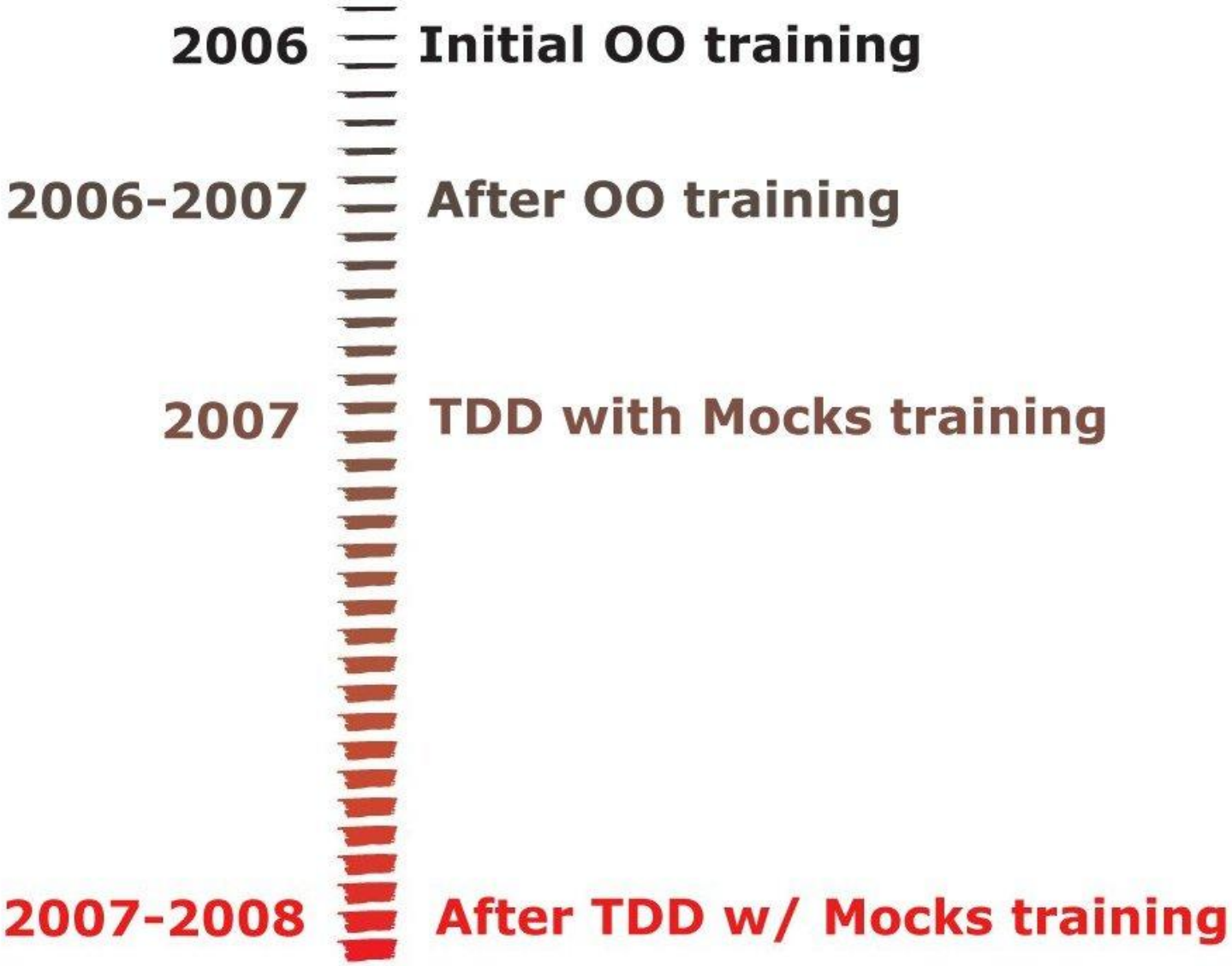
Luca Minudel

Exploratory study

based on :

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

http://github.com/lucaminudel


2006
Initial OO training
2006-2007
After OO training
2007
TDD with Mocks training
2007-2008
After TDD w/ Mocks training

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***

»

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 th
Software Engineering - XP20
be published in *XP eXamined*

Abstract

Unit testing is a fundamental p
difficult to test in isolation. It
and difficult to maintain and i
domain code and test suites. T
structure, and avoid polluting

Keywords: Extreme Programm

1 Introduction

"Once," said the Mo

Unit testing is a fundamental p
trivial code is difficult to test
time, and you want to be notif
because you are trying to test

We propose a technique calle
implementations that emulate
code which they test from insi
writing code stubs with two in
is usual, and we use our tests

Our experience is that develop
better structure of both domai
regular format that gives the d
should be written to make it e
technique to achieve this. We
cost of writing stub code.

In this paper, we first describe
the benefits and costs of Mock
brief pattern for using Mock O

2 Unit testing with Mo

An essential aspect of unit tes
you are 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
{sfreeman, npryce, tmackinnon, jwalnes} @thoughtworks.com

ABSTRACT

Mock Objects is an extension to Test
supports good Object-Oriented design
a coherent system of types within a co
less interesting as a technique for isol
libraries than is widely thought. This p
of using Mock Objects with an extende
and worst practices gained from ex
process. It also introduces jMock, a Jav
our collective experience.

Categories and Subject Desc
D.2.2 [**Software Engineering**]: Desig
Object-Oriented design methods

General Terms
Design, Verification.

