Continuous Refactoring

prophylaxis, surgery and spring-cleaning for source code

bbv
MAKING VISIONS WORK.

Urs Enzler

bbv Software Services AG

once upon a time. . .

quick poll

Code **refactoring** is the process of restructuring existing computer code—changing the factoring—without changing its external behavior.

Refactoring improves nonfunctional attributes of the software.

Wikipedia

tests

no behaviour change

reviews pair programming

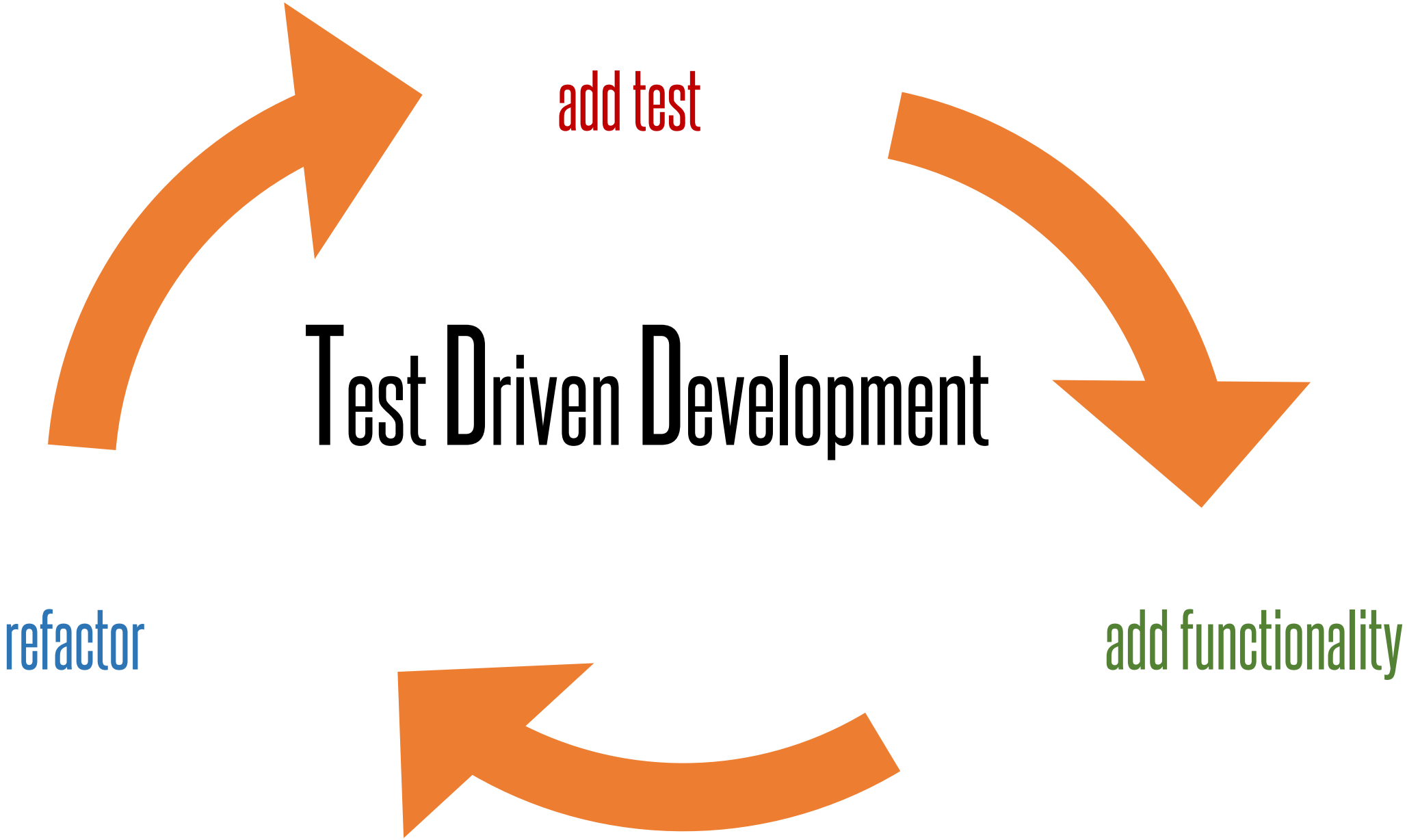
what is the boundary where no behaviour change occurs?

