



Luca Minudel



<http://github.com/lucaminudel>

Alan Turing:

«

The popular view that scientists proceed inexorably from well-established fact to well-established fact, never being influenced by any improved conjecture, is quite mistaken

*Provided it is made clear which are **proved facts** and which are **conjectures**, no harm can result*

*Conjectures are of great importance since they **suggest useful lines of research***

»

Conjectures

Based on :

- what observed in this experience
- data collected from experiments since now

DESIGN ~ TDD ?

Endo-Testing: Unit Testing with Mock Objects

Tim Mackinnon (Connextra), Steve Freeman (BBST), Philip Craig (Independent)
(tim.mackinnon@pobox.com, steve@m3p.co.uk, philip@pobox.com)

This paper was presented at the
Software Engineering - XP2004
be published in *XP eXamined*

Abstract

Unit testing is a fundamental part of software development. It is difficult to test in isolation. It is difficult to maintain and it is difficult to integrate with domain code and test suites. Test structure, and avoid polluting

Keywords: Extreme Programming

1 Introduction

"Once," said the Mock Object,

Unit testing is a fundamental part of software development. It is difficult to test trivial code is difficult to test in isolation. It is difficult to maintain and it is difficult to integrate with domain code and test suites. Test structure, and avoid polluting

We propose a technique called Mock Objects. It supports good Object-Oriented design. It is less interesting as a technique for isolating libraries than is widely thought. This paper describes the use of Mock Objects with an extended process. It also introduces jMock, a Java Mocking framework.

Our experience is that developing better structure of both domain and test code is a regular format that gives the domain code a better structure. We should be written to make it easier to achieve this. We should be written to make it easier to achieve this. We should be written to make it easier to achieve this.

In this paper, we first describe the benefits and costs of Mock Objects. We then describe a brief pattern for using Mock Objects.

2 Unit testing with Mock Objects

An essential aspect of unit testing is testing and where any simply and clearly as possible

Mock Roles, not Objects

Steve Freeman, Nat Pryce, Tim Mackinnon, Joe Walnes
ThoughtWorks UK
Berkshire House, 168-173 High Holborn
London WC1V 7AA

{sfree@thoughtworks.com, npryce@thoughtworks.com, tmackinnon@thoughtworks.com, jwalnes@thoughtworks.com}

ABSTRACT

Mock Objects is an extension to Test-Driven Development that supports good Object-Oriented design. It is less interesting as a technique for isolating libraries than is widely thought. This paper describes the use of Mock Objects with an extended process. It also introduces jMock, a Java Mocking framework.

Categories and Subject Descriptors: D.2.2 [Software Engineering]: Design and Analysis—Object-Oriented design methods

