Refactoring Legacy Code

@alvarobiz, 2015-06-29

INTRODUCTION TO LEGACY CODE

What is legacy code

- *[…] legacy code as code without tests. It is a good working definition, and it points to a solution […]*
- *M Feathers, in the preface of Working effectively with legacy code*

What is legacy code (II)

- *Legacy code is code without tests that provide trust to all your stakeholders*

What is tested code

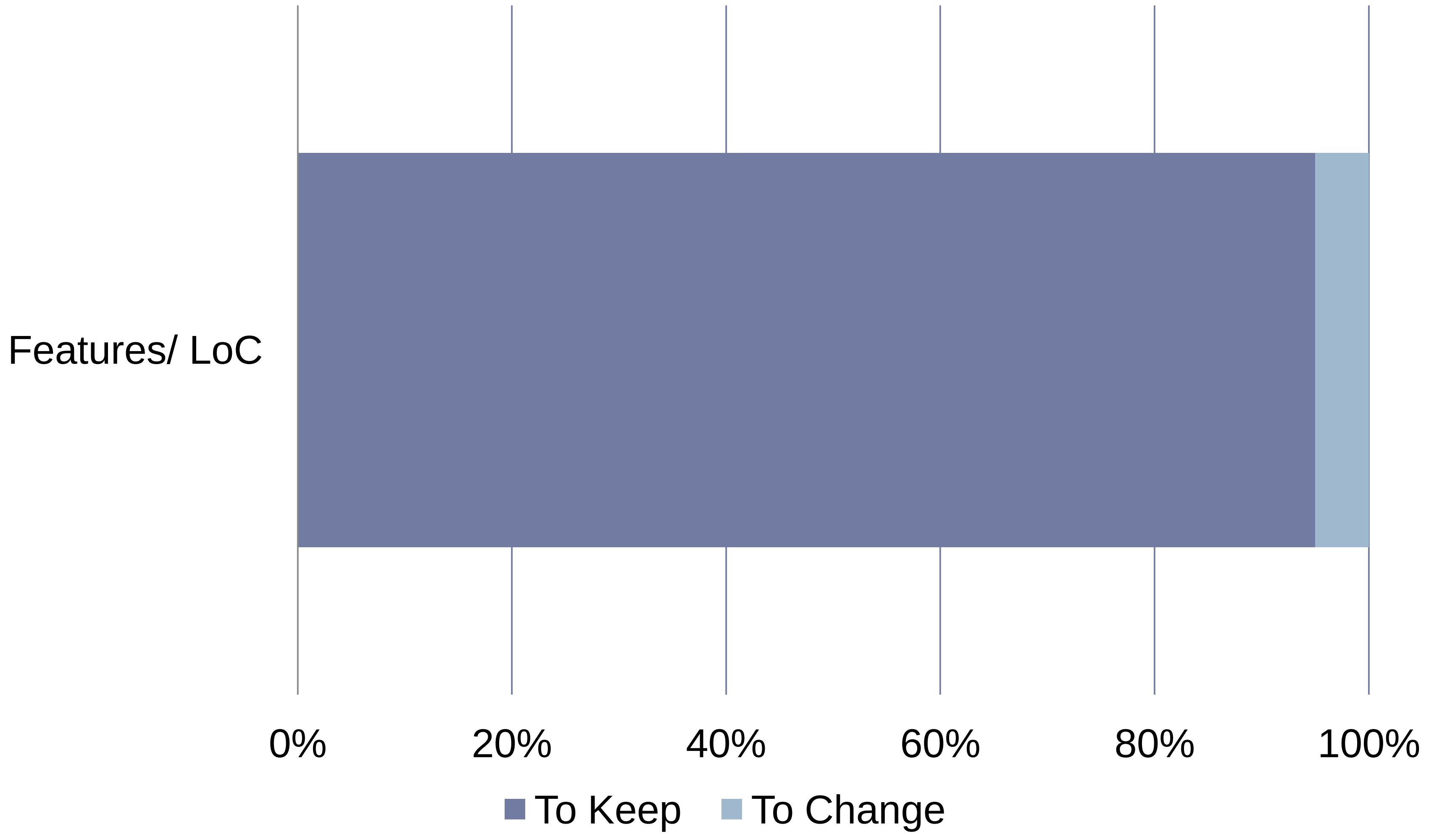
- *Trust (reliable)*
- *Cheap to modify*