Keywords
Test-Driven Development, Mock Objec

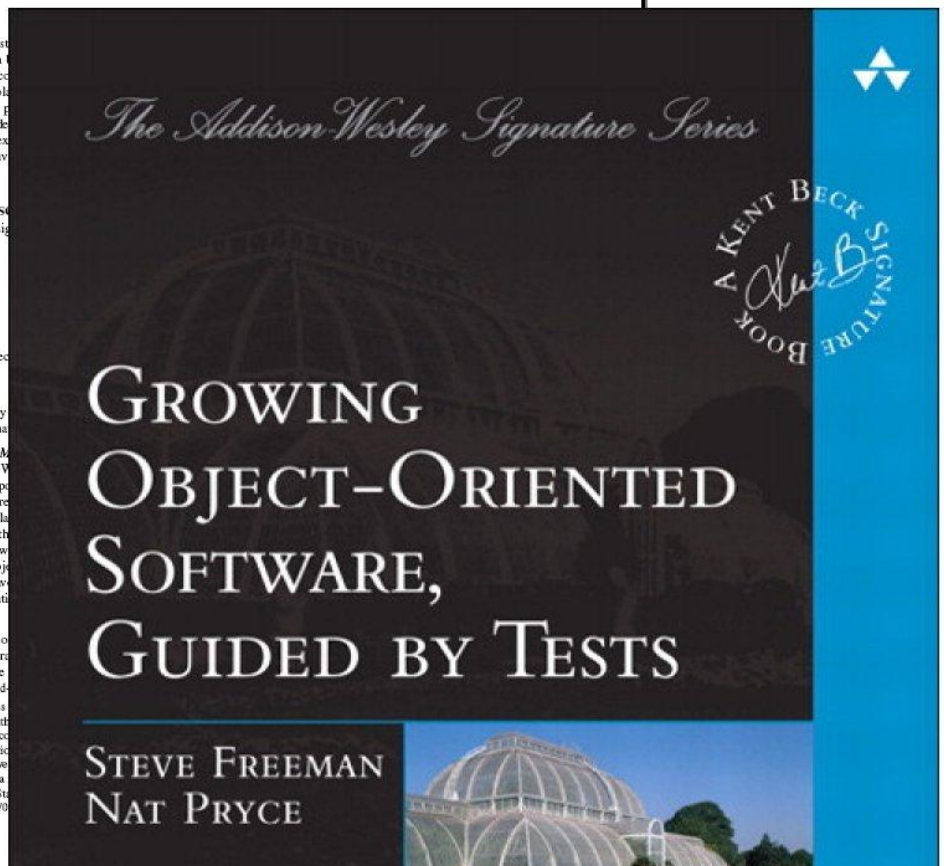
1. INTRODUCTION

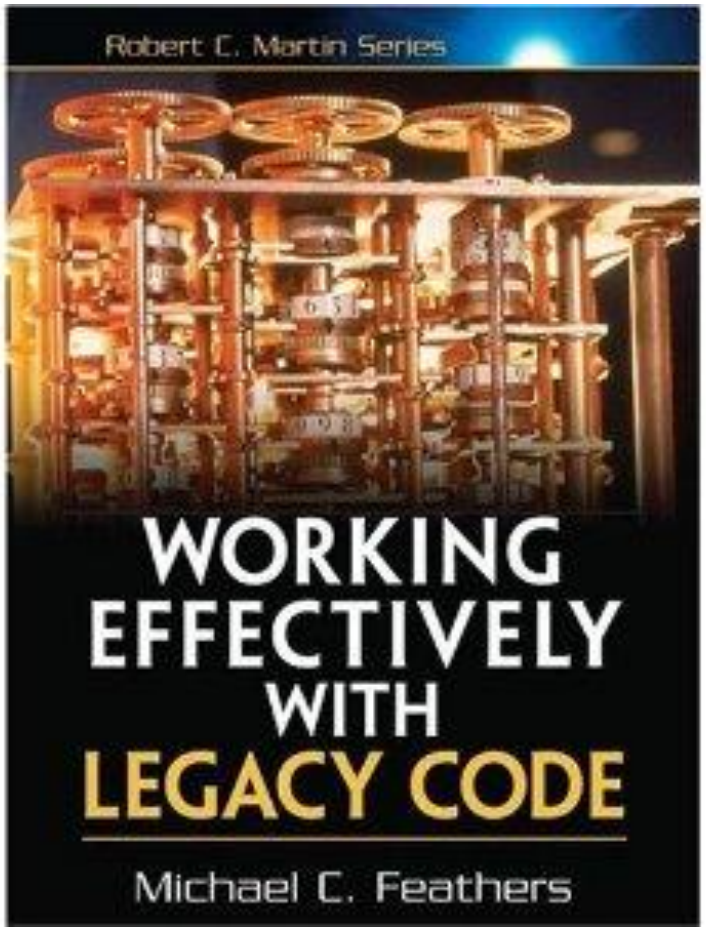
Mock Objects is misnamed. It is really
types in a system based on the roles tha

In [10] we introduced the concept of *M*
to support Test-Driven Development. W
better structured tests and, more impo
code by preserving encapsulation, re
clarifying the interactions between cla
how we have refined and adjusted th
experience since then. In particular, w
most important benefit of Mock Obje
called "interface discovery". We hav
framework to support dynamic generati
on this experience.

The rest of this section establishes o
Driven Development and good pra
Programming, and then introduces the
rest of the paper introduces Need-

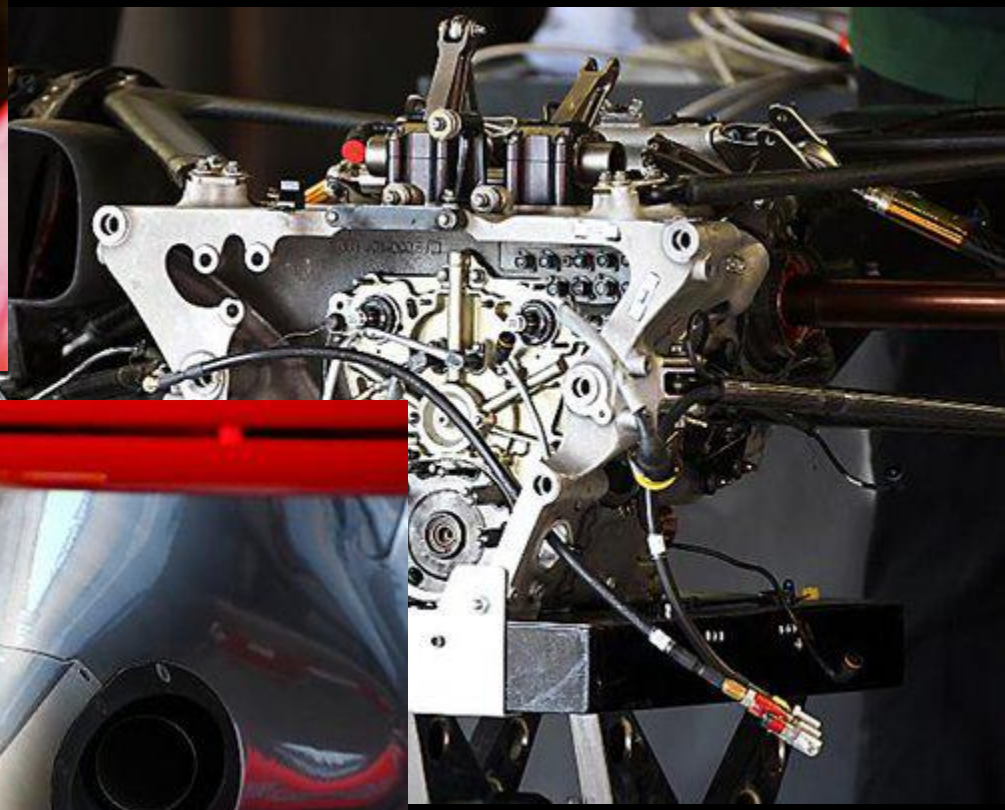
Permission to make digital or hard copies
personal or classroom use is granted with
are not made or distributed for profit or co
copies bear this notice and the full citatio
otherwise, or republish, to post on serve
requires prior specific permission and/or a
Conference'04, Month 1–2, 2004, City, St
Copyright 2004 ACM 1-58113-000-0/00/0



