

Agile Software Development

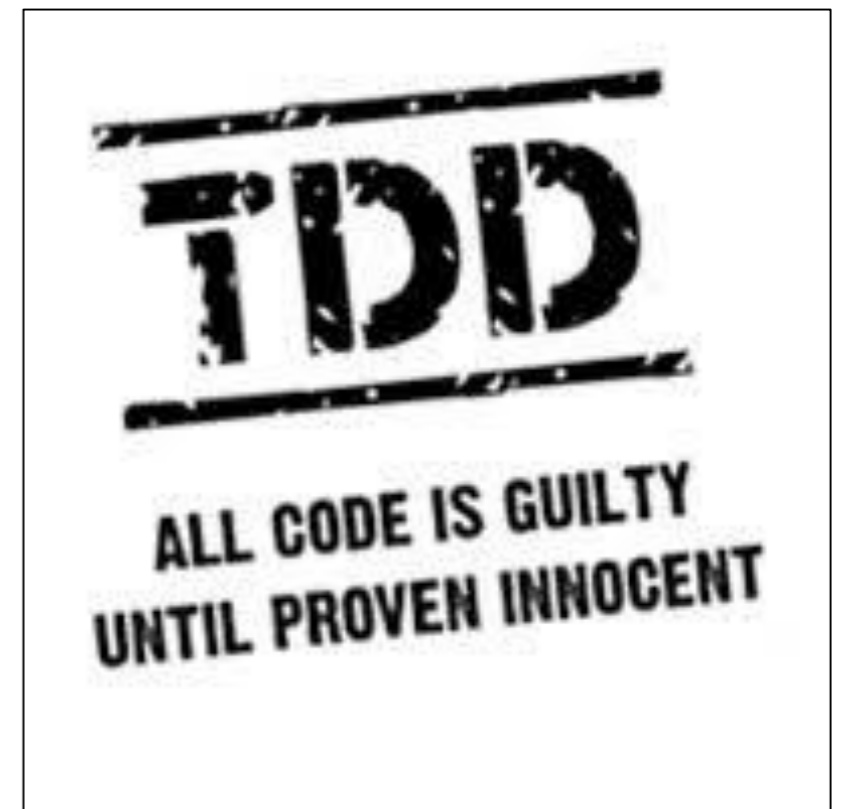
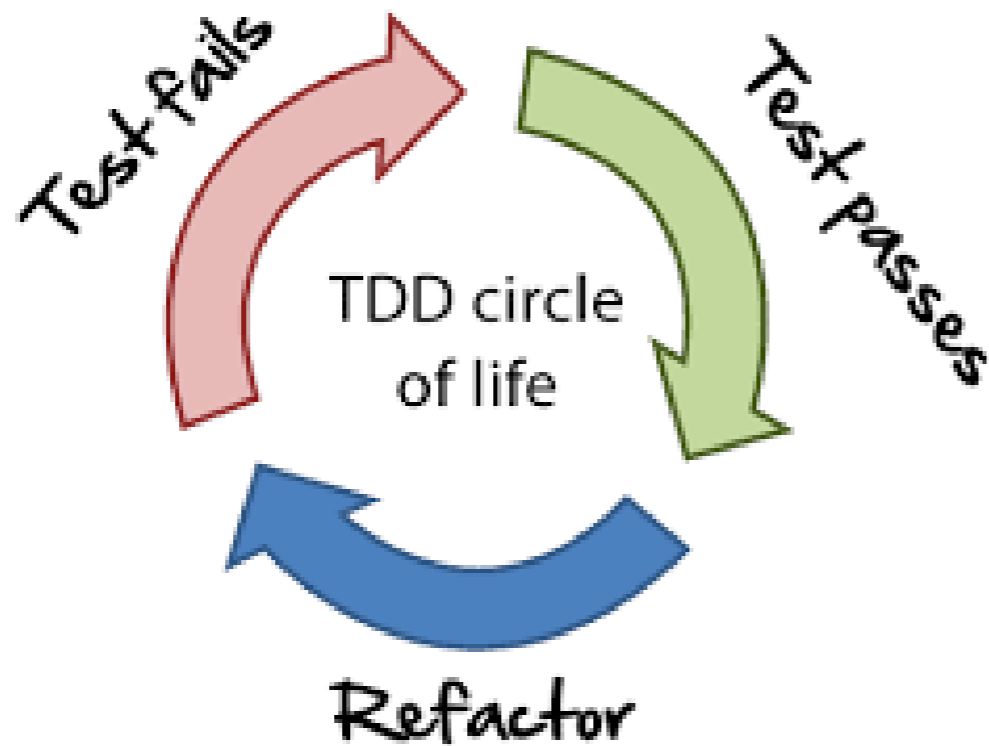
Produced
by

