

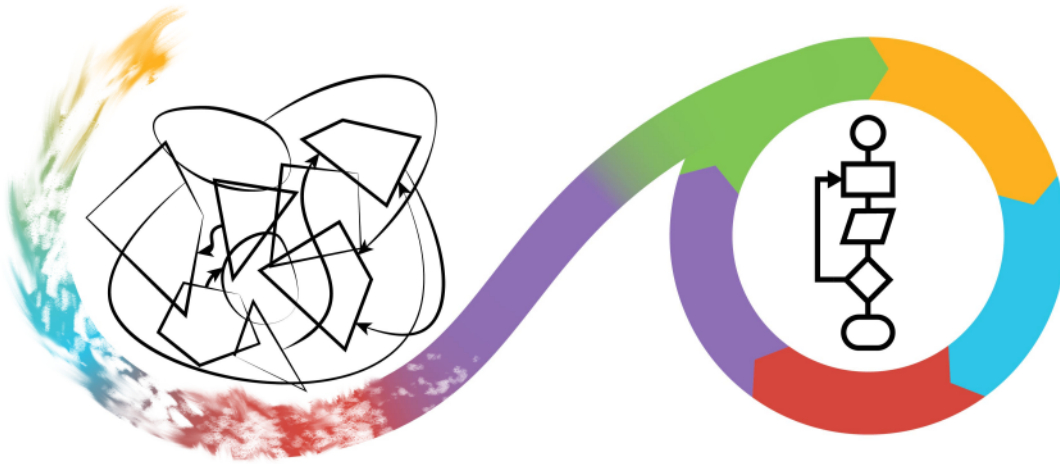
# Test-Driven Development

Richèl Bilderbeek

## 1 The Big Picture



[https://github.com/UPPMAX/programming\\_formalisms/blob/main/tdd/tdd\\_lecture/tdd\\_lecture.qmd](https://github.com/UPPMAX/programming_formalisms/blob/main/tdd/tdd_lecture/tdd_lecture.qmd)



### 1.1 Breaks

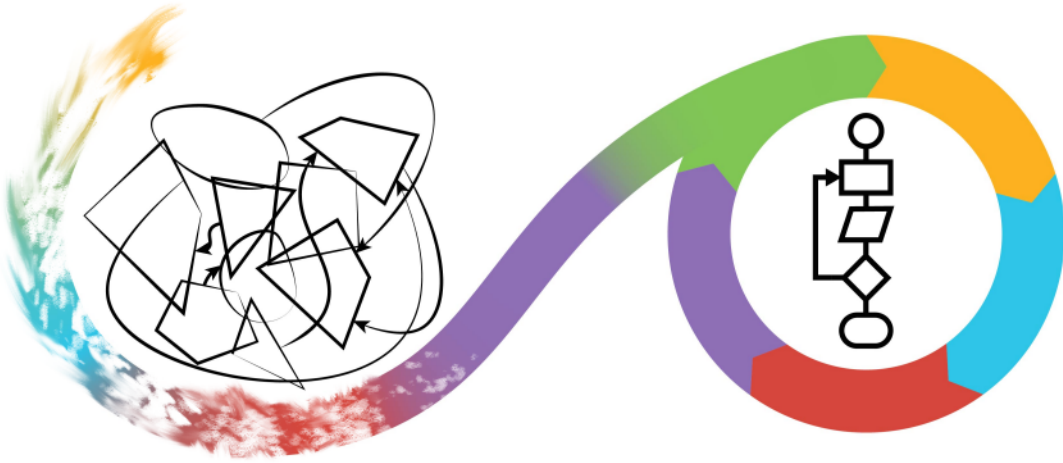
Please take breaks: these are important for learning. Ideally, do something boring (1)!

### 1.2 Schedule

Day	From	To	What
Wed	9:00	10:00	TDD
Wed	10:00	10:15	Break
Wed	10:15	11:00	TDD

Day	From	To	What
Wed	11:00	11:15	Break
Wed	11:15	12:00	TDD
Wed	12:00	13:00	Lunch

## 2 Growing code



### 2.1 Problem

How do you grow/develop your code?