CHANGING
LEGACY CODE

How to get from A to B

- *Shortest*
- *Safest*

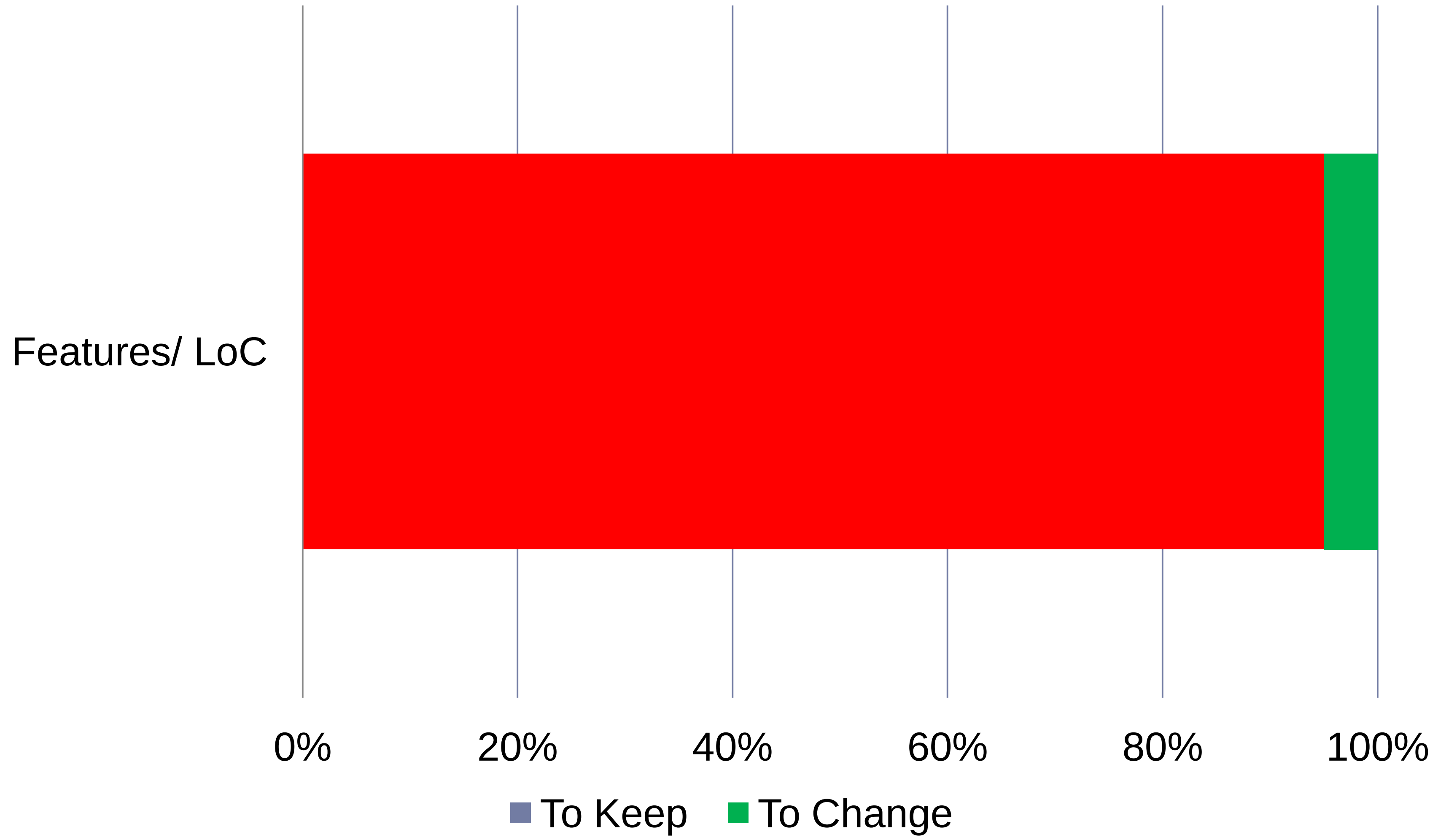
Any long-lived software



How to change from A to B

- *Keep old, risk new*
- *Risk old, keep new*

Any long-lived software (II)