system	system & constraint tests
feature component	specifications (ATDD / BDD)
class method	facts (TDD / unit tests)


tool


manual

times to refactor


add test
Test Driven Development
add functionality
refactor

peer reviews

before pushing

after finishing a user story

```
PS C:\Projects> git push
```

Litter pick-up

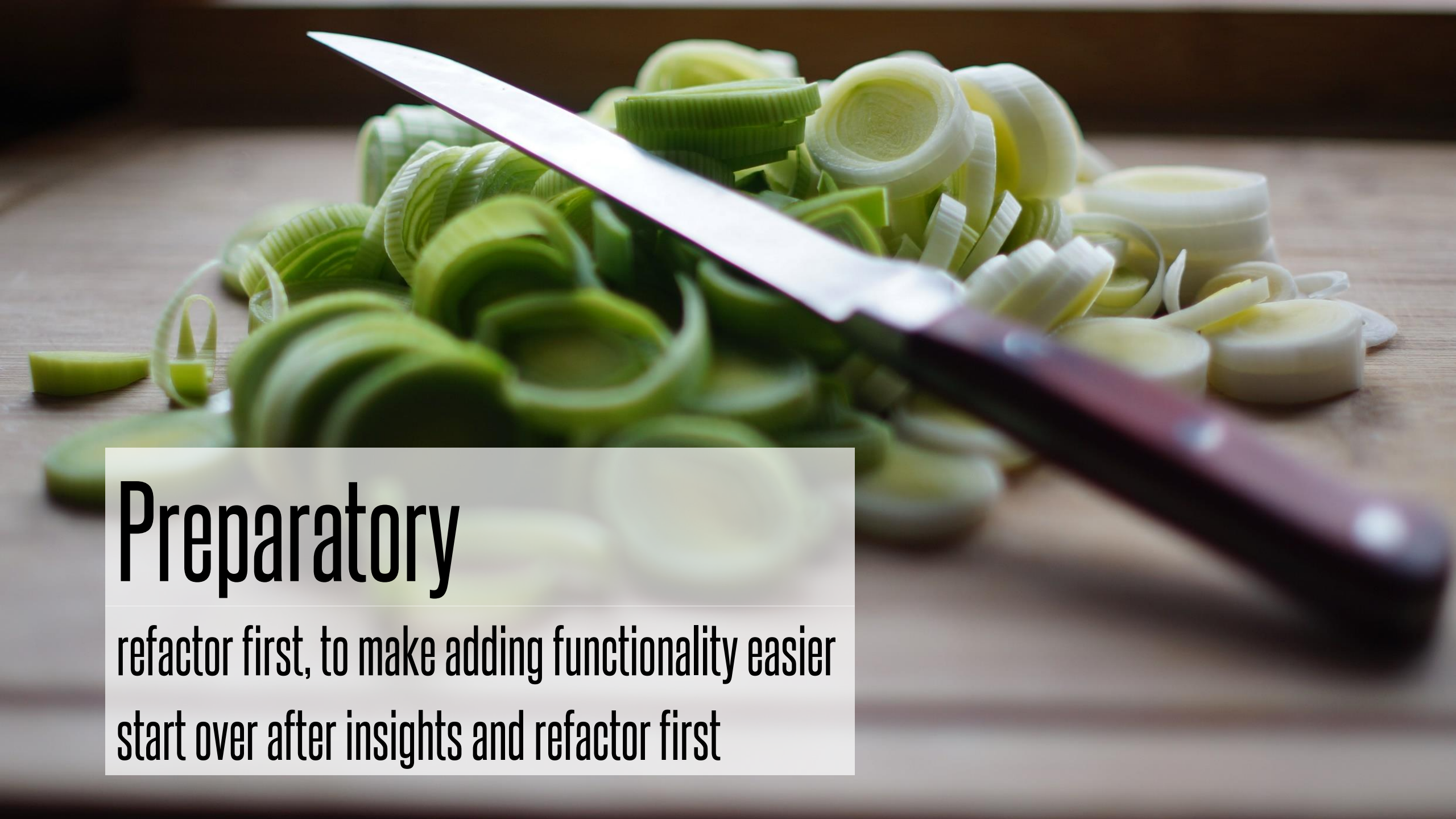
1. stash your changes
2. refactor
3. commit
4. un-stash

