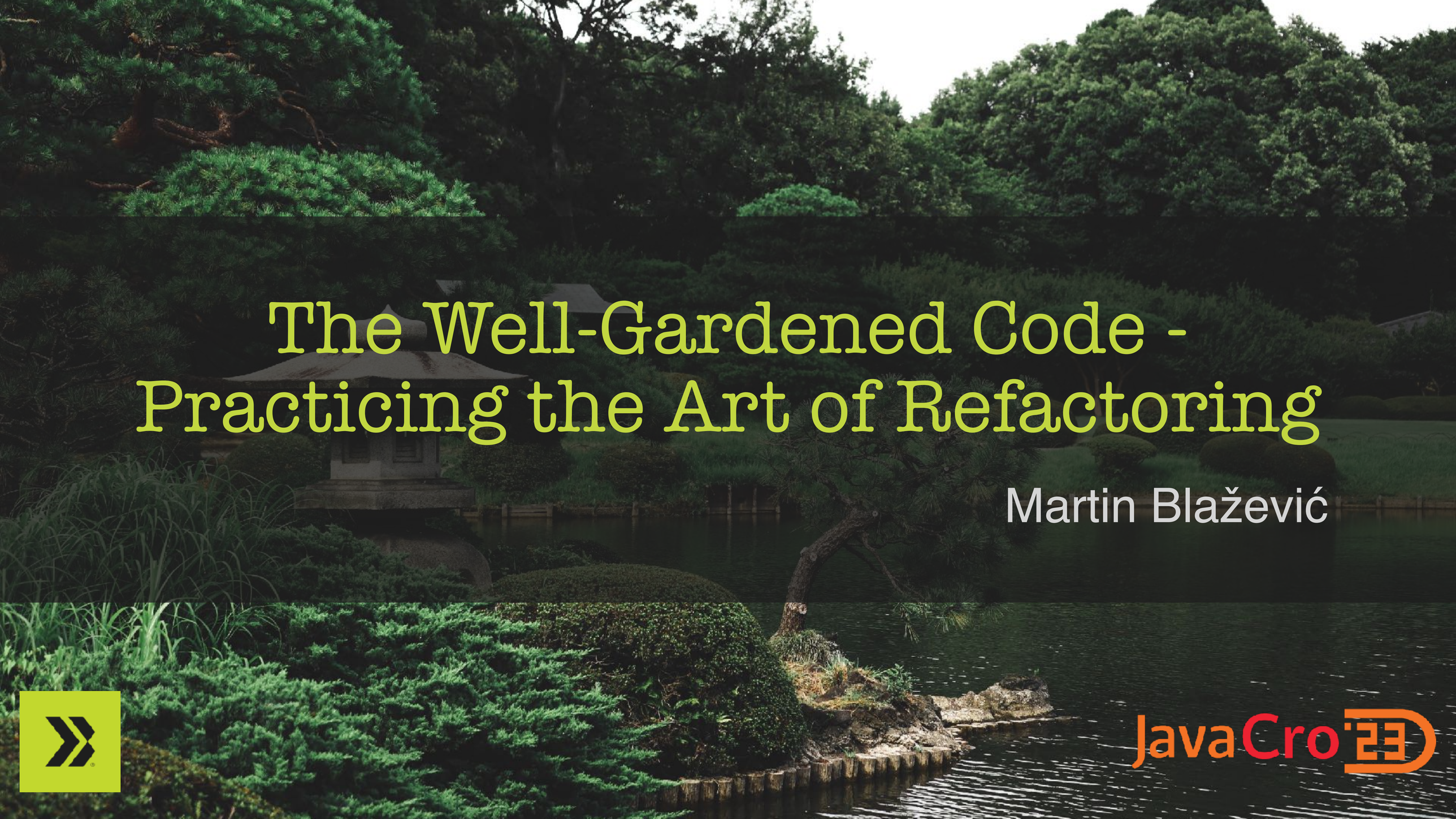


The Well-Gardened Code - Practicing the Art of Refactoring

Martin Blažević

JavaCro'23

Legacy

"Everyone must leave something behind when he dies, my grandfather said. A child or a book or a painting or a house or a wall built or a pair of shoes made. Or a garden planted. Something your hand touched some way so your soul has somewhere to go when you die, and when people look at that tree or that flower you planted, you're there."

Fahrenheit 451, Ray Bradbury

Wanna hear something scary?

Legacy Code

Legacy Code

Software Entropy

Legacy Code

"Walking on water and developing software from a specification are easy if both are frozen."

Edward V. Berard

Legacy Code

"All software becomes legacy software as soon as it's written."