THE LEGACY CODE CHANGE ALGORITHM

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Source: [Feathers04], Ch 2: Working with Feedback

Legacy Code Change, example

- Example: Ugly Trivia
- We want to remove the direct printing to the console (`System.out.printXXX`)

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Change points

- All direct invocations to System.out.printXXX

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Test points

- The console output

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Breaking dependencies

- Are the execution reproducible?

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Writing tests

- Capture the console output
- Save and verify it automatically

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Making changes

- Global search and replace “System.out.println” for “log” into an object
- Verify, commit

Refactoring

- Inject the logging collaborator
- Verify, commit

Refactoring

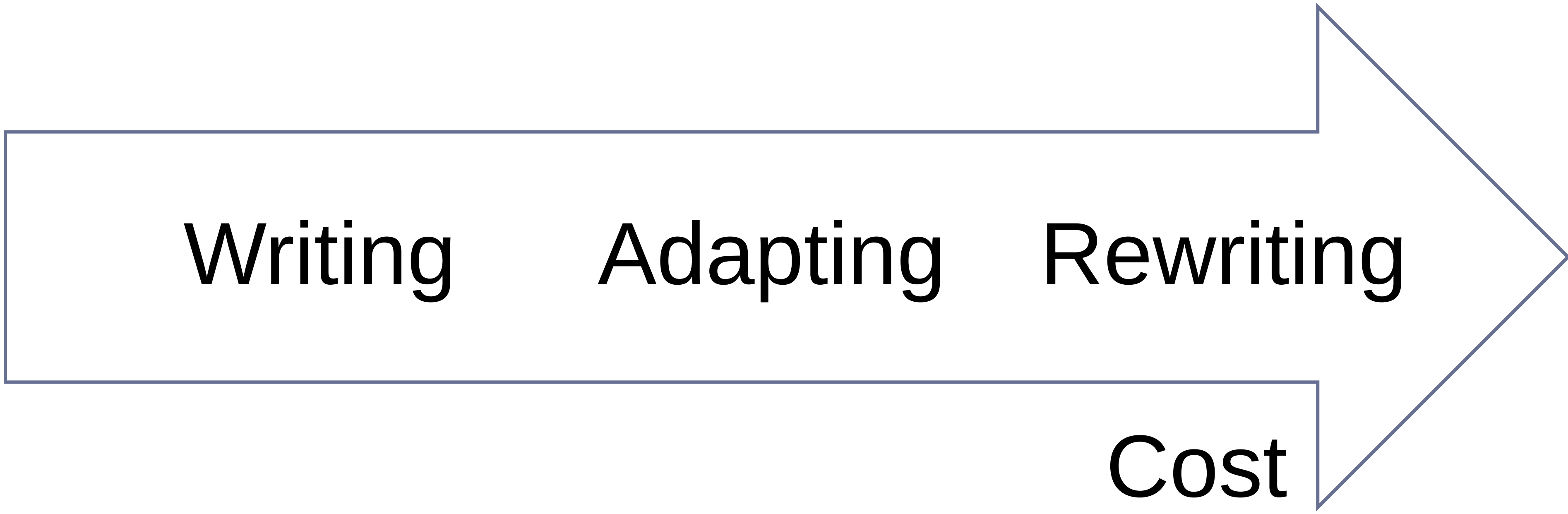
- DRY on the messages
- Verify, commit

The Legacy Code Change Algorithm

- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

Source: [Feathers04], Ch 2: Working with Feedback

The Legacy Code Change Algorithm



UNDERSTANDING LEGACY CODE

Sensing

- *[S]ense when we can’t access values our code computes*

Source: [Feathers04]:
Chapter 3: Sensing and Separation

Separating

- *[S]eparate when we can’t even get a piece of code into a test harness to run.*