General Terms: Design, Verification

Keywords

Test-Driven Development, Mock Objects

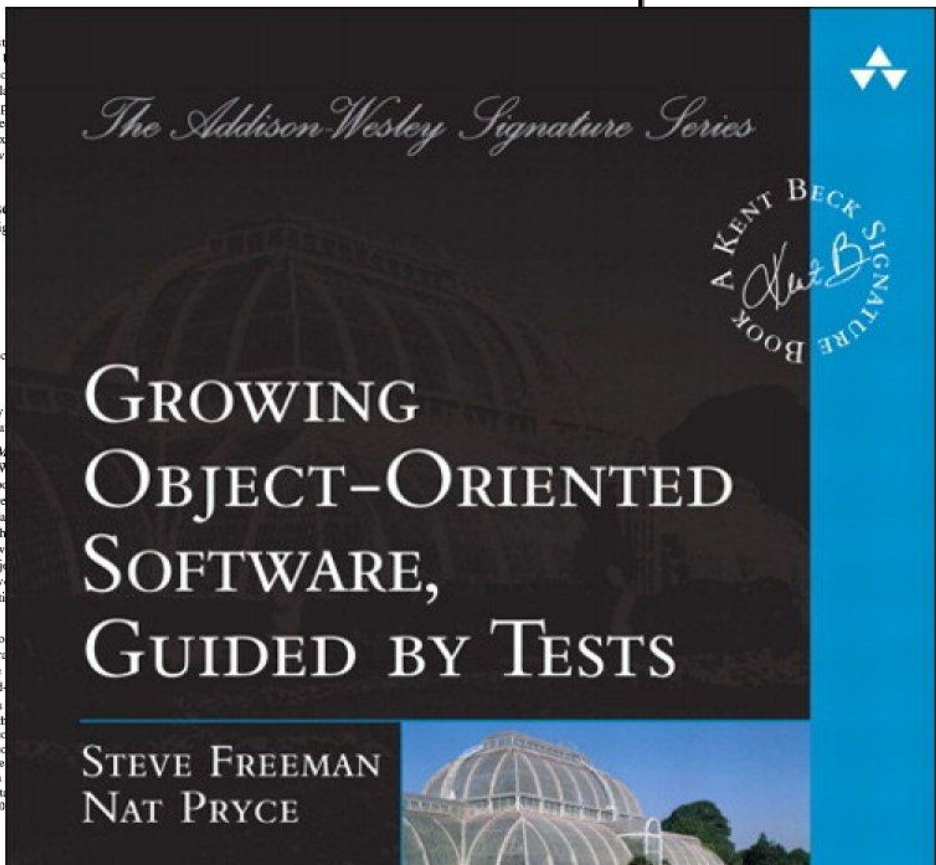
1. INTRODUCTION

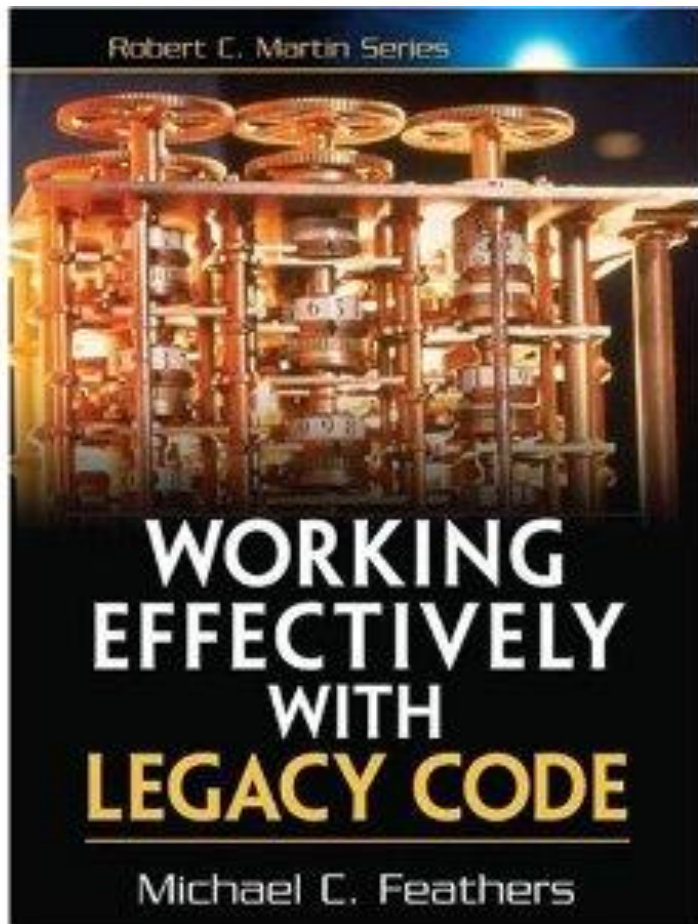
Mock Objects is misnamed. It is really a technique for isolating libraries in a system based on the roles that

In [10] we introduced the concept of Mock Objects to support Test-Driven Development. We have better structured tests and, more importantly, we have preserved encapsulation, reducing