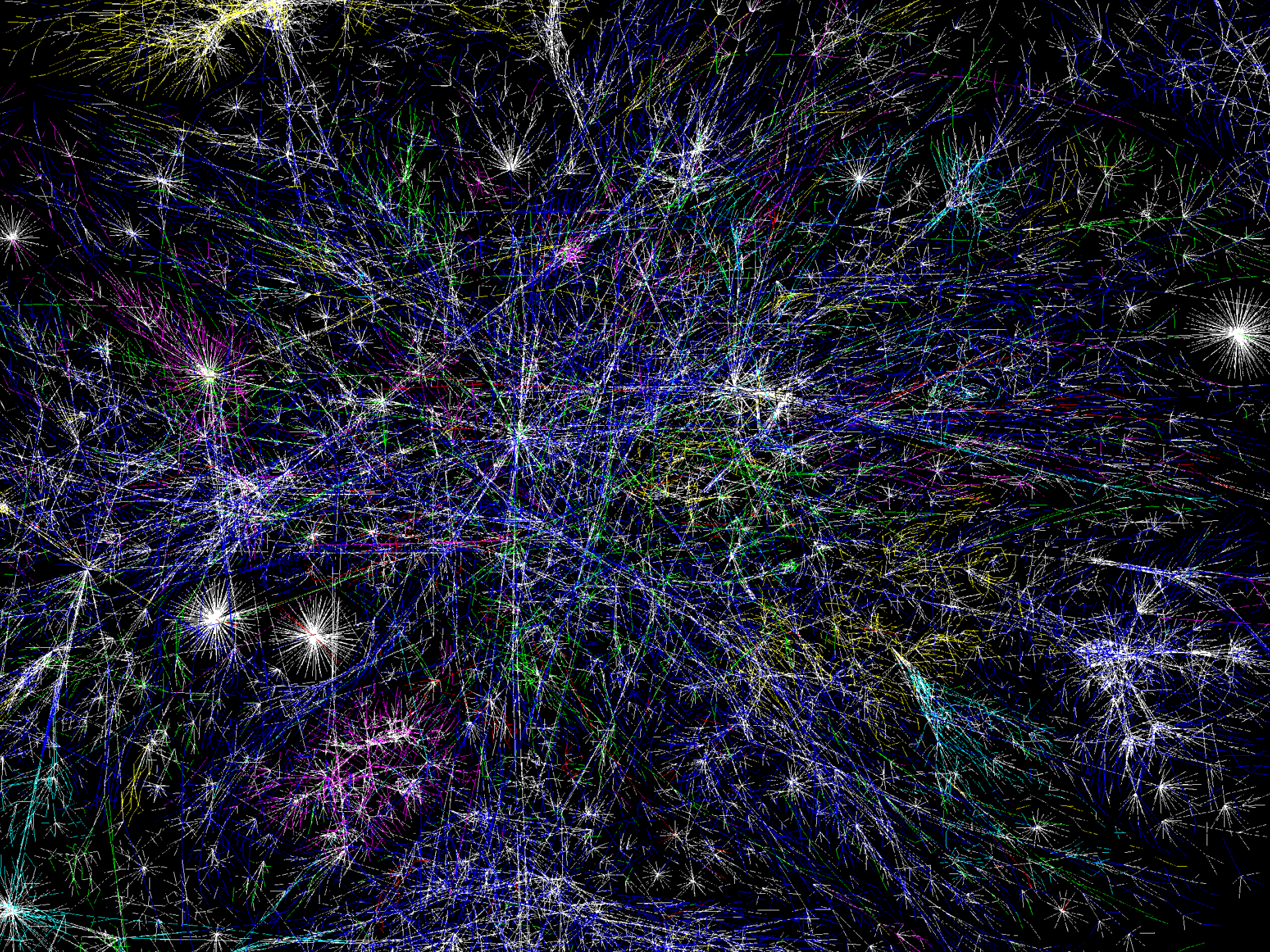


- Parametrize Constructor
- Extract Interface
- Introduce Instance Delegator
- Skin and Wrap the API
- Parametrize Method
- Adapt Parameter
- Responsibility-Based Extraction
- …

