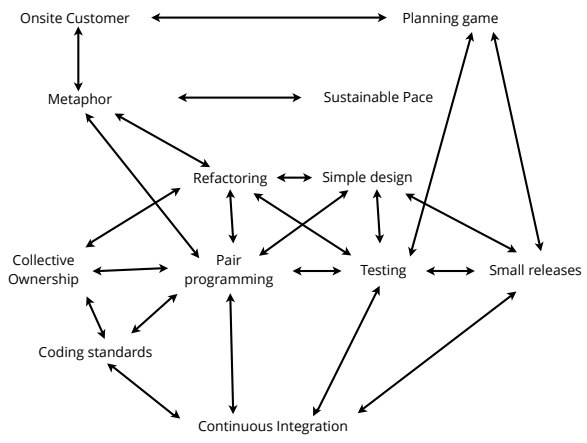
ThoughtWorks®

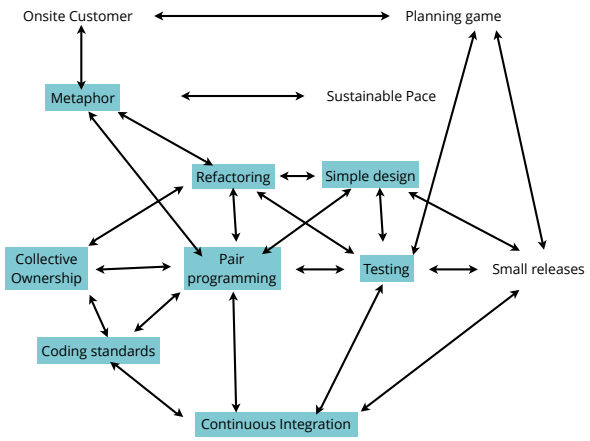
DEVELOPMENT PRACTICES

Text

AGENDA

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





4

TEST DRIVEN DEVELOPMENT

5

Write a failing test *Make it pass* *Make it better*

Red *Green* *Refactor*

6

TDD helps communicate intentions

TDD focuses on behaviour, not implementation

REFACTORING

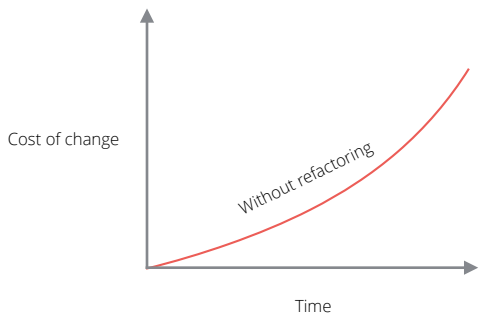
Refactoring is a behaviour preserving transformation

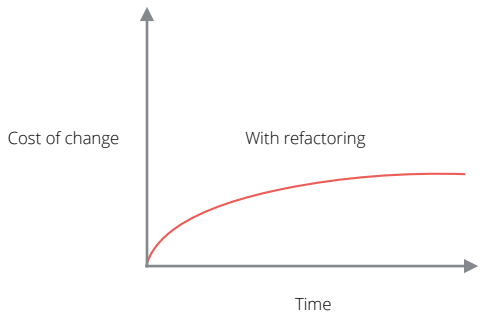
Functionally, the software is the same

Refactoring improves maintainability & extensibility

Refactoring vs redesign

WHY REFACTOR?





16

WHEN TO REFACTOR

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

17

WHEN NOT TO REFACTOR

- Code is too messy
- Near a deadline

18

DEMO

WHY PAIR

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

PAIRING STYLES

- Ping-pong
- Driver Navigator

PAIRING MYTHS

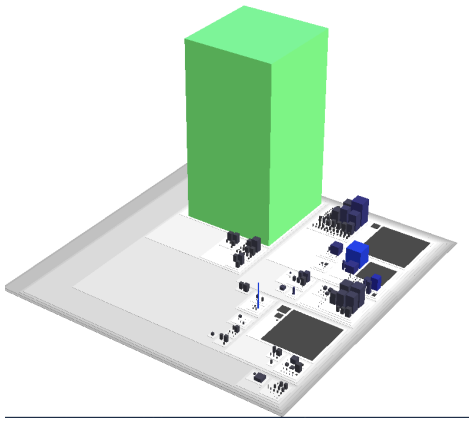
- It's distracting!
- It's costly!
- It hurts morale!
- Will it slow me down?