the interactions between classes. We have refined and adjusted the experience since then. In particular, we have the most important benefit of Mock Objects called "interface discovery". We have a framework to support dynamic generation of Mock Objects on this experience.

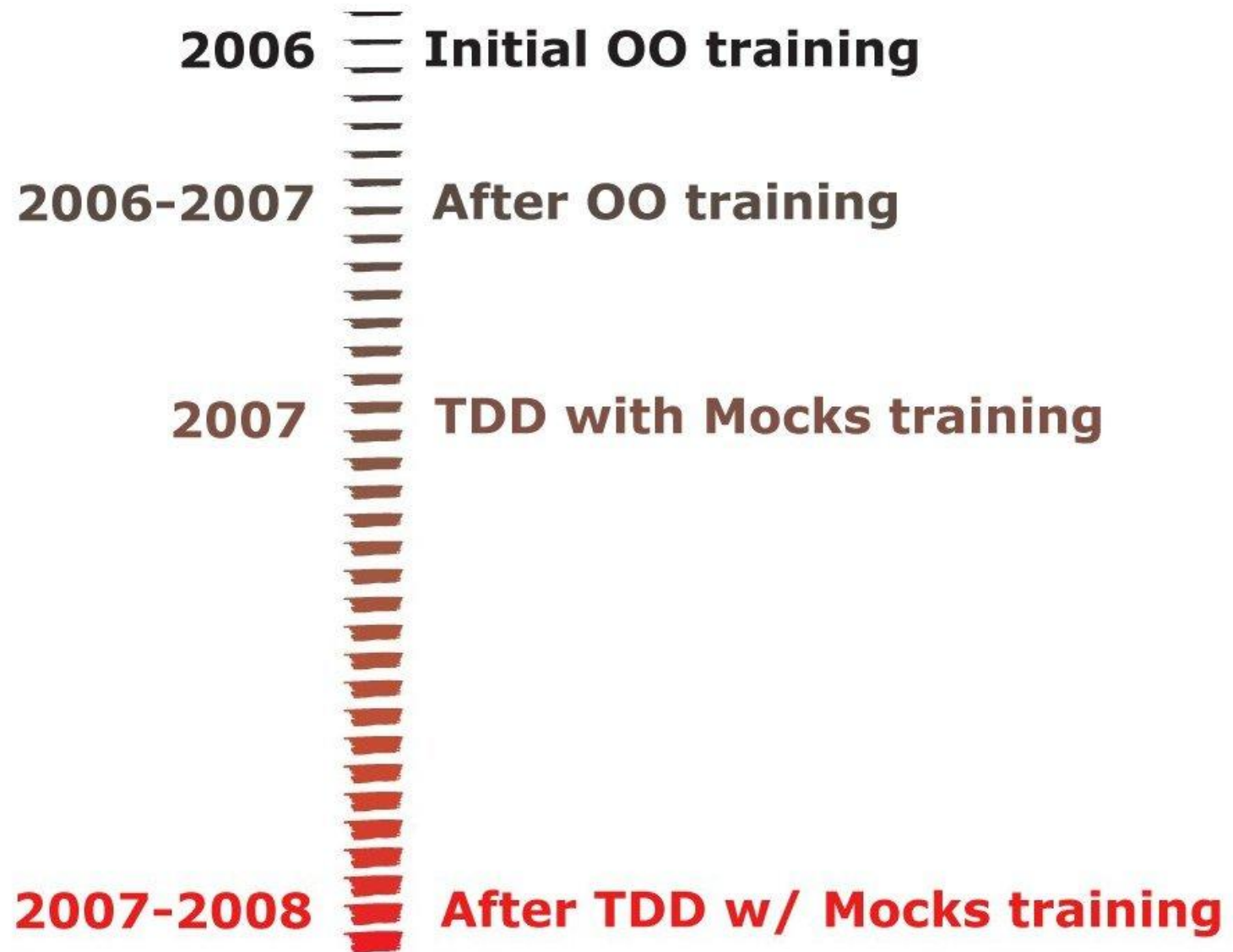
The rest of this section establishes the context of Test-Driven Development and good programming, and then introduces the rest of the paper introduces Needs.

