PAIR

- Fewer defects
- Less rework
- More creativity
- Easier to maintain

WHY PAIR

- Fewer defects
- Teaching / on boarding
- Less rework
- More creativity
- Easier to maintain


Business Analyst
Developer
Quality Analyst

PAIRING STYLES

- Ping-pong
- Driver Navigator

PAIRING MYTHS

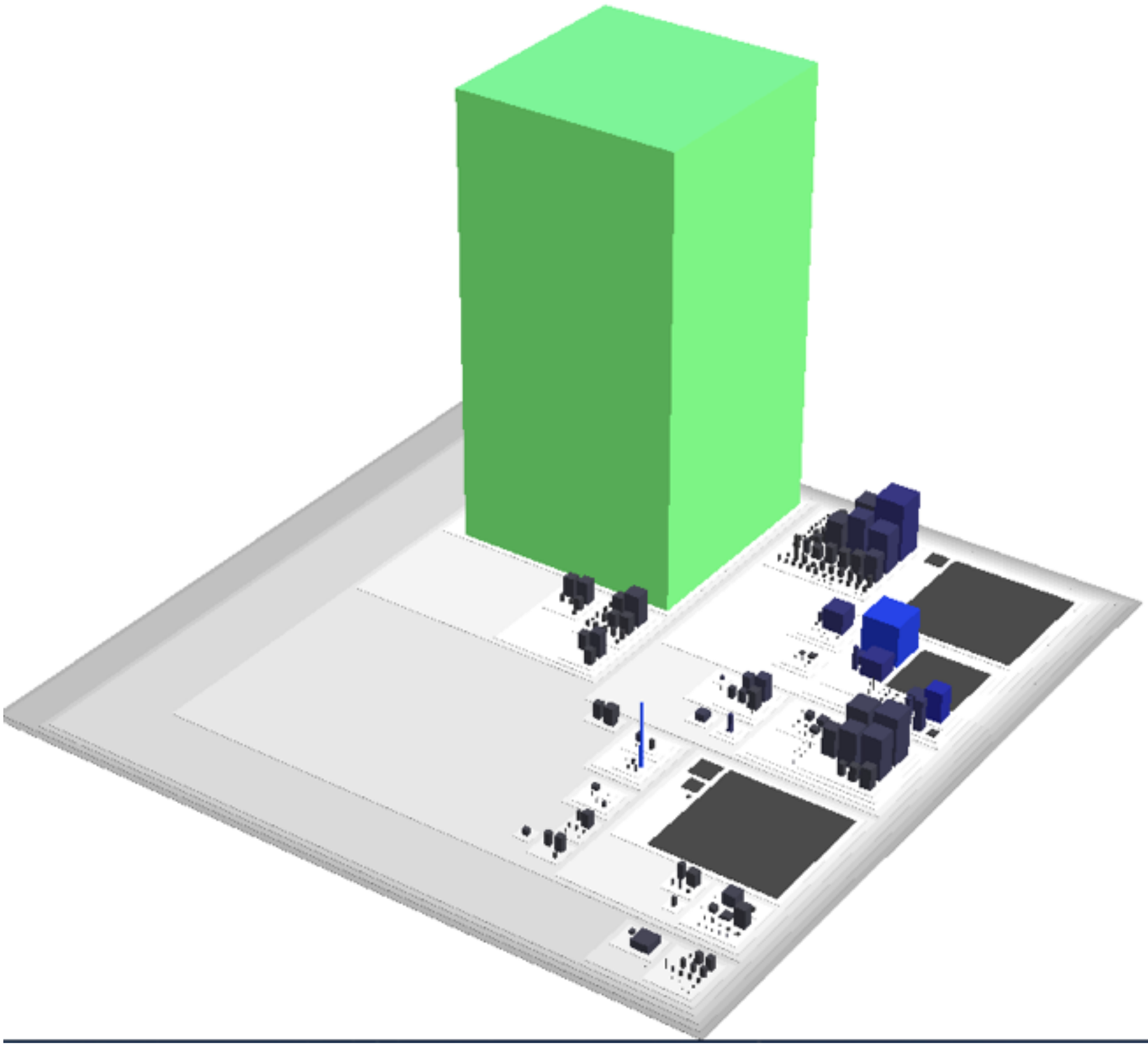
- It’s distracting!
- It’s costly!
- It hurts morale!
- Will it slow me down?

