



GOTO Copenhagen 2017
Conference Oct. 1-3, 2017

How do you Learn New Skills?

Emily Bache

 Follow us @gotocph



As a Professional Programmer - how do you learn new skills?

Emily Bache

@emilybache

<http://coding-is-like-cooking.info>

Emily Bache

Test Automation Specialist

Senior Consultant at Praqma

Author of

“The Coding Dojo Handbook”

@emilybache

emily.bache@praqma.com



PRAQMA

Programming



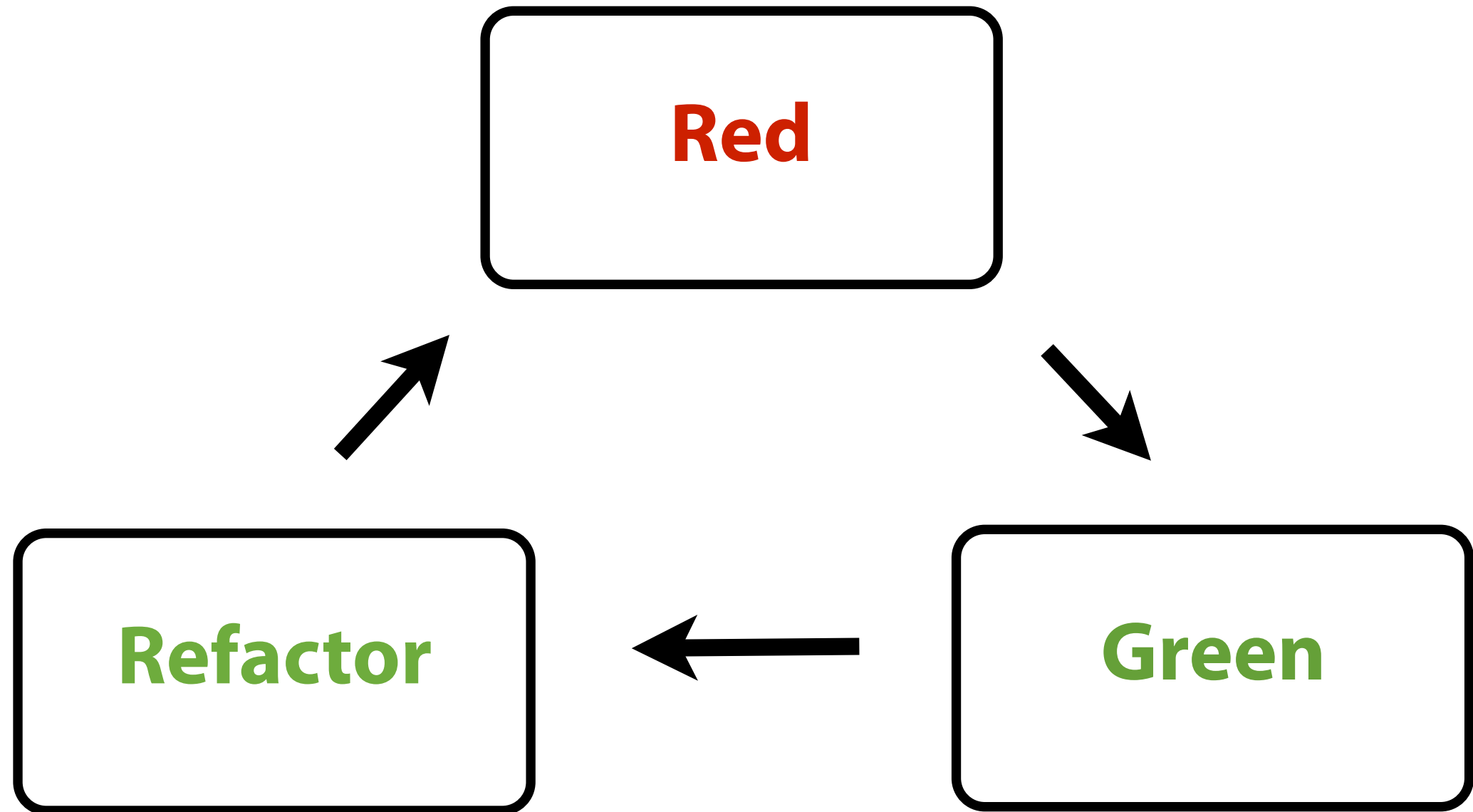
flickr: Matylda Czarnecka

Practical Coding Skills

- using IDE shortcuts
- Pair Programming
- Test Driven Development
- Designing good Test Cases
- Refactoring
- Working incrementally, committing code often
- Designing using SOLID principles
- Object Oriented Paradigm
- Functional Programming Paradigm
- ...



Test Driven Development



Why TDD?

refactoring support:
Make changes with
confidence



design verification:
the code does what
you think it does

design benefits:
isolated units should have
low coupling & high cohesion

work incrementally:
share your changes often,
maintain flow

Wonderful feeling of freedom & productivity!

Self-testing code

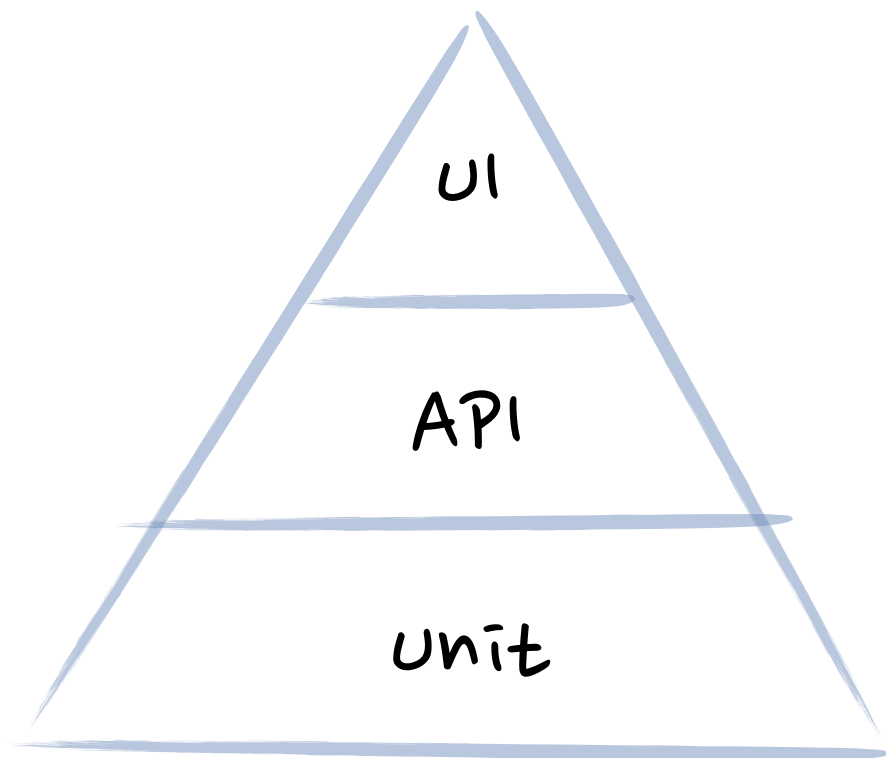
You have self-testing code when you can **run a series of automated tests** against the code base and be **confident** that, should the tests pass, your code is **free of any substantial defects**.

