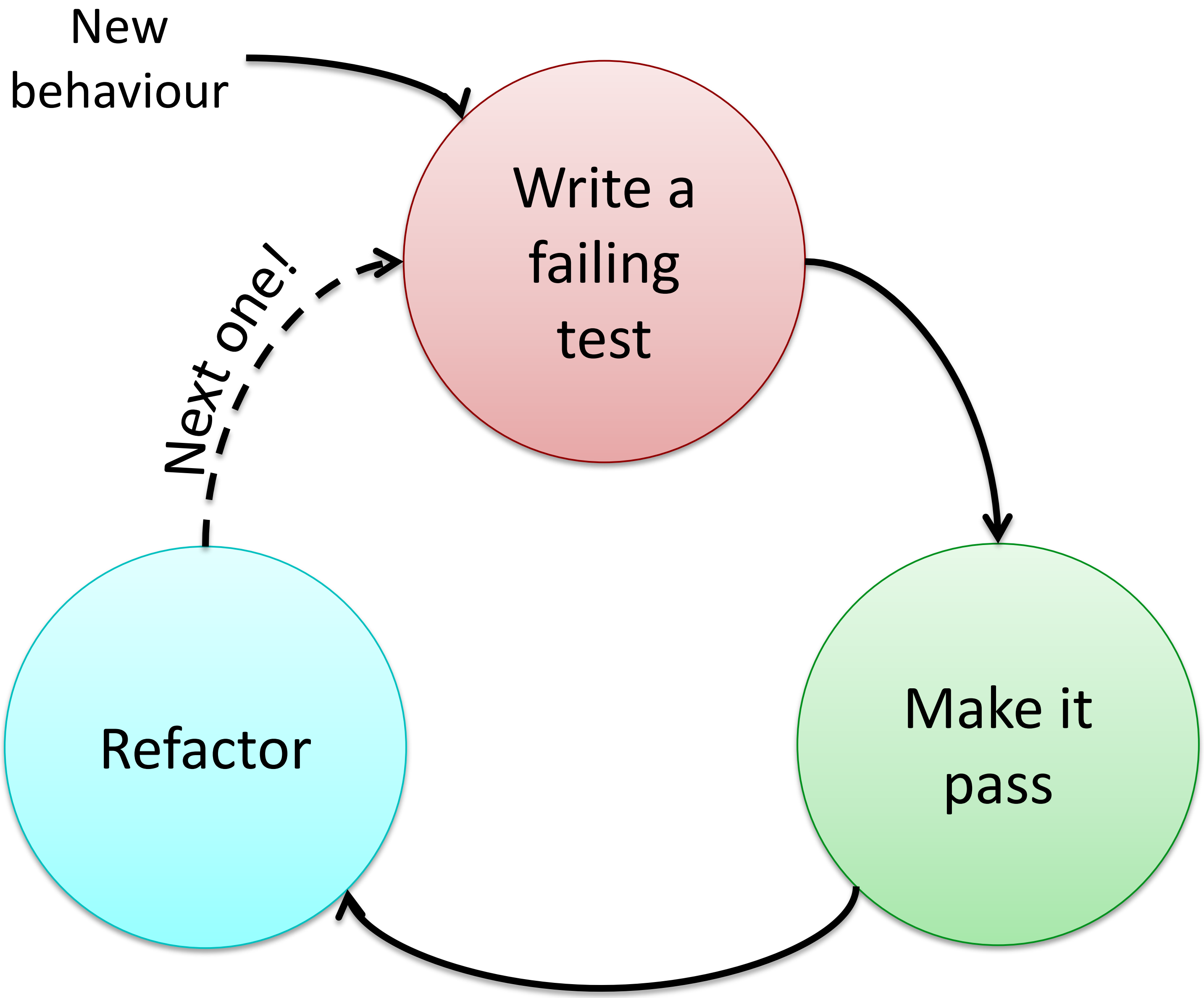
TDD
All the Things!

Liz Keogh
@lunivore
http://lizkeogh.com


New behaviour
Write a failing test
Make it pass
Refactor
Next one!

```
public bool WaitFor(AutomationElementWrapper element,
                SomethingToWaitFor check,
                TimeSpan timeout, FailureToHappenHandler failureHandler,
                IEnumerable<AutomationEventWrapper> events)
{
    Monitor.Enter(_waitingRoom);
    _triggeringEvent = null;

    DateTime started = DateTime.Now;
    var handlerRemovers = AddPulsingHandlers(events, element);

    bool checkPassed = true;
    while (!check(element, _triggeringEvent) &&
        DateTime.Now.Subtract(started).CompareTo(timeout) < 0)
    {
        checkPassed = false;
        Monitor.Wait(_waitingRoom, timeout);
    }
    Monitor.Exit(_waitingRoom);
    ClearPulsingHandlers(handlerRemovers);

    if (!checkPassed && !check(element, null))
    {
        failureHandler(element);
        return false;
    }
    return true;
}
```

```
public void ShouldWaitForEventsToOccur()
{
    // Given an automation element
    _window = LaunchPetShopWindow();
    var combo = _window.Find<ComboBox>("petFoodInput");

    // When we cause a slow event on that element
    new Thread(() =>
                   {
                       Thread.Sleep(200);
                       combo.Select("PetFood[Carnivorous]");
                   }).Start();

    // And we wait for the event
    var eventOccurred = false;
    new Waiter().WaitFor(
                    combo, (src, e) => {
                        eventOccurred = true;
                        return combo.Selection.Equals("PetFood[Carnivorous]");
                    },
                    new TimeSpan(0, 0, 1),
                    (ex) => Assert.Fail(),
    new List<AutomationEventWrapper> {
         new StructureChangeEvent(TreeScope.Element)});

    // Then we should be notified when the event occurs
    Assert.IsTrue(eventOccurred);
}
```

```
public void ShouldWaitForEventsToOccur()
{
```

```
// Given an automation element
 _window = LaunchPetShopWindow();
var combo =
    _window.Find<ComboBox>("petFoodInput");
```

```
// When we cause a slow event on that element
new Thread(() =>
    {
        Thread.Sleep(200);
        combo.Select("PetFood[Carnivorous]");
    }).Start();
```

```
// And we wait for the event
var eventOccurred = false;
new Waiter().WaitFor(combo, (src, e) =>
    {
        eventOccurred = true;
        return combo.Selection.Equals(
            "PetFood[Carnivorous]");
    }, new TimeSpan(0, 0, 1),
    (ex) => Assert.Fail(),
    new List<AutomationEventWrapper> {
        new StructureChangeEvent(
            TreeScope.Element)});
```

```
// Then we should be notified
// when the event occurs
Assert.IsTrue(eventOccurred);
```

}

ShouldWaitForEventsToOccur

Given an automation element

When we cause a slow event
on that element

And we wait for the event

Then we should be notified
when the event occurs.