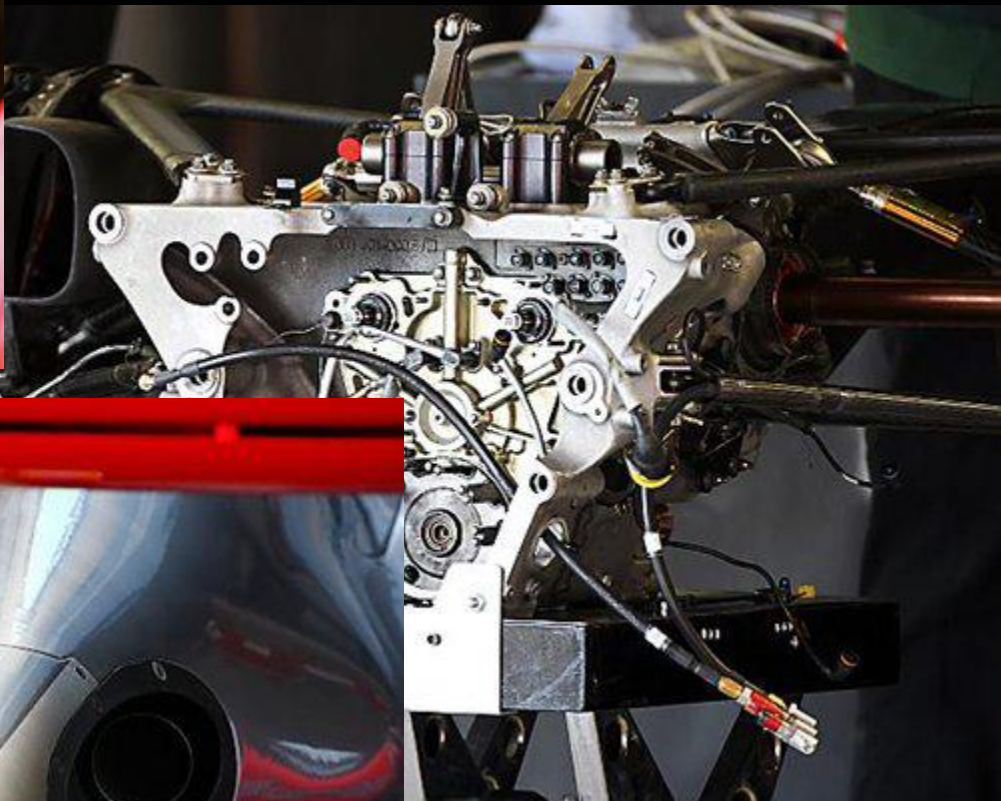
Permission to make digital or hard copies of this work for personal or classroom use is granted without fee, provided that the copies are not made or distributed for profit or commercial advantage, and that the copies bear this notice and the full citation to the source. Copyright 2004 ACM 1-58113-000-0/00/000000