CODE SMELLS

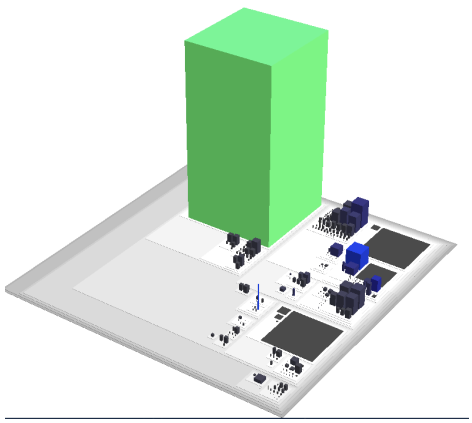
CODE SMELLS

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

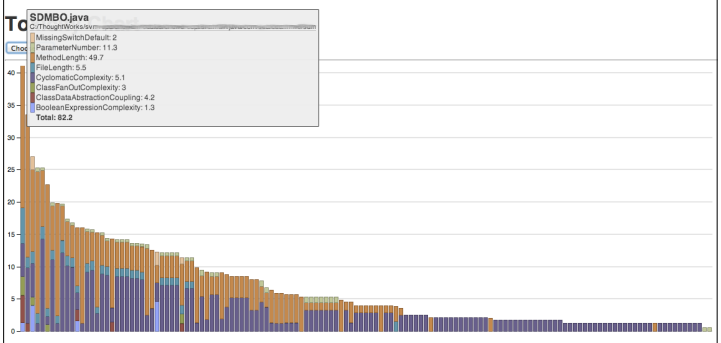
BIG SCARY CLASS



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.

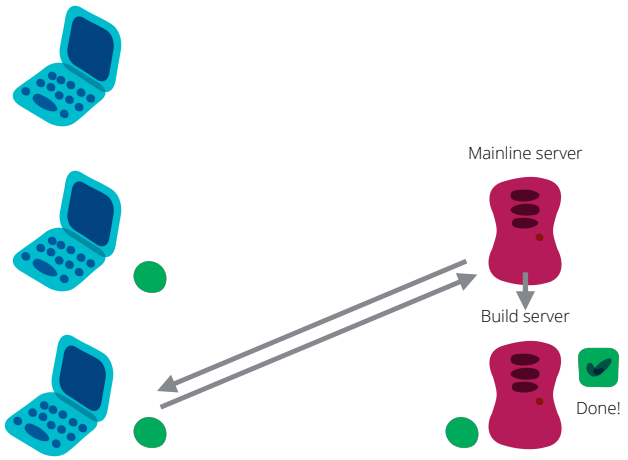
EVOLUTIONARY ARCHITECTURE

Growing software, guided by tests

EVOLUTIONARY ARCHITECTURE

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

CONTINUOUS INTEGRATION

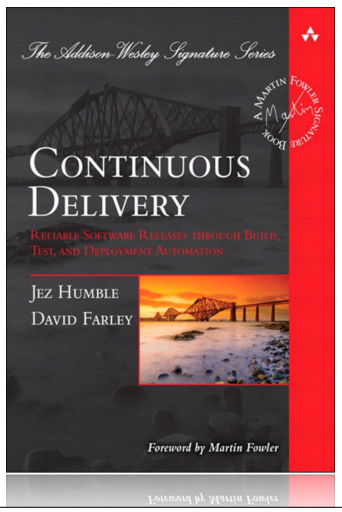


WHY PRACTISE CI?

- 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



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

37

38

AGILE DEVELOPERS

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

39

SUMMARY

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

QUESTIONS?