Dr. Siobhan Drohan (sdrohan@wit.ie)

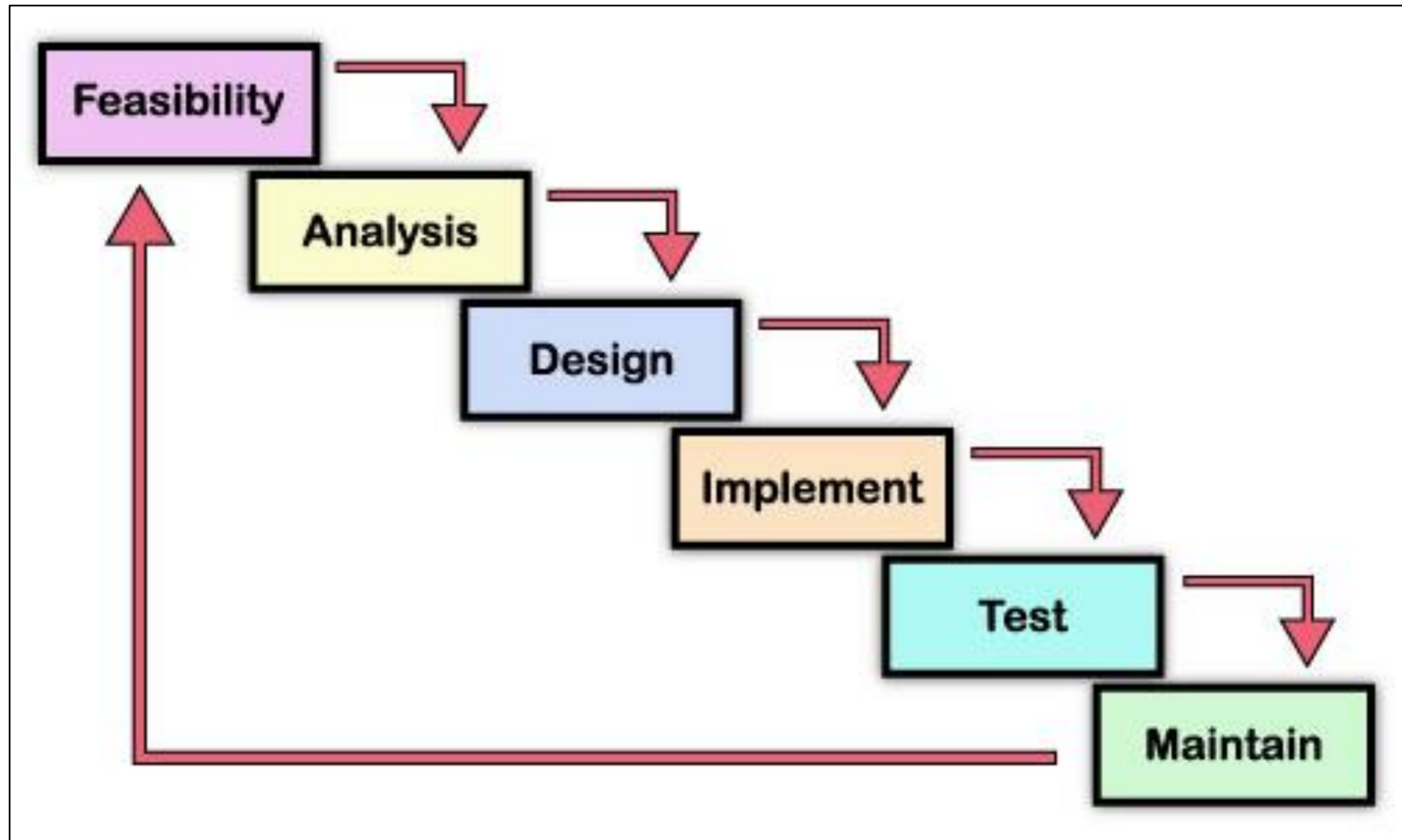
Eamonn de Leastar (edeleastar@wit.ie)