Examples

Given a context

When an event happens

Then an outcome *should* occur

Arrange

Act

Assert

Arrange

Given a context

When an event happens

Then an outcome *should* occur

Act

Given a context

When an event happens

Then an outcome *should* occur

Given a context

When an event happens

Then an outcome *should* occur

Assert

An Example of an Example

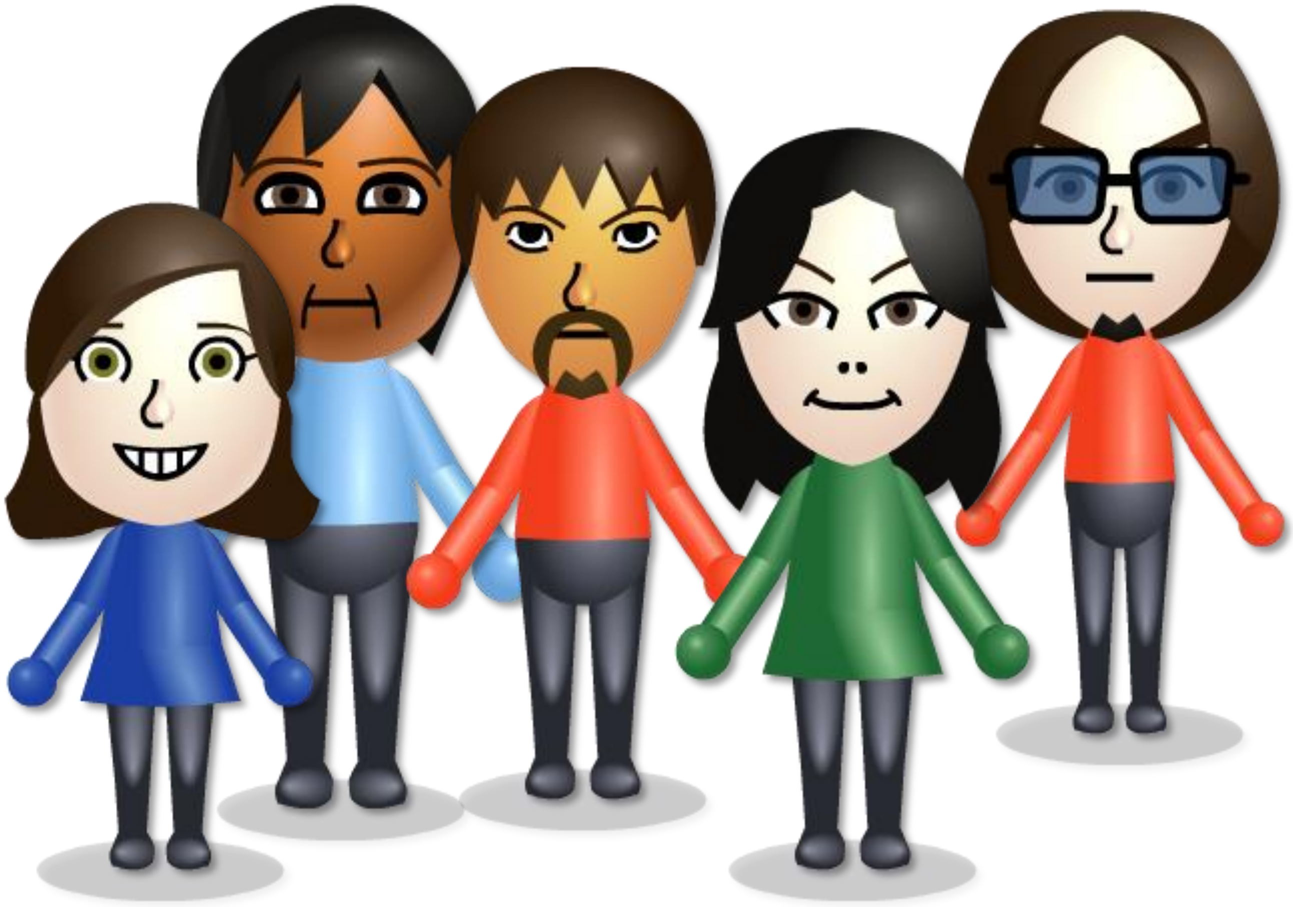
Given Fred has bought a microwave

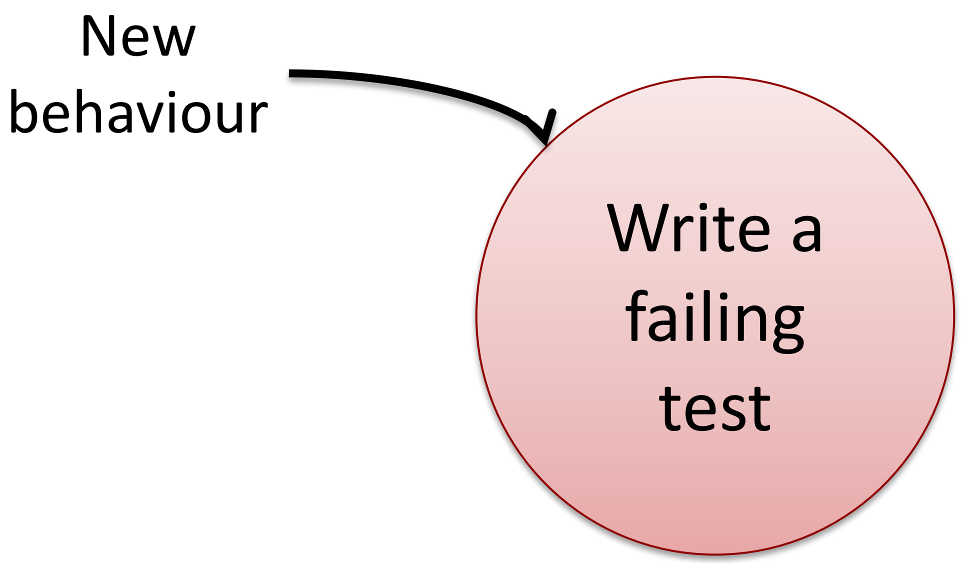
And the microwave cost £100

When we refund the microwave

Then Fred *should* be refunded £100.