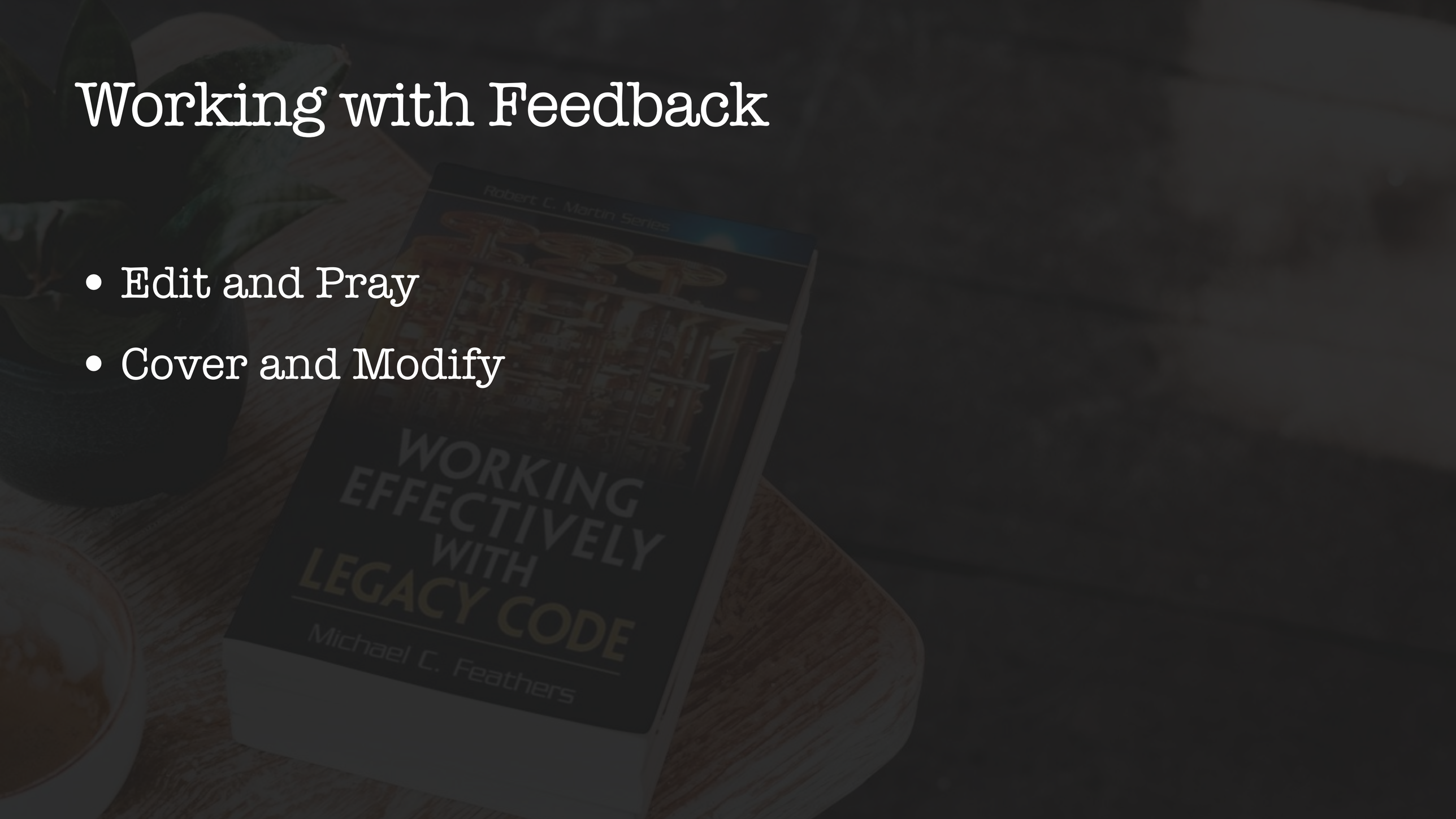
The Pragmatic Programmer

The Broken Window

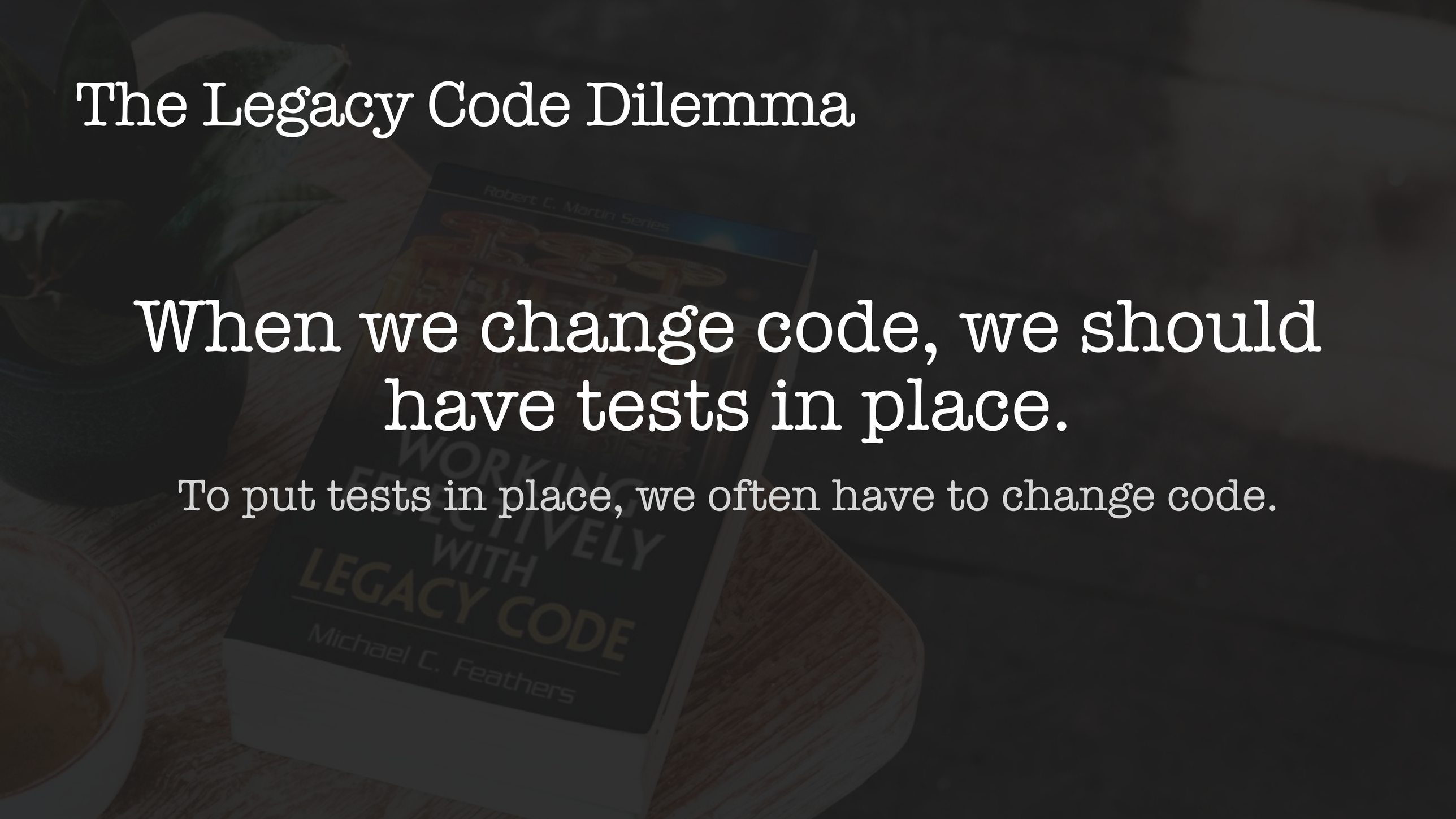
Technical Debt

Working with Feedback

- Edit and Pray
- Cover and Modify



The Legacy Code Dilemma



When we change code, we should have tests in place.

To put tests in place, we often have to change code.