DIF IN
FFA
MIX
GRP
DD
MF
REV
TRO



BRIDGESTONE
BRIDGESTONE



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

D. H. Ackley et al. 2002

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

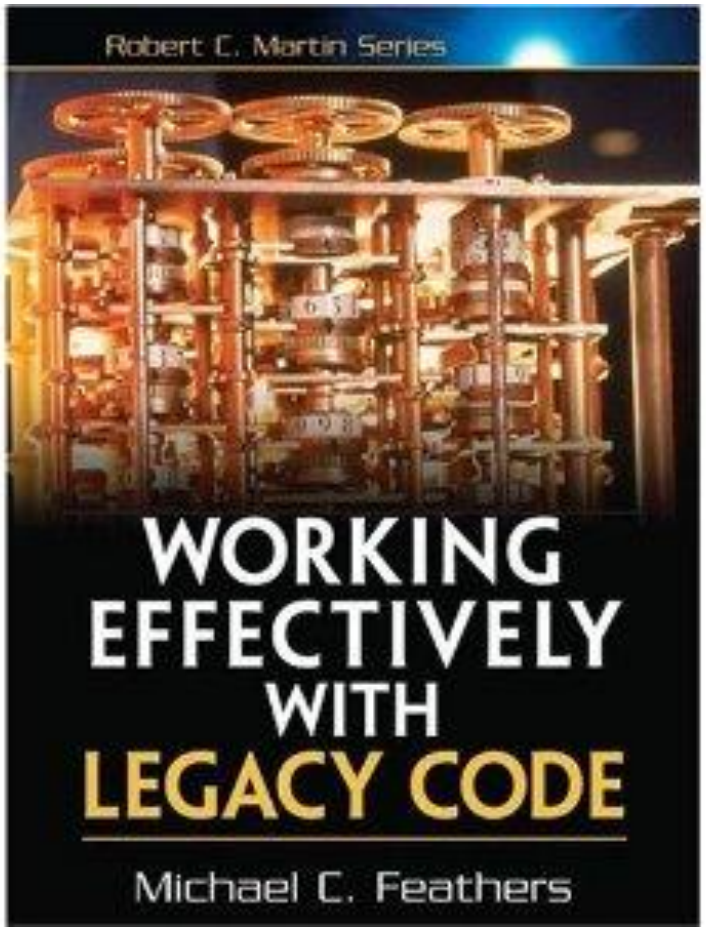
No technique can survive inadequately trained developers - Steve Freeman