Let’s TDD a person!




New behaviour
Write a failing test

Feedback

Context in which they act

Given a context

When an event happens

Then an outcome *should* occur

Feedback

Action they take

Given a context

When an event happens

Then an outcome *should* occur

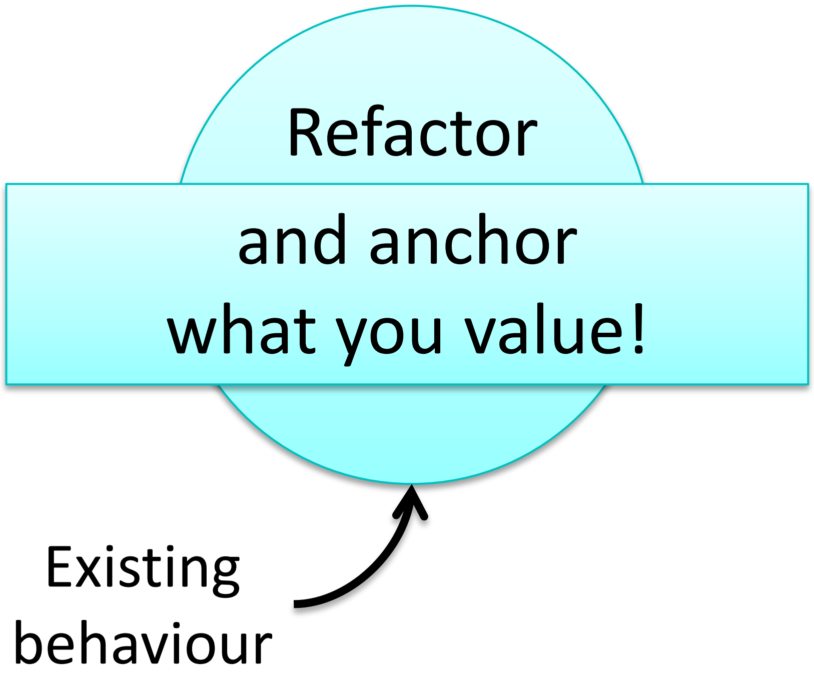
Feedback

Given a context

When an event happens

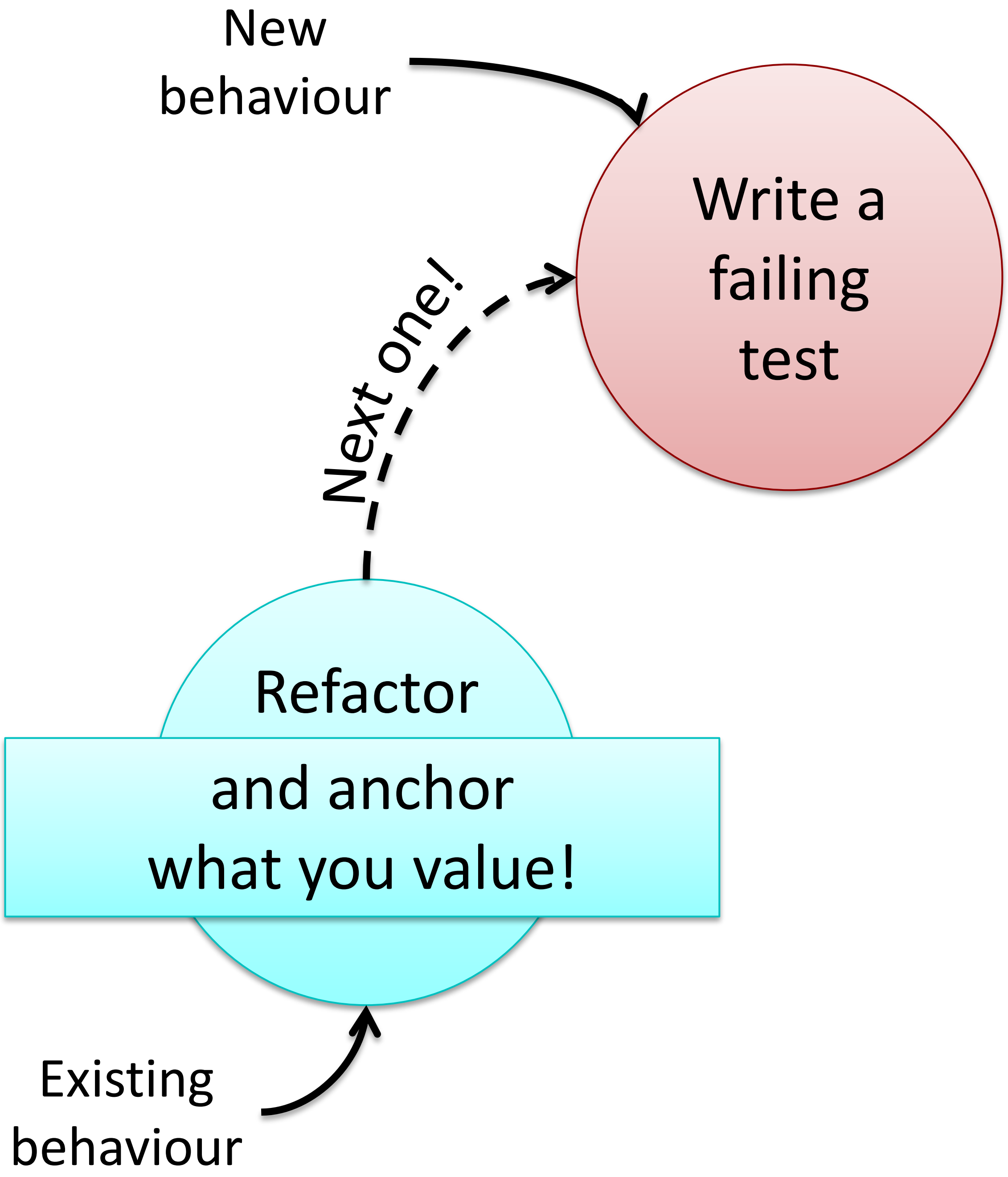
Then an outcome *should* occur