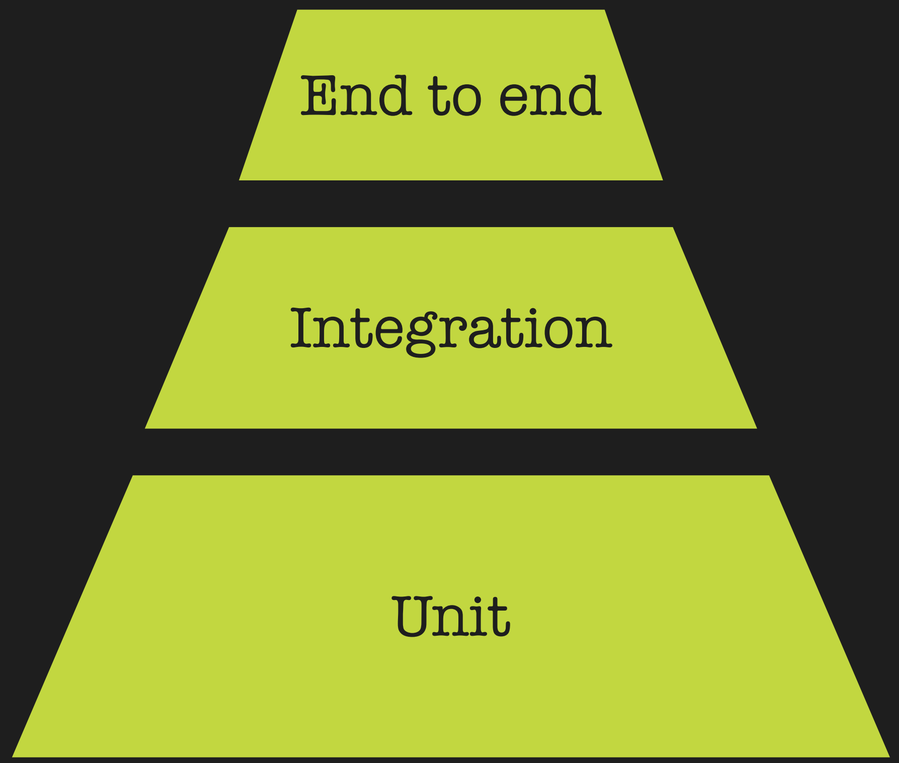
The Legacy Code Change Algorithm

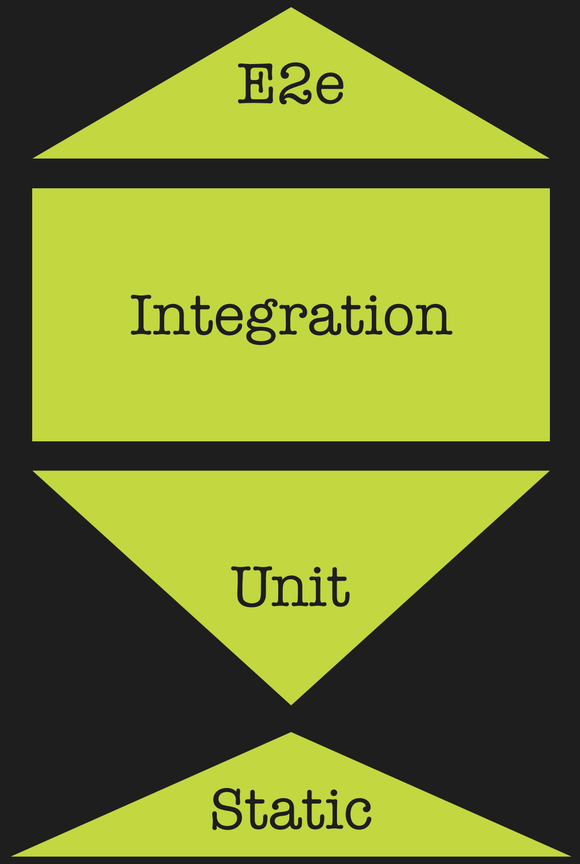
- Identify change points.
- Find test points.
- Break dependencies.
- Write tests.
- Make changes and refactor.

characterization tests

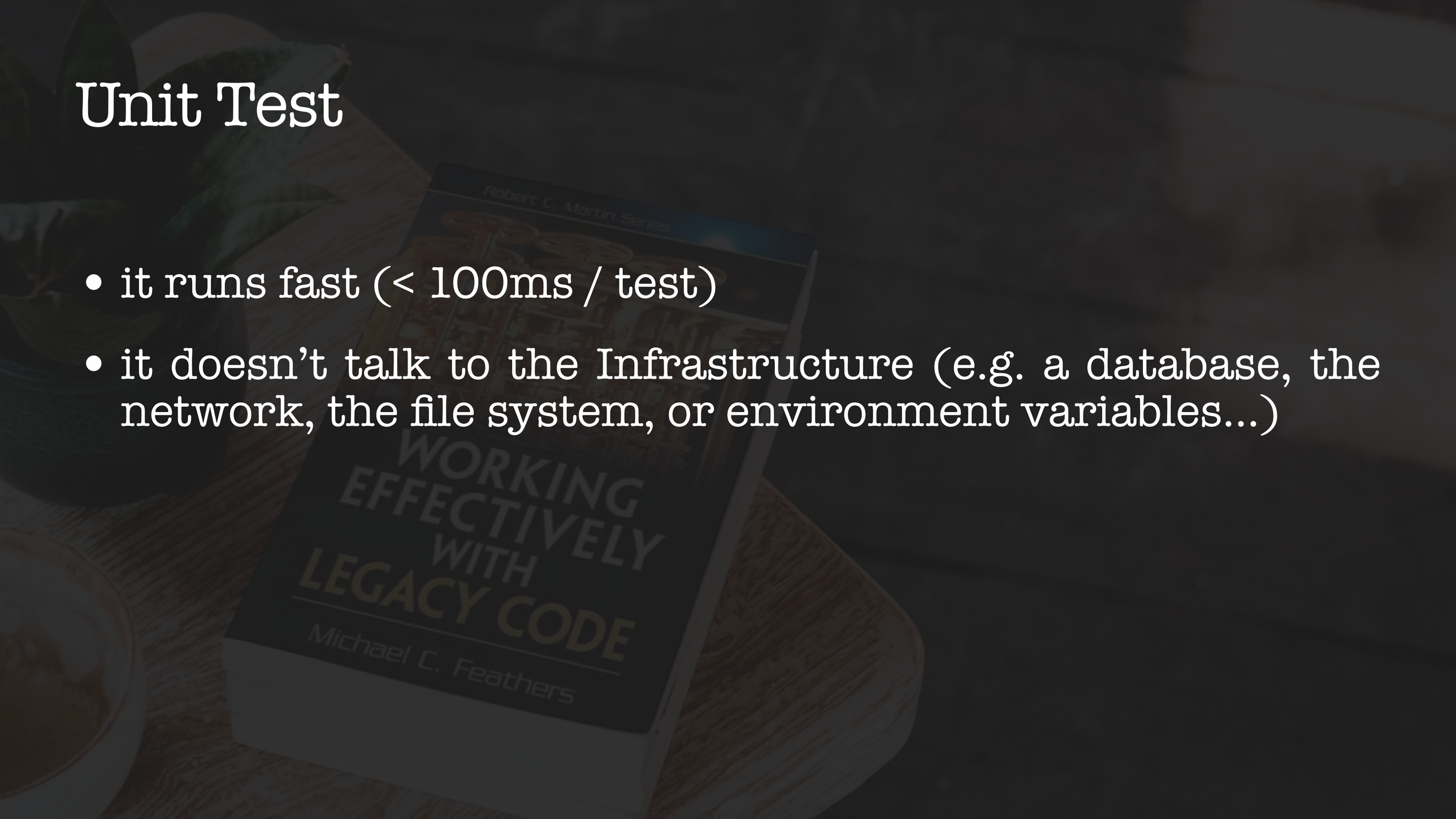
```
assertEquals(9, calculate(2, 2));
assertEquals(4, calculate(2, 2))
assertEquals(5, calculate(2, 3))
```

Pyramid or Trophy?





Unit Test



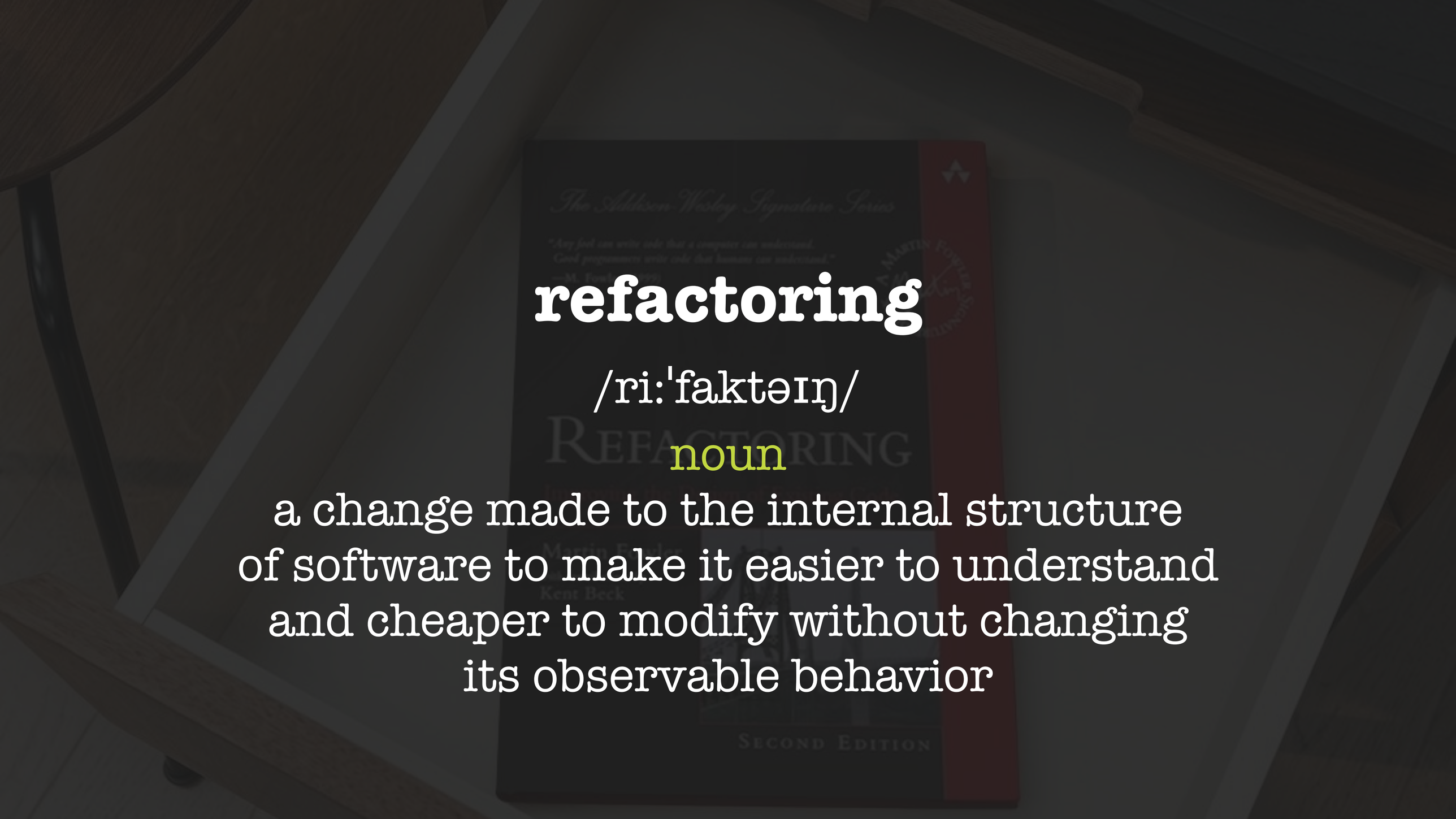
- it runs fast (< 100ms / test)
- it doesn't talk to the Infrastructure (e.g. a database, the network, the file system, or environment variables...)