20<sup>th</sup> ANNIVERSARY EDITION

# The Pragmatic Programmer

your journey to mastery

DAVID THOMAS  
ANDREW HUNT



## 2.2 Newbie developers

‘Just start somewhere’



## 2.3 Experienced developers

Work systematically



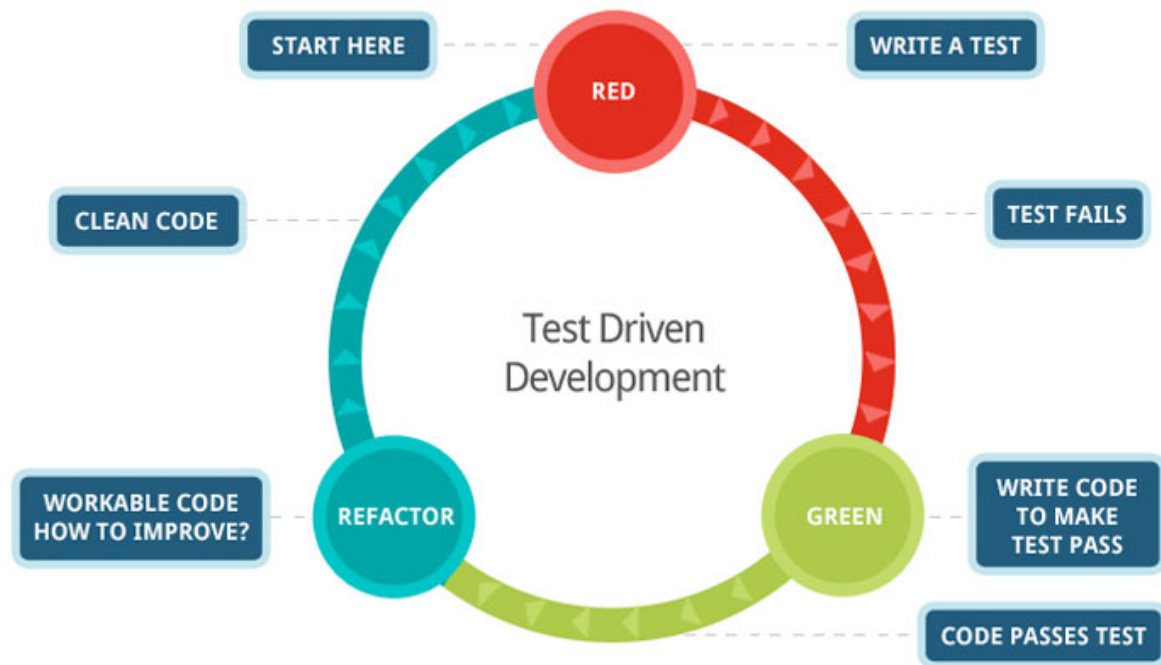


## 2.4 TDD

Short for 'Test-driven development' A systematic way to grow code



## 2.5 TDD cycle



## 3 Example exercise: `is_zero`

- Only observe, no type-along!
- Ask questions on the go! When in doubt: ask that question!
- Time: 15 minutes

### 3.1 Example exercise: `is_zero`

- Function name: `is_zero`
- Output:
  - Returns `True` if the input is zero
  - Returns `False` if the input is not zero
  - Gives an error when the input is not a number
- Zen Of Python: 'Errors should never pass silently'

### 3.2 Example exercise: `is_zero` technical

- Use the GitHub repository for the learners of this course
- Work on the main branch
- Work in a file called `learners/[your_name]/is_zero.py`

### 3.3 Live demo (15 minutes)

- Only observe, no type-along!

Videos:

- [Python video for 'is\\_zero'](#), from 3:02
- [R video for 'is\\_one'](#)
- Or see also slides beyond end

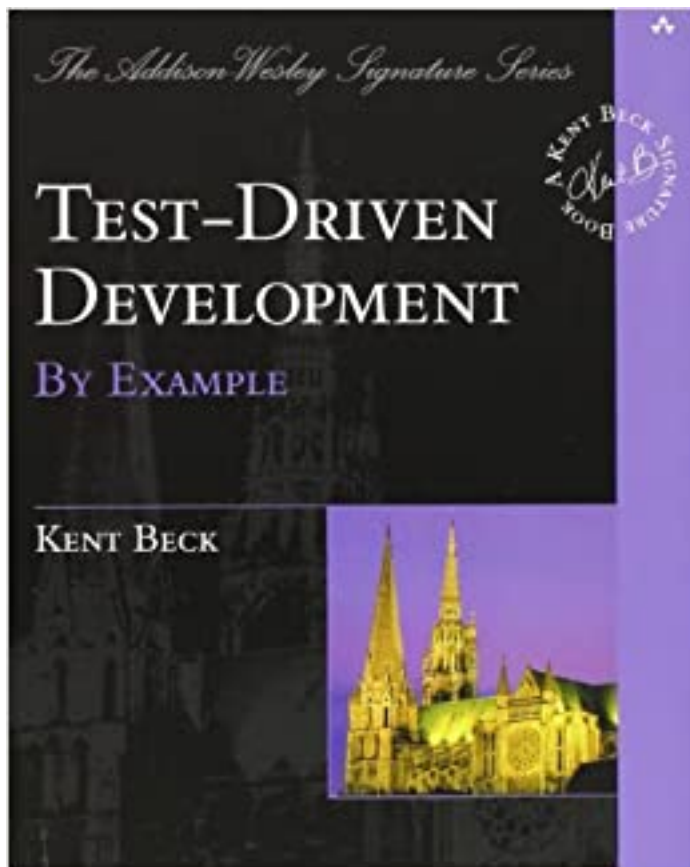
### 3.4 Reflection

Q: Do developers really do this?