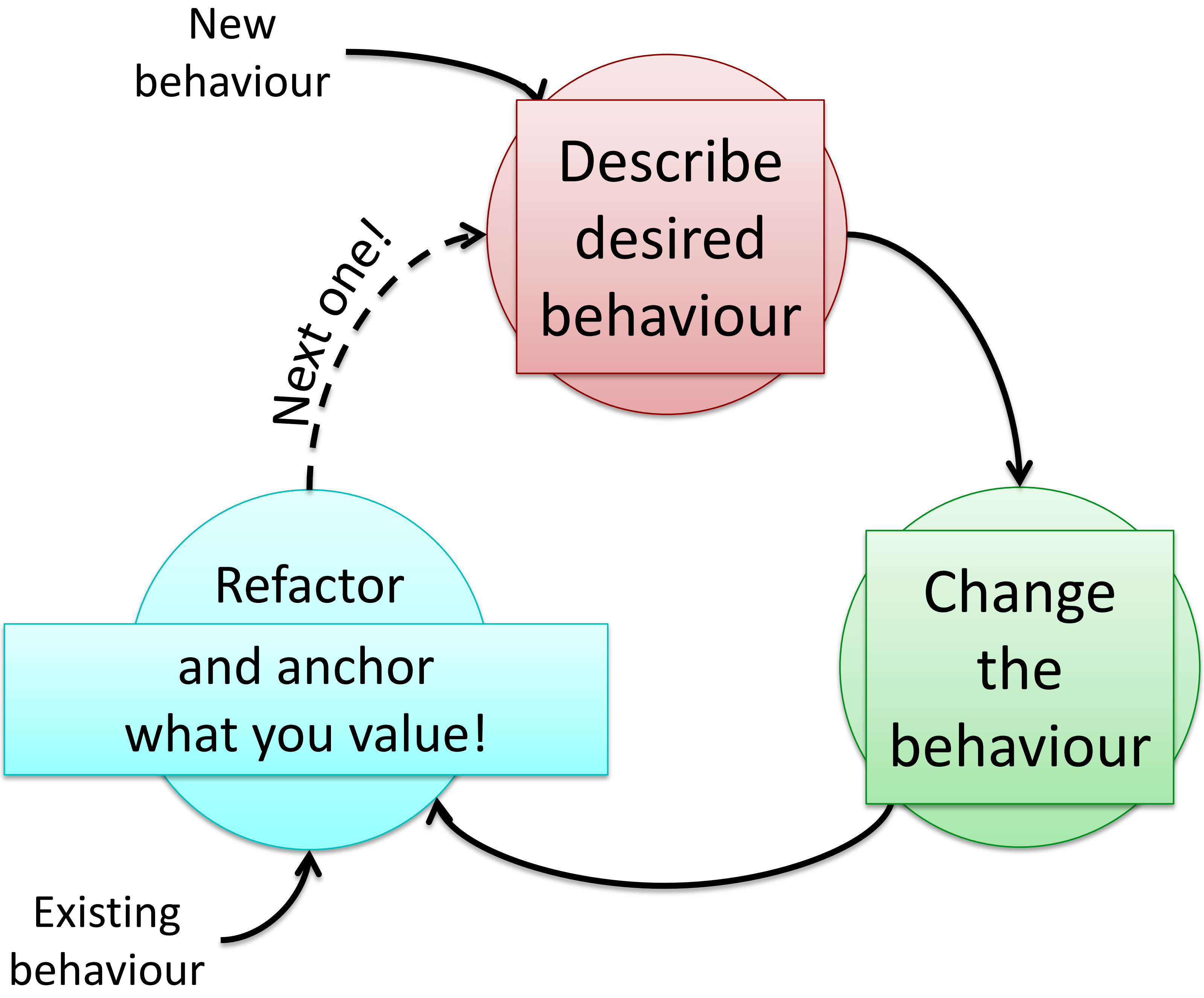
Outcomes


Refactor
and anchor
what you value!
Existing
behaviour

Number 1 rule of ~~feedback~~: changing behaviour

Anchor what you value!


New behaviour
Write a failing test
Next one!
Refactor
and anchor what you value!
Existing behaviour


New behaviour
Describe desired behaviour
Change the behaviour
Refactor and anchor what you value!
Next one!
Existing behaviour