Write Tests for People.

Gerard Meszaros

Test Early, Test Often, Test Automatically.

The Pragmatic Programmer

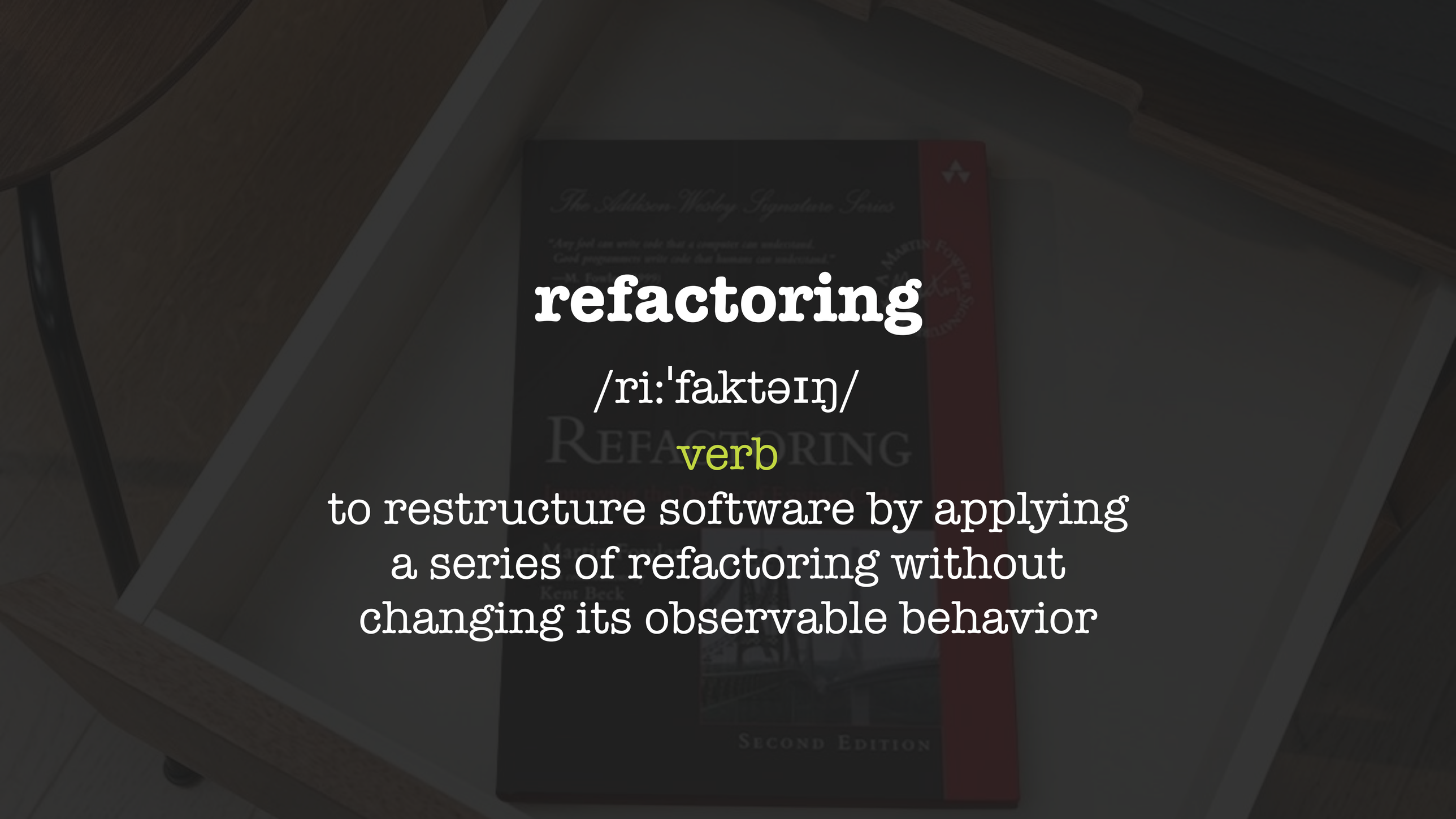


refactoring

/riːˈfaktəɪŋ/

noun

a change made to the internal structure
of software to make it easier to understand
and cheaper to modify without changing
its observable behavior



refactoring

/riːˈfaktəɪŋ/

verb

to restructure software by applying
a series of refactoring without
changing its observable behavior

