Introduction to TDD

Dario Campagna

Let’s start with

Development

Now add **Test Driven**

Software development practice

Clean code that works

Test first

Small steps, fast feedback

Example of Rotten Code

TerrariaClone class from the GitHub repository TerrariaClone.

- > 6500 lines of code
- > 1300 lines of code for init() method
- Deeply nested if and for statements
- Many other code smells

```
3056                if (ic != null) {
3057                    if (ic.type.equals("workbench")) {
3058                        for (ux=0; ux<3; ux++) {
3059                            for (uy=0; uy<3; uy++) {
3060                                if (mousePos[0] >= ux*40+6 && mousePos[0] < ux*40+46 &&
3061                                    mousePos[1] >= uy*40+inventory.image.getHeight()+46 &&
3062                                    mousePos[1] < uy*40+inventory.image.getHeight()+86) {
3063                                    checkBlocks = false;
3064                                    if (mouseClicked) {
3065                                        mouseNoLongerClicked = true;
3066                                        moveItemTemp = ic.ids[uy*3+ux];
3067                                        moveNumTemp = ic.nums[uy*3+ux];
3068                                        if (moveItem == ic.ids[uy*3+ux]) {
3069                                            moveNum = (short)inventory.addLocationIC(ic, uy*3+ux, moveItem, moveNum, moveDur);
3070                                            if (moveNum == 0) {
3071                                                moveItem = 0;
3072                                            }
3073                                        }
3074                                        else {
3075                                            inventory.removeLocationIC(ic, uy*3+ux, ic.nums[uy*3+ux]);
3076                                            if (moveItem != 0) {
3077                                                inventory.addLocationIC(ic, uy*3+ux, moveItem, moveNum, moveDur);
3078                                            }
3079                                            moveItem = moveItemTemp;
3080                                            moveNum = moveNumTemp;
3081                                        }
3082                                    }
3083                                }
3084                            }
3085                        }
3086                        if (mousePos[0] >= 4*40+6 && mousePos[0] < 4*40+46 &&
3087                            mousePos[1] >= 1*40+inventory.image.getHeight()+46 &&
3088                            mousePos[1] < 1*40+inventory.image.getHeight()+86) {
3089                            checkBlocks = false;
3090                            if (mouseClicked) {
3091                                if (moveItem == ic.ids[9] && moveNum + ic.nums[9] <= MAXSTACKS.get(ic.ids[9])) {
3092                                    moveNum += ic.nums[9];
3093                                    inventory.useRecipeWorkbench(ic);
3094                                }
3095                                if (moveItem == 0) {
3096                                    moveItem = ic.ids[9];
3097                                    moveNum = ic.nums[9];
3098                                    if (TOOLDURS.get(moveItem) != null) {
3099                                        moveDur = TOOLDURS.get(moveItem);
3100                                    }
3101                                    inventory.useRecipeWorkbench(ic);
3102                                }
```

Rotting code is

Rigid

Fragile

Inseparable

Opaque

Why does it rot?