— Martin Fowler



<http://www.martinfowler.com/bliki/SelfTestingCode.html>

Agile Testing Pyramid



a few tests for the whole stack

many 'service' tests, may be public or test-specific API

majority are unit tests

TDD gives more than self-testing code

refactoring support:
Make changes with
confidence



design verification:
the code does what
you think it does

design benefits:
isolated units should have
low coupling & high cohesion

work incrementally:
share your changes often,
maintain flow

Wonderful feeling of freedom & productivity!

Kent Beck in 2009

The business problem with Max is that there just aren't enough of us who rely on tests minute-by-minute. Even allowing for my marketing and sales mistakes, the data suggests that there are at most a few thousand Java programmers actively practicing TDD. That's not a large enough market to sustain a business, especially as the market turned out to be price sensitive. Even if every Max subscriber successfully convinced 100 of their friends to sign up, there wouldn't have been enough revenue.

<http://www.threeriversinstitute.org/blog/?p=291>

Learning to Ski



flickr user nonanet

Cross Country skiing



Snowplow



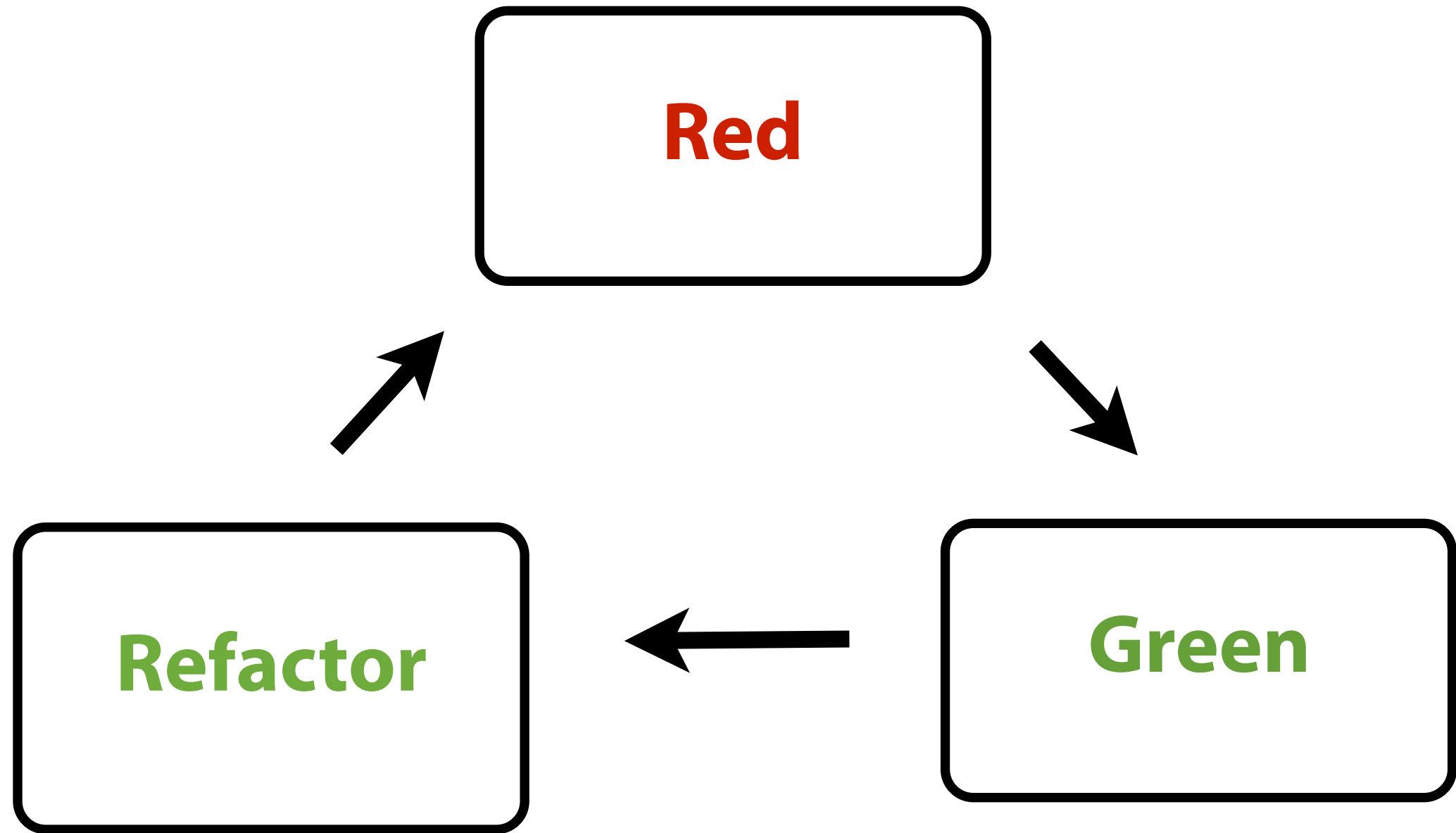
flickr: MichaEli

Parallel turns



flickr user nonanet

Test Driven Development



Training Course



image: <http://www.flickr.com/photos/fboyd/>

Learn on-the-job



image: flickr user Lisamarie Babik

Pair Program & Get Stuff Done

The Coding Dojo

Dojo = The place
you go to learn



What happens at a dojo meeting?

- 5 – 15 or so coders
- write code, collaborate, discuss



A Regular Coding Dojo



- Your team meets for a coding dojo every so often
- practice skills you can use in production code