. . .

A: Yes (2)(3)





## Modern C++ Programming with Test-Driven Development

Code Better,  
Sleep Better



Jeff

Edited by Mic

### 4 Exercise 1: `is_even`

- Time: 30 mins

#### 4.1 Exercise 1: `is_even` (30 mins)

- Develop a function called `is_even`
- Output:
  - Returns `True` if the input is even
  - Returns `False` if the input is not even
  - Gives an error when the input is not a number
- Try to be **exemplary**
- We'll discuss a random commit history

#### 4.2 Exercise 1: `is_even` technical (30 mins)

- Use the GitHub repository for the learners of this course
- Work on the main branch
- Create a file called `learners/[your_name]/is_even.py`

Done? Write `is_odd`, then `is_probability`.

#### 4.3 Exercise 1 feedback

- ☐ Ask for a volunteer for feedback
  - If none: pick a random folder
- ☐ Discuss history

#### 4.4 Reflection

Q: Does this really save time?

...

A: No, it takes longer

Study	Extra time	Effect
(4)	16%	18% more black-box tests pass
(5)	15%	2x higher code quality
(6)	15-35%	40%-90% less defects

## 4.5 Reflection

Q: Why do TDD?

. . .

A:

- TDD makes developers more productive (7)
- TDD increases quality of the code (7) (8) (9)
  - There are plenty of costly programming mistakes documented!
- TDD helps shape the project architecture (10)
- TDD helps better modularisation (11)
- TDD works great with Xtreme programming and CI

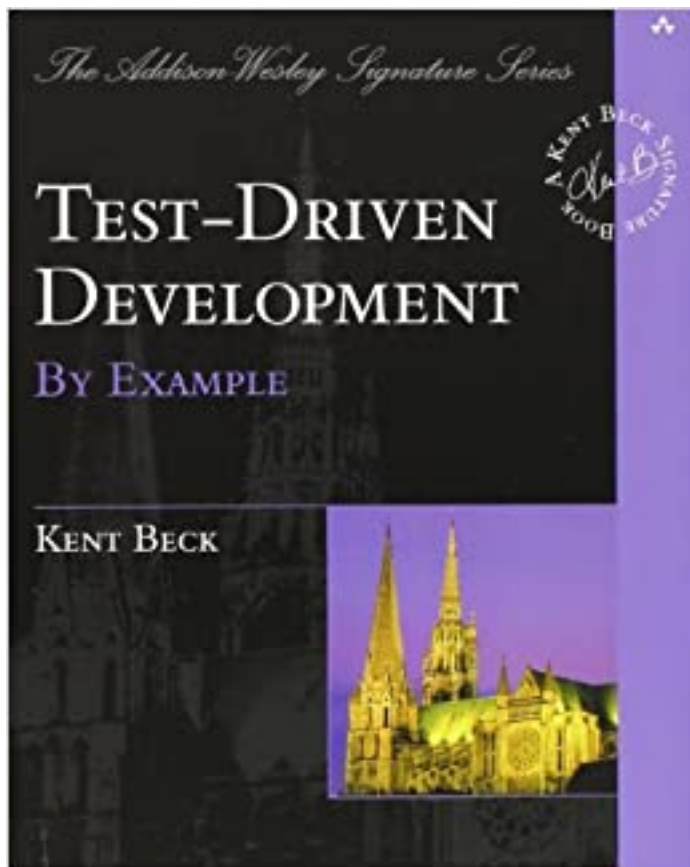
## 4.6 Reflection

Q: How many tests should I write?

. . .