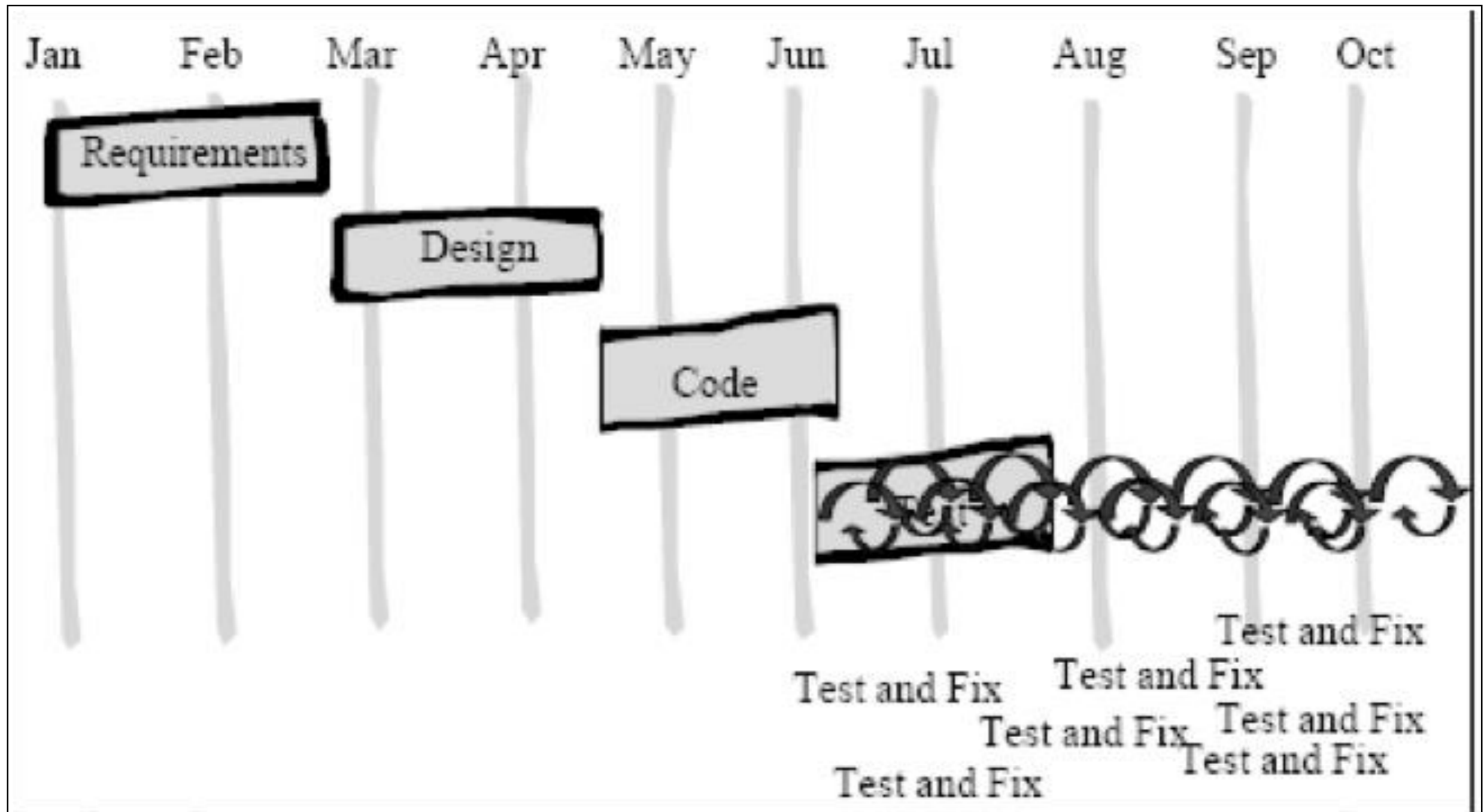
Agile and Test Driven Development (TDD)