or

1. write a task

Comprehension

code reflects your understanding



Preparatory

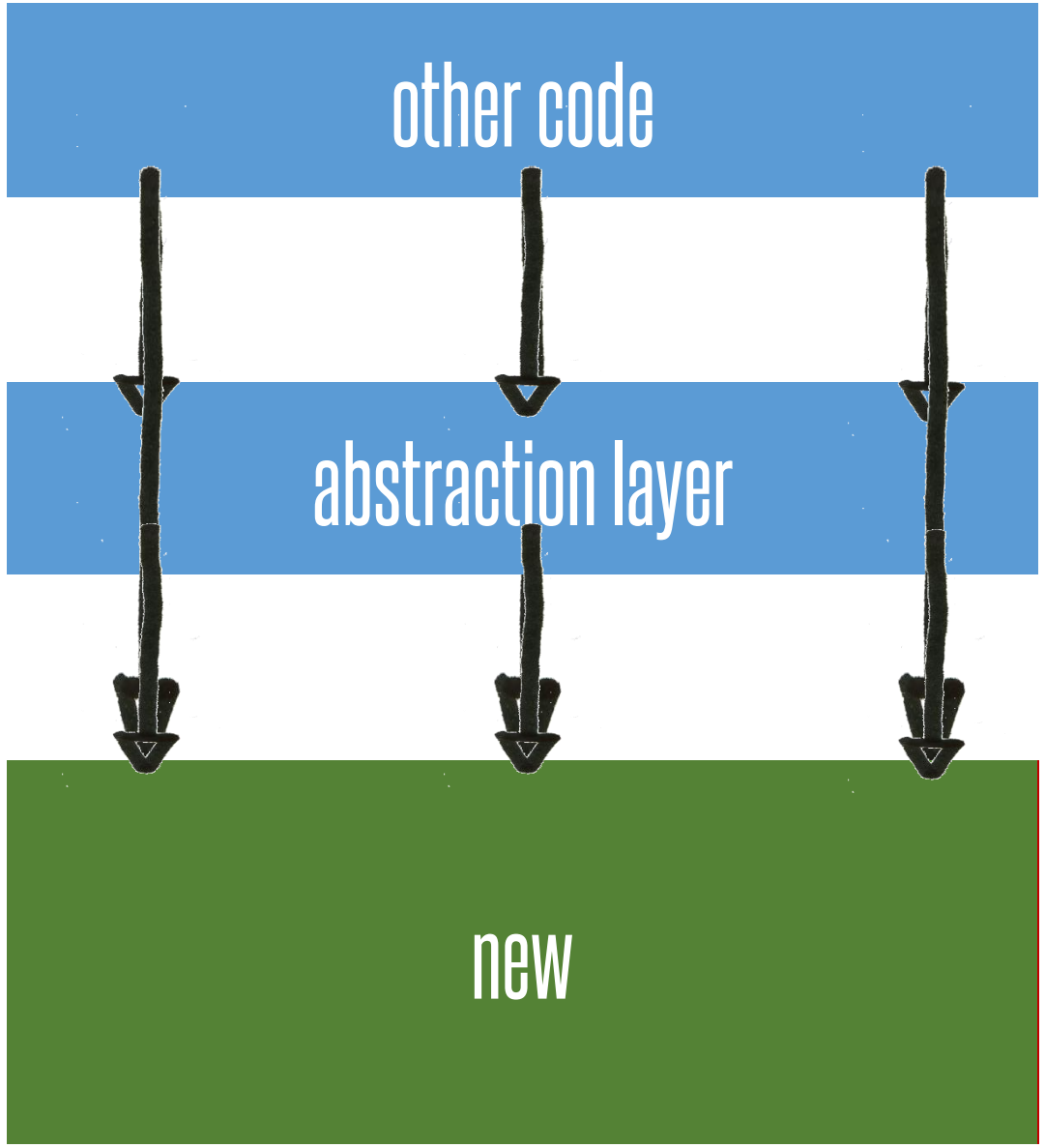
refactor first, to make adding functionality easier