The sandwich model

Start with something good

Say something bad

Finish with something good

The sandwich model done right

Anchor what you value

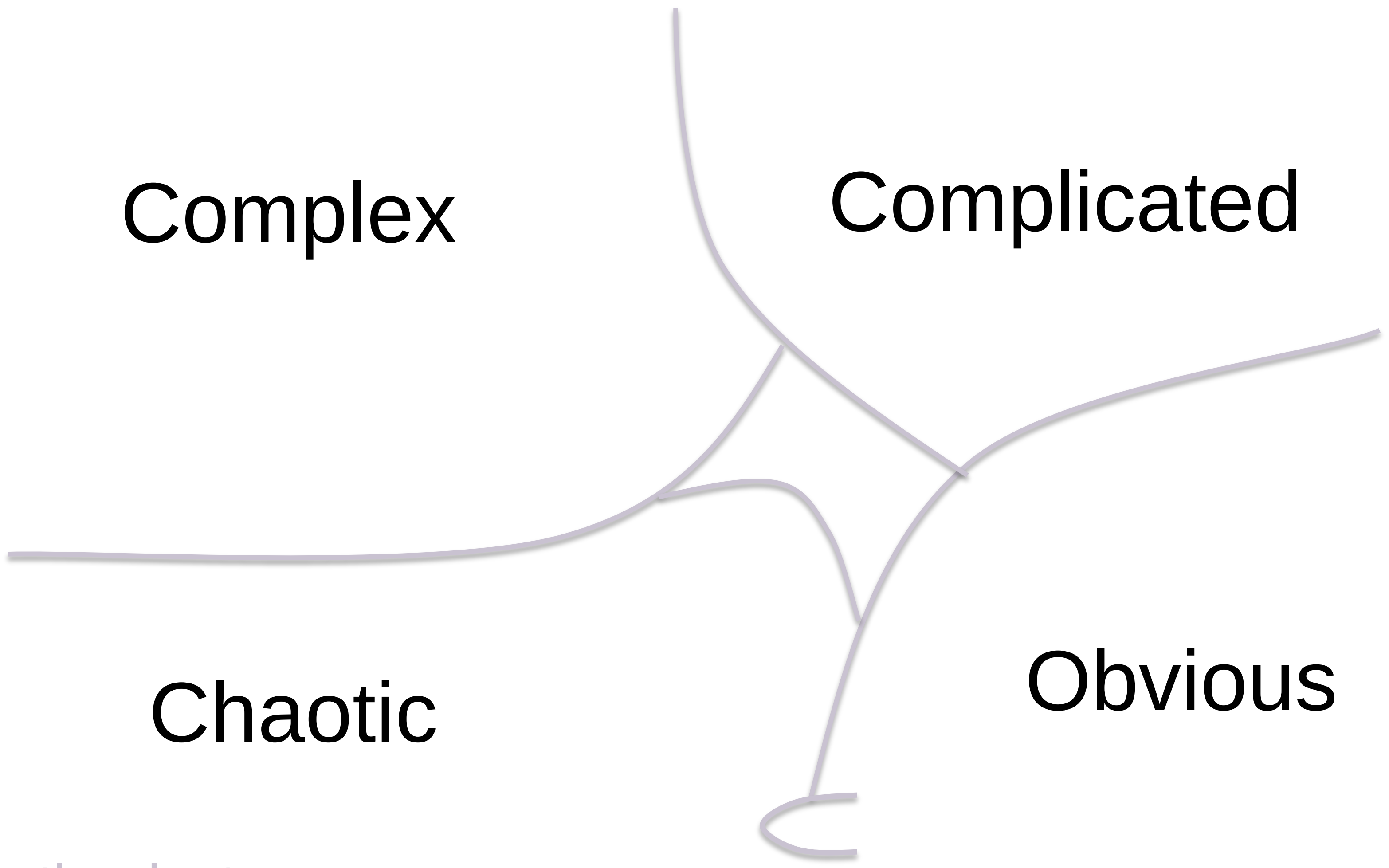
Describe desired behaviour

THEN change the behaviour

(People can do this bit themselves!)

What about refactoring?

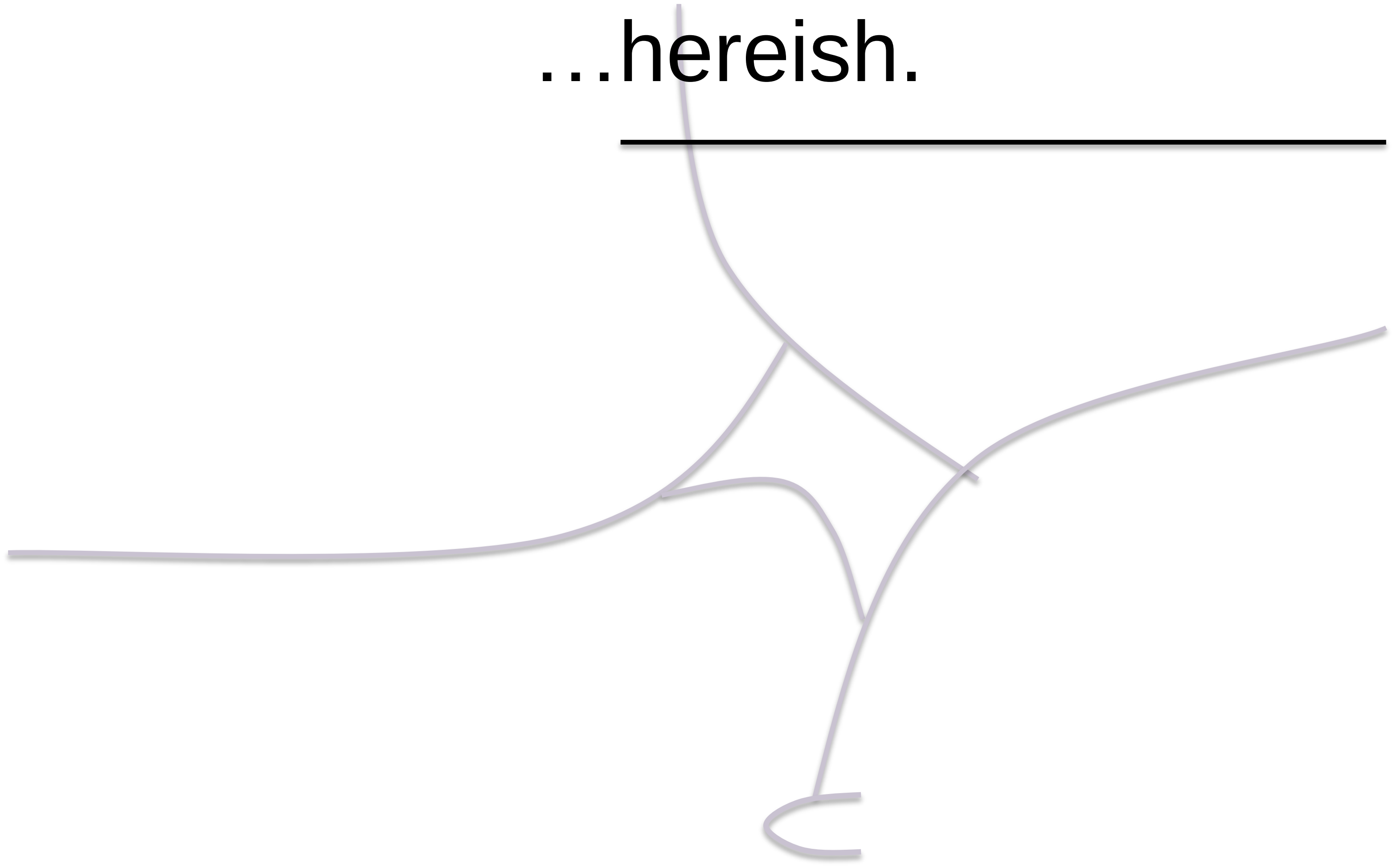
Cynefin



With thanks to
David Snowden and Cognitive Edge

BDD and TDD work really well…

…hereish.