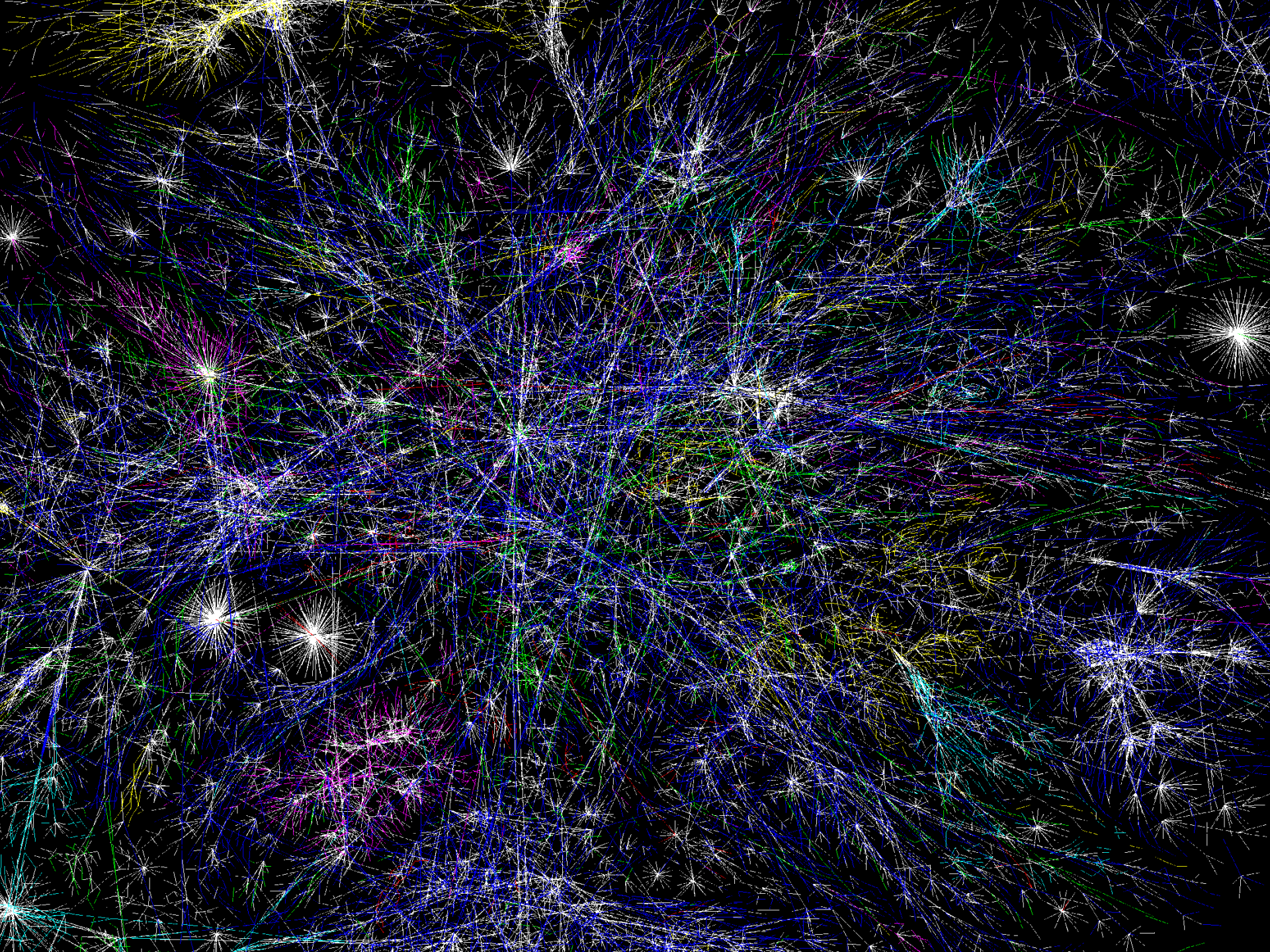


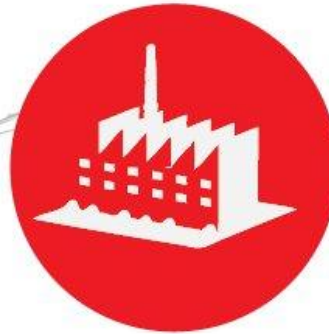


- Parametrize Constructor
- Extract Interface
- Introduce Instance Delegator
- Skin and Wrap the API
- Parametrize Method
- Adapt Parameter
- Responsibility-Based Extraction
- ...