CODE SMELLS

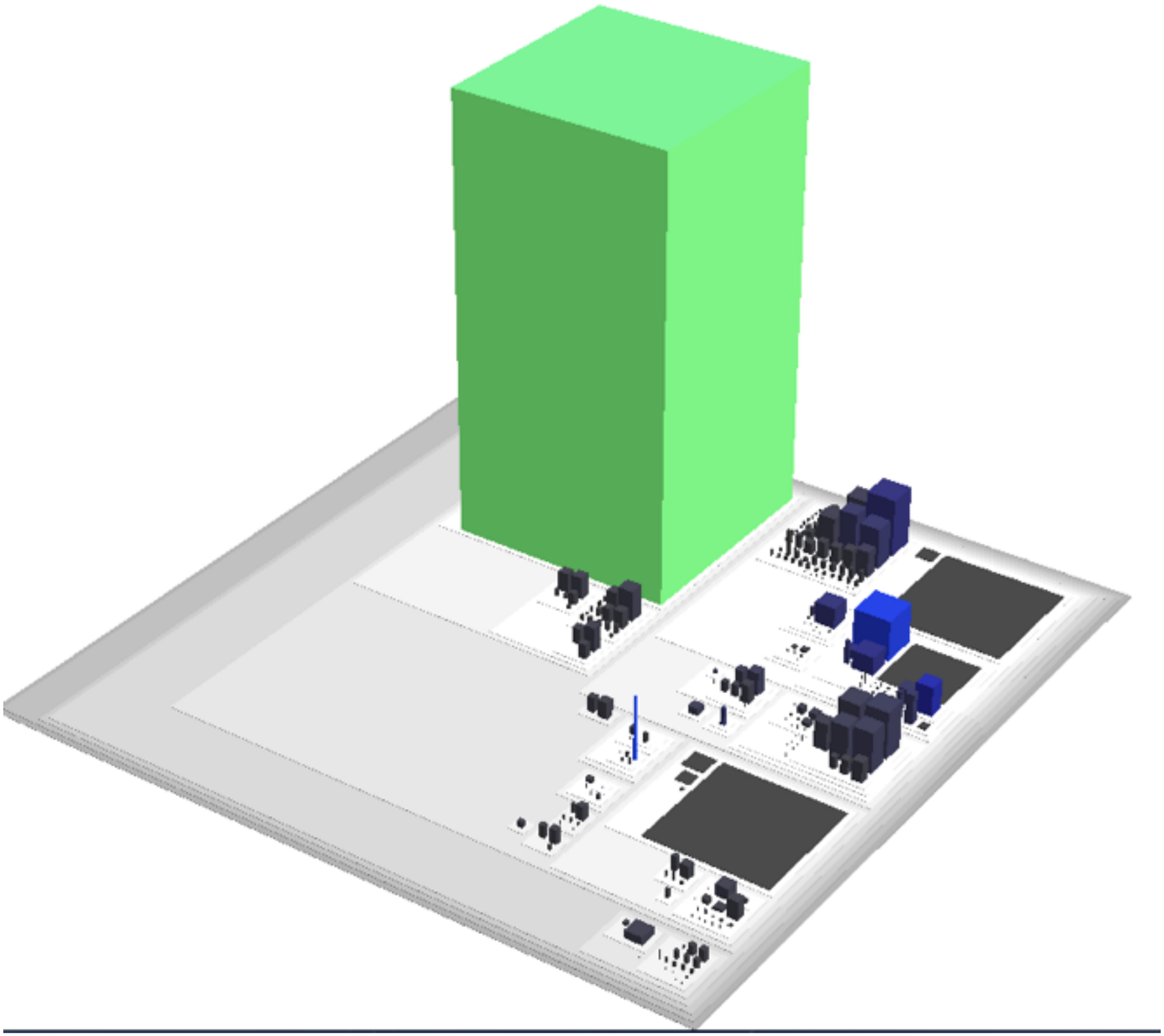
CODE SMELLS

- Long methods
- Lots of parameters
- Code comments
- Magic numbers
- Duplication