Whenever we do anything

new

we will make

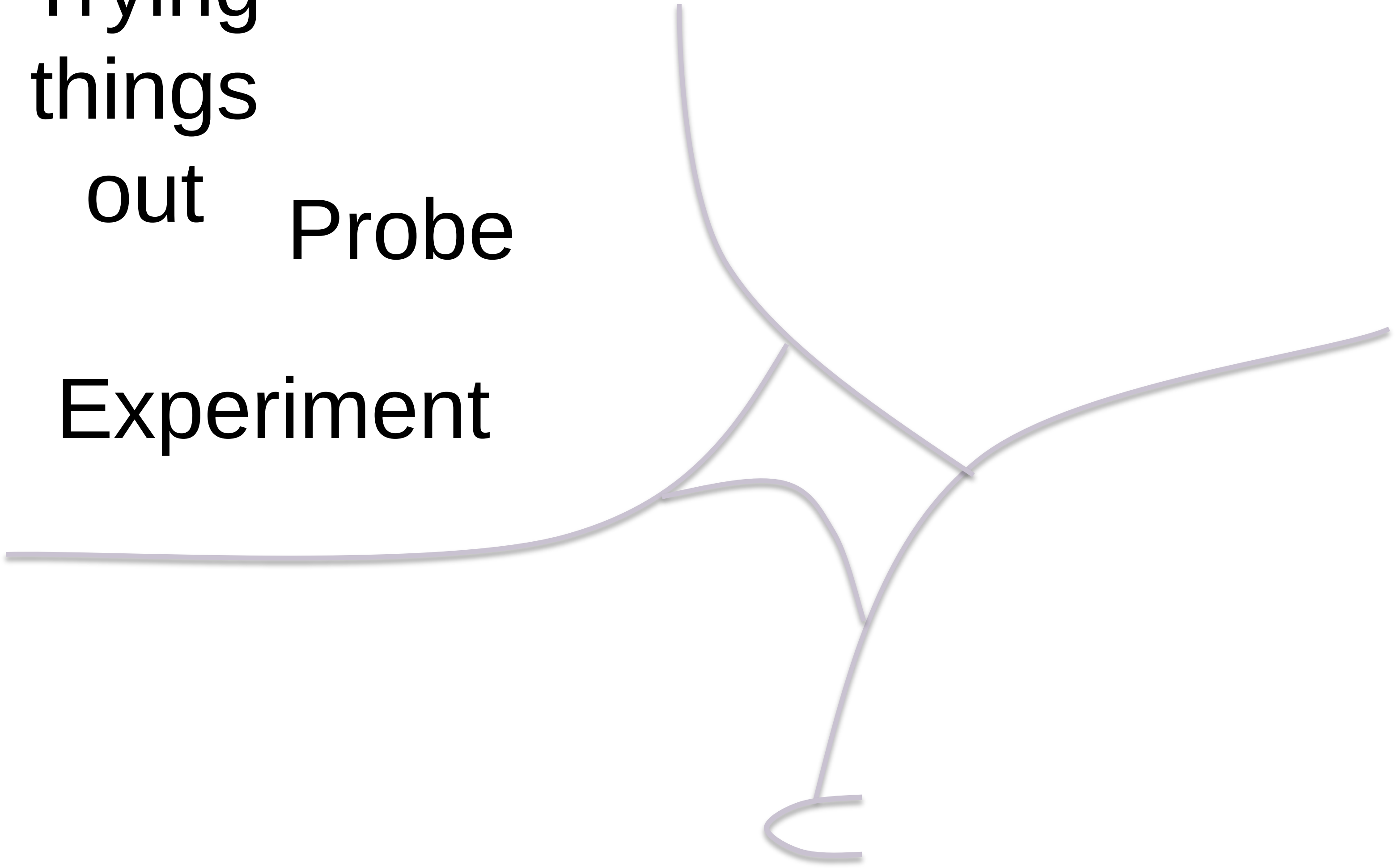
discoveries

Cynefin

Trying things out

Probe

Experiment



Refactoring code

Make it run

Get the thing that's new working

Make it right

Separate concerns

Make it fast

so you can do more!

Good code

Focus on responsibilities and strengths

Is generous on input, strict on output

Is easy to understand and work with

You can trust clean code.