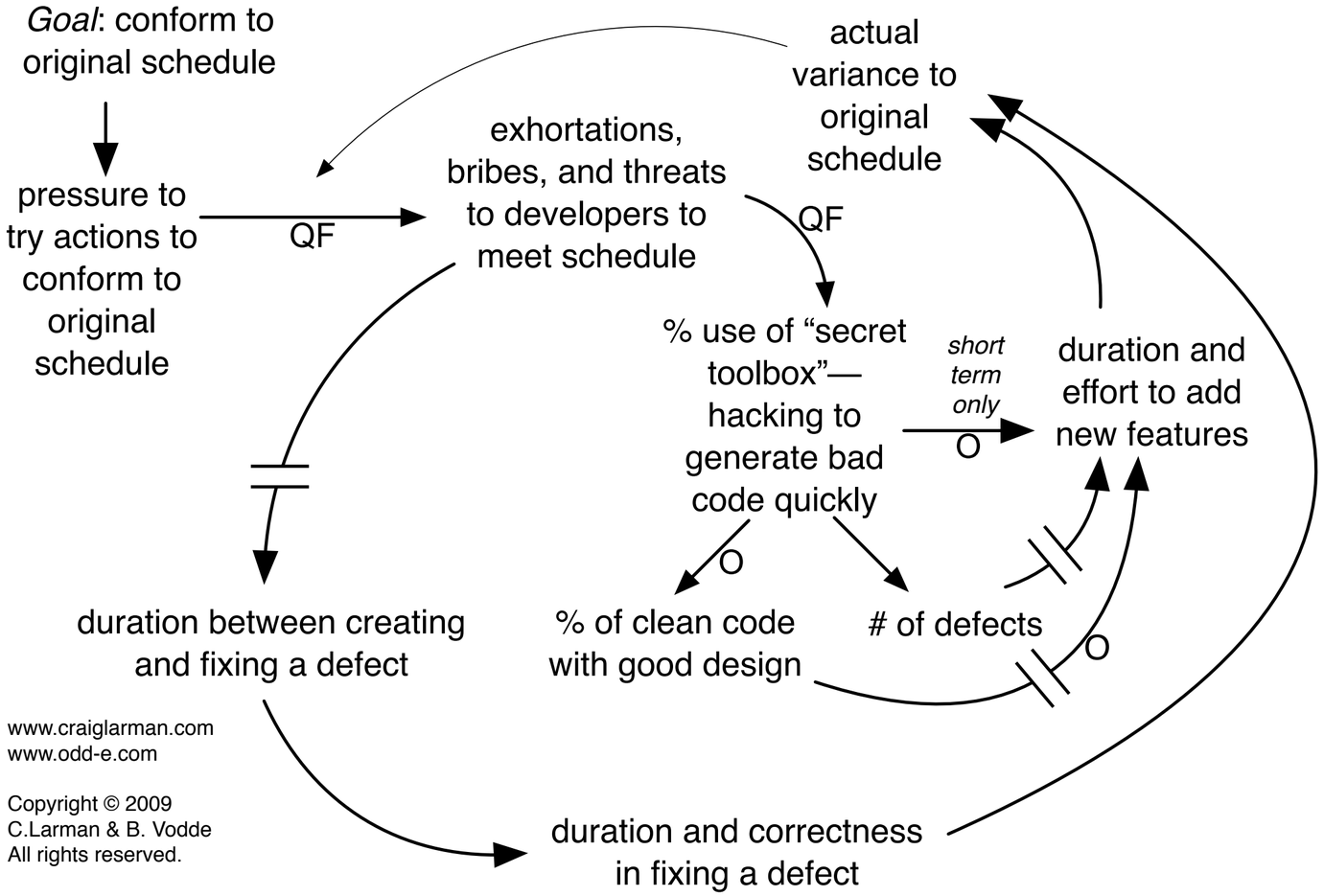
We have no time to clean it

We need to go “faster”