start over after insights and refactor first

Long term refactoring

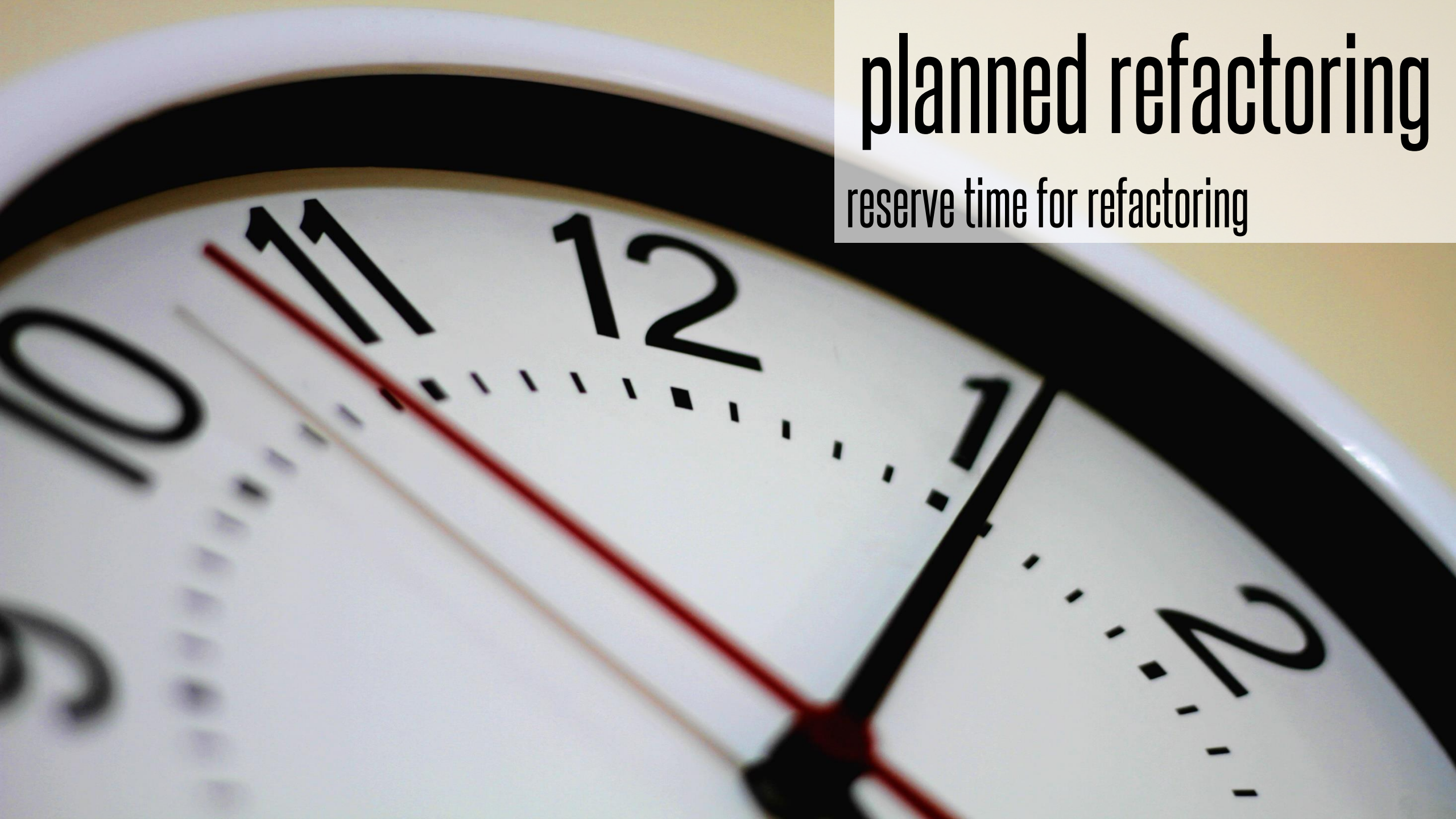
split into steps

use abstraction layer to support current and replacement implementation


other code
abstraction layer
new

planned refactoring

reserve time for refactoring



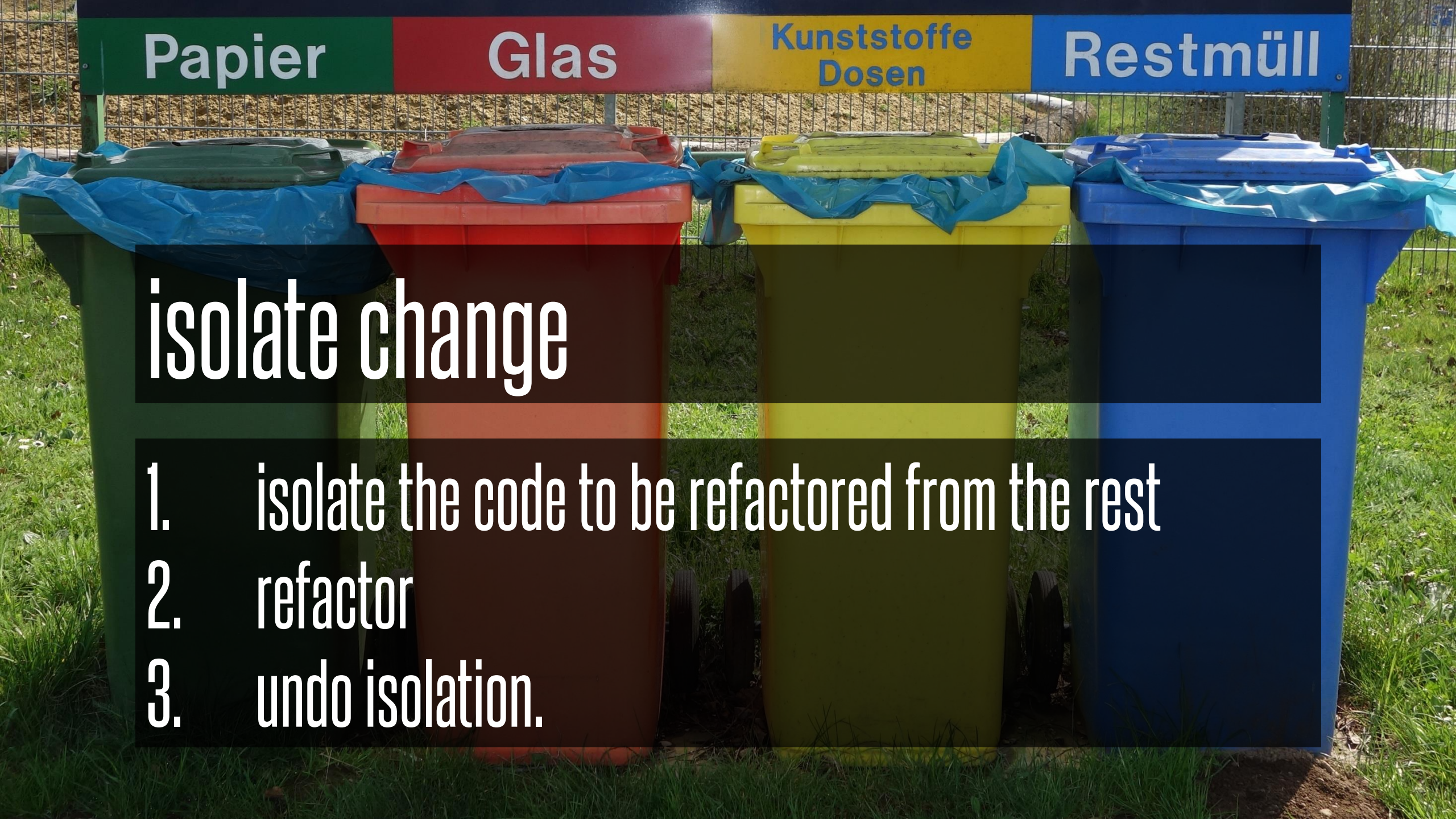
refactoring patterns

small refactorings

Only refactor in small steps with working code in-between so that you can keep all loose ends in your head. Otherwise, defects sneak in.

reconcile differences – unify similar code

Change both pieces of code stepwise until they are identical. Then extract.