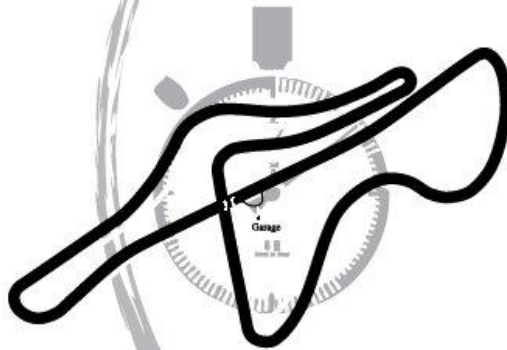




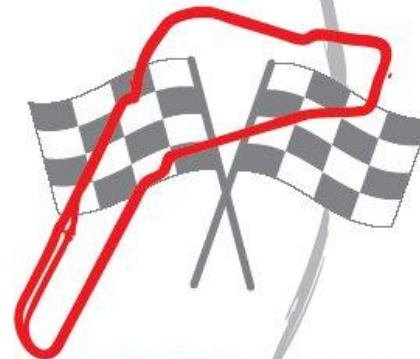




HOME FACTORY

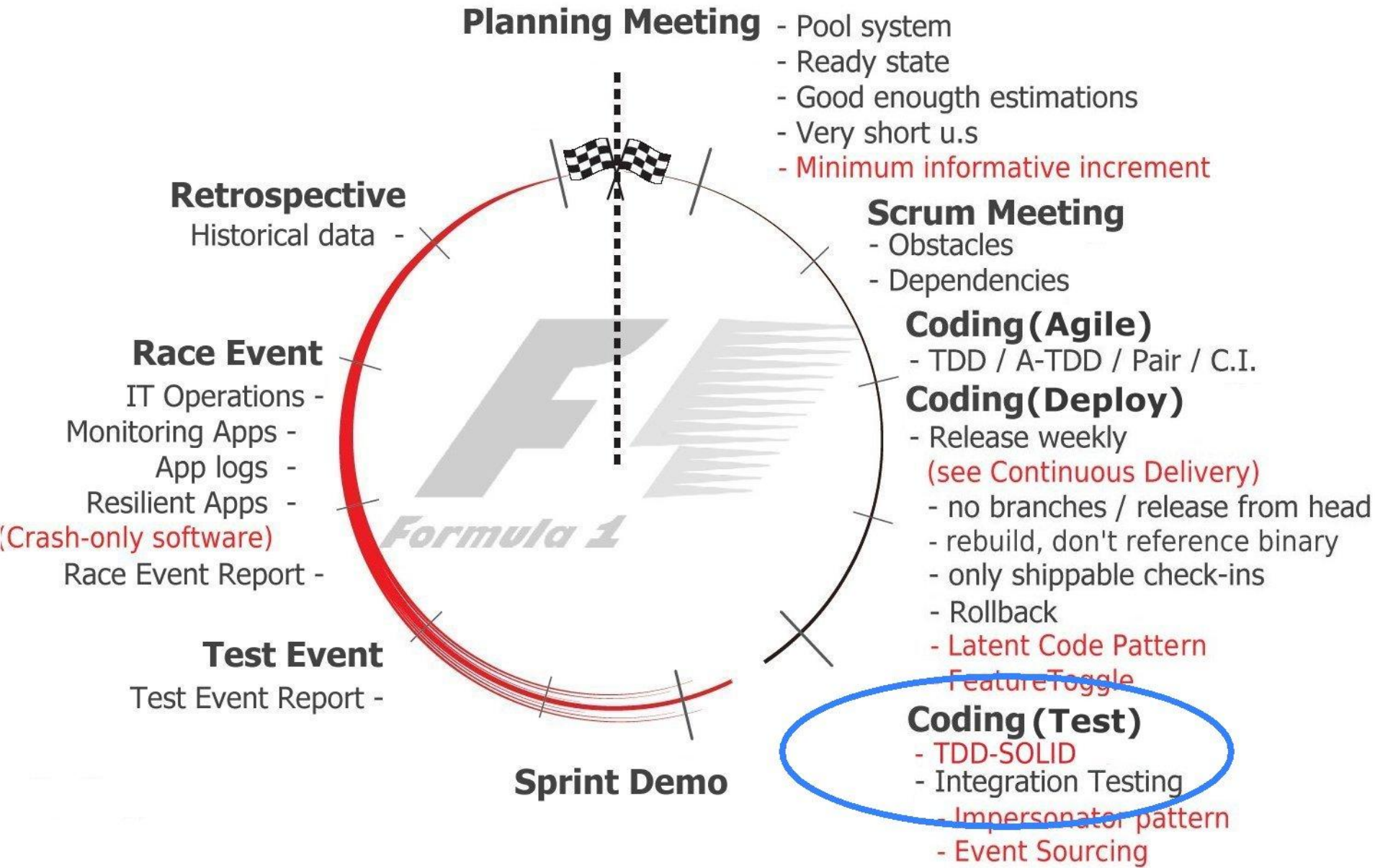


TEST TRACK



RACE TRACK

INSPECT-ADAPT



*All science is experiential; but all experience must be related back to and derives its validity from the **conditions and context** of consciousness in which it arises, i.e., the totality of our nature - Wilhelm Dilthey*

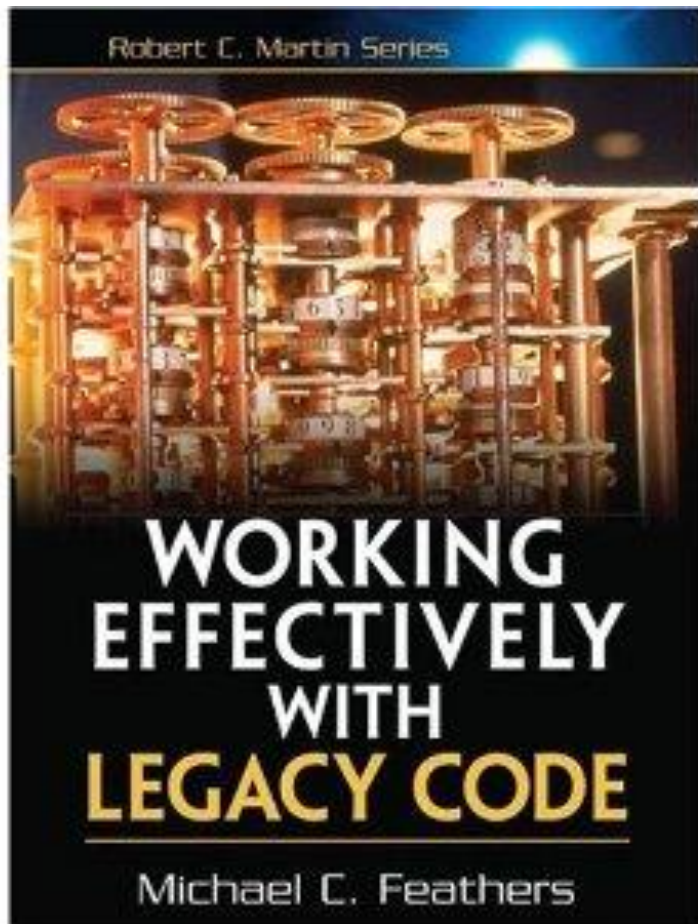
*No technique can survive inadequately
trained developers* - Steve Freeman

1st Law of Software Process:

*Process only allows us to do things we
already know how to do* - Philip G. Armour

TDD does not drive towards good design, it drives away from a bad design. If you know what good design is, the result is a better design - Nat Pryce

TDD doesn't drive good design. TDD gives you immediate feedback about what is likely to be bad design - Kent Beck



- Parametrize Constructor
- Extract Interface
- Introduce Instance Delegator
- Skin and Wrap the API
- Parametrize Method
- Adapt Parameter
- Responsibility-Based Extraction
- ...