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

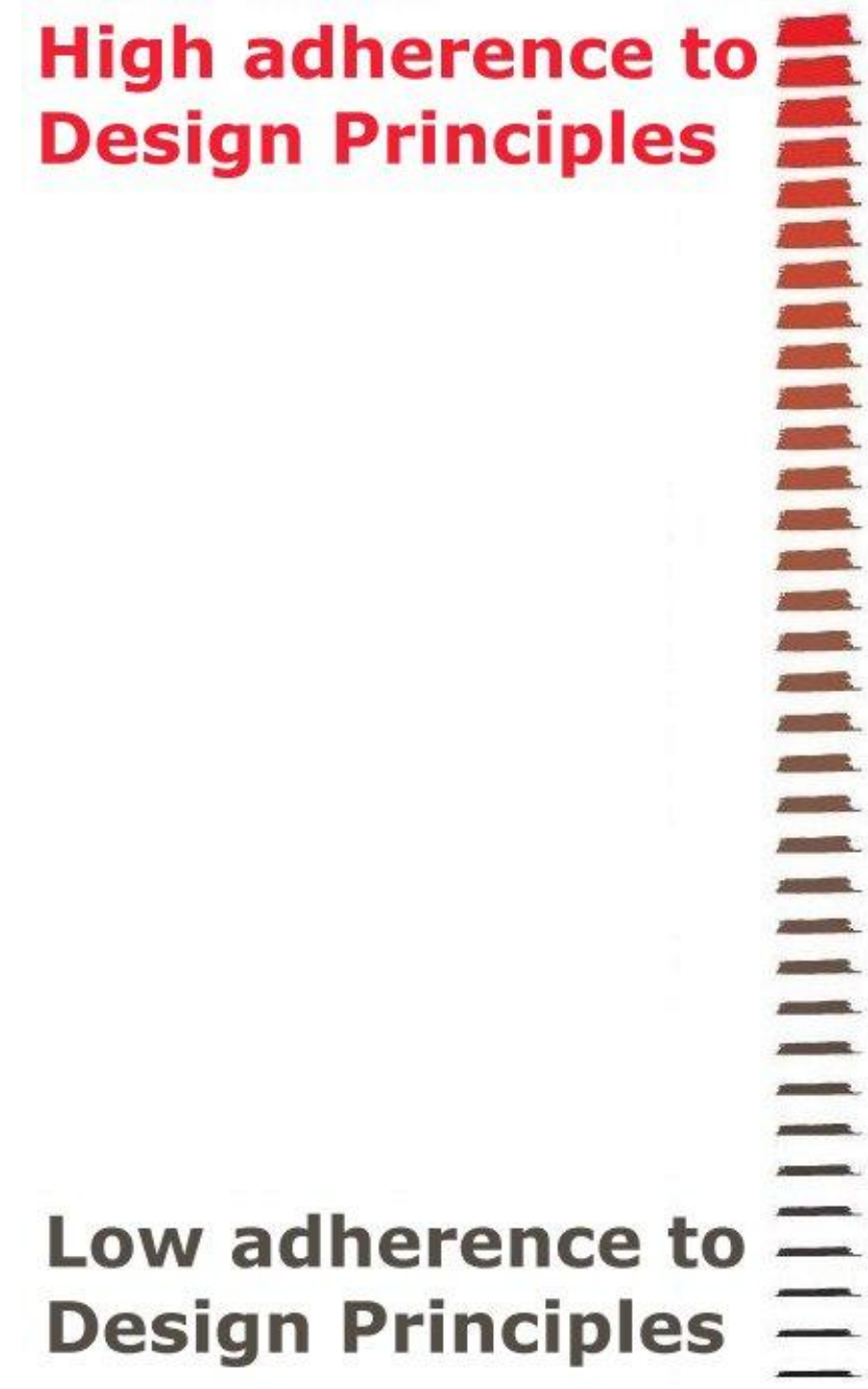
SRP – ISP

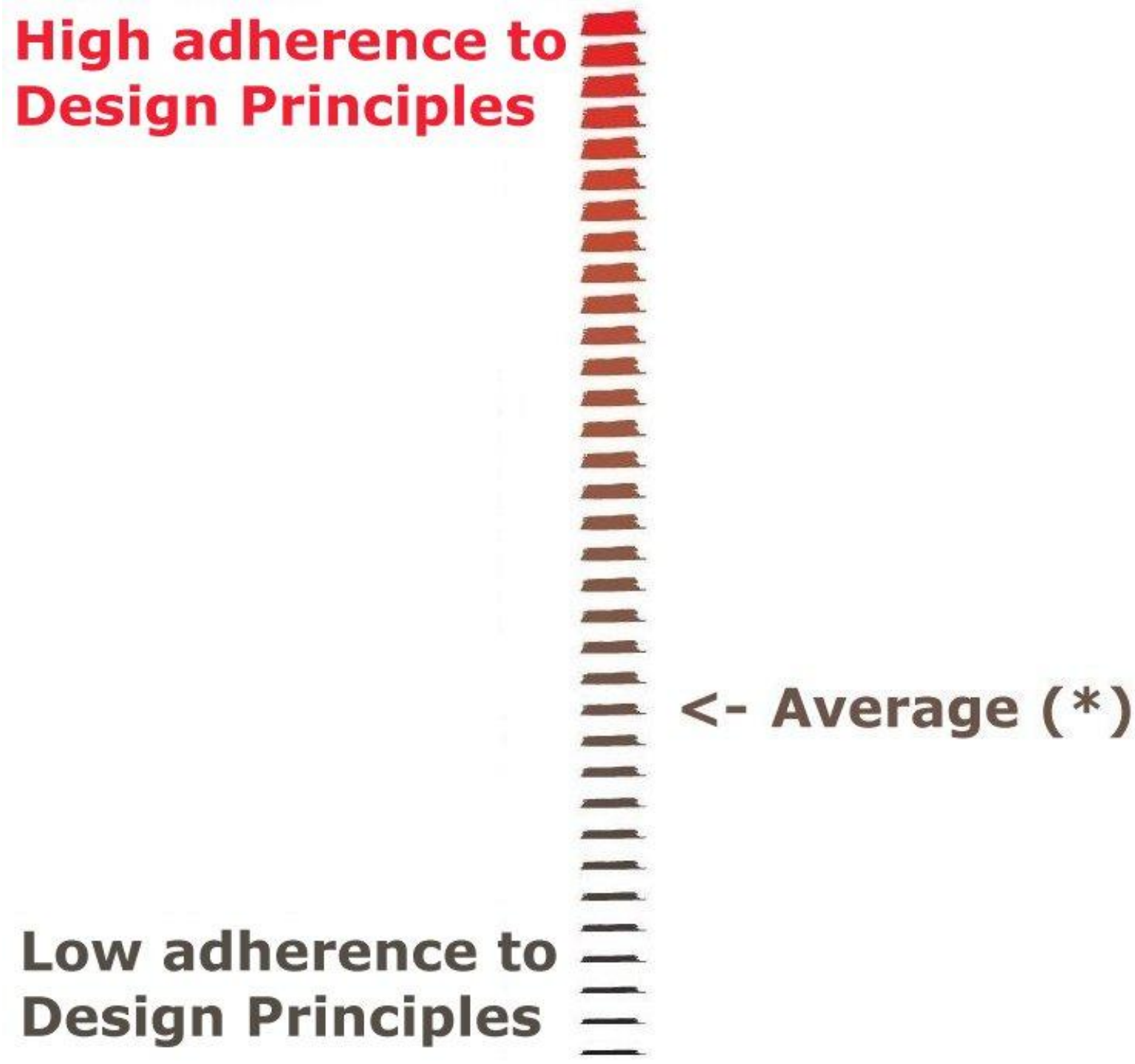
LSP

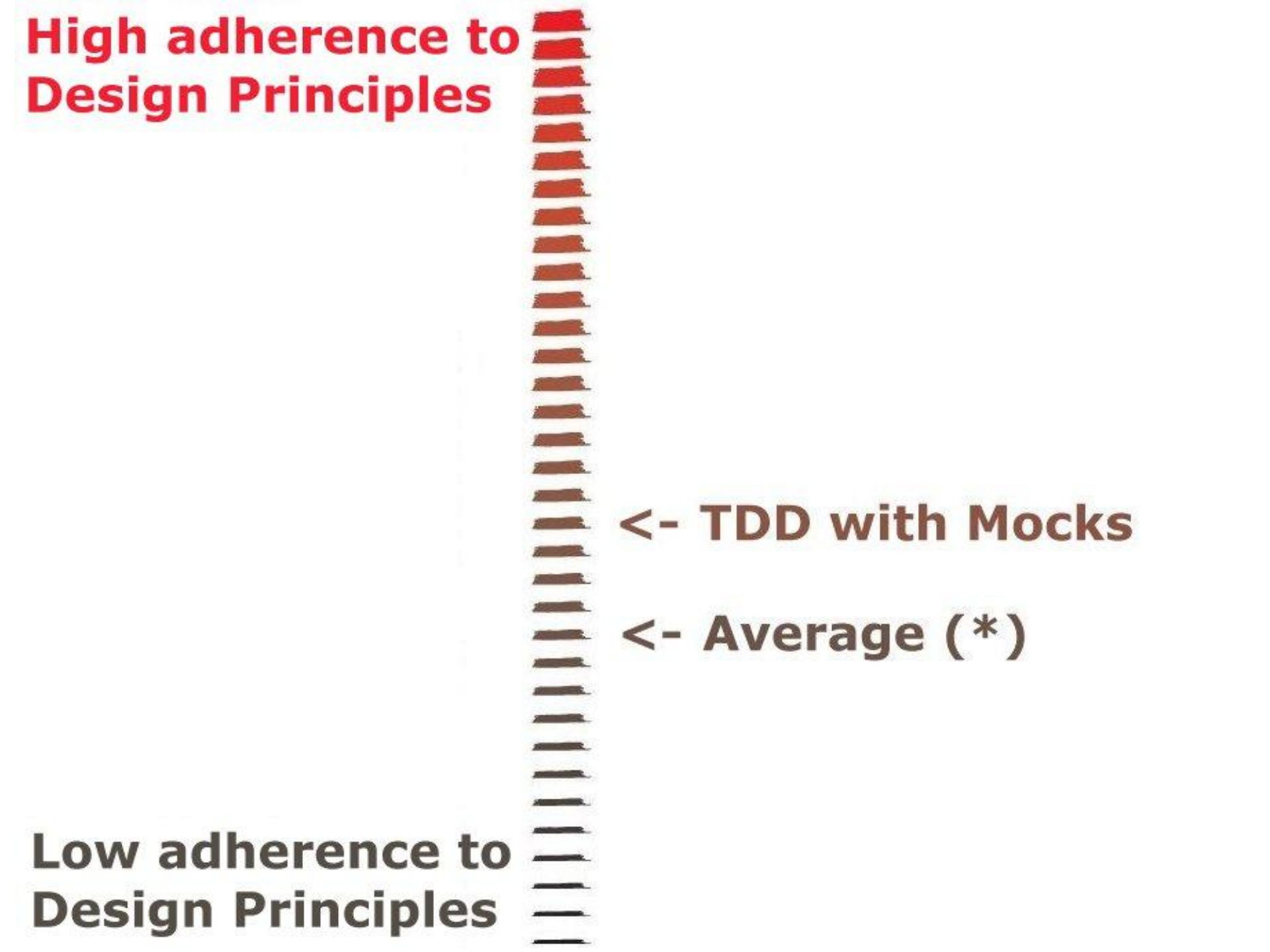
LoD

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

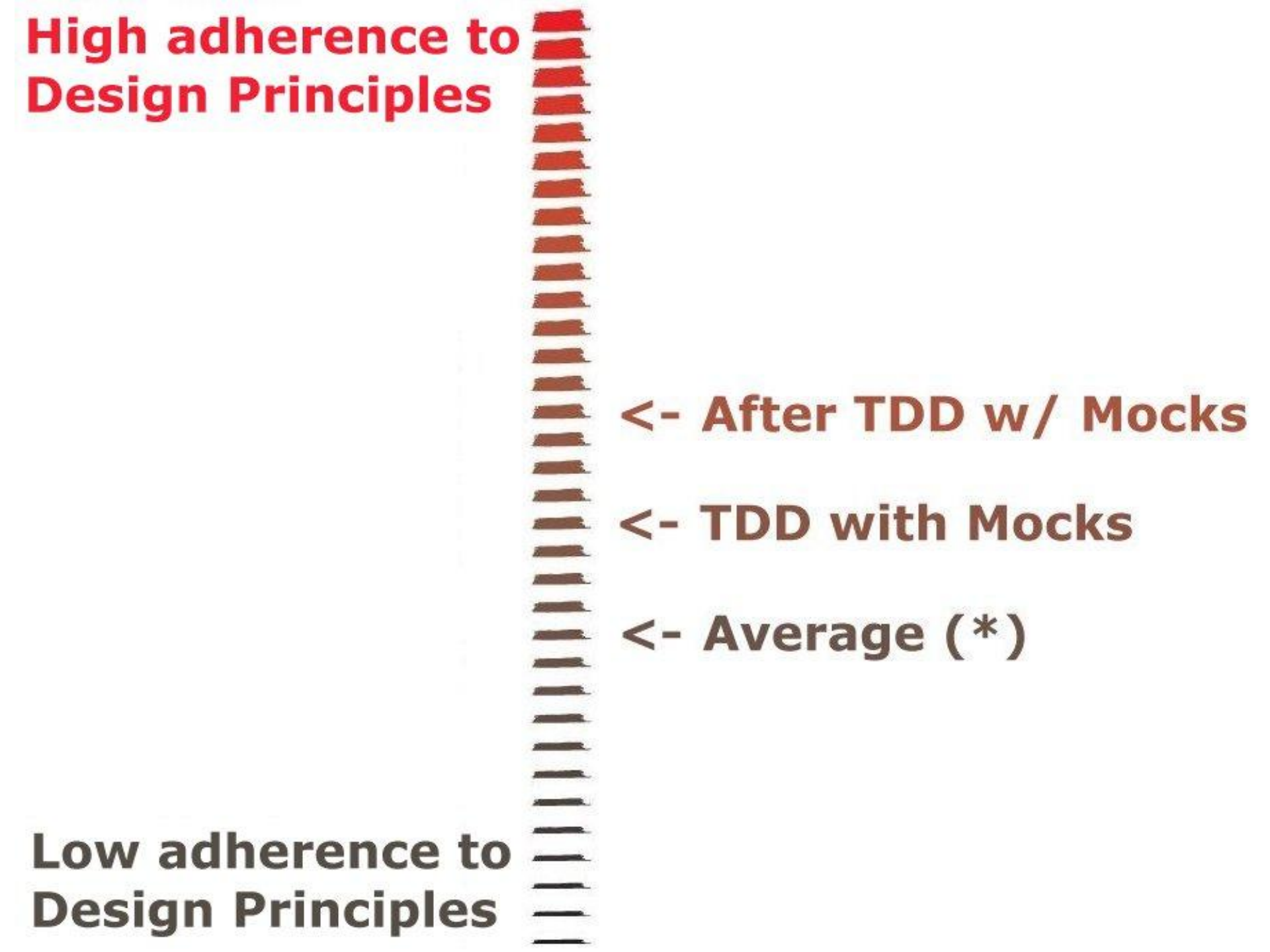