change

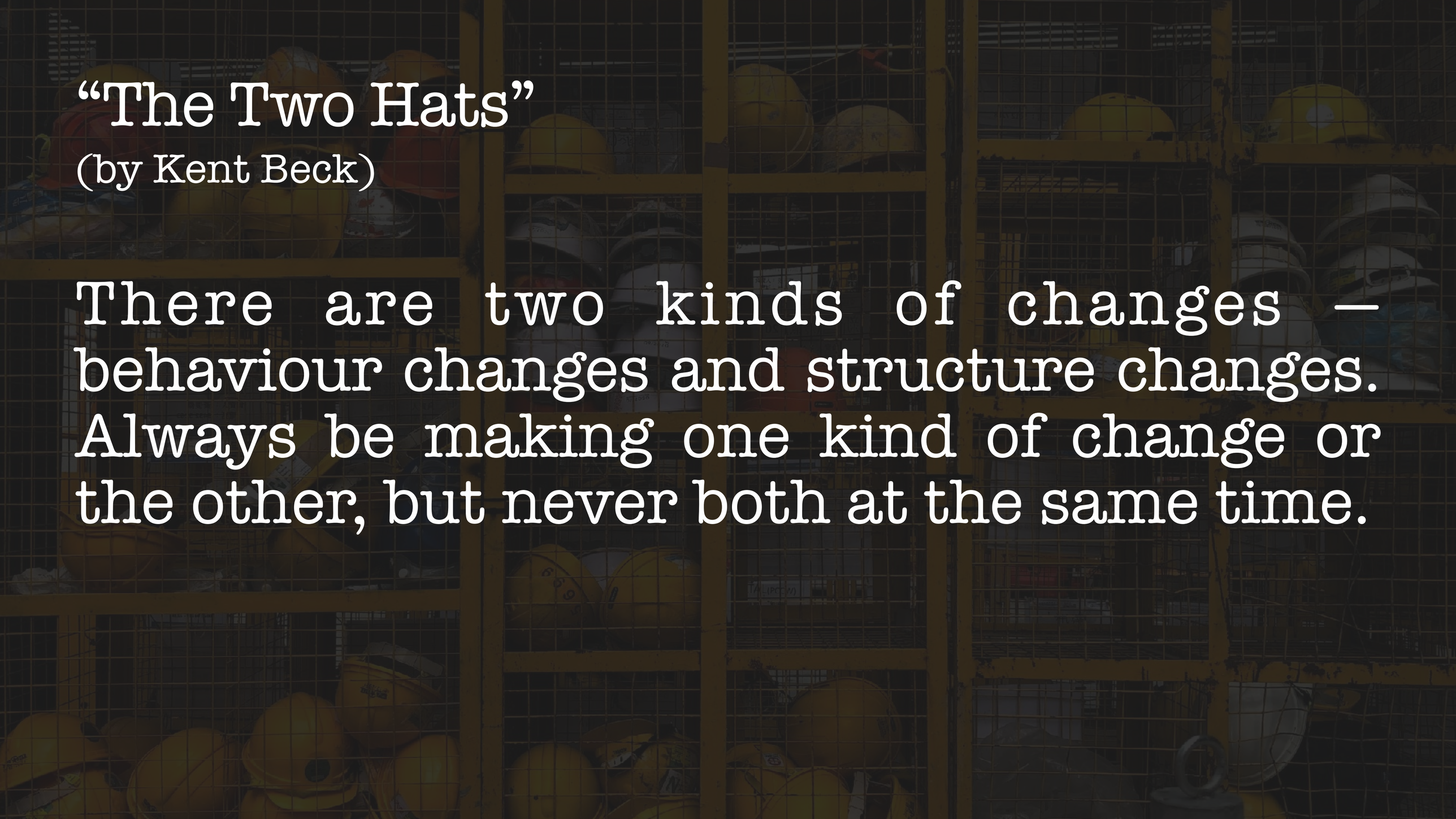
改善

good

kaj
~~kai~~zen

/ kai'zen/

noun (from Japanese)
improvement, change for the better



“The Two Hats”

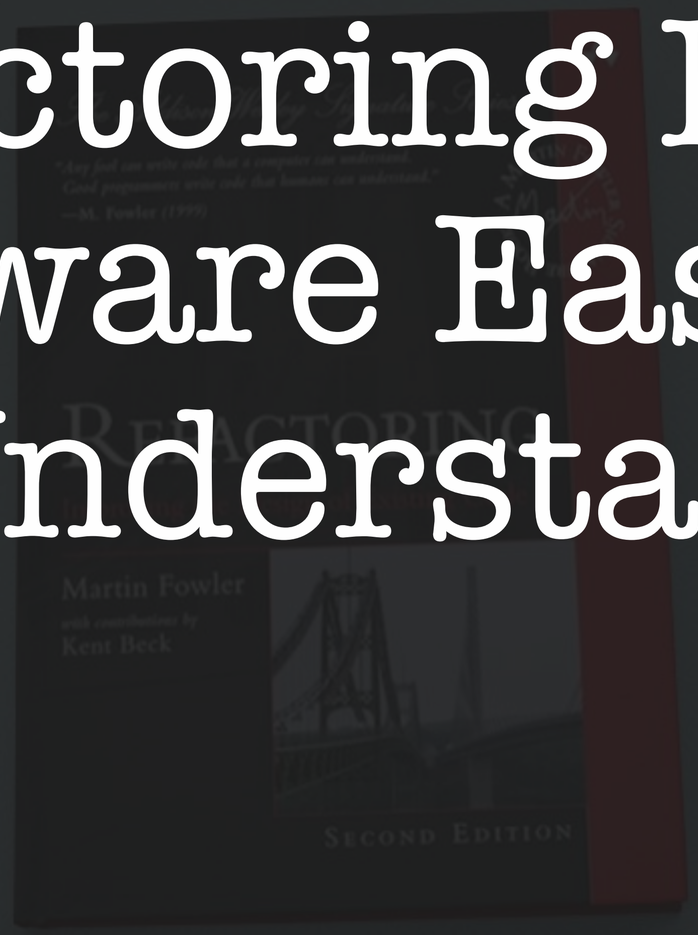
(by Kent Beck)

There are two kinds of changes — behaviour changes and structure changes. Always be making one kind of change or the other, but never both at the same time.

Why Should We Refactor?

Refactoring Improves the Design of Software

Why Should We Refactor?



Refactoring Makes Software Easier to Understand

Why Should We Refactor?

Refactoring Helps Us Find Bugs

Why Should We Refactor?

Refactoring Helps Us Program Faster

When Should We Refactor?

"The first time you do something, you just do it. The second time you do something similar, you wince at the duplication, but you do the duplicate thing anyway. The third time you do something similar, you refactor."

Don Roberts

When Should We Refactor?

Comprehension Refactoring: Making Code Easier to Understand

“By refactoring I move the understanding from my head into the code itself.”

Ward Cunningham

When Should We Refactor?

Preparatory Refactoring:
Making It Easier to Add a Feature

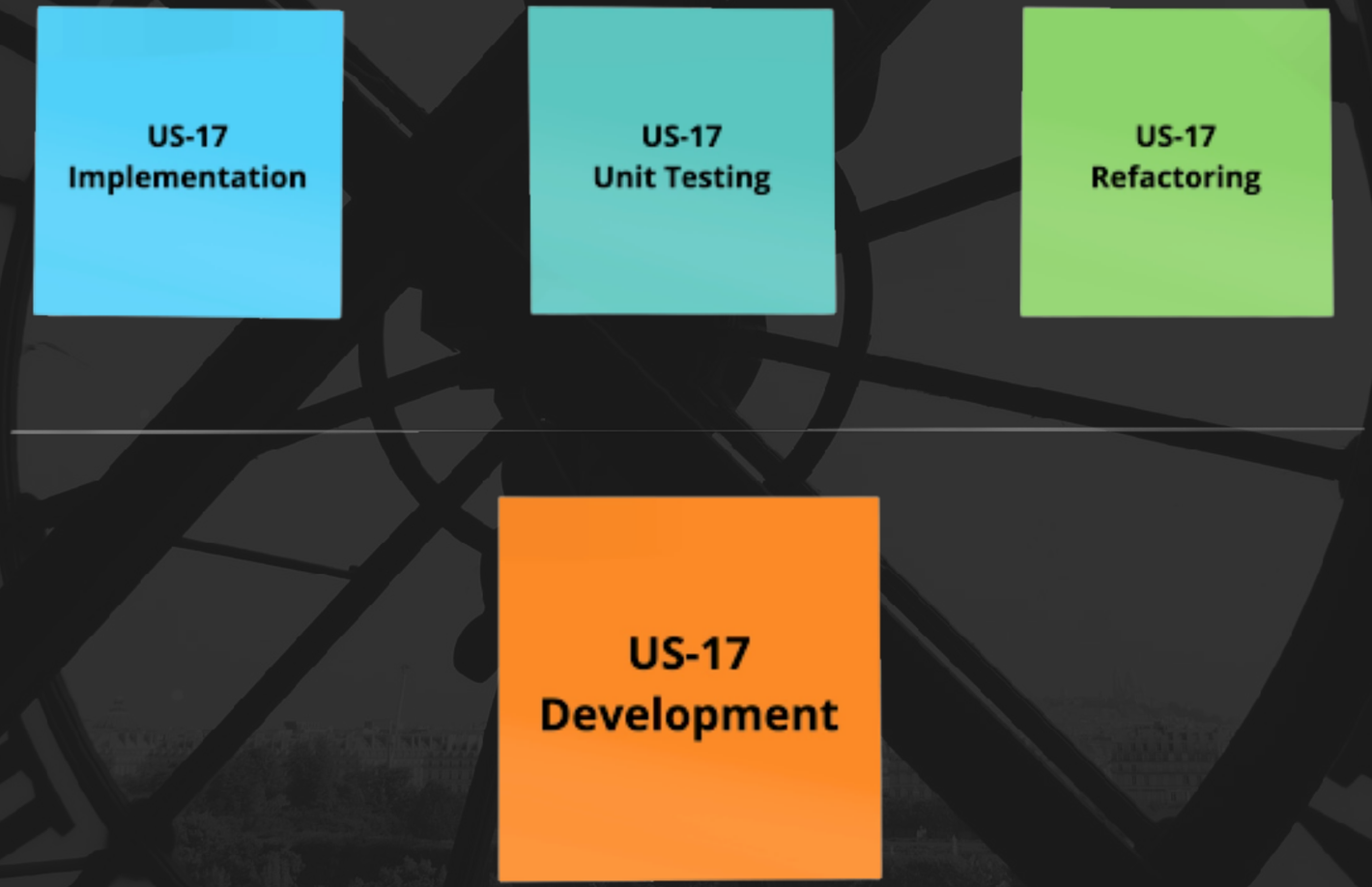
“For each desired change, make the change easy (warning: this may be hard), then make the easy change.”