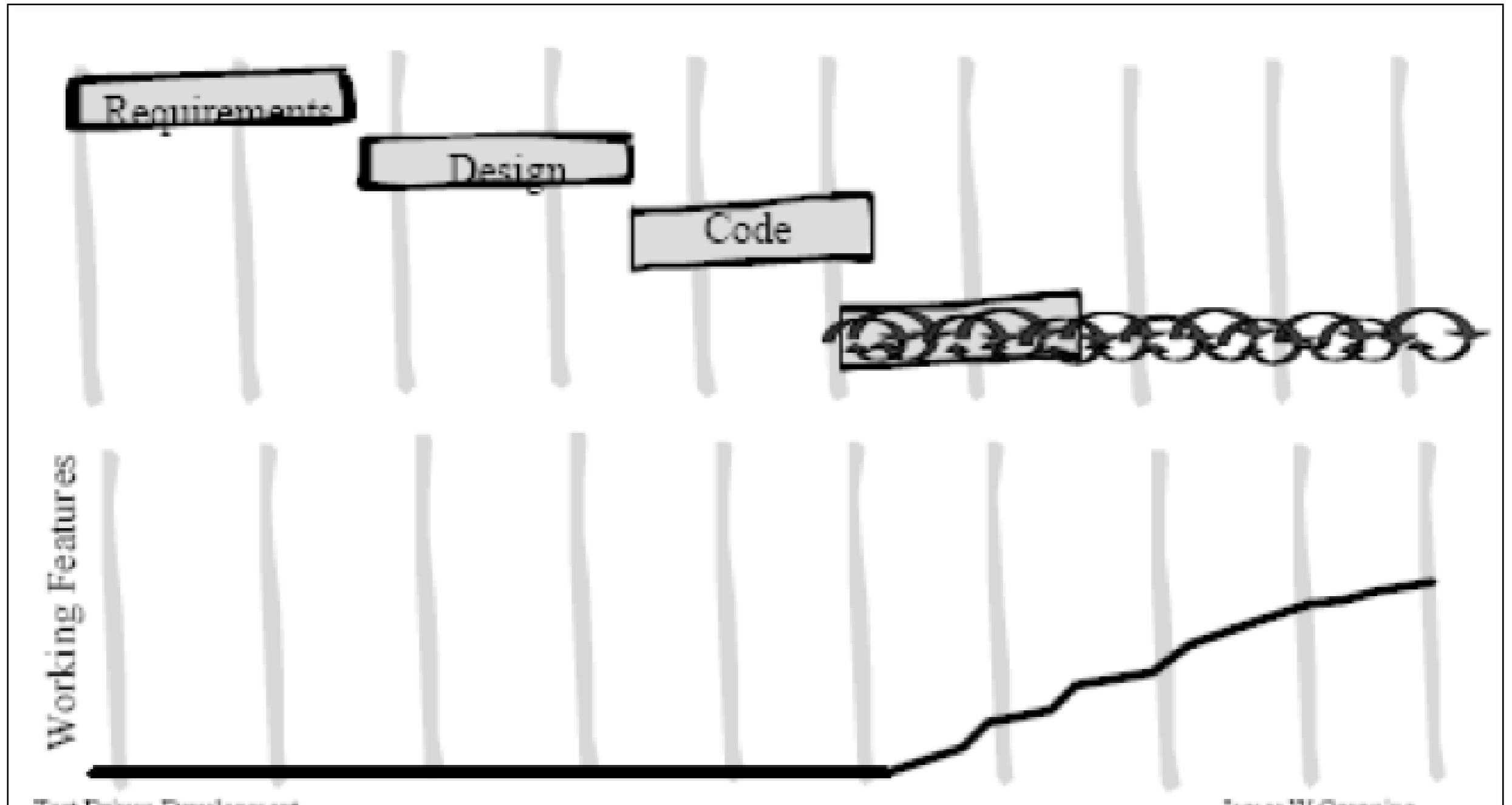
Waterfall - development approach



Waterfall - development approach



Waterfall - Working Features



THE NEW PRODUCT WATERFALL



HOW DO WE CHART OUR ENTIRE COURSE IF WE DON'T KNOW WHAT'S AHEAD?