We are afraid of breaking it

Fear prevents us to clean it

The “Faster is Slower” Dynamic



Clean code

Easy to understand

Easy to evolve

Easy to maintain

Sustains delivery pace

Knowledge Gap Causes Fear

Fear makes you tentative

Fear makes you want to communicate less

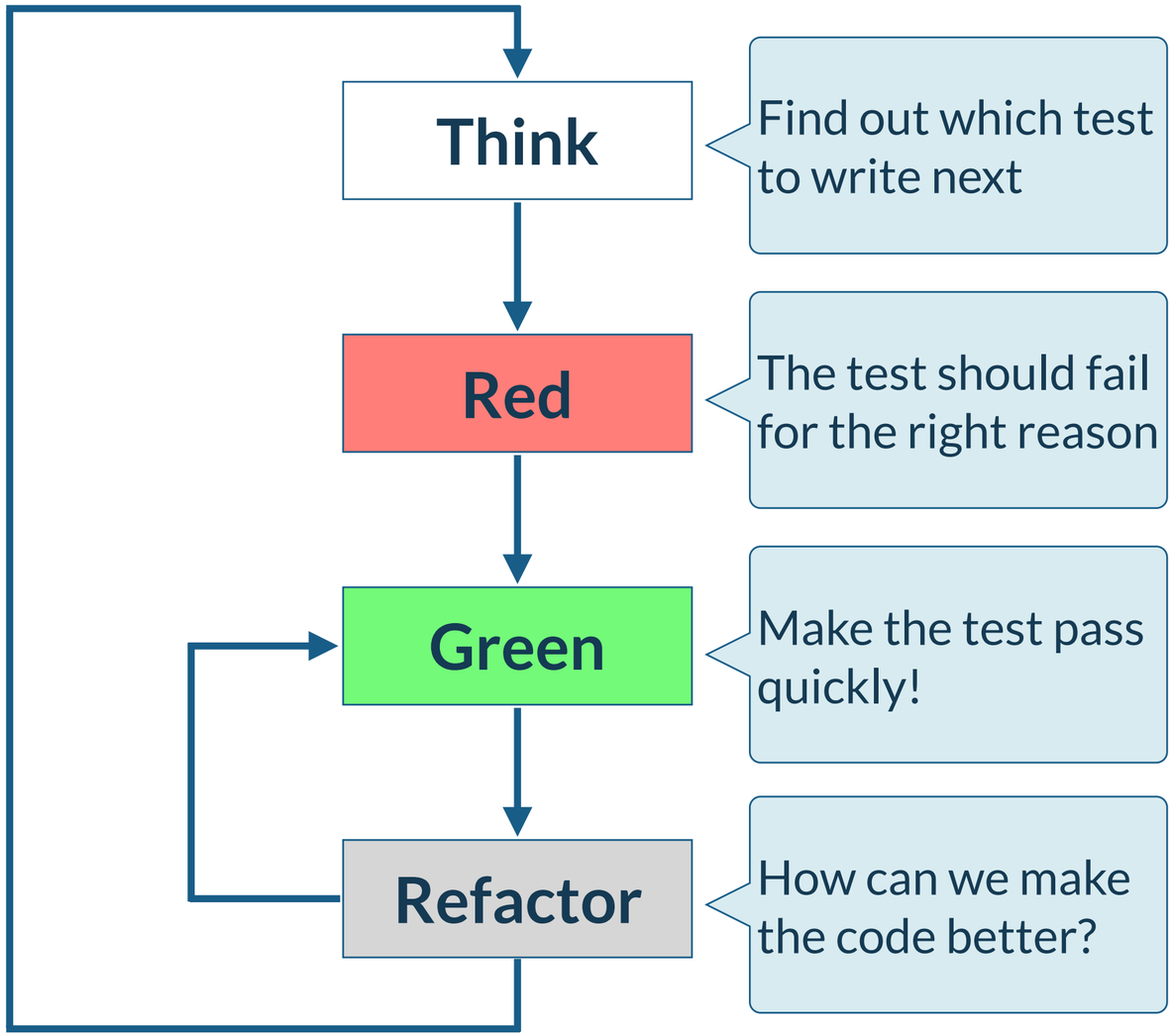
Fear makes you shy away from feedback

We want to begin learning concretely as quickly as possible

We want to communicate more clearly

We want to search out helpful, concrete feedback

Test Driven Development Cycle



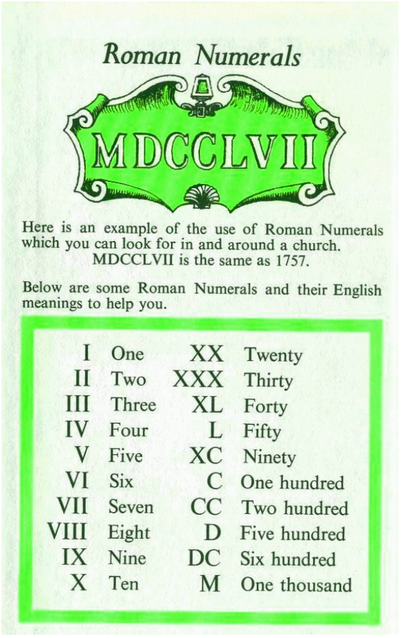
Test Infrastructure

Things we need to practice TDD	Why?
Automated build	To run the tests, as fast as possible
Test framework	To build the test suite
Assertion library	To check test status
Arrange/Act/Assert	A way to write tests

Let’s try it!

We want to write a program that convert integer numbers (in decimal notation) into their Roman numeral equivalents.

1. What test should we write?
2. Write the test and see it fail.
3. Make the test pass, quickly!
4. Eliminate duplication and improve expressiveness.
5. Back to 1.



Roman Numerals

MDCCLVII

Here is an example of the use of Roman Numerals which you can look for in and around a church. MDCCLVII is the same as 1757.

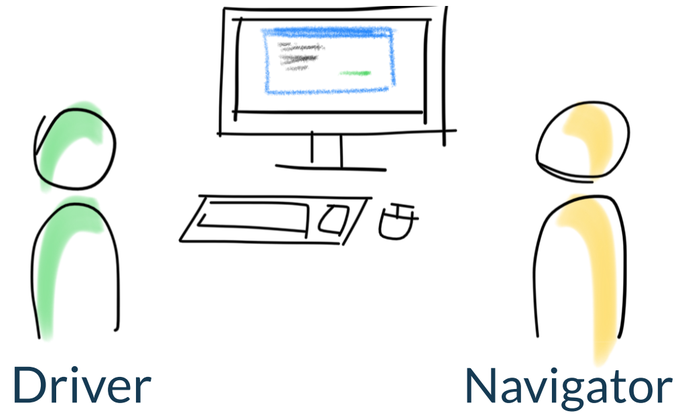
Below are some Roman Numerals and their English meanings to help you.