Co-Evolution / Attractors / Barriers

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

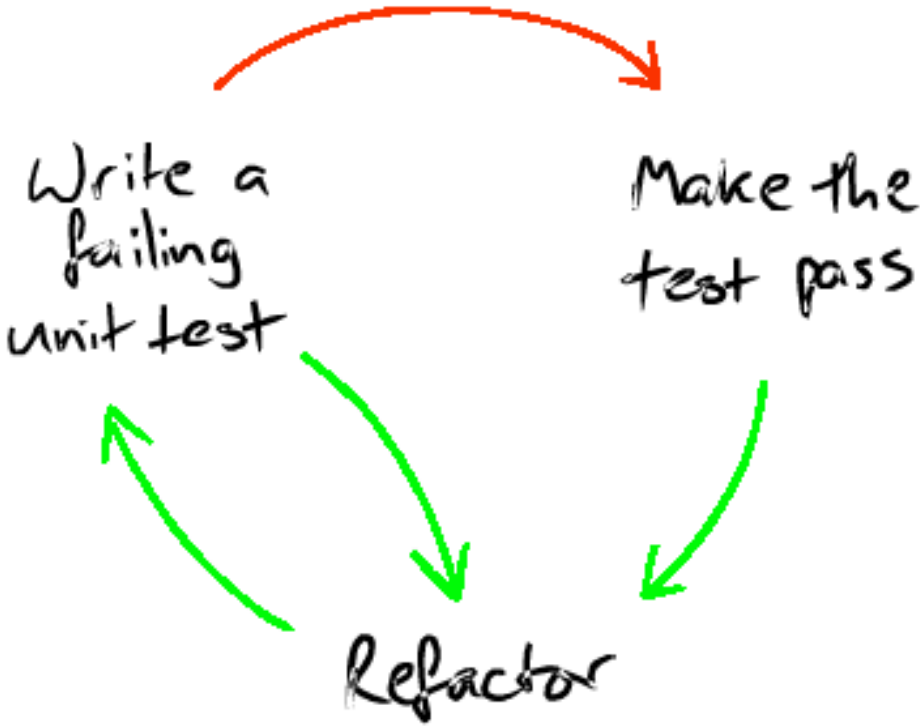

High adherence to Design Principles
<- After TDD w/ Mocks
<- TDD with Mocks
<- Average (*)
Low adherence to Design Principles