Kent Beck

Refactor Early, Refactor Often.

The Pragmatic Programmer

What Do I Tell My Manager?



What Do I Tell My Manager?

Dictionary

Definitions from Oxford Languages · Learn more

deadline

/ˈdɛdlʌɪn/

noun

1. the latest time or date by which something should be completed.
   "the deadline for submissions is Friday 5th February"

   Similar: time limit | limit | finishing date | finishing time | target date | target time

2. **HISTORICAL**
   a line drawn around a prison beyond which prisoners were liable to be shot.

When Should We Not Refactor?

"There is nothing so useless as doing efficiently that which should not be done at all."

Peter Drucker

When Should We Not Refactor?

Slowing Down
New Features?

When Should We Not Refactor?

Code Ownership

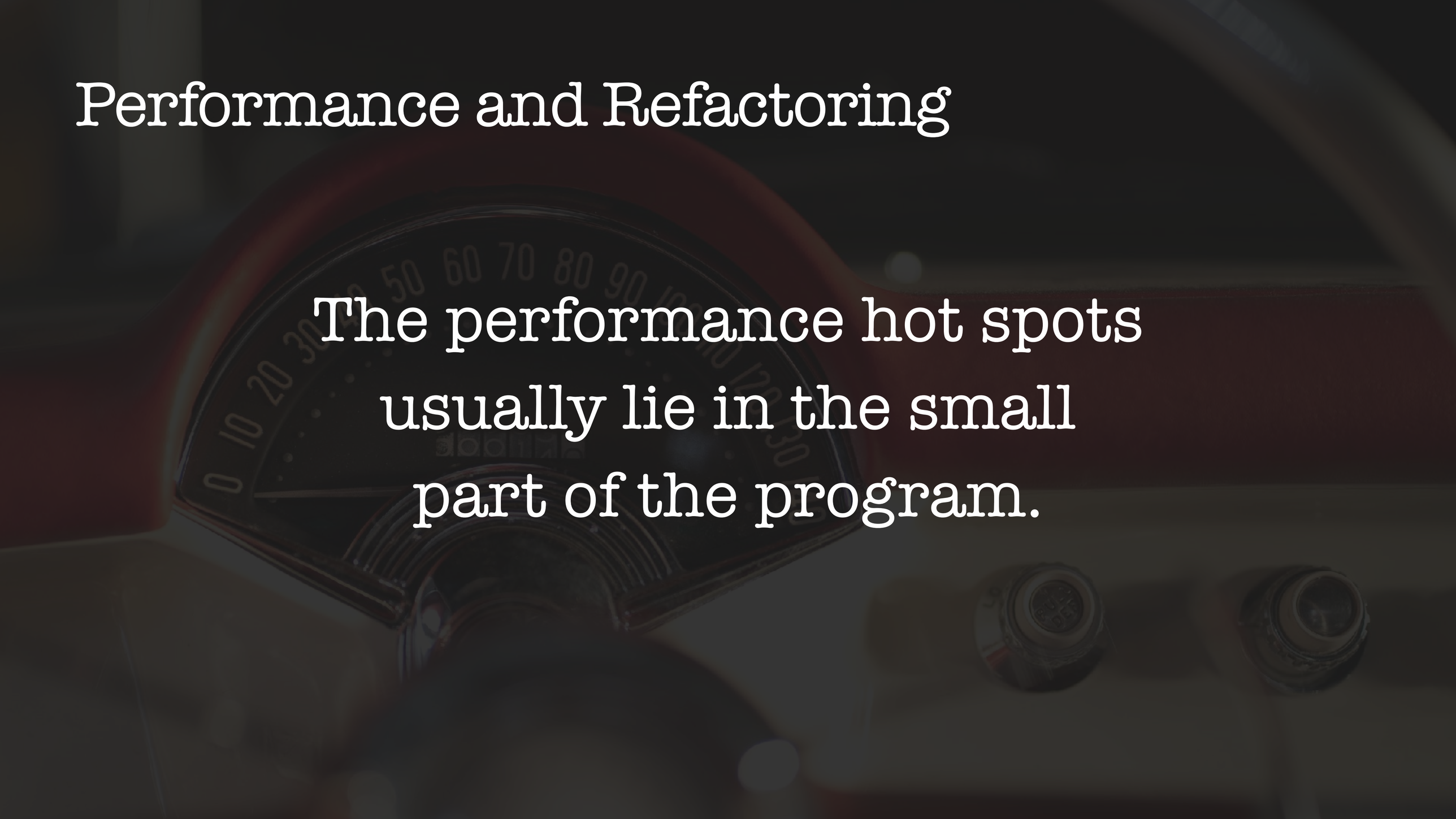
Boundaries

Rewrite or Refactor?

"Avoid the temptation to rewrite everything."

Rajith Attapattu

Performance and Refactoring



The performance hot spots
usually lie in the small
part of the program.

Performance and Refactoring

Don't guess, measure!

(with the proper tool)

Code Smells

Code Smells


LONG PARAMETER LIST
LARGE CLASS
MYSTERIOUS NAME
MESSAGE CHAINS
COMMENTS
LOOPS
INSIDER TRADING
SHOTGUN SURGERY
DUPLICATED CODE
PRIMITIVE OBSESSION
SPECULATIVE GENERALITY
LONG FUNCTION
DATA CLASS
FEATURE ENVY
DATA CLUMPS
MUTABLE DATA
REPEATED SWITCHES
GLOBAL DATA
DIVERGENT CHANGE
REFUSED BEQUES
LAZY ELEMENT

How to Refactor?