Source: [Feathers04]:
Chapter 3: Sensing and Separation

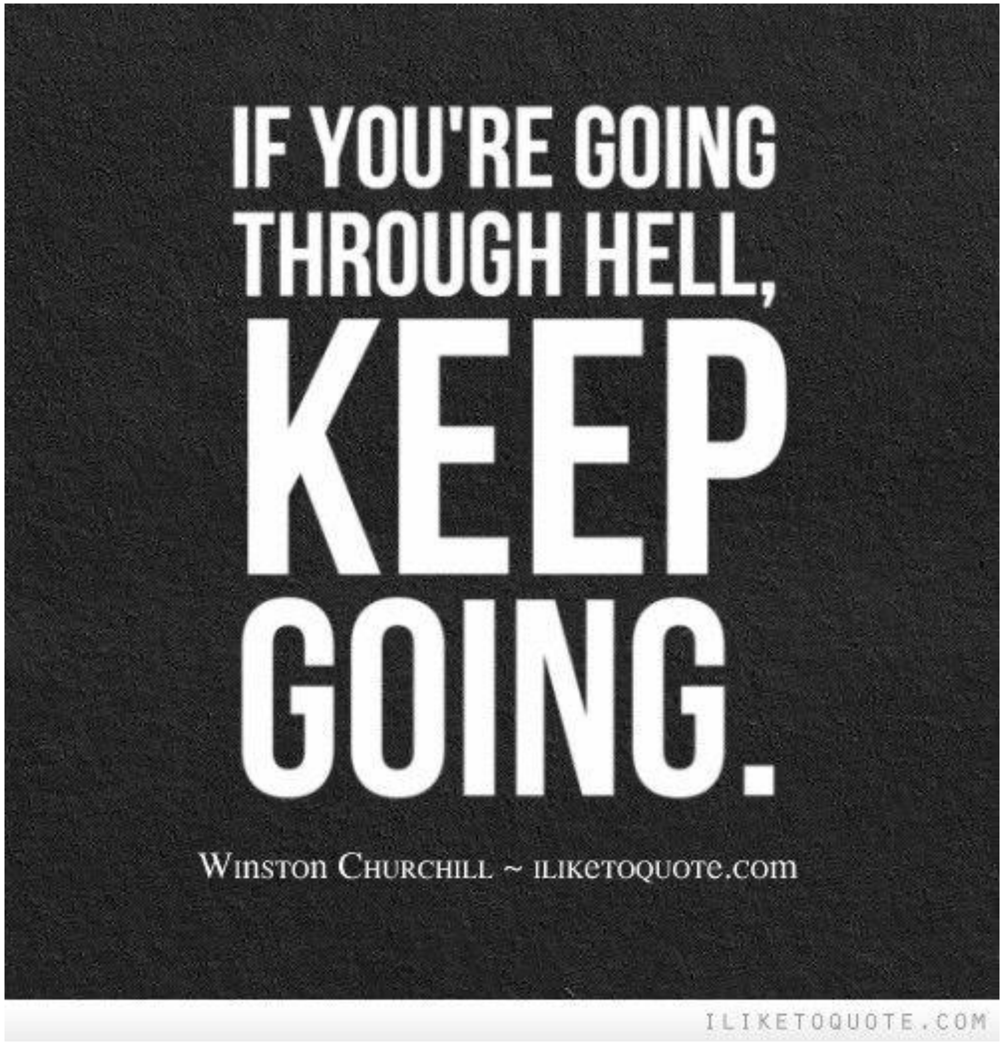
Solutions

- Fake Objects
- Mock Objects
- Seams
- Dependency breaking

Source: [Feathers04]:
Chapter 3: Sensing and Separation

EXPERIENCES WITH LEGACY CODE

Motivation (I)


IF YOU'RE GOING THROUGH HELL, KEEP GOING.