I	One	XX	Twenty
II	Two	XXX	Thirty
III	Three	XL	Forty
IV	Four	L	Fifty
V	Five	XC	Ninety
VI	Six	C	One hundred
VII	Seven	CC	Two hundred
VIII	Eight	D	Five hundred
IX	Nine	DC	Six hundred
X	Ten	M	One thousand

Let’s try Pair Programming too

Two people working together, at the same computer, solving the same problem.

- XP practice for software development
- Two roles: driver and navigator
- The pair swaps fluidly between the roles
- Improve code quality, spread knowledge, training on the job, …



Should you **always** practice
Test Driven Development?

Tests in TDD

Should be…

Isolated and composable

Fast and automated

Behavioral and structure-insensitive

Specific and deterministic

Inspiring and predictive

Beware of

Databases

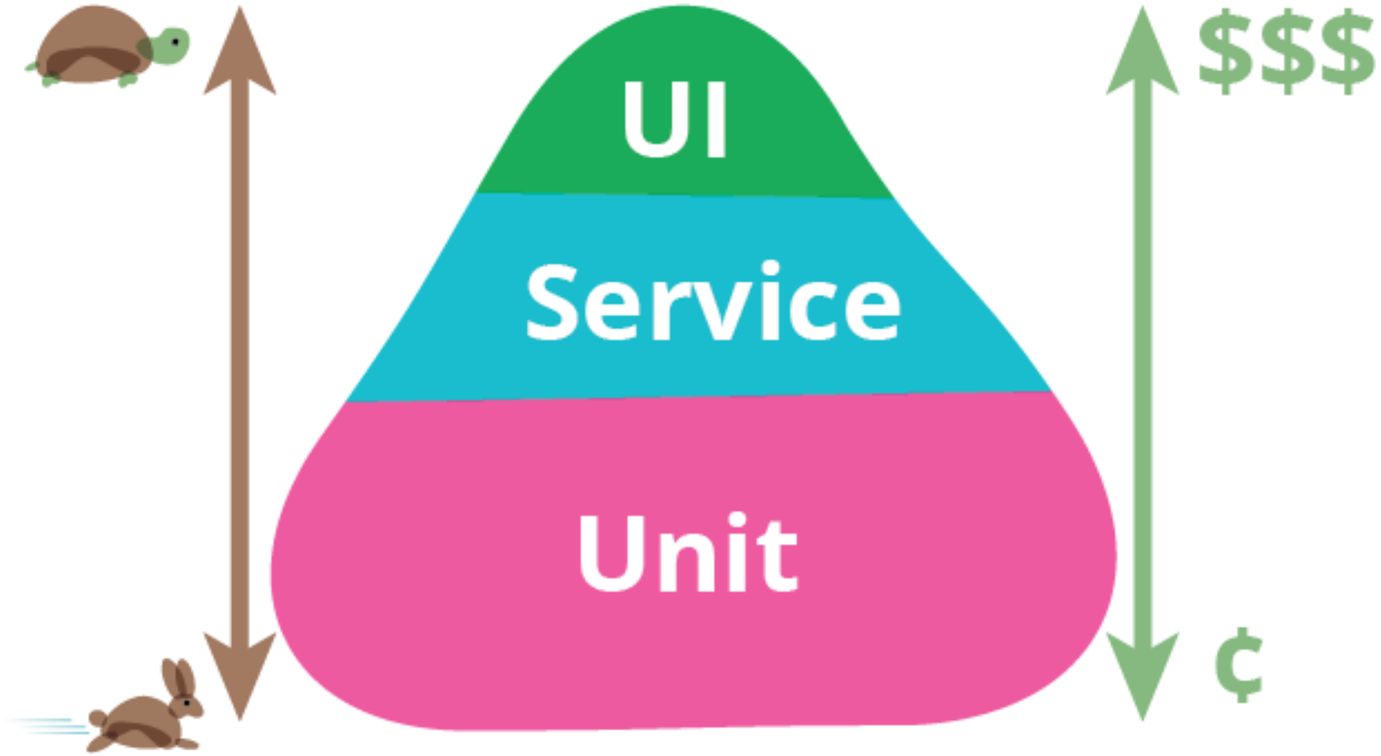
Network communications

File system

Other shared fixtures

Configurations

Test Pyramid



https://martinfowler.com/bliki/TestPyramid.html

References

Test Driven Development by Example
Kent Beck

Extreme Programming Explained
Kent Beck

Clean Code
Robert C. Martin