Dojo Principles

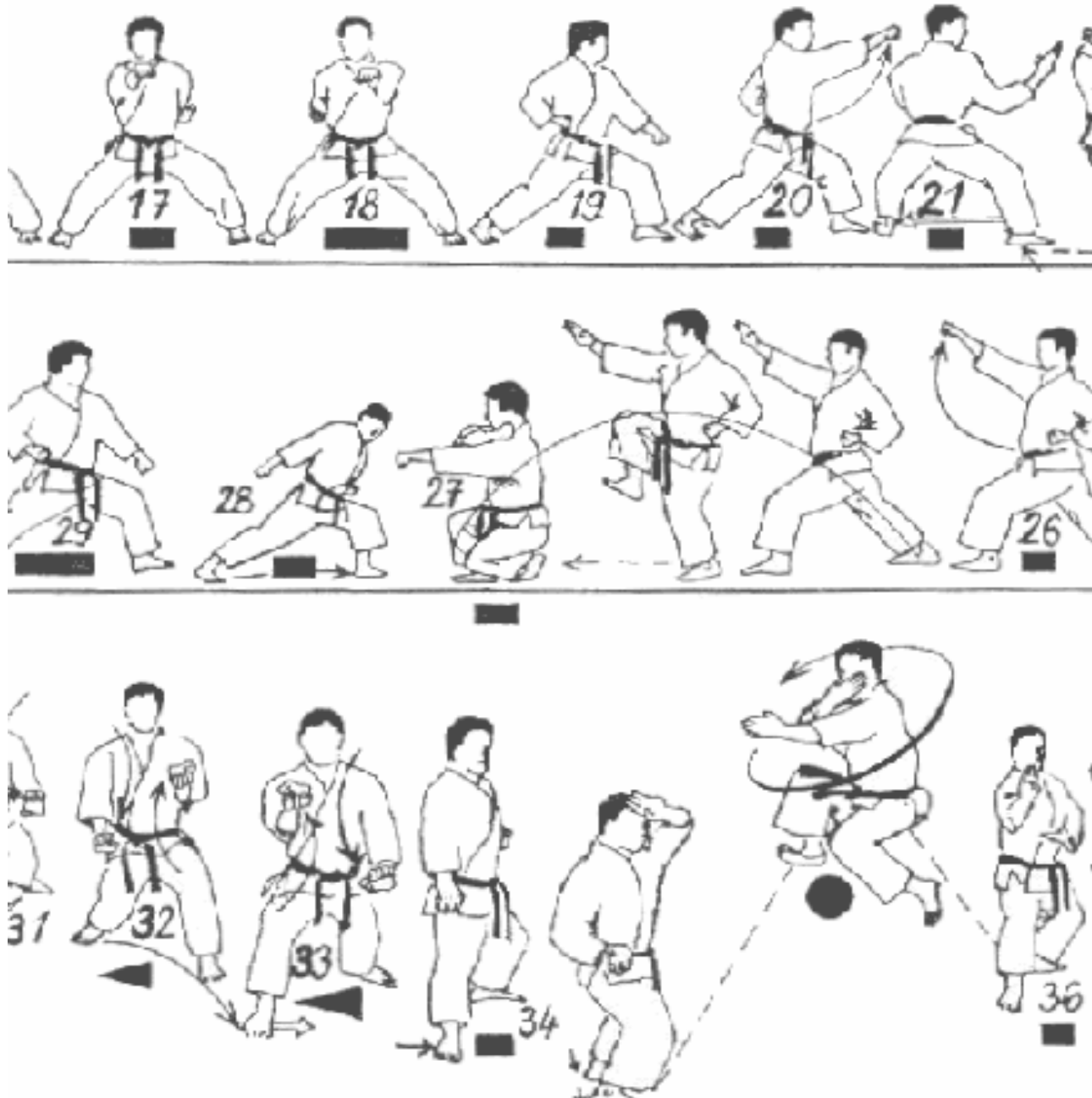
- **The first rule of the dojo:**
 - You can't discuss a technique without code
 - You can't show code without tests

Code without tests simply doesn't exist!



The Dojo Principles: http://bossavit.com/dojo/archives/2005_02.html

Code Kata



- In martial arts, a “Kata” is a sequence of moves that you learn.
- Dave Thomas proposed the idea of the “Code Kata”
 - <http://codekata.pragprog.com/>
- small exercises, that you repeat

Code Kata - Leap Years

Write a function that returns *true* or *false* depending on whether its input integer is a leap year or not.

A leap year is divisible by 4, but is not otherwise divisible by 100, unless it is also divisible by 400.

Examples:

1996 --> true

2001 --> false

2000 --> true

1900 --> false

2 kinds of Practice

Incidental Practice

- Repeatedly doing something you can already do, and improving at it

Deliberate Practice

Incidental Practice: Good Habits

**“I’m not a great programmer;
I’m just a good programmer
with great habits.”**

- Kent Beck



image taken by Martin Fowler at XP2002

quote from p57, “Refactoring” by Martin Fowler

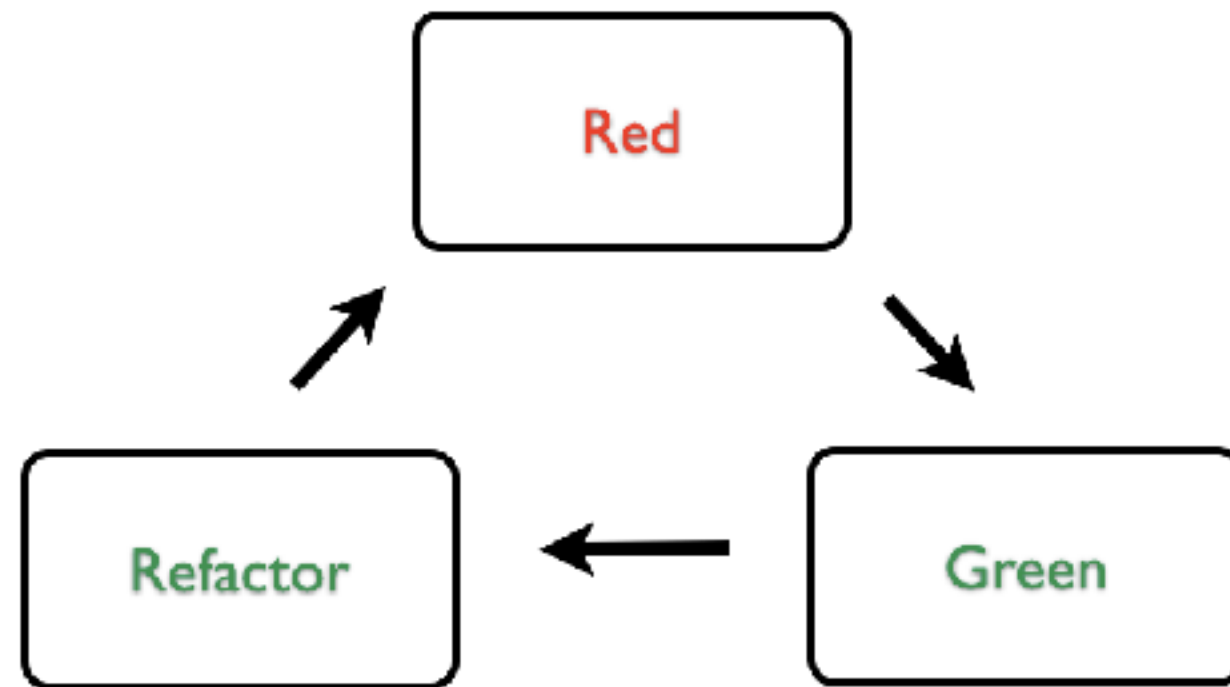
US Military training philosophy

“Under pressure, you don’t rise to the occasion, you
sink to the level of your training.

That’s why we train so hard”