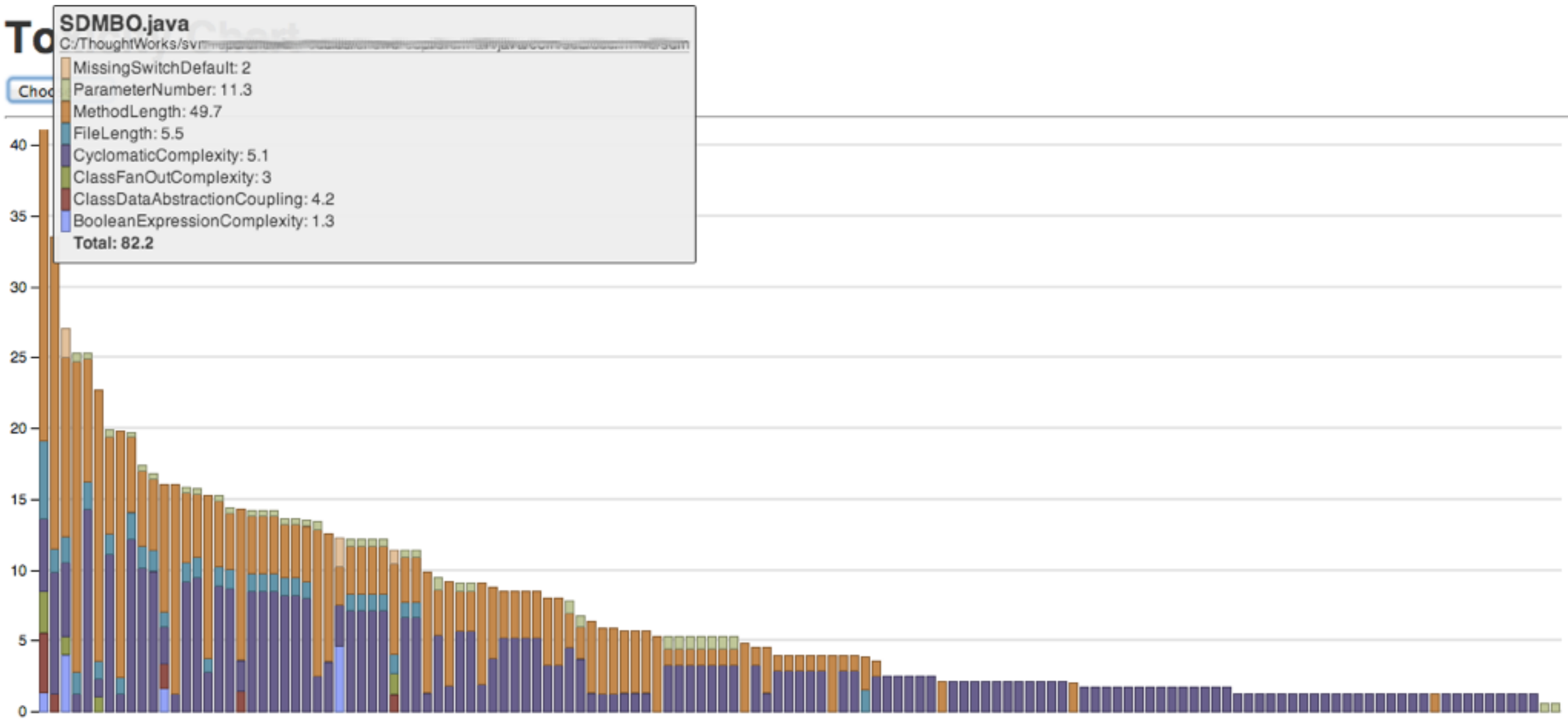
BIG SCARY CLASS



ANAEMIC MODEL



TOXIC CODE



Hover over the bars to see detailed information about the score for the class.

The y-axis is fixed with 41 as a maximum to make it easier to compare charts for different code-bases.