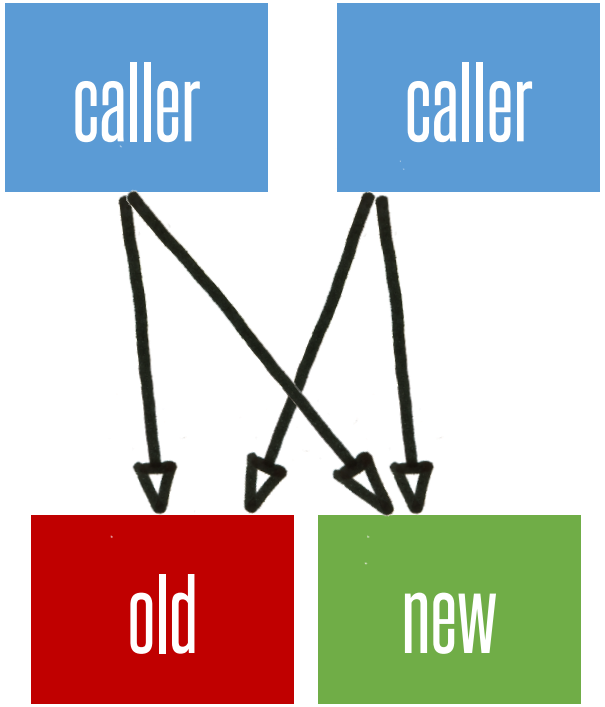
isolate change

1. isolate the code to be refactored from the rest
2. refactor
3. undo isolation.

temporary parallel implementation

1. introduce a new parallel implementation.
2. Switch one caller after the other.
3. Remove old solution when no longer needed.

This way you can refactor with only one red test at a time.



migrate data

Move from one representation to another by temporary duplication of data structures.

```
foo(int i, string s)
```

```
foo(int i, string s, MyStruct m)
```