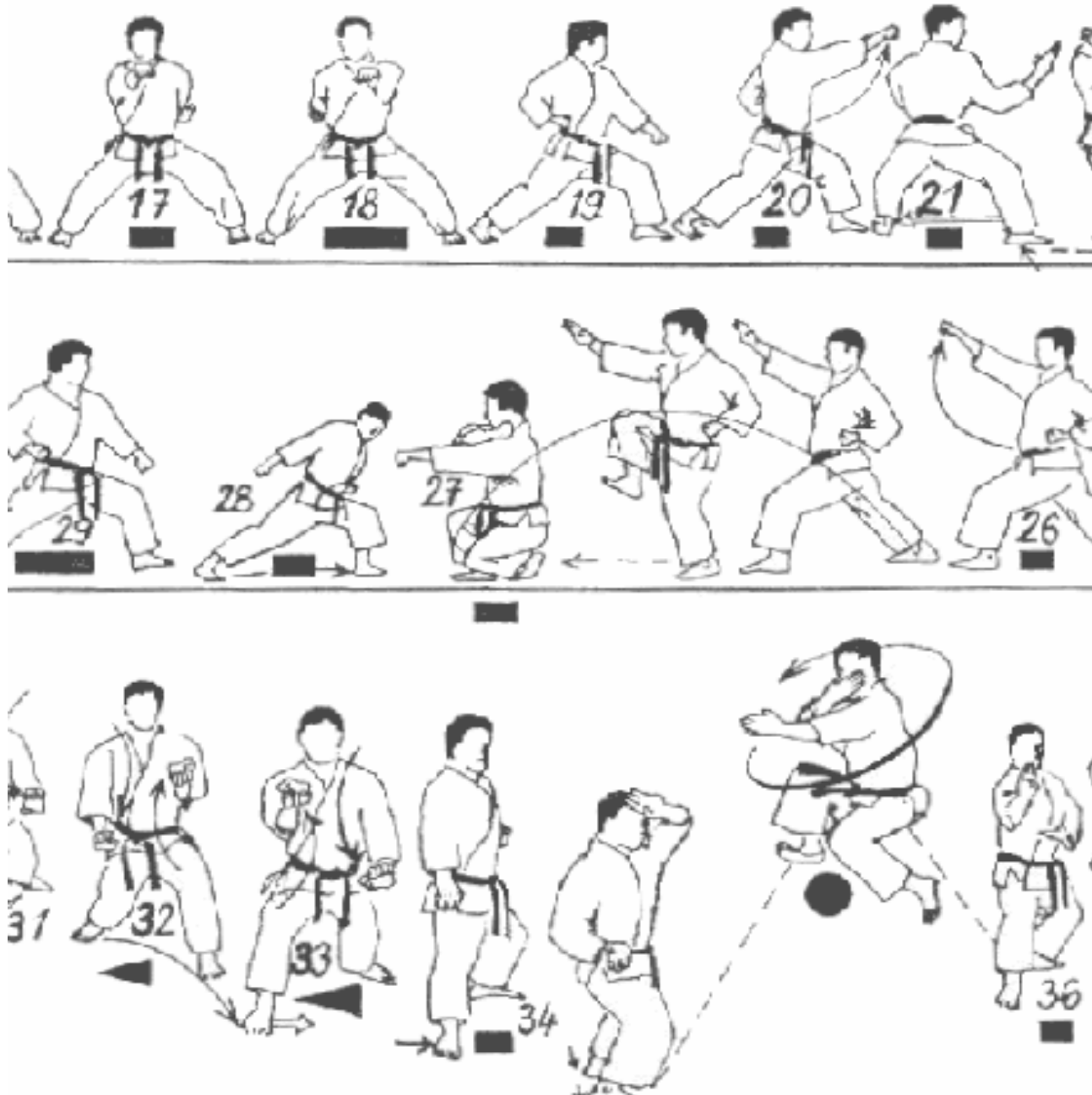
-- A US Navy Seal

Experience TDD



- Experience what TDD feels like when it works
- Recognise problems well suited to it

Repeatedly practice the same Katas



- In Karate there are some katas everyone learns
- There are some popular software katas lots of people have done

Practice

Incidental Practice:

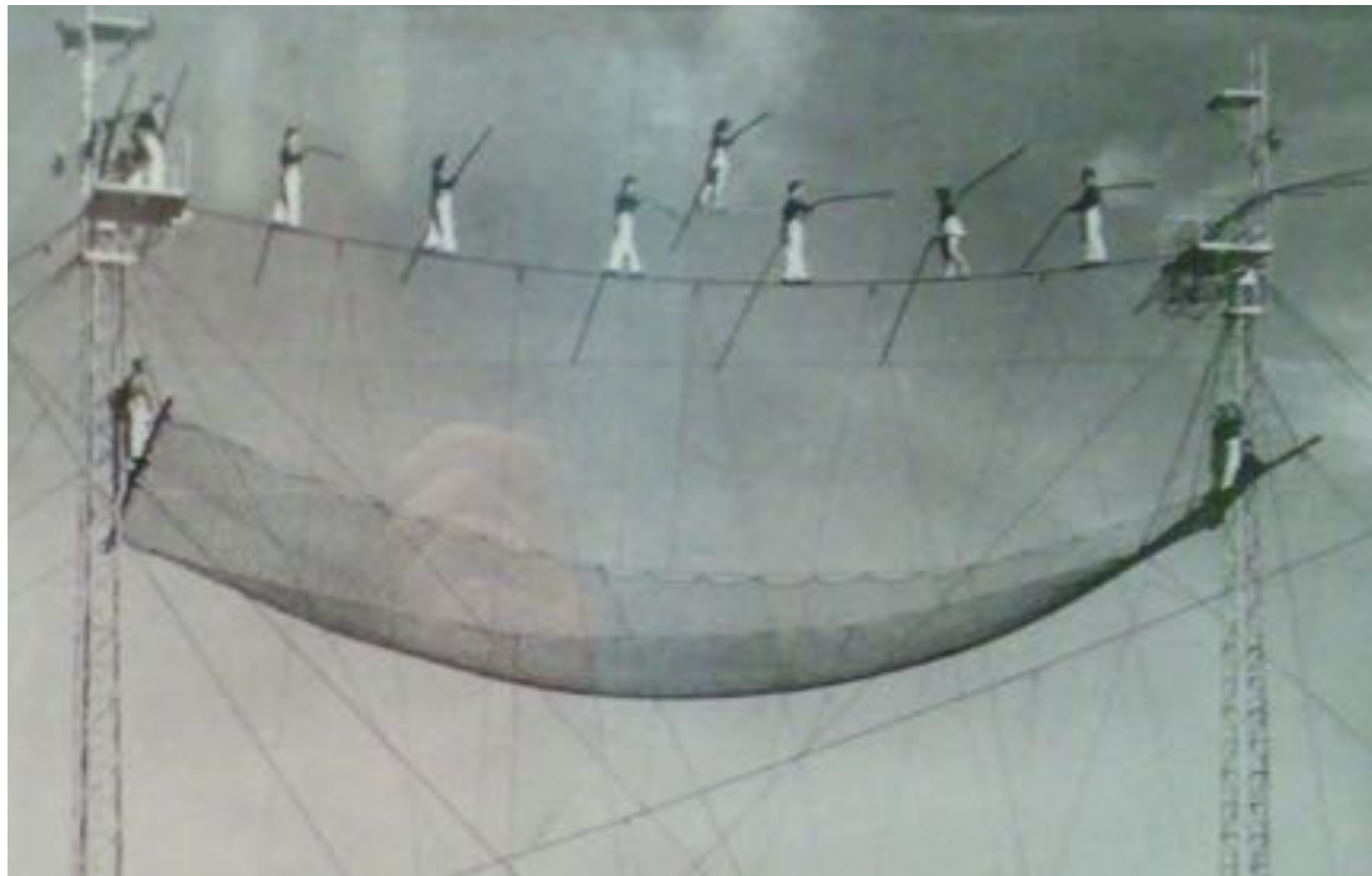
- Repeatedly doing something you can already do, and improving at it

Deliberate Practice:

- Trying to do something you can't comfortably do
- breaking down a skill into components you practice separately