OCP – DIP

SRP – ISP

LSP

LoD

Downsides observed in the use of mocks ?

Difficulties experienced with TDD and mocks ?

**High adherence to
Design Principles**

**Low adherence to
Design Principles**



**High adherence to
Design Principles**



<- Average (*)

**Low adherence to
Design Principles**

**High adherence to
Design Principles**



<- TDD with Mocks

<- Average (*)

**Low adherence to
Design Principles**

Co-Evolution / Attractors / Barriers

Prof. Sugata Mitra speculation

education is a self organizing system where learning is an emergent phenomenon

Stafford Beer quote

If you wish to tell someone how to reach the top of a mountain that is shrouded in mist, the heuristic 'keep going up' will get him there

Software Systems Evolution

*To the degree that a software system is large and distributed enough that there is **no effective single point of control**, we must expect **evolutionary forces***

...

*The strategies that we adopt to understand, control, interact with, and influence **the design of computational systems** will be different once we understand them as **ongoing evolutionary processes***

**High adherence to
Design Principles**



<- After TDD w/ Mocks

<- TDD with Mocks

<- Average (*)

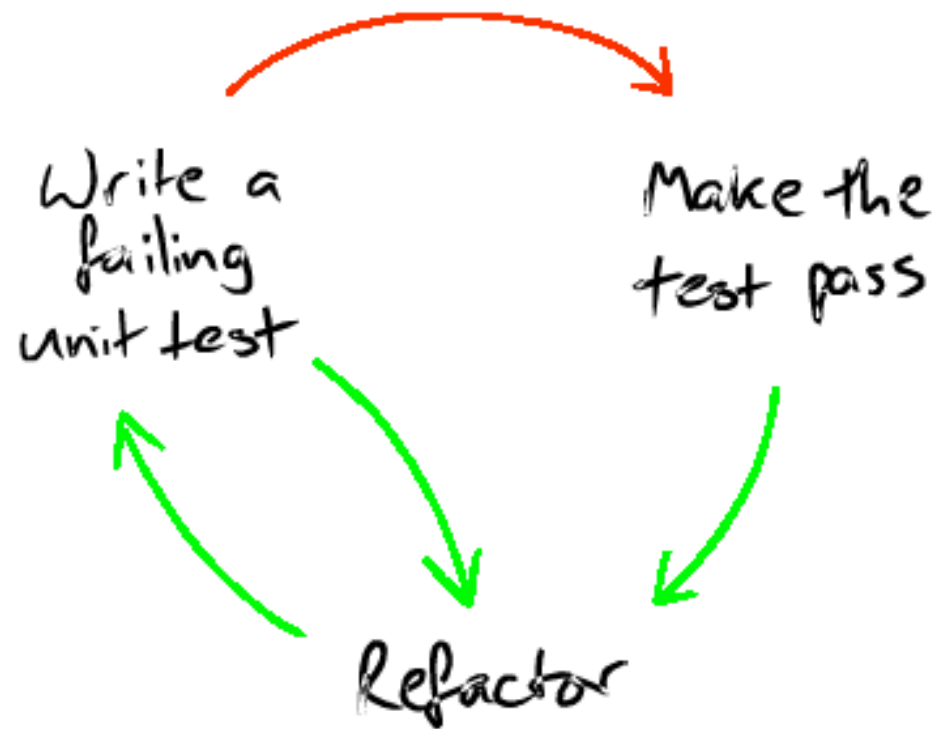
**Low adherence to
Design Principles**

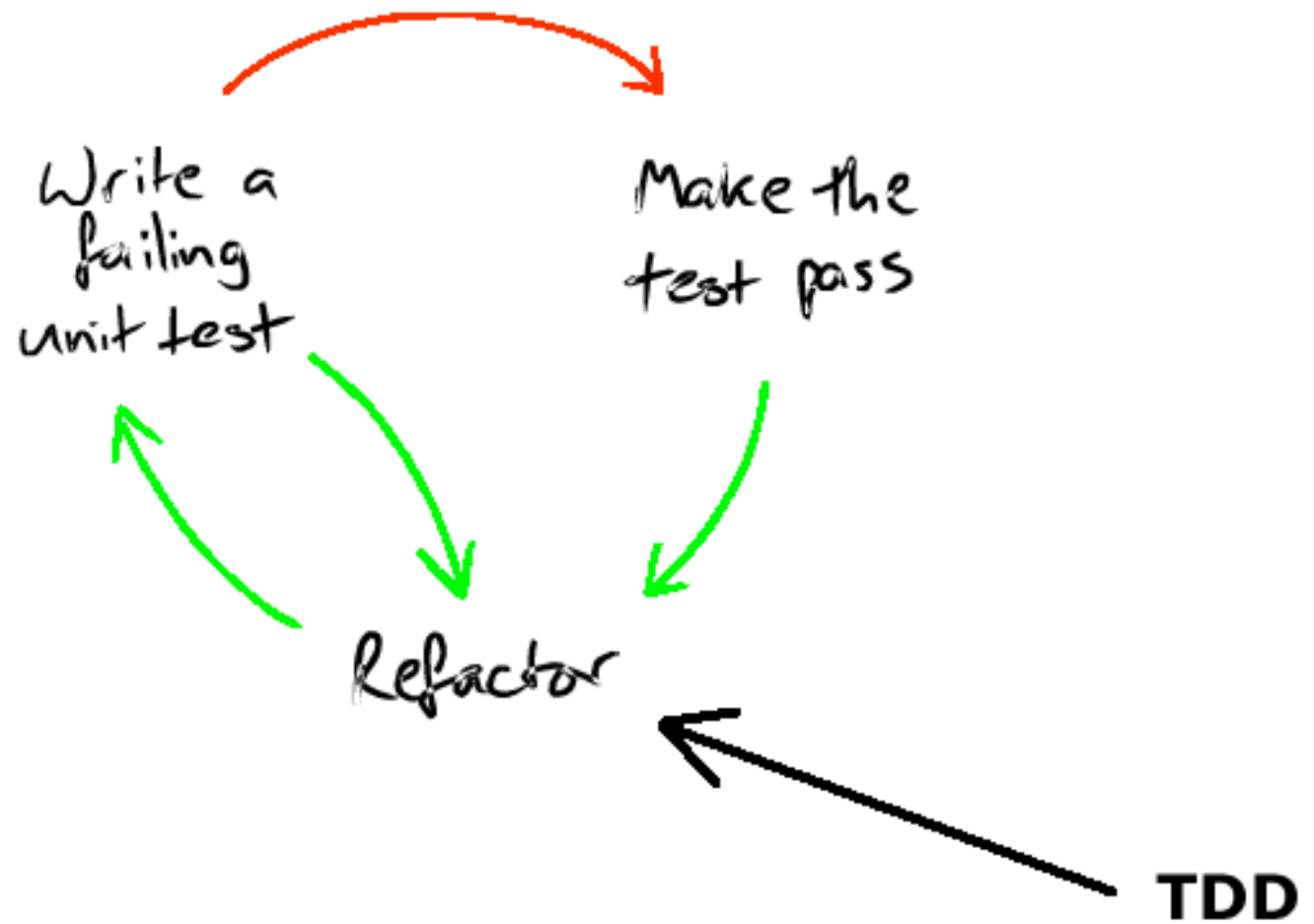
Norwegian Developer Conference 2010

Session 'The Deep Synergy Between Testability and Good Design'

Michael Feathers:

writing tests is another way to look the code and locally understand it and reuse it, and that is the same goal of good OO design. This is the reason of the deep synergy between testability and good design.