EVOLUTIONARY ARCHITECTURE

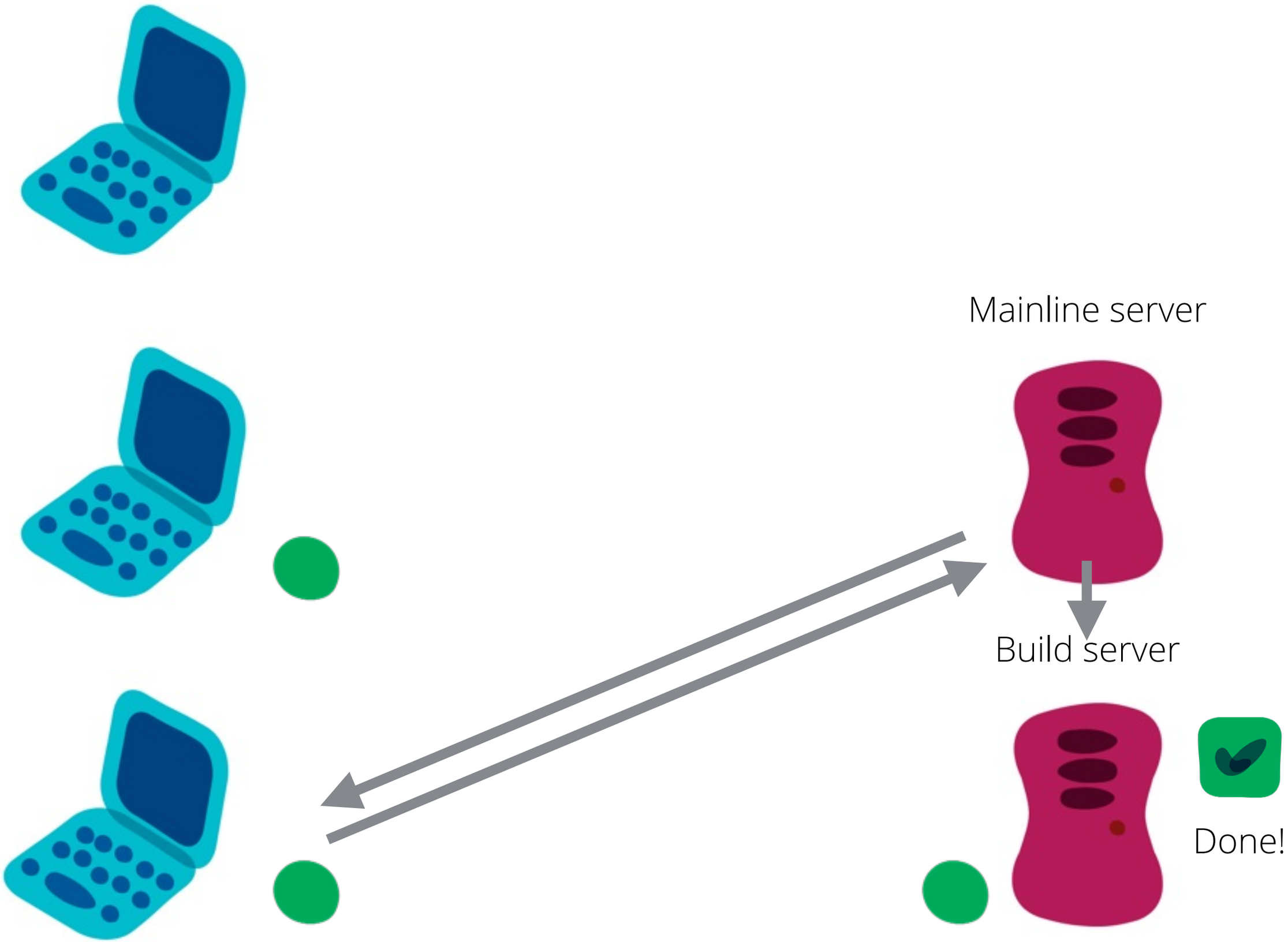
Growing software, guided by tests

EVOLUTIONARY ARCHITECTURE

- Incremental, not Big Design Up Front
- Simple design
- YAGNI

BIG DESIGN UP FRONT

CONTINUOUS INTEGRATION


Mainline server
Build server
Done!

WHY PRACTISE CI?

- Quick feedback on problems
- Gets the most out of automated testing
- Tests, builds deployments using production-like setup
- Reduces waste due to manual integration
- Provides a safety net allowing changes to be made with confidence

CONTINUOUS INTEGRATION PRACTICES

- Keep the build fast: < 10 minutes
- Keep the build green
- Fix breakages within 10 minutes
- Commit as frequently as possible
- Avoid branching during development
- Trunk based development with feature toggles

INFORMATION RADIATORS

- Large, visible, easily understood
- Changes periodically, useful information
- Shows project progress, build status
- To make effort visible, to make a point
- For stakeholders, teams, passers-by

Build #526 (Mar 26, 2012 4:02:04 PM)

Keep this build forever

Delete this build

add description

Started 18 hr ago
Took 6 min 21 sec

Changes

1. MEDTZIN-1105 bug-fix Se corrige ordenamiento de las notas del Expediente Completo. (commit: 16d8075fba05ee5d071d90824c7b8b2b73bed486) (detail)