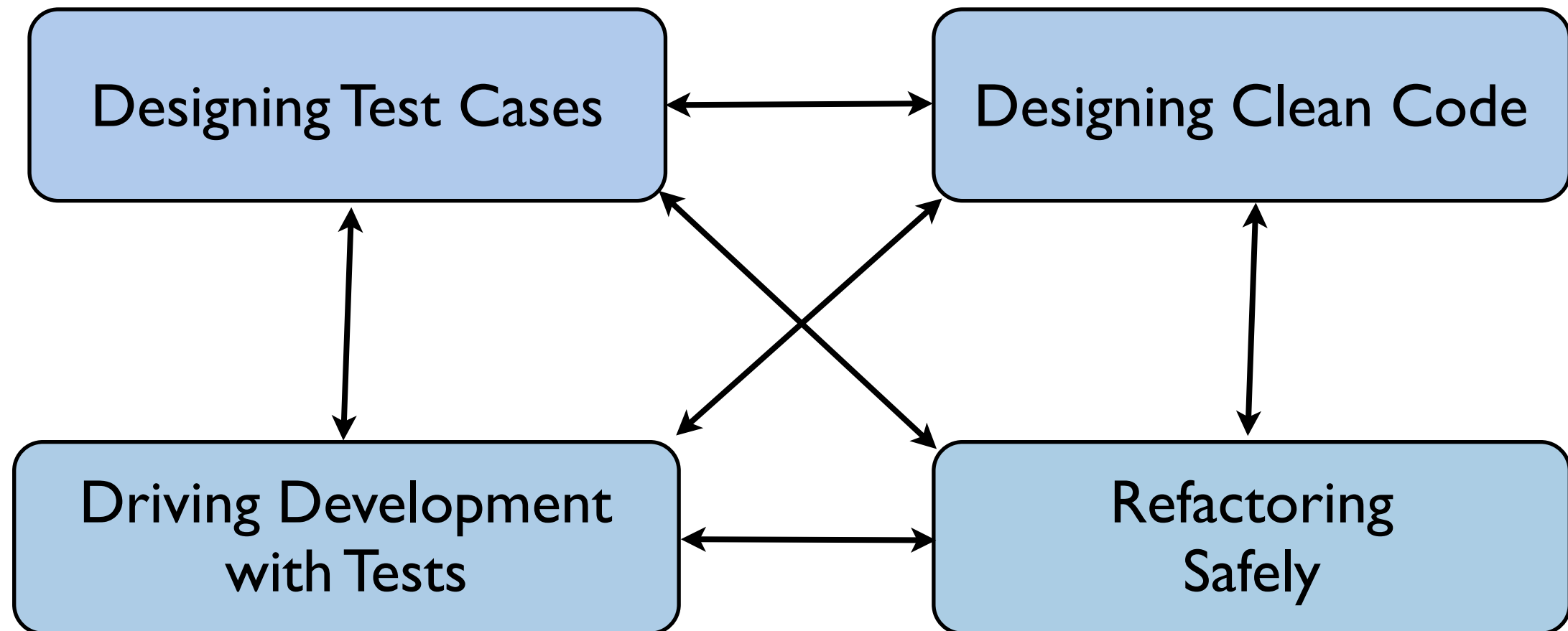
Deliberate Practice

- Need to feel safe
- Need to feel motivated



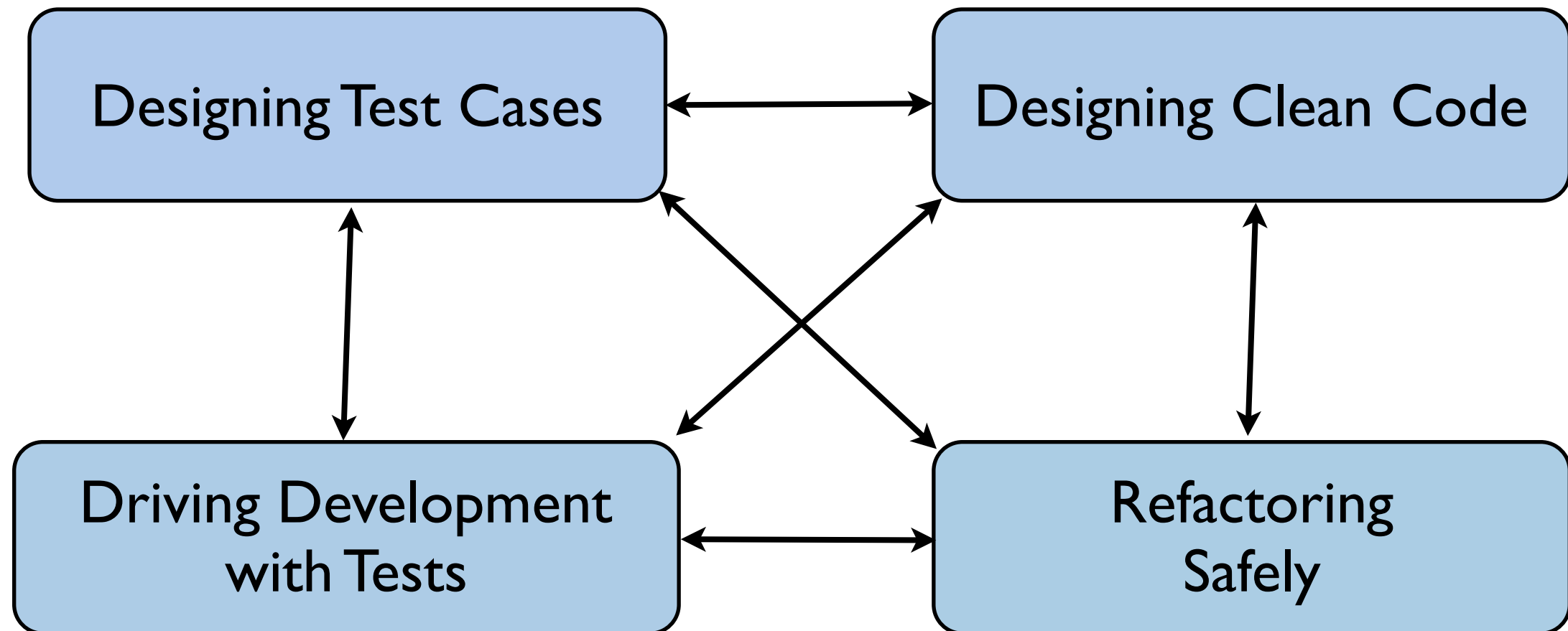
flickr user antony_mayfield

TDD skills



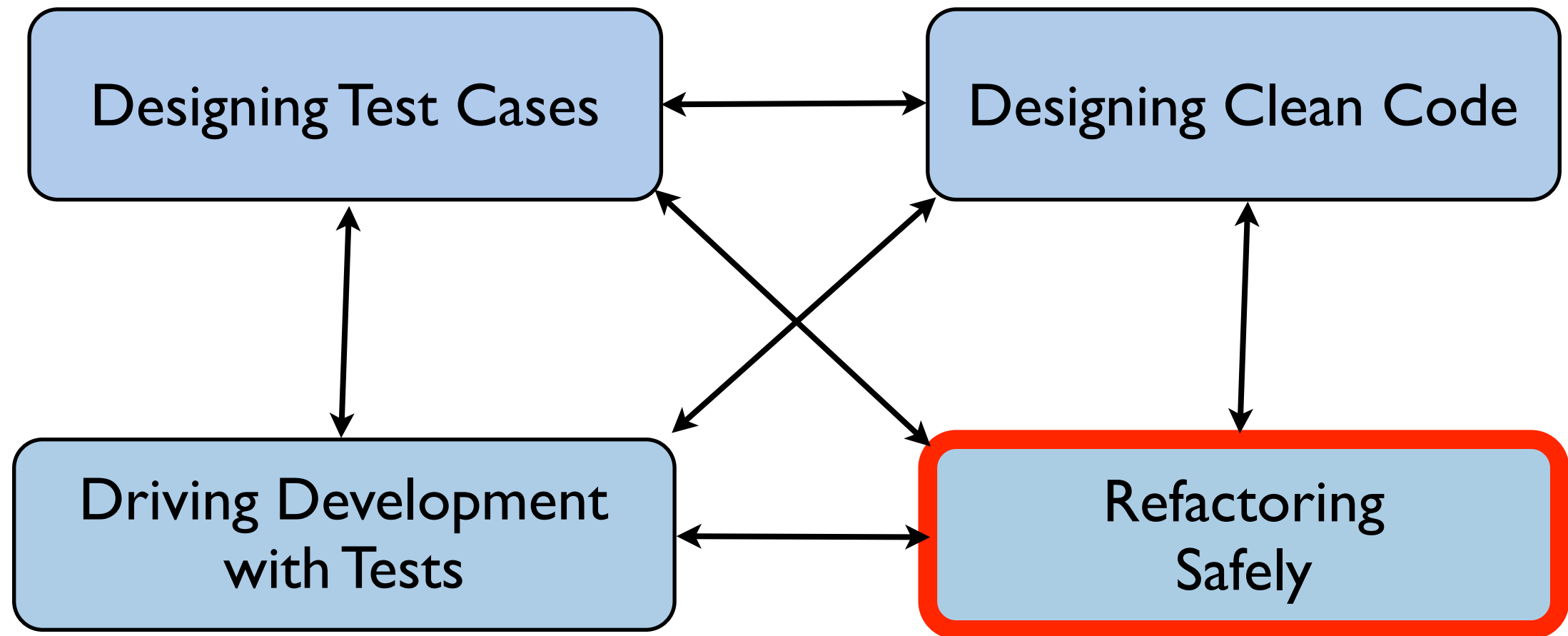
TDD is a composite skill

TDD skills



In the dojo we can focus on one at a time

TDD skills

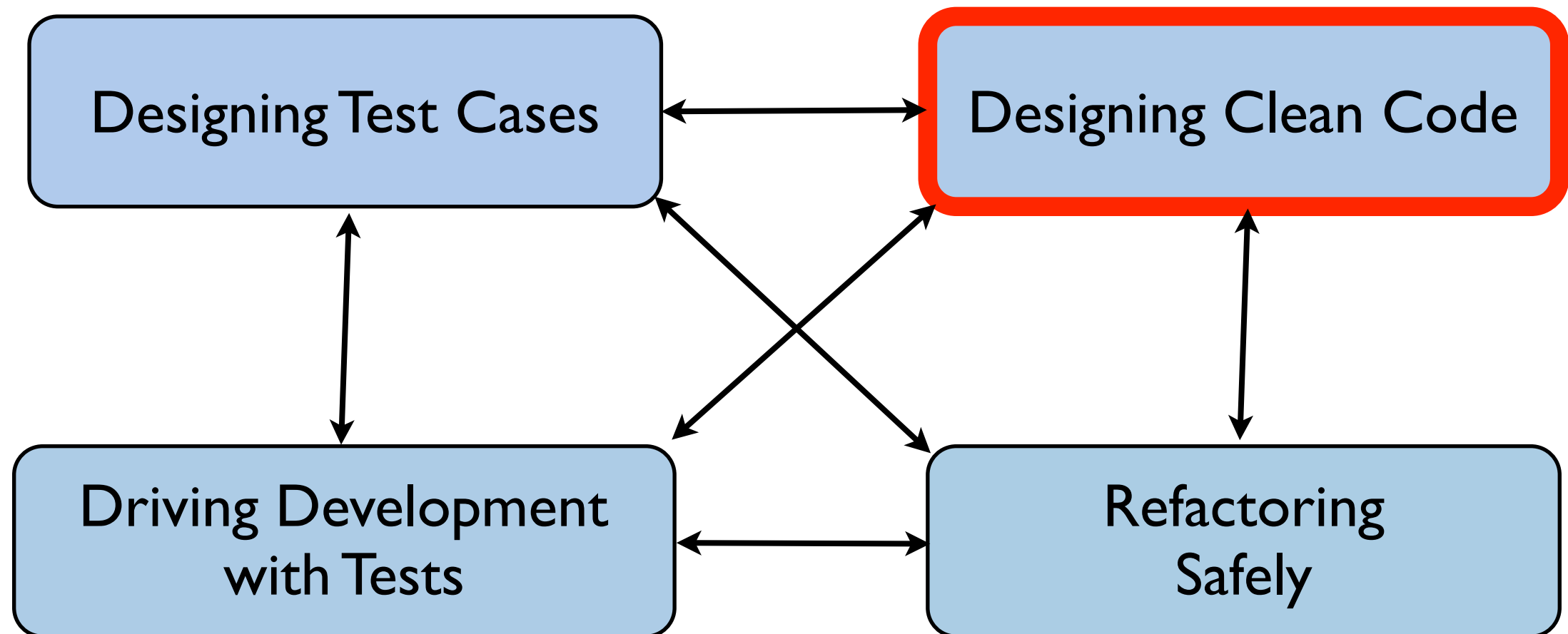


Refactoring Katas: Tennis, Yatzy, Gilded Rose...

Tennis Refactoring Kata

```
public String getScore() {
    String score = "";
    int tempScore=0;
    if (m_score1==m_score2)
    {
        switch (m_score1)
        {
            case 0:
                score = "Love-All";
                break;
            case 1:
                score = "Fifteen-All";
                break;
            case 2:
                score = "Thirty-All";
                break;
            default:
                score = "Deuce";
                break;
        }
    }
    else if (m_score1>=4 || m_score2>=4)
    {
        int minusResult = m_score1-m_score2;
        if (minusResult==1) score = "Advantage player1";
        else if (minusResult ==-1) score = "Advantage player2";
        else if (minusResult>=2) score = "Win for player1";
        else score = "Win for player2";
    }
}
```