WINSTON CHURCHILL ~ ILIKETOQUOTE.COM

ILIKETOQUOTE.COM

Motivation (II)

- *I have nothing to offer but blood, toil, tears and sweat.*

  W. Churchill

How to get from A to B

- One step at a time

How to get from A to B

- Low-hanging fruit

How to get from A to B

- Refactor relentlessly

How to get from A to B

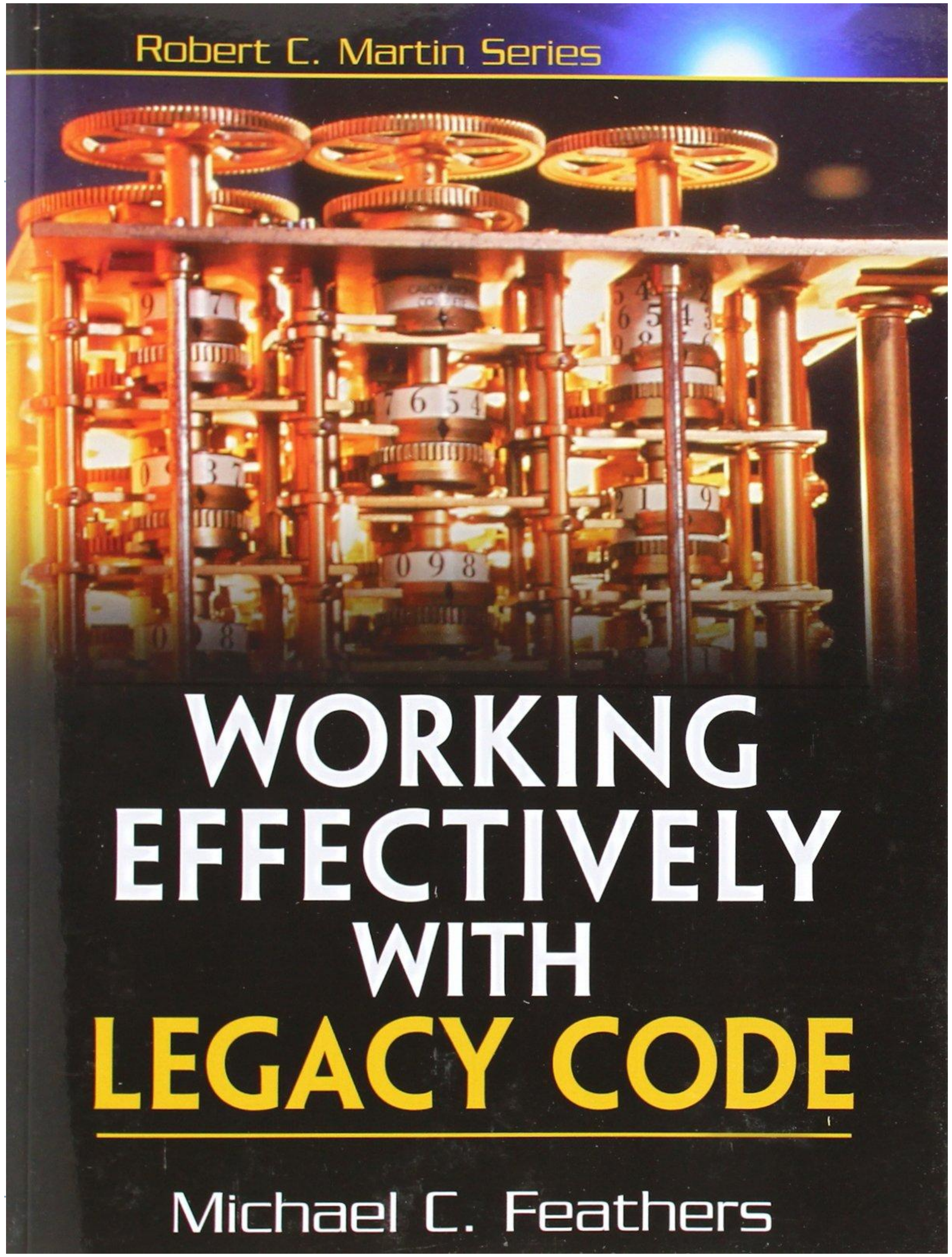
- Do not stop the presses, prefer along the way