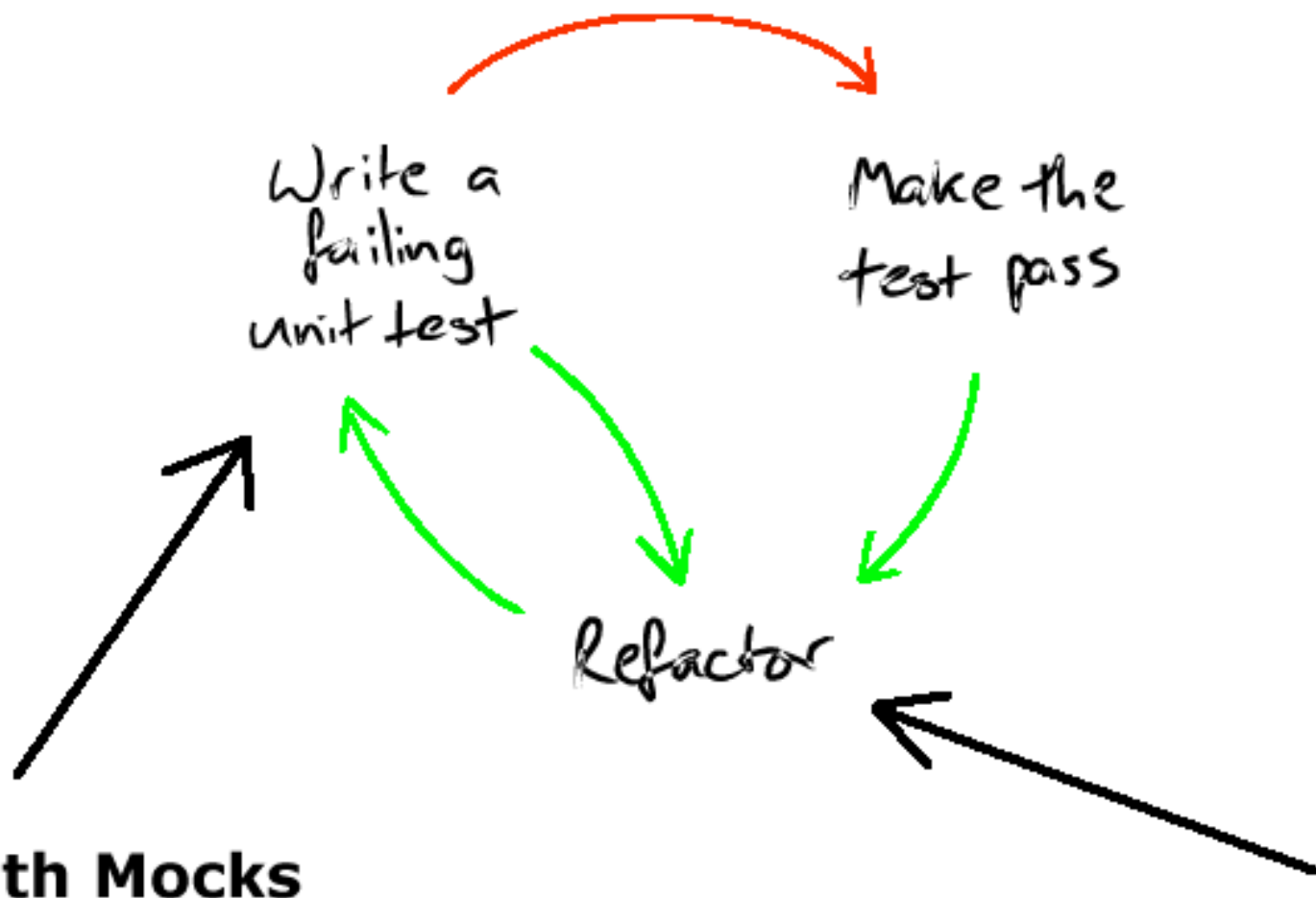
Write a
failing
unit test

Make the
test pass

Refactor

TDD with Mocks

TDD



After TDD with Mocks

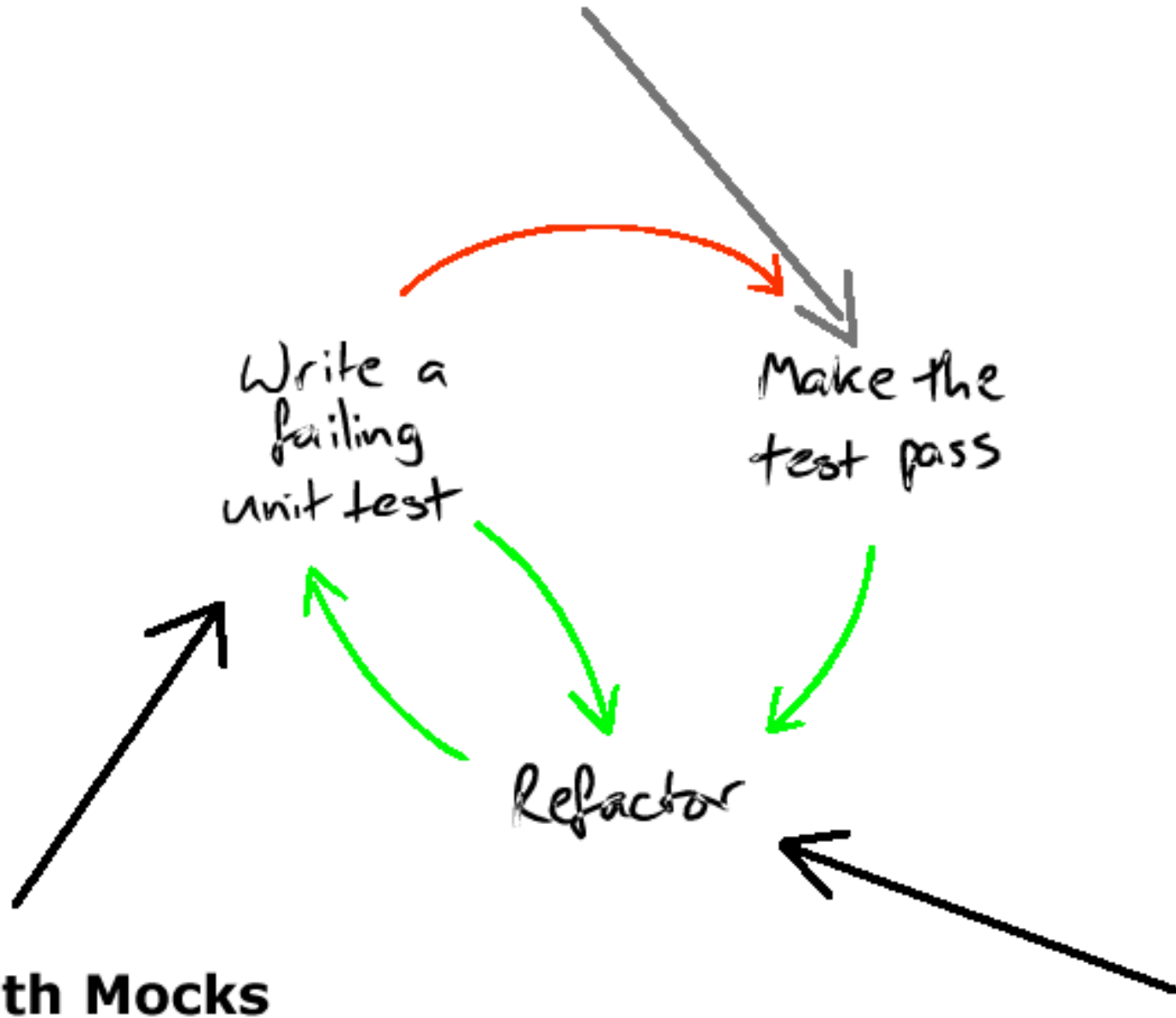
Write a
failing
unit test

Make the
test pass

Refactor

TDD with Mocks

TDD



Conclusions: does TDD w/ Mocks

- encourage good design ?
- promote good design and conformance to design principles ?
- in some degree lead to conformance to design principles even in the absence of an explicit policy to do so ?
- in some degree lead to learn and deeply understand and apply the design principle ?

Code / Paper / Slides :

<http://github.com/lucaminudel>

Feedback / Comments / Questions :

Luca Minudel - tdd@minudel.it