TDD skills



SOLID principles Katas: Tyre Pressure, Leaderboard

Tyre Pressure Kata

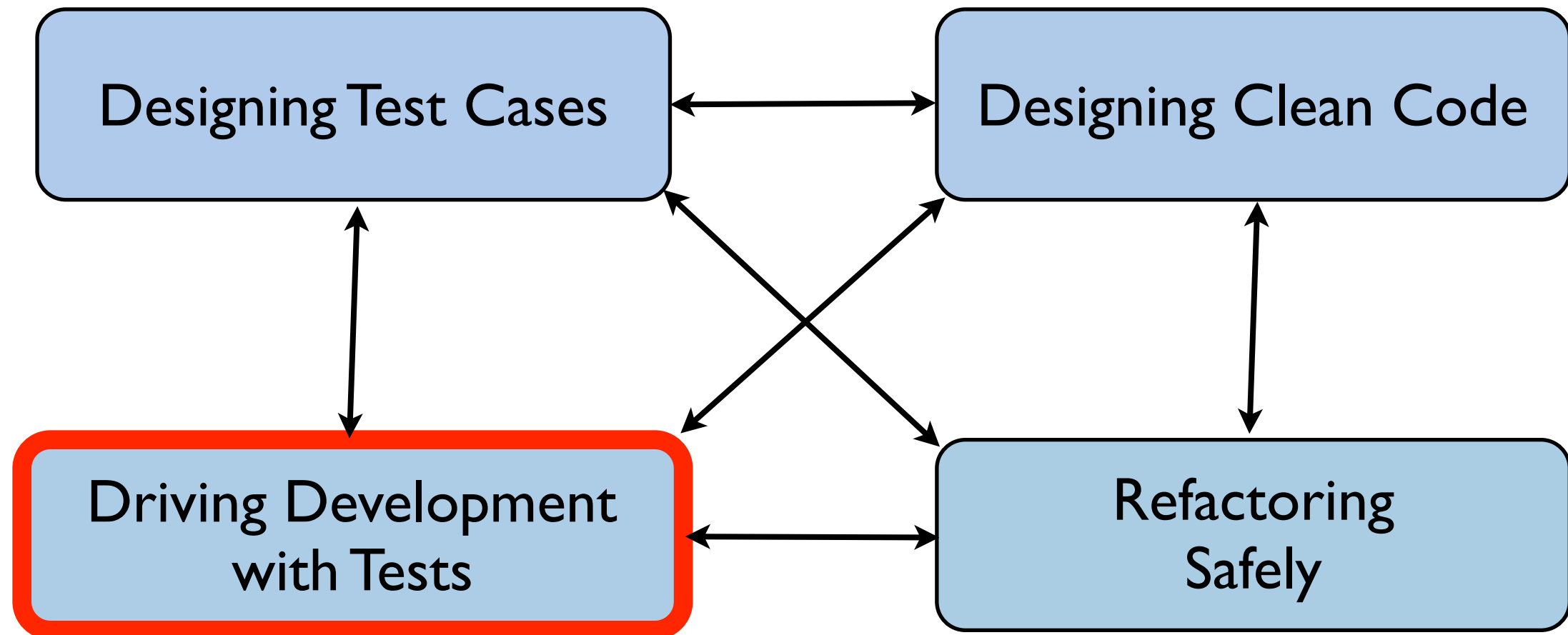
```
class Alarm(object):

    def __init__(self):
        self._low_pressure_threshold = 17
        self._high_pressure_threshold = 21
        self._sensor = Sensor()
        self._is_alarm_on = False

    def check(self):
        psi_pressure_value = self._sensor.pop_next_pressure_psi_value()
        if psi_pressure_value < self._low_pressure_threshold \
            or self._high_pressure_threshold < psi_pressure_value:
            self._is_alarm_on = True

    @property
    def is_alarm_on(self):
        return self._is_alarm_on
```

TDD skills



Diamond Kata, ISBN

Diamond Kata

Given a letter print a diamond starting with 'A' with the supplied letter at the widest point.

For example: `print-diamond 'E'` prints

```
  A
 B B
C   C
D   D
E   E
D   D
C   C
 B B
  A
```

For example: `print-diamond 'C'` prints

```
  A
 B B
C   C
 B B
  A
```

Diamond Kata

```
class DiamondIncrementalSpec extends FlatSpec with Matchers {

  "Diamond A" should "have one line containing A" in {
    Diamond.print('A') should be ("A")
  }

  "letter sequence" should "give the list of letters on each line in the diamond" ignore {
    //new Diamond('A').char_sequence should be (Seq('A'))
    //new Diamond('B').char_sequence should be (Seq('A', 'B', 'A'))
    //new Diamond('C').char_sequence should be (Seq('A', 'B', 'C', 'B', 'A'))
    //new Diamond('D').char_sequence should be (Seq('A', 'B', 'C', 'D', 'C', 'B', 'A'))
  }

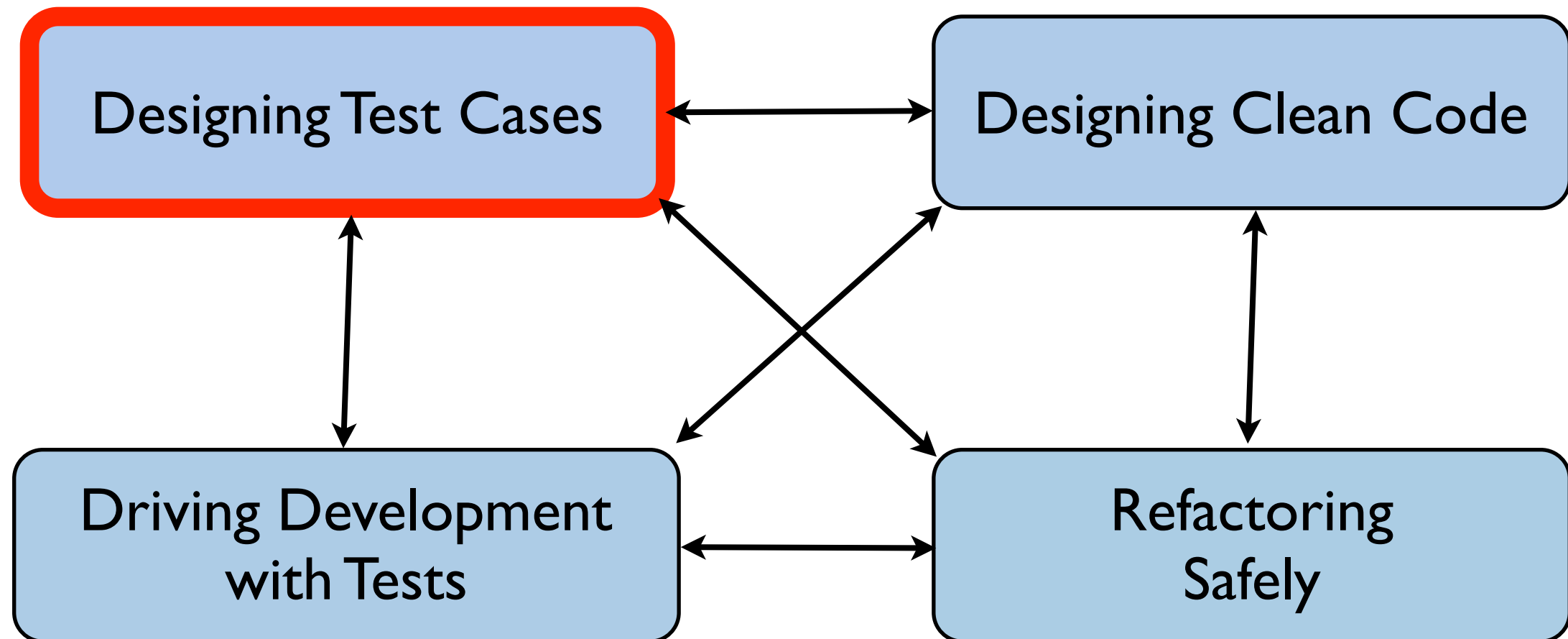
  "indents" should "give a list of indents for each line of the diamond" ignore {
    //new Diamond('A').indents should be (Seq(0))
    //new Diamond('B').indents should be (Seq(1, 0, 1))
  }
}
```

```
class DiamondIterativeSpec extends FlatSpec with Matchers {

  "Diamond A" should "have one line containing A" in {
    Diamond.print('A') should be ("A")
  }

  "Diamond B" should "print a char sequence for the top half" ignore {
    Diamond.print('B') should be ("AB")
  }
}
```