A: Until you cannot break your function anymore (2)(3)

The  
Pragmatic  
Programmers



## Modern C++ Programming with Test-Driven Development

Code Better,  
Sleep Better



Jeff

Edited by Mic

### 5 Exercise 2: `is_odd`

- Time: 30 mins

## 5.1 Exercise 2: `is_odd` (30 mins)

- Develop a function called `is_odd`
- Output:
  - Returns `True` if the input is odd
  - Returns `False` if the input is not odd
  - Gives an error when the input is not a number
- Try to be **exemplary**
- We'll discuss a commit history after the exercise

## 5.2 Exercise 2: `is_odd`, technical (30 mins)

- Use the GitHub repository for the learners of this course, e.g. [https://github.com/programming-formalisms/programming\\_formalisms\\_project\\_autumn\\_2023](https://github.com/programming-formalisms/programming_formalisms_project_autumn_2023)
- Work on the main branch
- Use a file called `learners/[your_name]/is_odd.py`

Done? Try exercise 3: `is_probability`.

## 5.3 Exercise 2 feedback

- ☐ Ask for a volunteer for feedback
  - If none: pick a random folder
- ☐ Discuss history

## 6 Exercise 3: `is_probability`

- Time: 30 mins

### 6.1 Exercise 3: `is_probability` (30 mins)

- Develop a function called `is_probability`
- Output:
  - Returns `True` if the input is in range `[0.0, 1.0]`
  - Returns `False` if the input is outside that range
  - Gives an error when the input is not a floating point number



- Try to be **exemplary**
- We'll discuss a commit history after the exercise

## 6.2 Exercise 3: `is_probability`, technical (30 mins)

- Use the GitHub repository for the learners of this course
- Work on the main branch
- Use a file called `learners/[your_name]/is_probability.py`

## 6.3 Exercise 3 feedback

- ☐ Ask for a volunteer for feedback
  - If none: pick a random folder
- ☐ Discuss history

## 6.4 Extra exercises

Done?

Exercise	Function name	Function purpose
4	<code>is_number</code>	Determines if an object is a number
5	<code>are_numbers</code>	Determines if an object is a list of numbers
S1	<code>is_roman_number</code>	Determine if a string is a roman number
S2	<code>is_prime</code>	Determine if a number is a prime number

## 6.5 Bottom line

- This session, we wrote **unit tests**
- It is only those your boss may read
- The literature assumes a responsible programmer writes tests, in C++ (12), R (13) and Python (14)

20<sup>th</sup> ANNIVERSARY EDITION

# The Pragmatic Programmer

your journey to mastery

DAVID THOMAS  
ANDREW HUNT



## 6.6 Weaknesses

- We only test manually
- We only test on our computer
- We are not sure if our functions are tested completely
- We do not test the code for style
- We should consider using a testing framework

These are addressed in the session called 'Testing' :-)

## 6.7 Questions?

Questions?

## 6.8 The End



## 6.9 TDD cycles in text

- In both Python and R

## 6.10 First example: `is_zero`

- Function name: `is_zero`
- Output:
  - Returns `True`/`TRUE` if the input is zero
  - Returns `False`/`FALSE` if the input is not zero
  - Gives an error when the input is not a number

## 6.11 Cycle 1, red: write a test that breaks



```
assert is_zero(0)
```



```
library(testthat)
expect_true(is_zero(0))
```

code that is not run, uses `is_zero`, as a worm cannot run.

## 6.12 Cycle 1, green: make the test pass



```
def is_zero(number):
    return True

assert is_zero(0)
```



```
library(testthat)

is_zero <- function(number) {
  TRUE
}

expect_true(is_zero(0))
```

## 6.13 Cycle 1, blue: refactor and commit

```
git add .
git commit -m "Add stub of 'is_zero'"
git push
```



#### 6.14 Cycle 2, red: write a test that breaks



```
assert is_zero(0)
assert not is_zero(42)
```





```
expect_true(is_zero(0))  
expect_false(is_zero(42))
```

## 6.15 Cycle 2, green: make the test pass



```
def is_zero(x):  
    return x == 0  
  
assert is_zero(0)  
assert not is_zero(42)
```

indent of 2 is non-standard, see [PEP 8](#)



```
library(testthat)  
  
is_zero <- function(number) {  
    number == 0  
}  
  
expect_true(is_zero(0))  
expect_false(is_zero(42))
```

## 6.16 Cycle 2, blue: refactor and commit

```
git add .  
git commit -m "'is_zero' responds correctly to numbers"  
git push
```