Refactoring people

Try it out

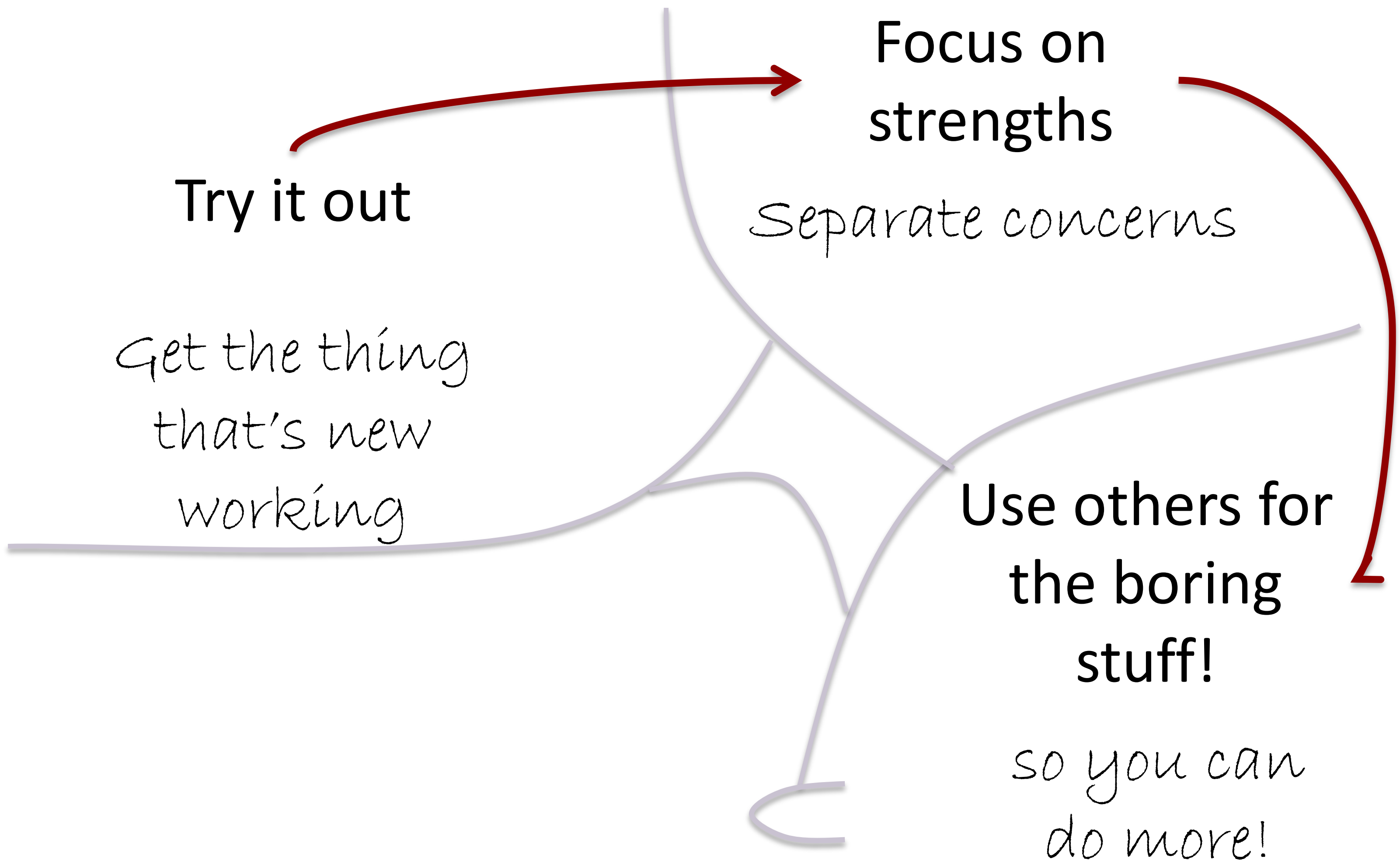
Get the thing that's new working

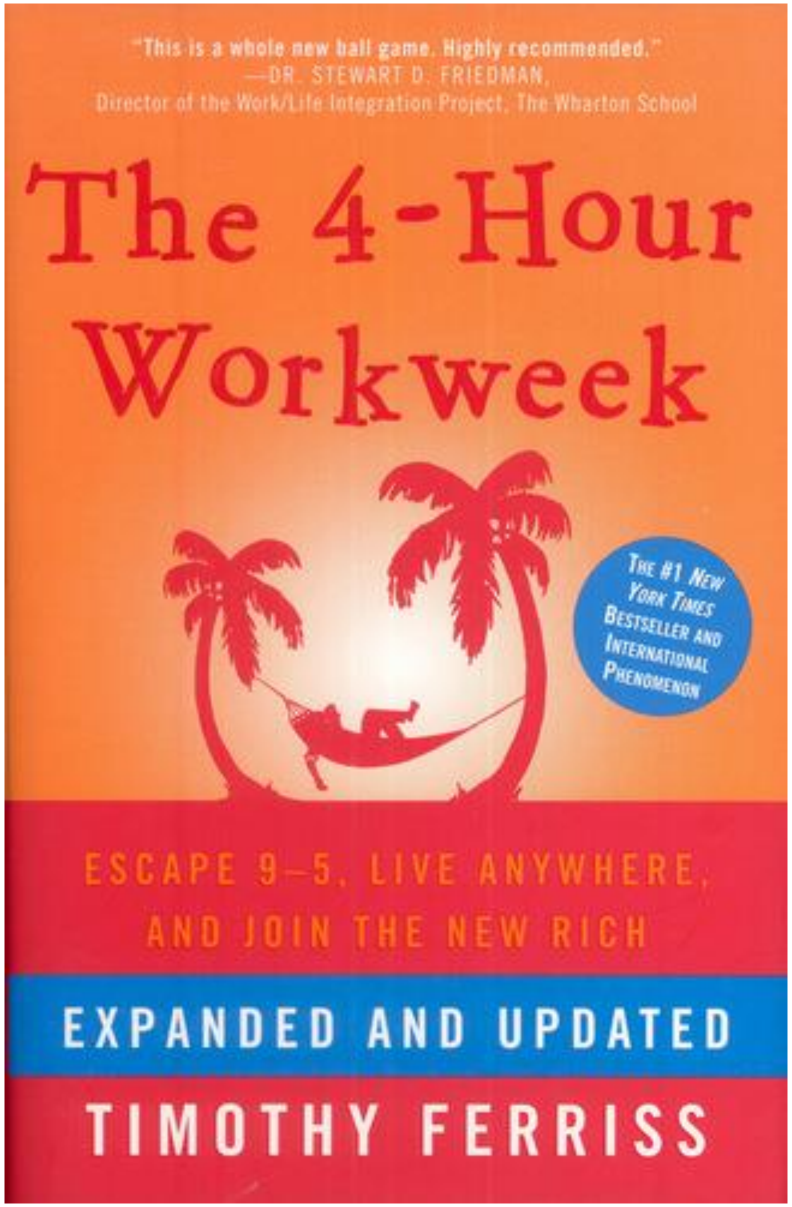
Focus on strengths

Separate concerns

Use others for the boring stuff!

so you can do more!





"This is a whole new ball game. Highly recommended."
—DR. STEWART D. FRIEDMAN,
Director of the Work/Life Integration Project, The Wharton School