TDD skills



Gilded Rose, Functional Code

Functional Code Kata

```
public static double atan2(double y, double x) {  
    if (y + x == y) {  
        return y >= 0 ? HALF_PI : -HALF_PI;  
    }  
    y = atan(y / x);  
    if (x < 0) {  
        if (y <= 0) {  
            return y + PI;  
        } else {  
            return y - PI;  
        }  
    }  
    return y;  
}
```

[Pull requests](#)[Issues](#)[Gist](#)

Emily Bache

emilybache



Bache Consulting



Göteborg, Sweden



<http://coding-is-like-cooking.info>

[Contributions](#)[Repositories](#)[Public activity](#)

Popular repositories

[GildedRose-Refactoring-Kata](#)

Starting code for the GildedRose Refactoring ...

161 ★

[Tennis-Refactoring-Kata](#)

Starting code for a Refactoring Code Kata on t...

69 ★

[Racing-Car-Katas](#)

The starting code for several code katas on a r...

55 ★

[KataTrainReservation](#)

A Kata exercise. This one involves writing cod...

23 ★

[Yatzy-Refactoring-Kata](#)

Starting code for a Refactoring Code Kata on t...

14 ★

<https://github.com/emilybache>

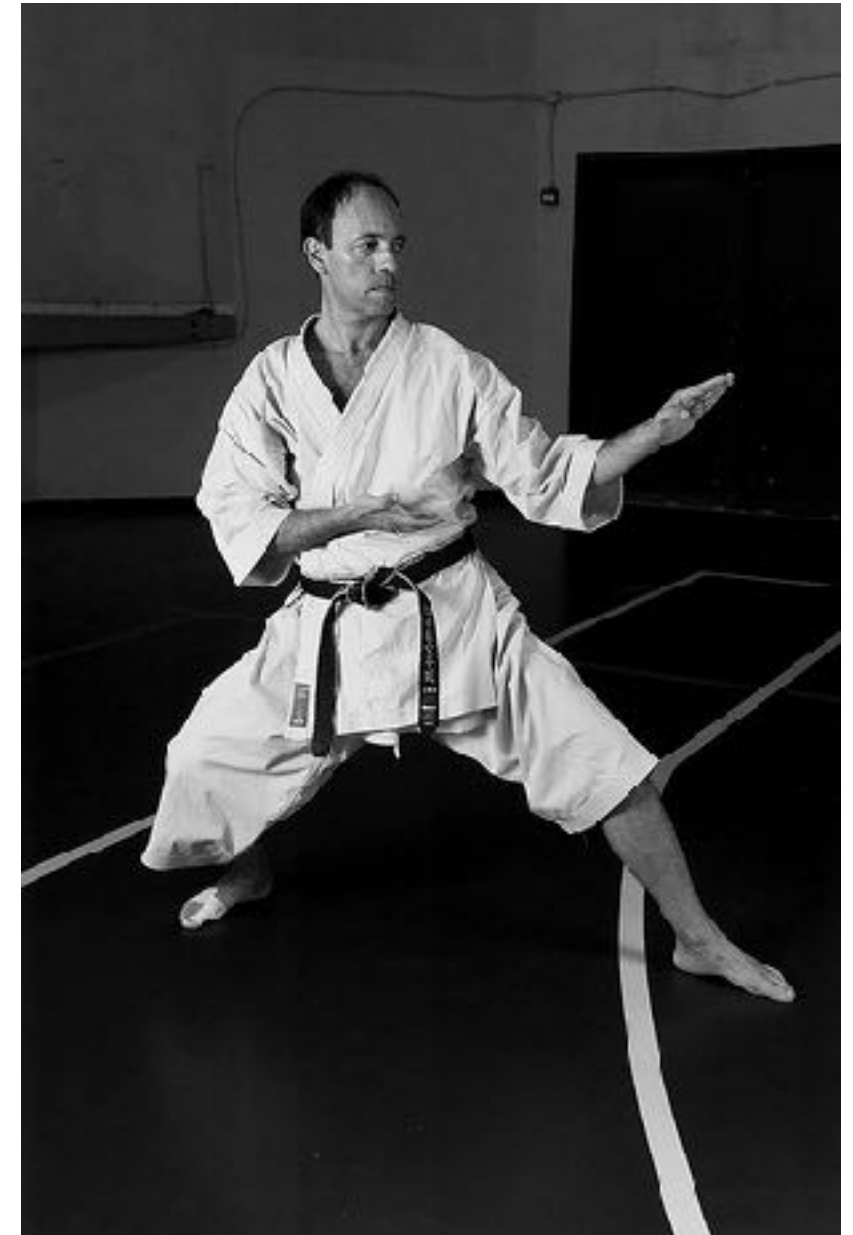
Essential Dojo Elements



- **Hold an introduction and retrospective**
- **Write tests as well as code**
- **Show your working**
- **Be a facilitator**

Dojo Principles: Mastery

- If it seems hard, find someone who can explain it.
- If it seems easy, explain to those who find it hard.
- No-one has mastery in all areas: everyone will both teach & learn.



*Sensei Henri Canditan
by flickr user Flavio~*

Coding Dojos in practice

- **Paris Dojo**
- **Ruby & Python User Groups**
- **Consultant flown-in for 5 or 6 sessions**
- **(semi) Regularly at Pagero**



Coding Dojos @ Pagero



Once a month



One or Two hours

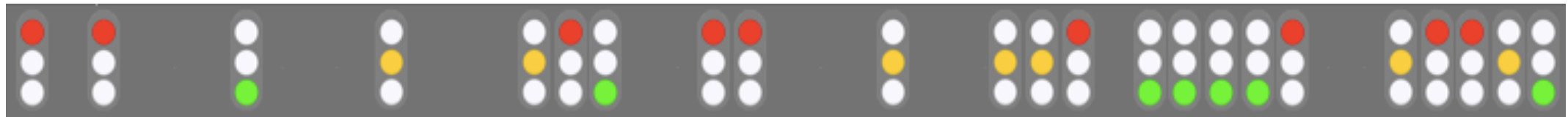
- **Standard structure**
 - Short intro
 - 45 minutes coding
 - short retrospective
 - (optional) repeat Kata

PAGERO

Improving at TDD

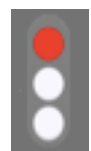


<http://cyber-dojo.org/>

TDD at first dojo



After 6 dojos



 Tests failing  Compiler/Syntax Error  Tests Passing

Coding Dojo Summary

- **Better coding skills**
- **Teach and Learn in a Group**
- **Incidental & Deliberate Practice**
- **Have Fun!**



goto;
copenhagen

GOTO Copenhagen 2017
Conference Oct. 1-3, 2017



**Click 'Rate Session'
to rate session
and ask questions.**

 Follow us @gotocph



As a Professional Programmer - how do you learn new skills?

Emily Bache

@emilybache

<http://coding-is-like-cooking.info>