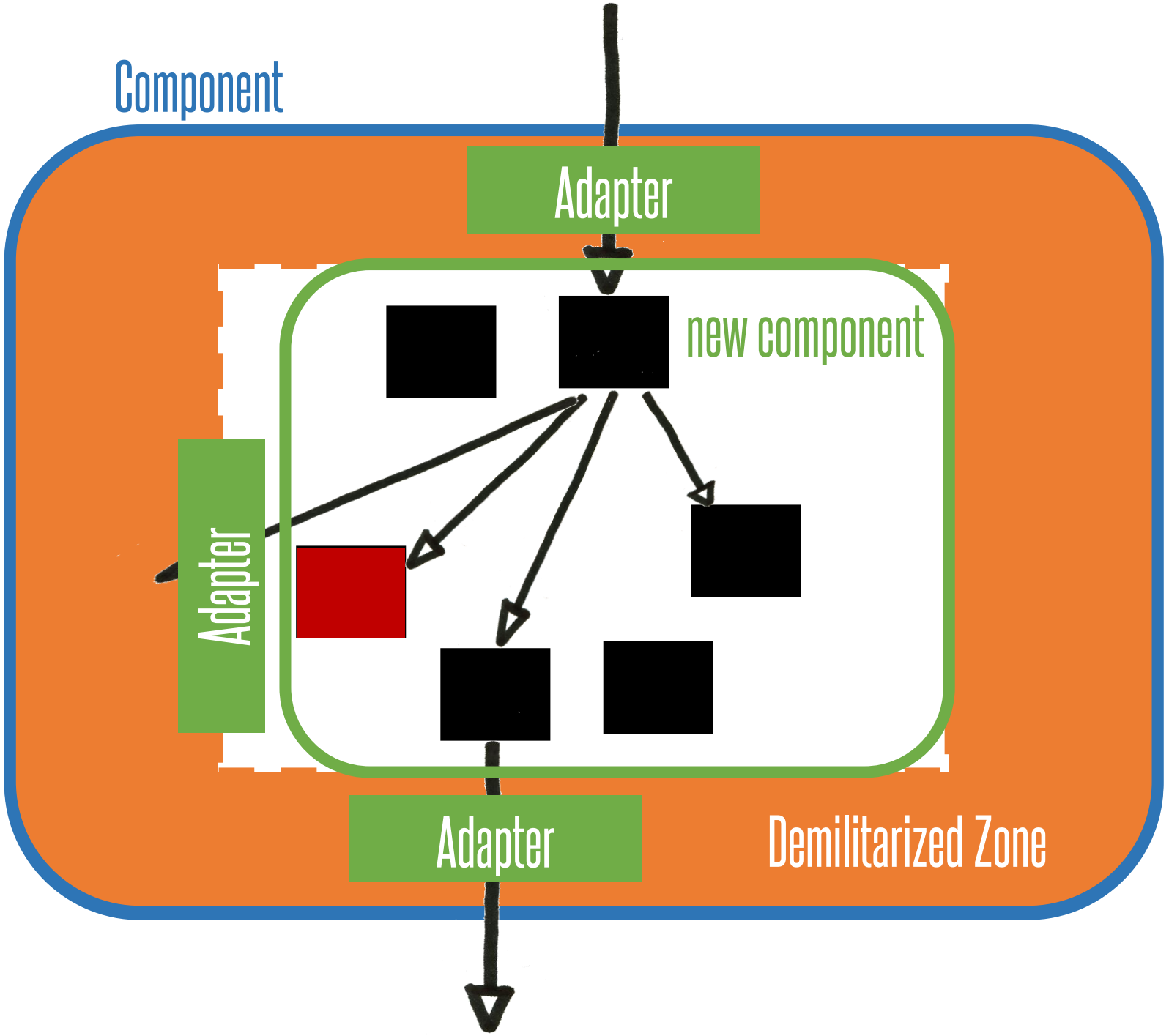
```
foo(MyStruct m)
```

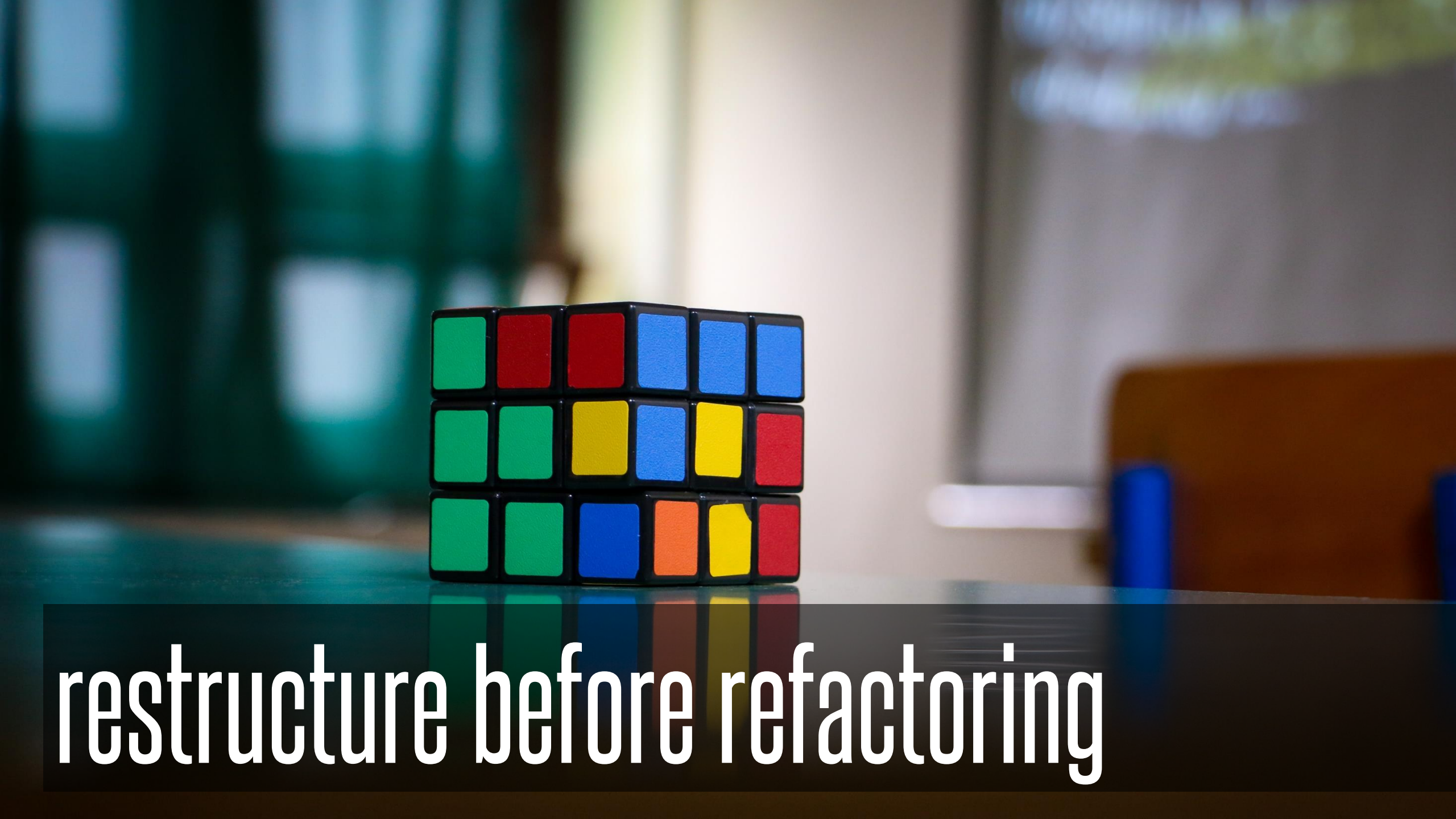
MyStruct contains i and s

demilitarized zone for components

Introduce an internal component boundary and push everything unwanted outside of the internal boundary into the demilitarized zone between component interface and internal boundary.