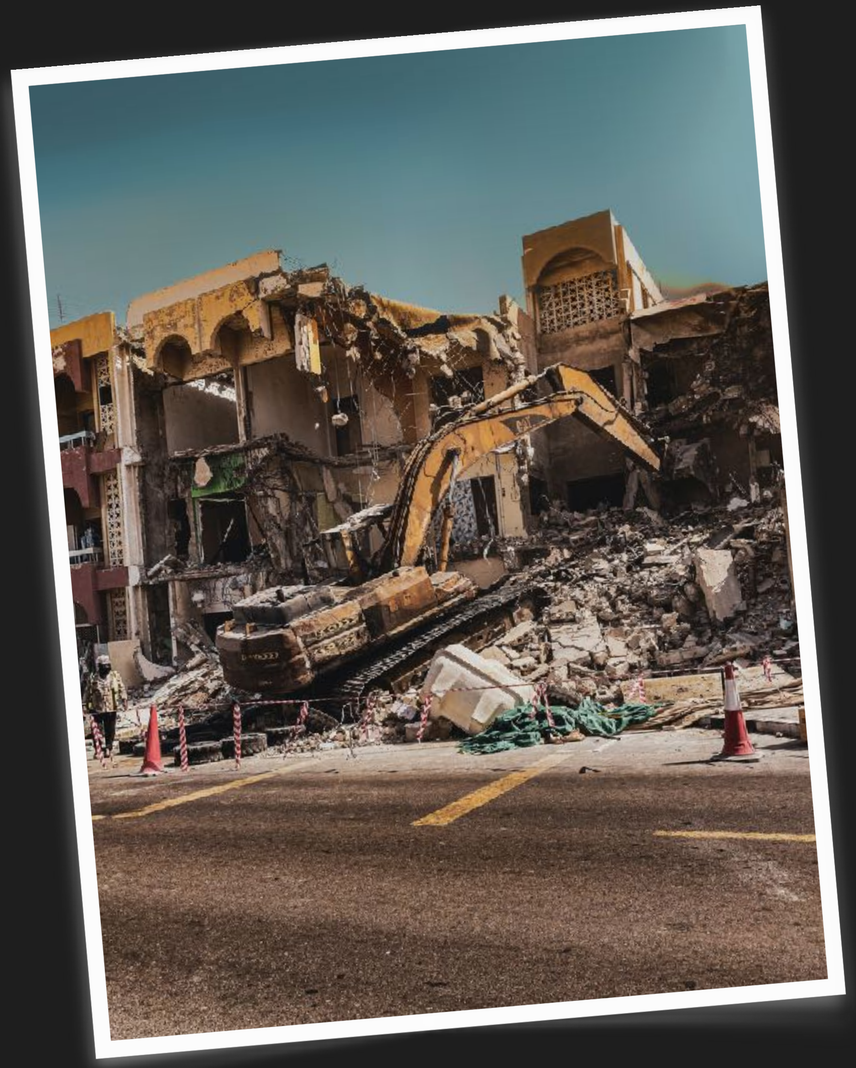
Automated Refactoring

Master your IDE

Disciplined Refactoring

Refactoring changes the programs in small steps, so if you make a mistake, it is easy to find where the bug is.

Disciplined Refactoring



Vs.

Disciplined Refactoring

Use micro-commits

Disciplined Refactoring

Write useful commit messages

Try writing down the message for your next commit

Scratch Refactoring

Catalogue of Refactoring

https://refactoring.com/catalog/

https://refactoring.guru/refactoring/catalog

The most common refactoring operations

https://medium.com/@aserg.ufmg/what-are-the-most-common-refactorings-performed-by-github-developers-896b0db96d9d

(Continuous) Renaming

It’s hard to get names right the first time - use the best name you can think of now, and don’t hesitate to rename it later.

(Continuous) Renaming - The Stroop Effect

BLACK RED

PURPLE WHITE

GREEN YELLOW

BROWN GRAY

BLUE ORANGE

Extract

method

class

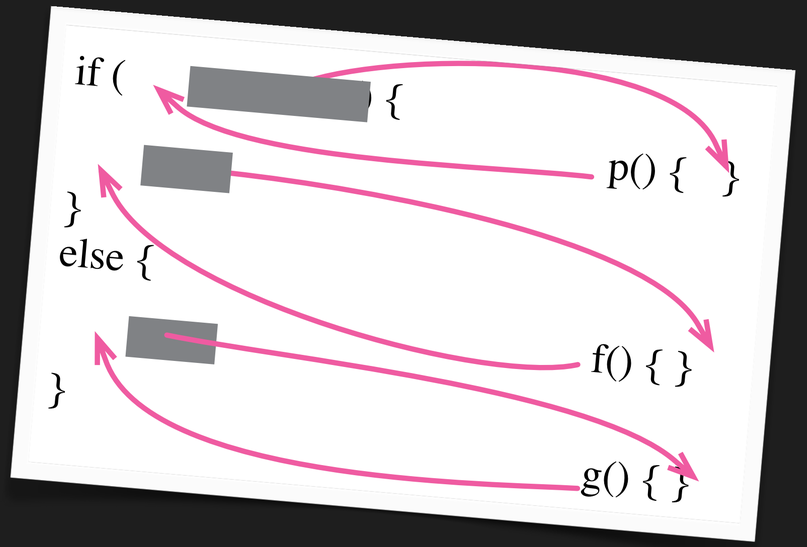
interface

superclass

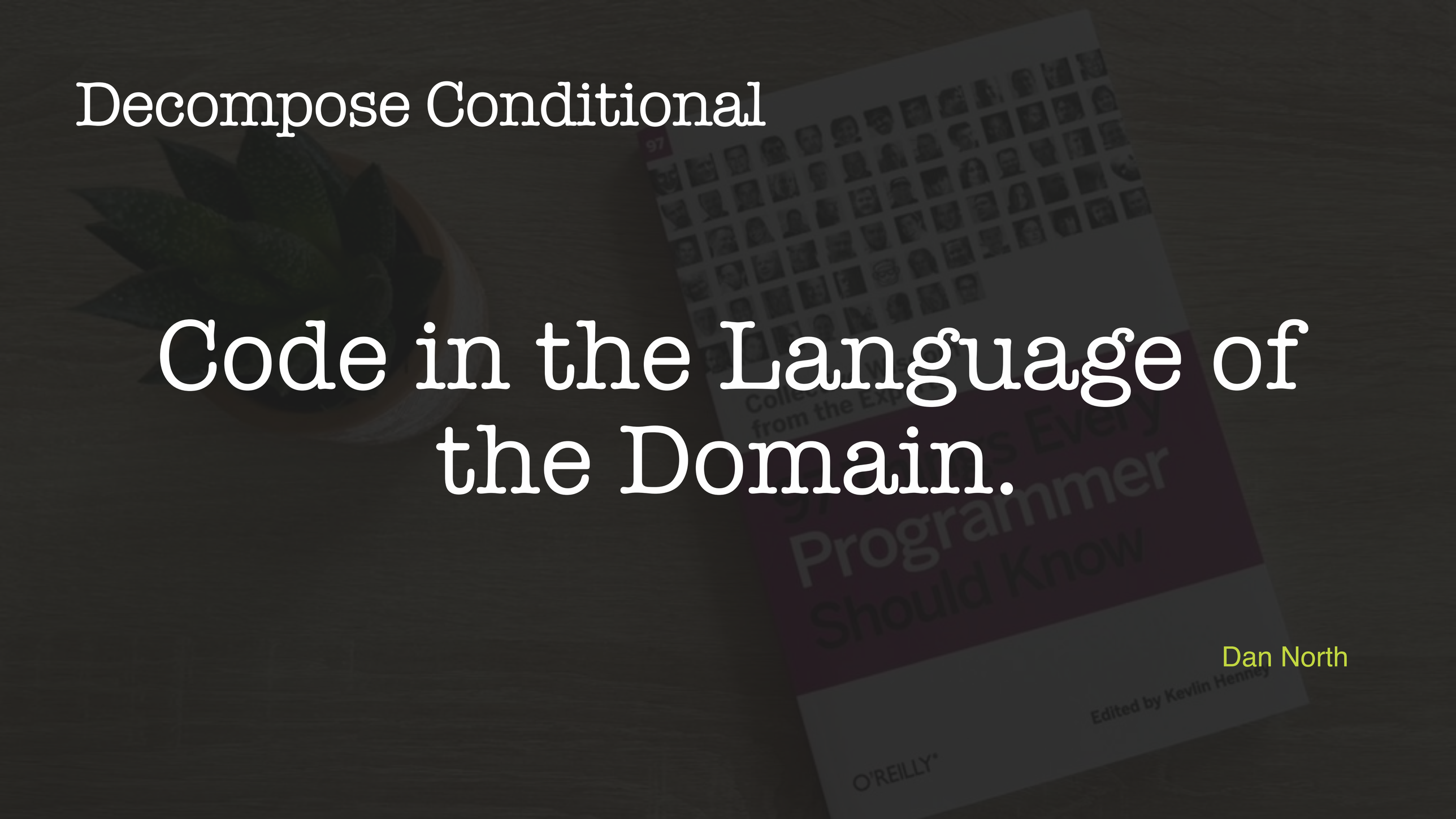
Decompose Conditional

```
if (date.before(SUMMER_START) || date.after(SUMMER_END)) {
    charge = quantity * winterRate + winterServiceCharge;
}
else {
    charge = quantity * summerRate;
}
```

```
if (isSummer(date)) {
    charge = summerCharge(quantity);
}
else {
    charge = winterCharge(quantity);
}
```