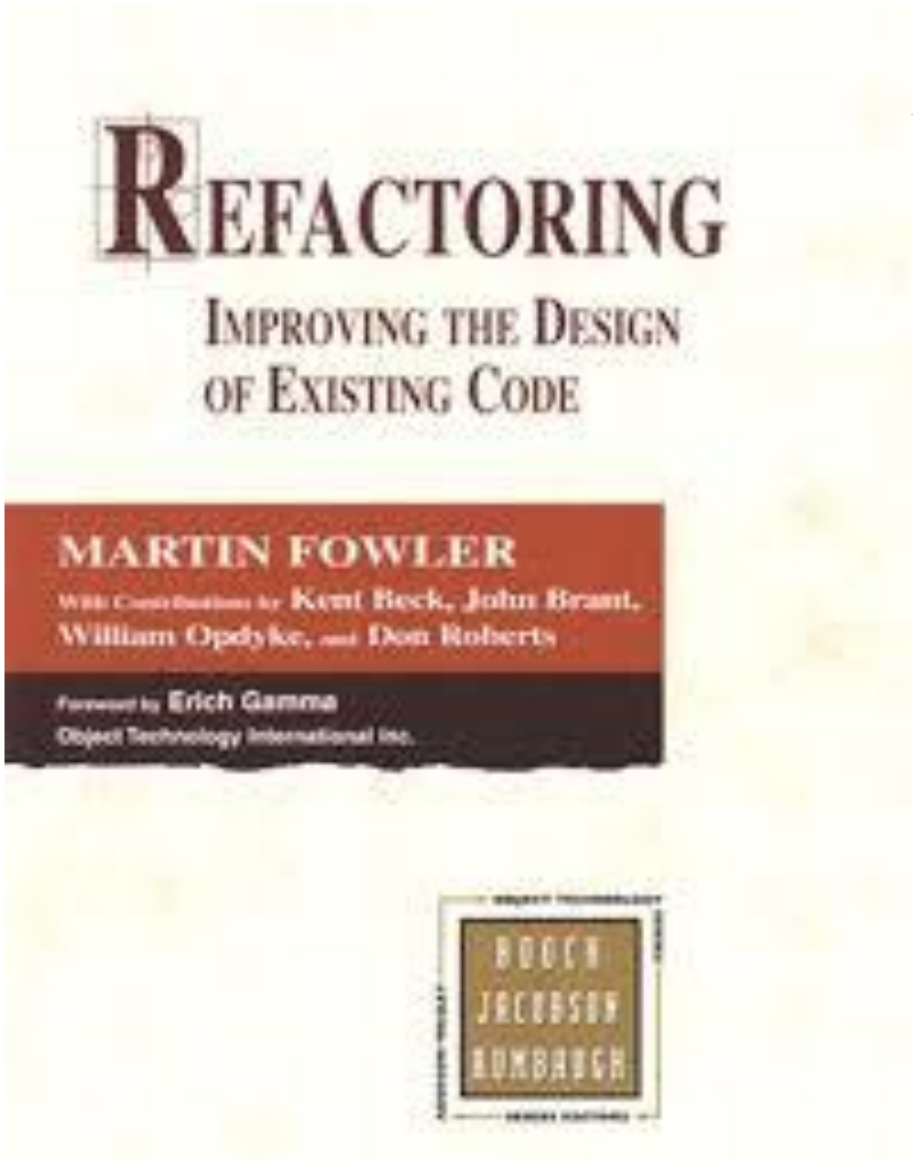
How to get from A to B

- Make it usable, then forget for a while

IF YOU HAVE TO SPEND 90% OF YOUR TIME BABYSITTING LEGACY CODE
INSTRUCTO
YOU'RE GONNA HAVE A BAD TIME
livememe.com

GETTING (PROFESSIONAL) HELP


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers



REFACTORING

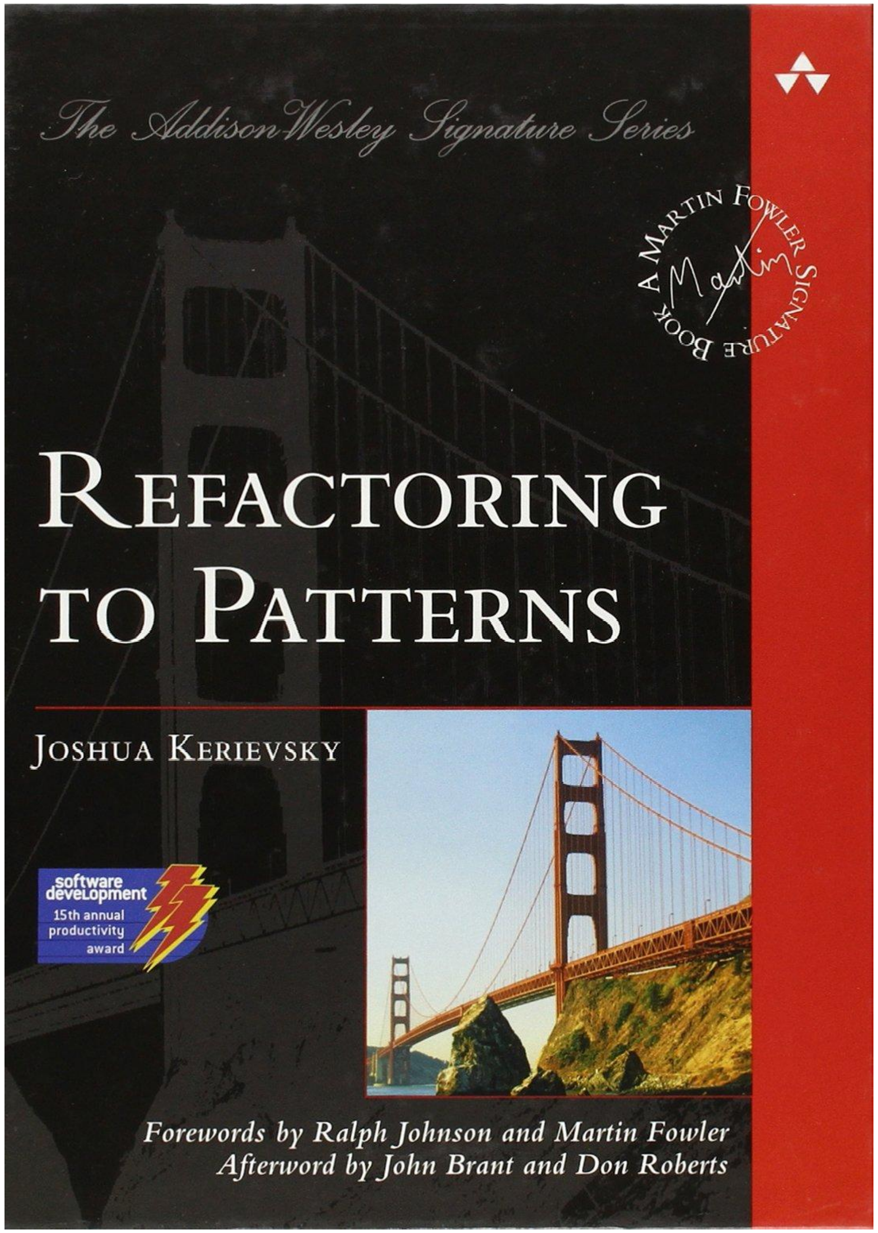
IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER

With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts

Foreword by Erich Gamma
Object Technology International Inc.

BOOCH JACOBSON RUMBAUGH



The Addison Wesley Signature Series

A MARTIN FOWLER SIGNATURE BOOK

REFACTORING TO PATTERNS

JOSHUA KERIEVSKY

software development 15th annual productivity award

Forewords by Ralph Johnson and Martin Fowler
Afterword by John Brant and Don Roberts

Bibliography

- [Feathers04]: Feathers, M. **Working Effectively with Legacy Code**, ISBN-13**:** 007-6092025986. Amazon
- [Refactoring]: Fowler, M with Beck, Brant, Opdyke, and Roberts. Refactoring: Improving the Design of Existing Code,ISBN-13: 978-0201485677 Official page
- [Kerievsky04]: Kerievsky, J. Refactoring to Patterns. ISBN-13: 078-5342213355 Official page