The 4-Hour Workweek

THE #1 NEW YORK TIMES BESTSELLER AND INTERNATIONAL PHENOMENON

ESCAPE 9–5, LIVE ANYWHERE, AND JOIN THE NEW RICH

EXPANDED AND UPDATED

TIMOTHY FERRISS

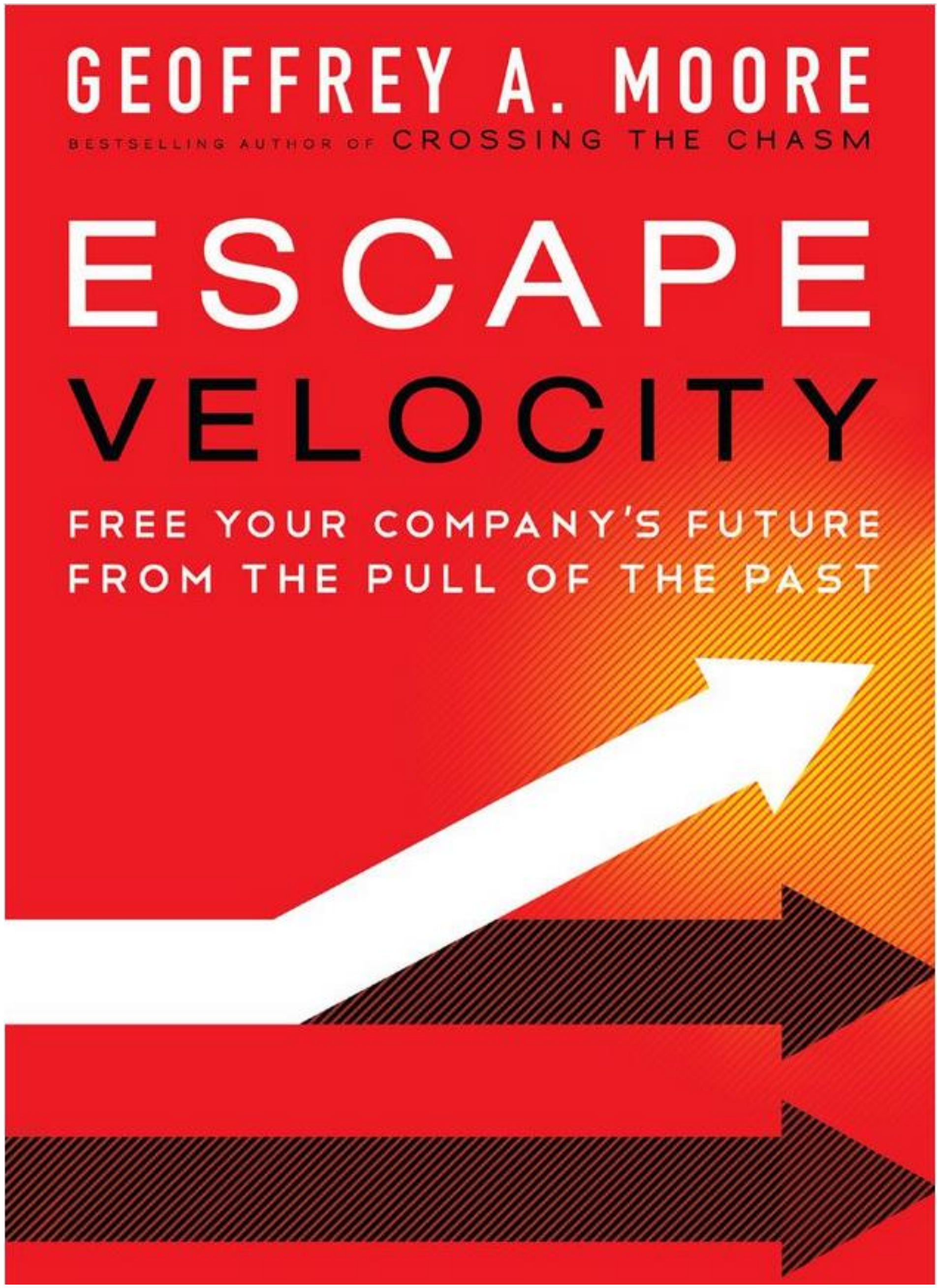
Good people

Focus on responsibilities and strengths

Are generous in listening, honest in speaking

Are easy to understand and work with

You can trust good people.



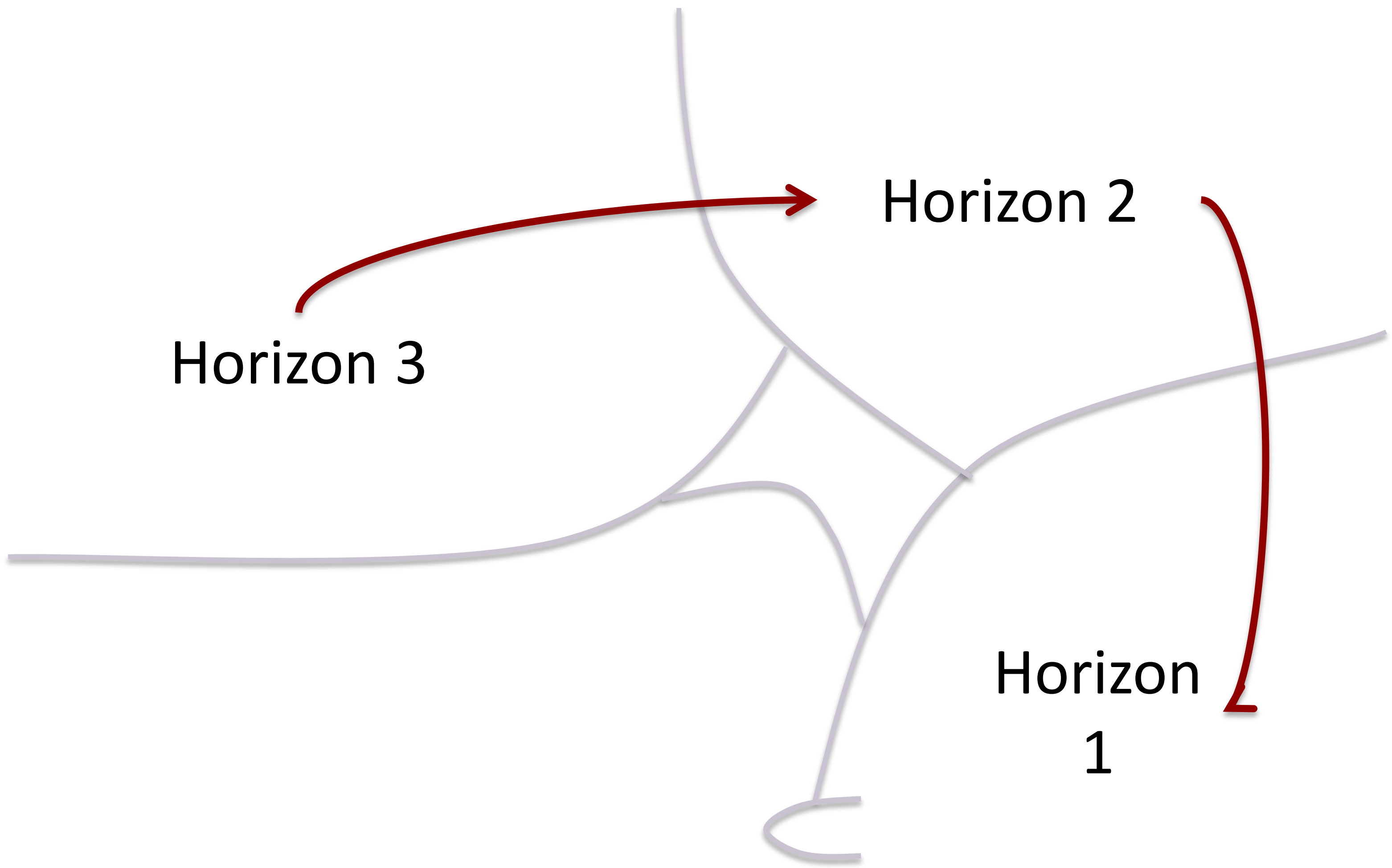
GEOFFREY A. MOORE

BESTSELLING AUTHOR OF CROSSING THE CHASM

ESCAPE VELOCITY

FREE YOUR COMPANY'S FUTURE FROM THE PULL OF THE PAST

Refactoring at scale



Horizon 2
fights for budget
with Horizon 1.

Focusing on strengths
fights for time
with the boring stuff.

Focus on
strengths.

Get rid of the
boring stuff.

Anchor the behaviour you value.

In unfamiliar scenarios, create options –
make it safe-to-fail.

Change code; help people change themselves.

Some experiments should fail.

Do the things which make you
different.

Liz Keogh
http://lizkeogh.com
@lunivore