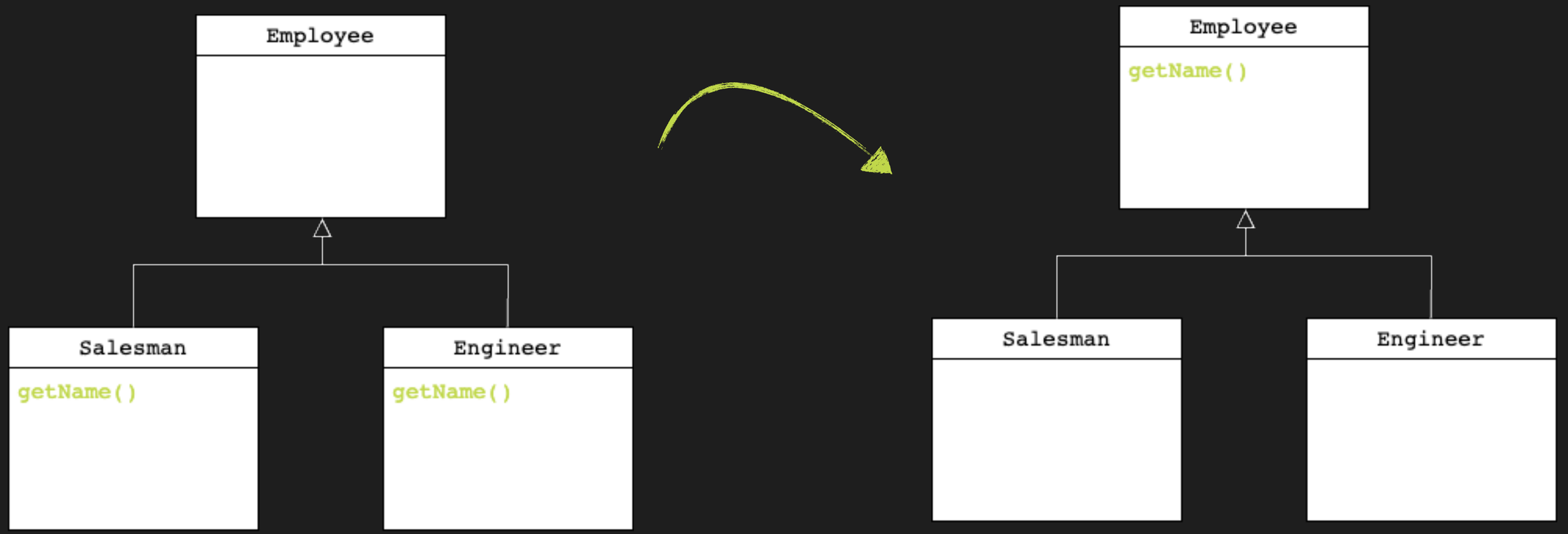
Decompose Conditional



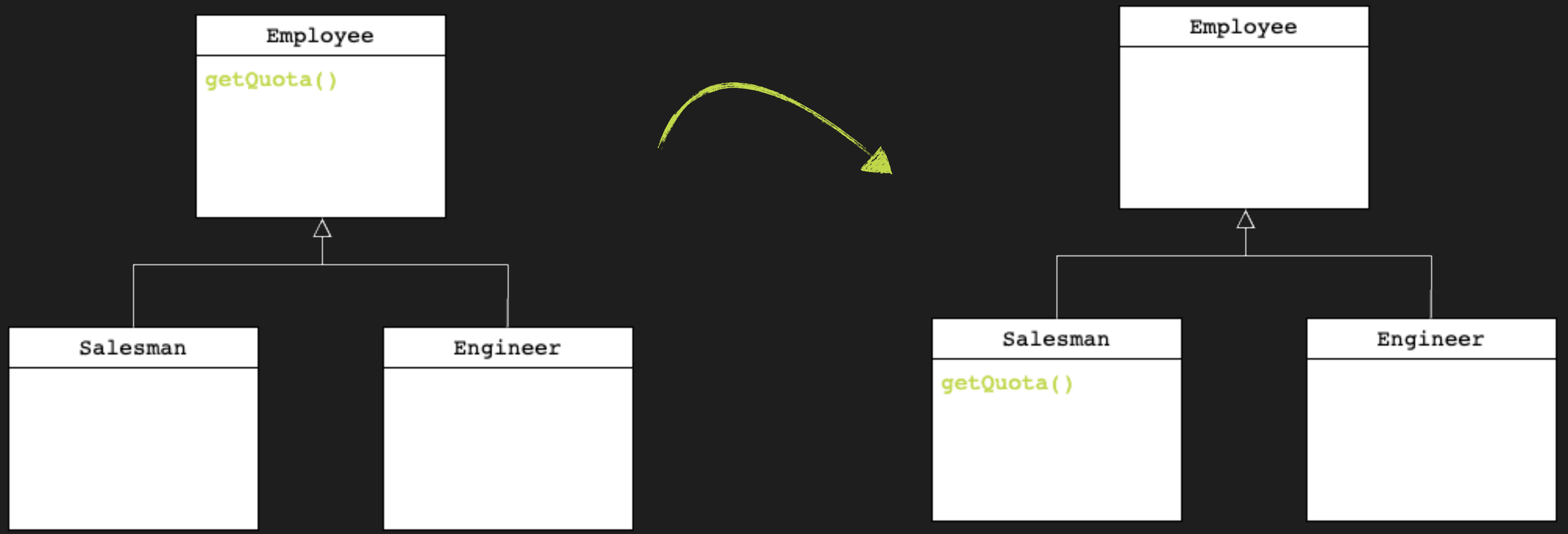
Code in the Language of the Domain.

Dan North

Pull Up Method



Push Down Method



Remove Dead Code

Less is more.

Delete and document what has been deleted, why, and where it can be found.

Comments

Comment Only What the Code Cannot Say.

Kevlin Henney


What will be our legacy?
Thank You for Your Attention