Started by an SCM change

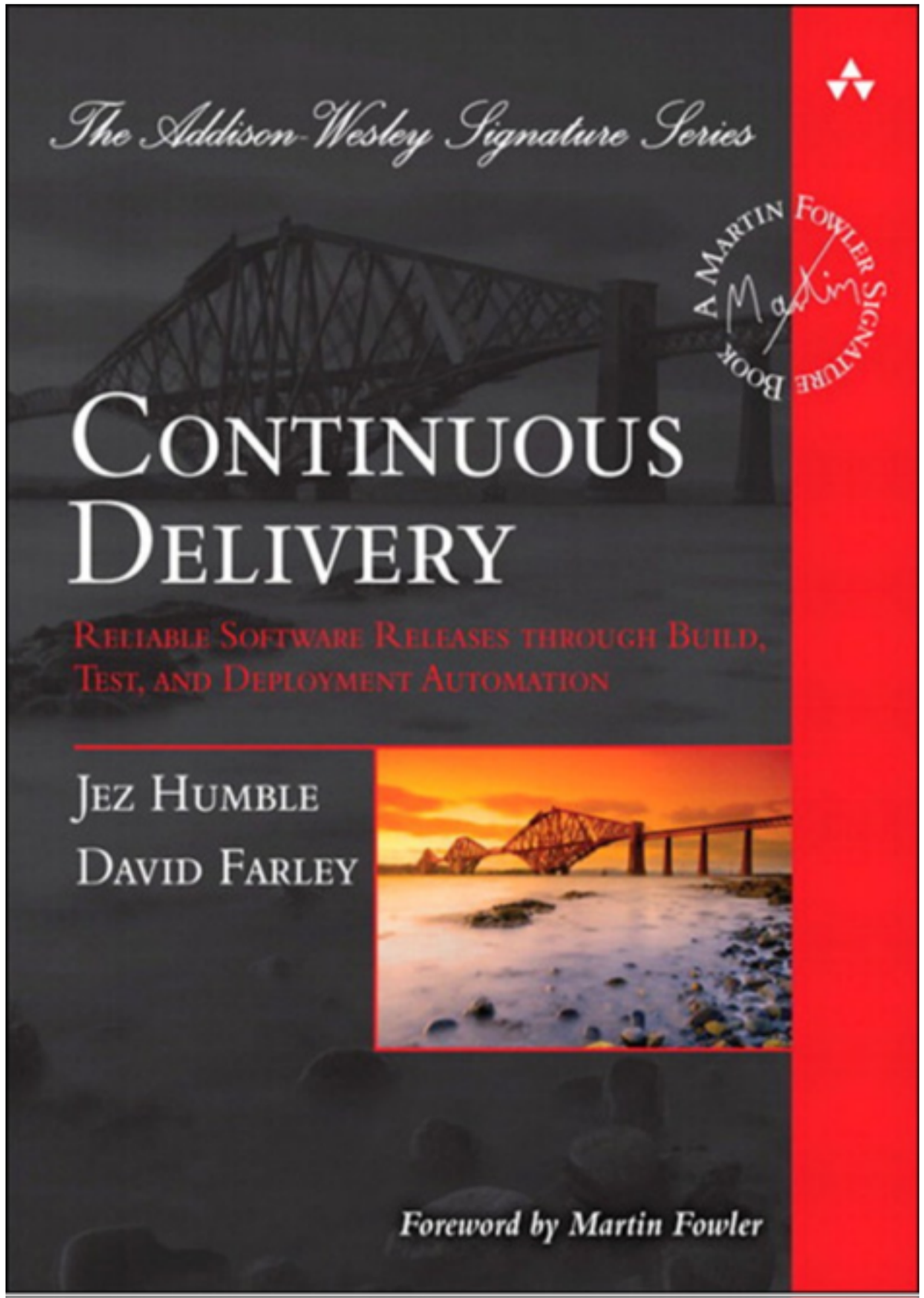
Revision: 16d8075fba05ee5d071d90824c7b8b2b73bed486

- origin/HEAD
- origin/master

There is no Esc key on Chuck Norris' keyboard, because no one escapes Chuck Norris.

The build was worth 1 points


BUILD
BUSTER
QUOTE
BOARD


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

CONTINUOUS DELIVERY

- Automate almost everything
- Everything in version control
- Repeatable process for releasing software
- Build quality in
- “Done” means released
- Everyone is responsible for delivery

AUTOMATE EVERYTHING

- Unit tests, integrations tests
- Acceptance tests
- Deployment
- Environment provisioning
- Configuration
- et cetera

COLLECTIVE OWNERSHIP

AGILE DEVELOPERS

- Can work with any part of the stack
- Can work on any story
- Usually have deep expertise in certain areas
- "T-Shaped"

SUMMARY

- Test Driven Development
- Automated Testing
- Refactoring
- Evolutionary Architecture
- Continuous Integration
- Continuous Delivery

QUESTIONS?