Then refactor the component interface to match the internal boundary and eliminate the demilitarized zone.


Component
Adapter
new component
Adapter
Adapter
Demilitarized Zone


restructure before refactoring

Structure101 Studio for .Net - <untitled>*

File Project Tag Tools Help

Model: Model 1 (2)

Model | Rules | Views | Summary

Structural over-complexity

100 % Tangled 0

Unstructured

Structured

% Fat 100

Tangles | Fat items

Item	Size
SensorLegacy	1'292

Option

SensorLegacy

Program

Reports

Contexts

Vhpt

Events

Notifier

Base

Split classes (1) Notables

#C	S...	Name
2	340	SensorLegacy.Vhpt.Doo...

Map contents

Depends | Feedback

Items | Descendants | Tags

Item

SensorLegacy

SensorLegacy.Base

SensorLegacy.Contexts

Actions for "Model 1"

Action list | Refactorings | Class map

#	Action
2	Remove Sb
1	Remove AAttribute

Dependency breakout

Select a dependency in the graph (or a cell in the matrix) to view its breakout

Excluded from project: 0 Excluded from XS: 0 Transformed classes: 0

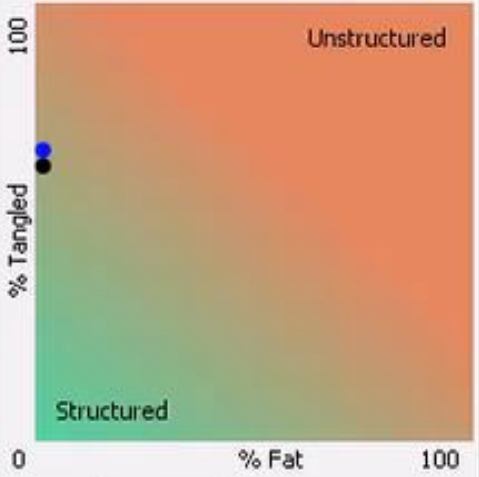
Structure101 Studio for .Net - <untitled>*

File Project Tag Tools Help

Model: Model 1 (2)

Model | Rules | Views | Summary

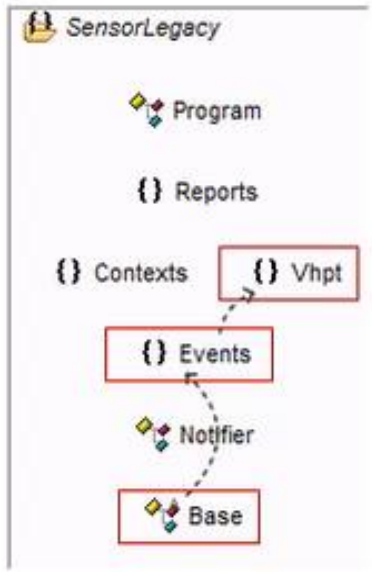
Structural over-complexity



Tangles | Fat items

Item	Size
SensorLegacy	1'292

Option



Dependency breakout

Select a dependency in the graph (or a cell in the matrix) to view its breakout

Split classes (1) Notables

#C	S...	Name
2	340	SensorLegacy.Vhpt.Doo...

Map contents

Depends | Feedback | Items | Descendants | Tags

Item
SensorLegacy
SensorLegacy.Base
SensorLegacy.Contexts

Actions for "Model 1"

Action list | Refactorings | Class map

#	Action
2	Remove Sb
1	Remove AAttribute

Excluded from project: 0 Excluded from XS: 0 Transformed classes: 0

can there be too much refactoring?

sure

but unlikely


you can always refactor
but there had better be a test around it

Urs Enzler

urs.enzler@bbv.ch

twitter: @ursenzler
blog: www.planetgeek.ch
www.bbv.ch/blog
OSS lead: Appccelerate
user group: www.dotnet-zentral.ch