## 7 Breaks

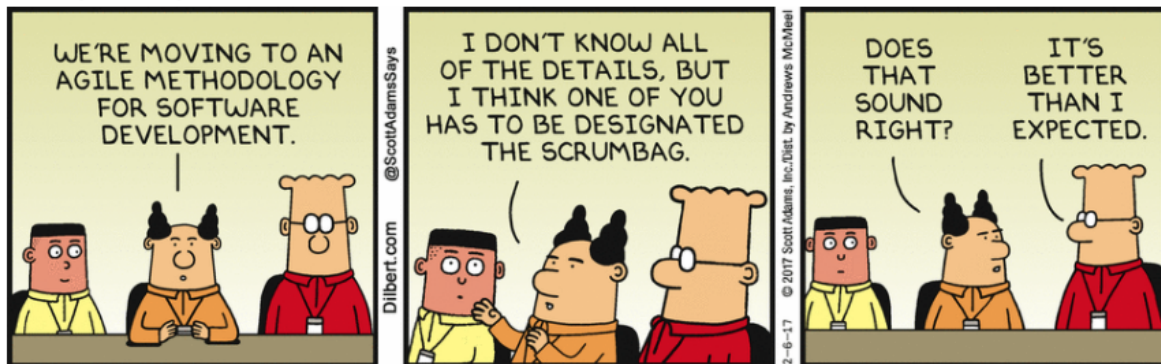
I put the break slides in the end

## 7.1 Break 1: 10:00-10:15

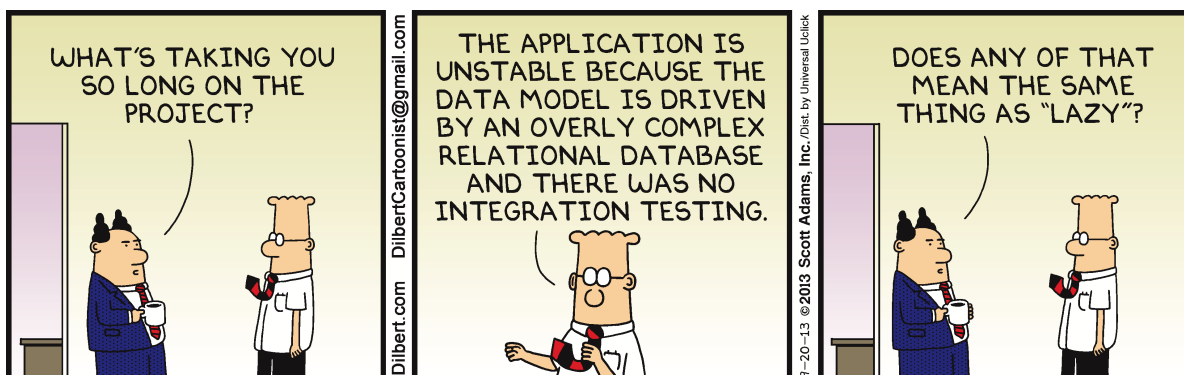


## 7.2 Break 2: 11:00-11:15

Monday February 06, 2017 *Agile Methodology*



## 7.3 Lunch: 12:00-13:00



## References

1. Newport C. Deep work: Rules for focused success in a distracted world. Hachette UK; 2016.
2. Beck K. Test driven development: By example. Addison-Wesley Professional; 2022.
3. Langr J. Modern c++ programming with test-driven development: Code better, sleep better. Modern C++ Programming with Test-Driven Development. 2013;1–368.
4. George B, Williams L. A structured experiment of test-driven development. Information and software Technology. 2004;46(5):337–42.
5. Bhat T, Nagappan N. Evaluating the efficacy of test-driven development: Industrial case studies. In: Proceedings of the 2006 ACM/IEEE international symposium on empirical software engineering. 2006. p. 356–63.
6. Nagappan N, Maximilien EM, Bhat T, Williams L. Realizing quality improvement through test driven development: Results and experiences of four industrial teams. Empirical Software Engineering. 2008;13:289–302.
7. Erdogmus H, Morisio M, Torchiano M. On the effectiveness of the test-first approach to programming. IEEE Transactions on software Engineering. 2005;31(3):226–37.
8. Alkaoud H, Walcott KR. Quality metrics of test suites in test-driven designed applications. International Journal of Software Engineering Applications (IJSEA). 2018;2018.
9. Janzen DS, Saiedian H. Test-driven learning: Intrinsic integration of testing into the CS/SE curriculum. Acm Sigcse Bulletin. 2006;38(1):254–8.
10. Mayr H. Projekt engineering: Ingenieurmäßige softwareentwicklung in projektgruppen. Hanser Verlag; 2005.
11. Madeyski L, información G de sistemas de. Test-driven development: An empirical evaluation of agile practice. Springer; 2010.
12. Stroustrup B, Sutter H, et al. C++ core guidelines. Web Last accessed February. 2018;
13. Wickham H. Advanced R. CRC press; 2019.
14. Van Rossum G, Warsaw B, Coghlan N. PEP 8–style guide for Python code. Python org. 2001;1565.