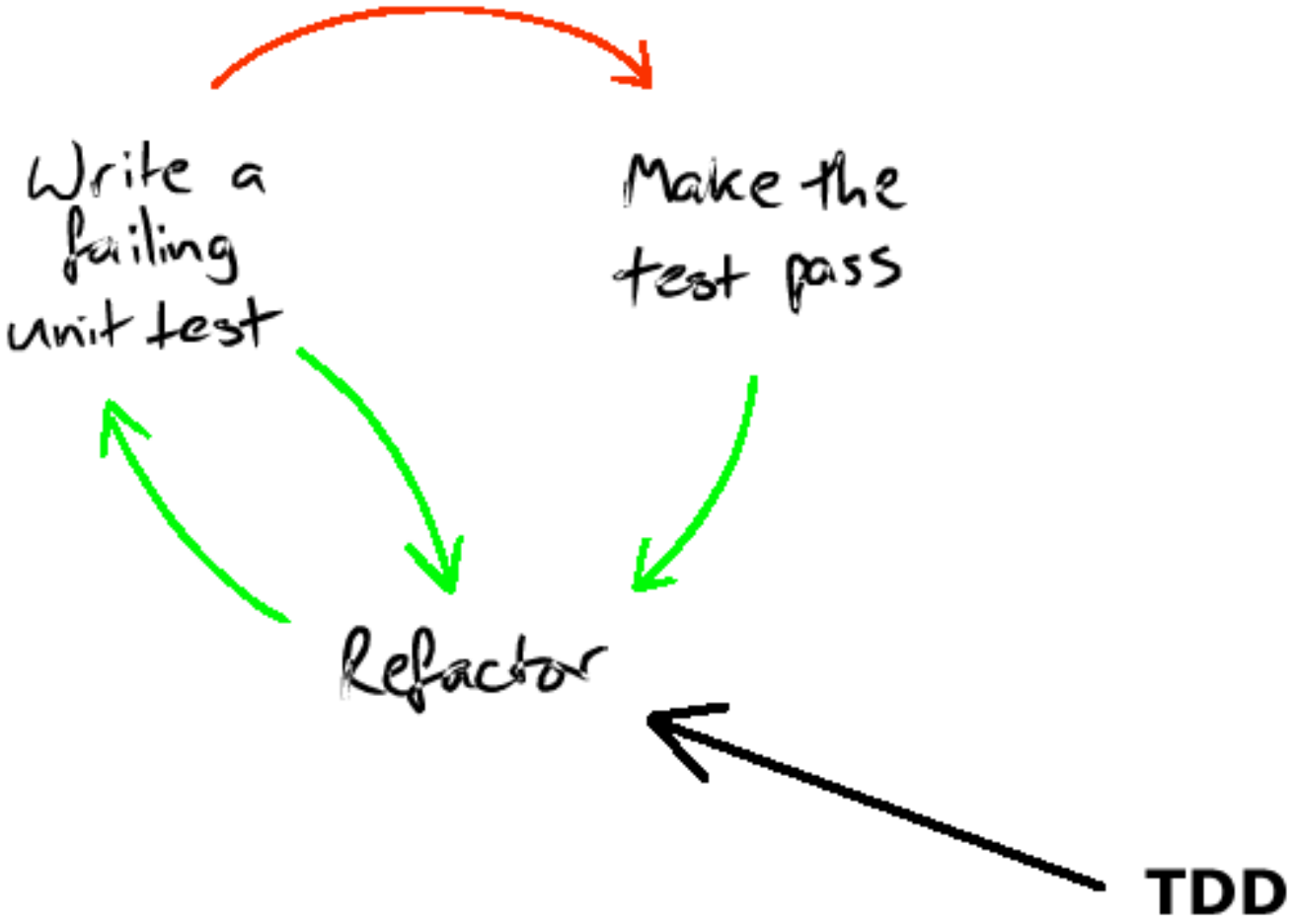
Norwegian Developer Conference 2010

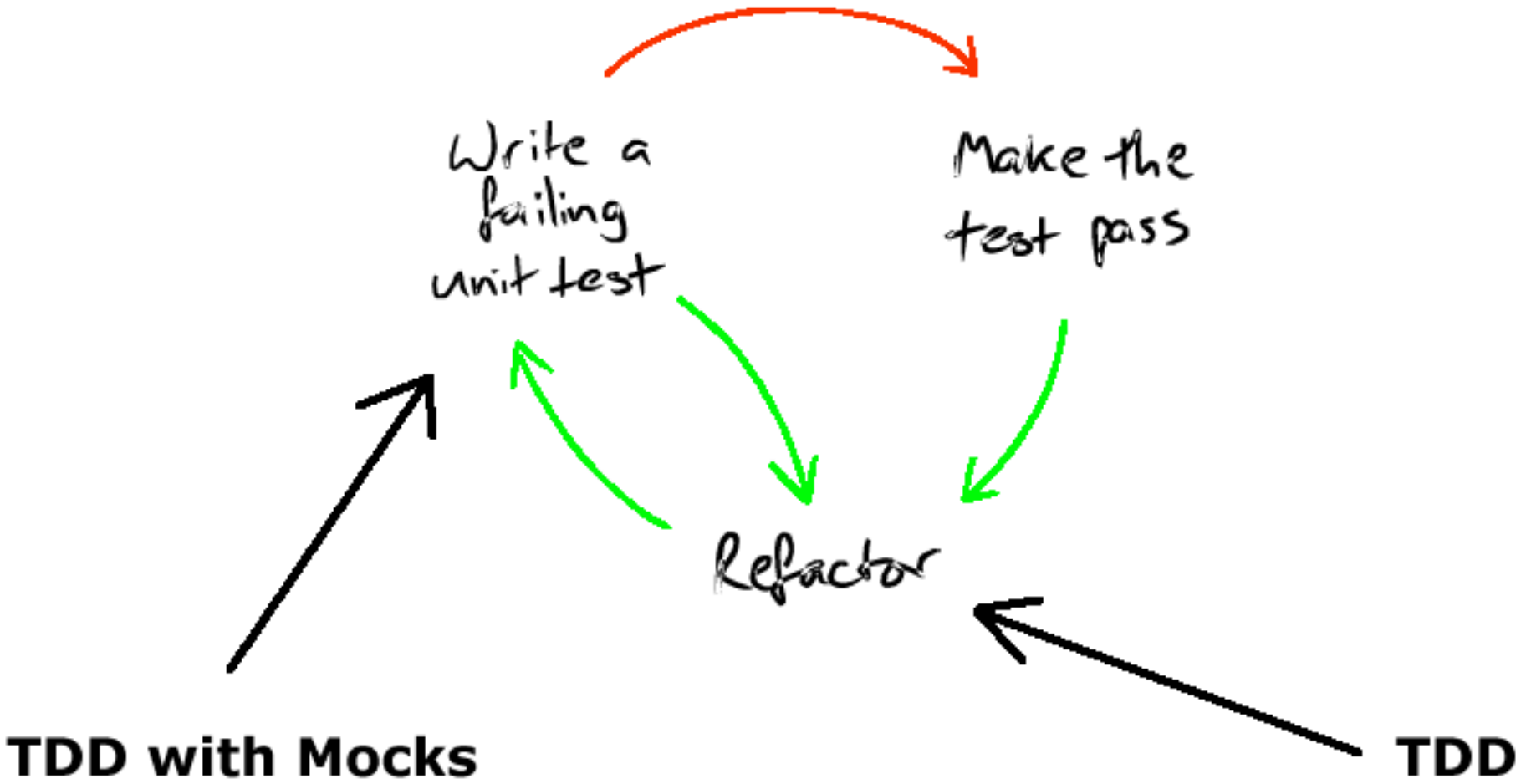
Session ‘The Deep Synergy Between Testability and Good Design’

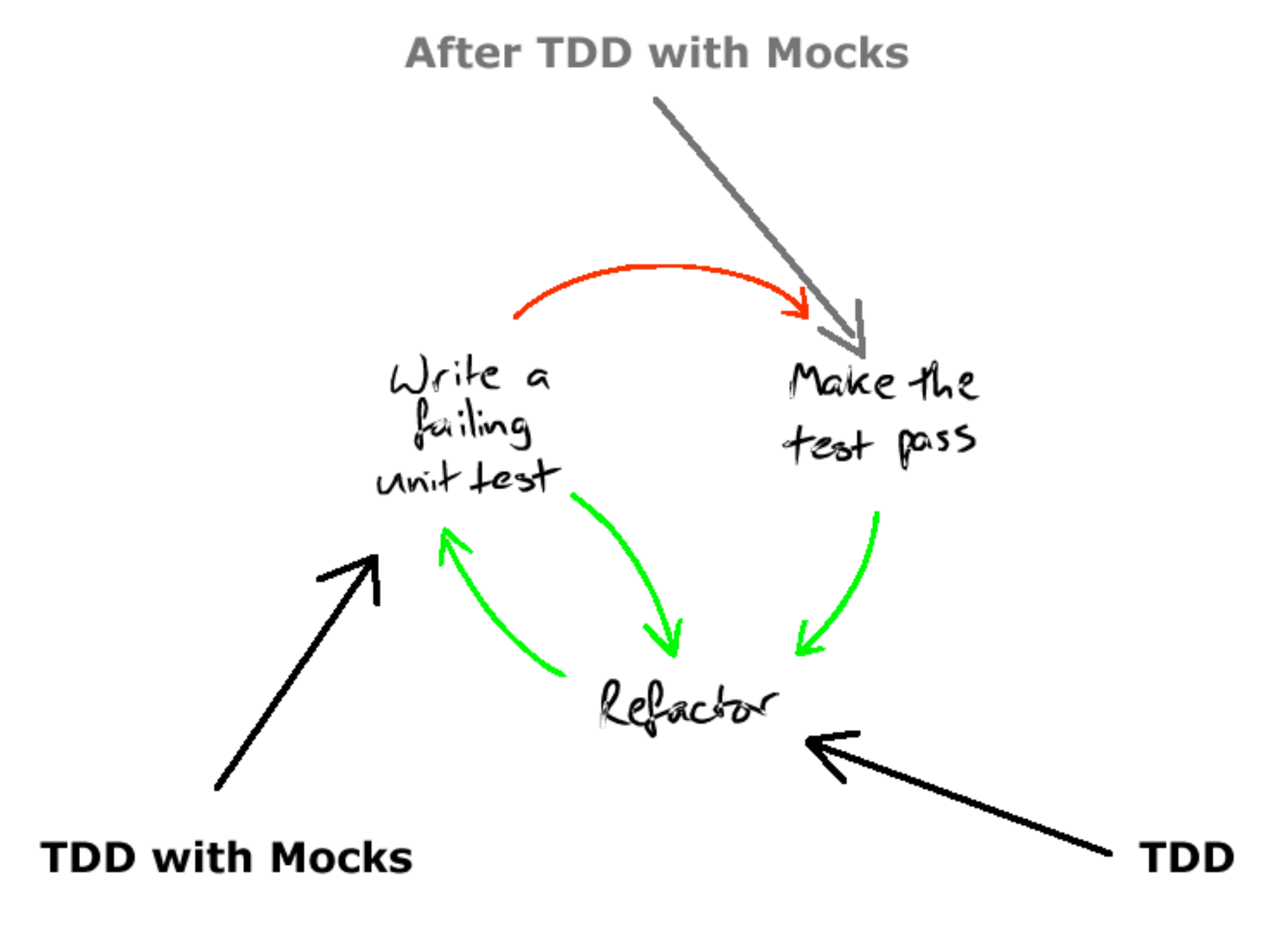
Michael Feathers:

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


Write a failing unit test
Make the test pass
Refactor
TDD


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

Formula 1
Formula 1