PLAN



WHATEVER HAPPENS, JUST KEEP PADDLING!

BUILD

I WISH WE'D DESIGNED FOR THIS SCENARIO UPFRONT



TEST

PATCH IT AS BEST WE CAN. NO TIME TO CHANGE COURSE NOW



LAUNCH

Waterfall

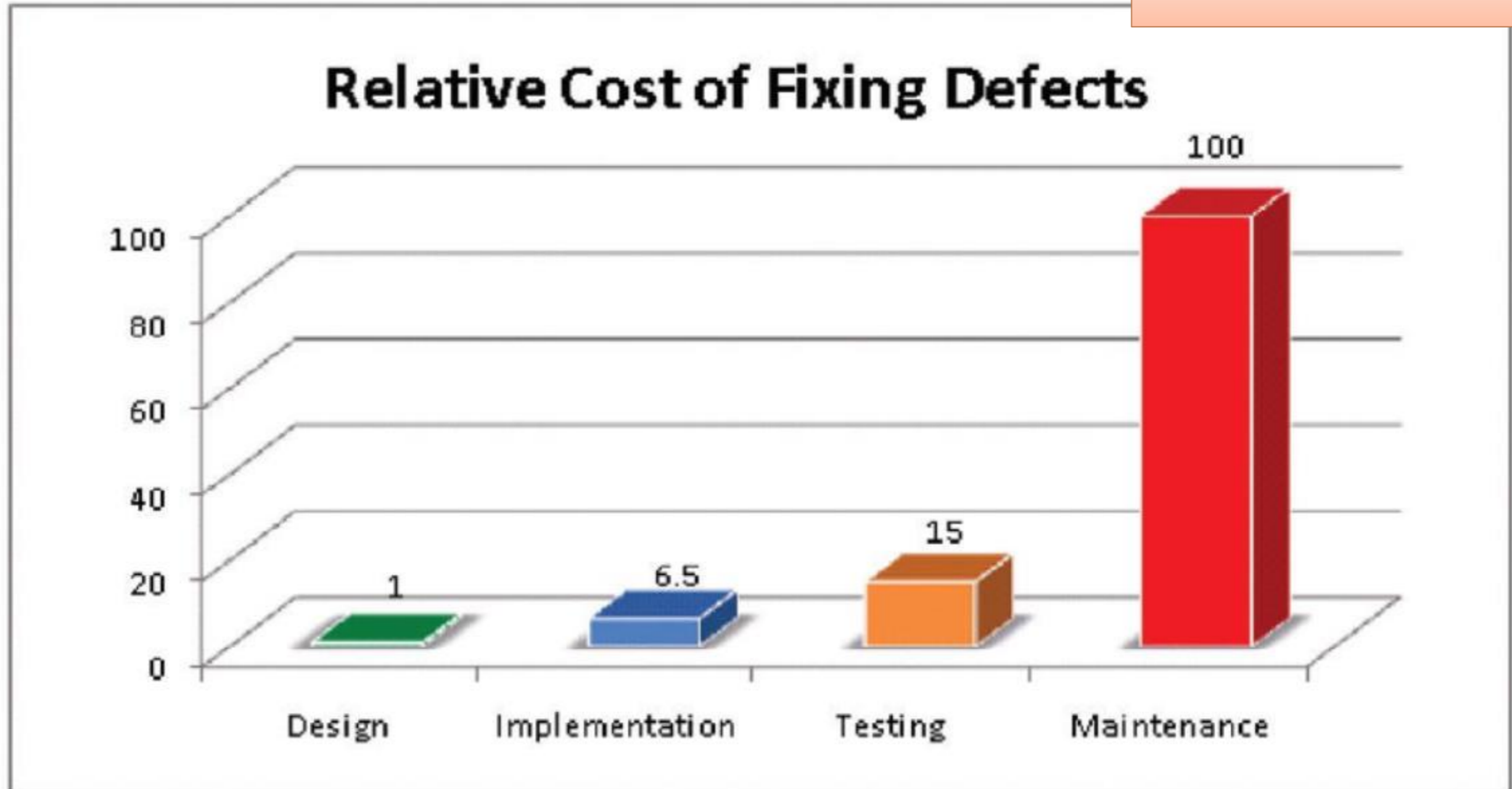
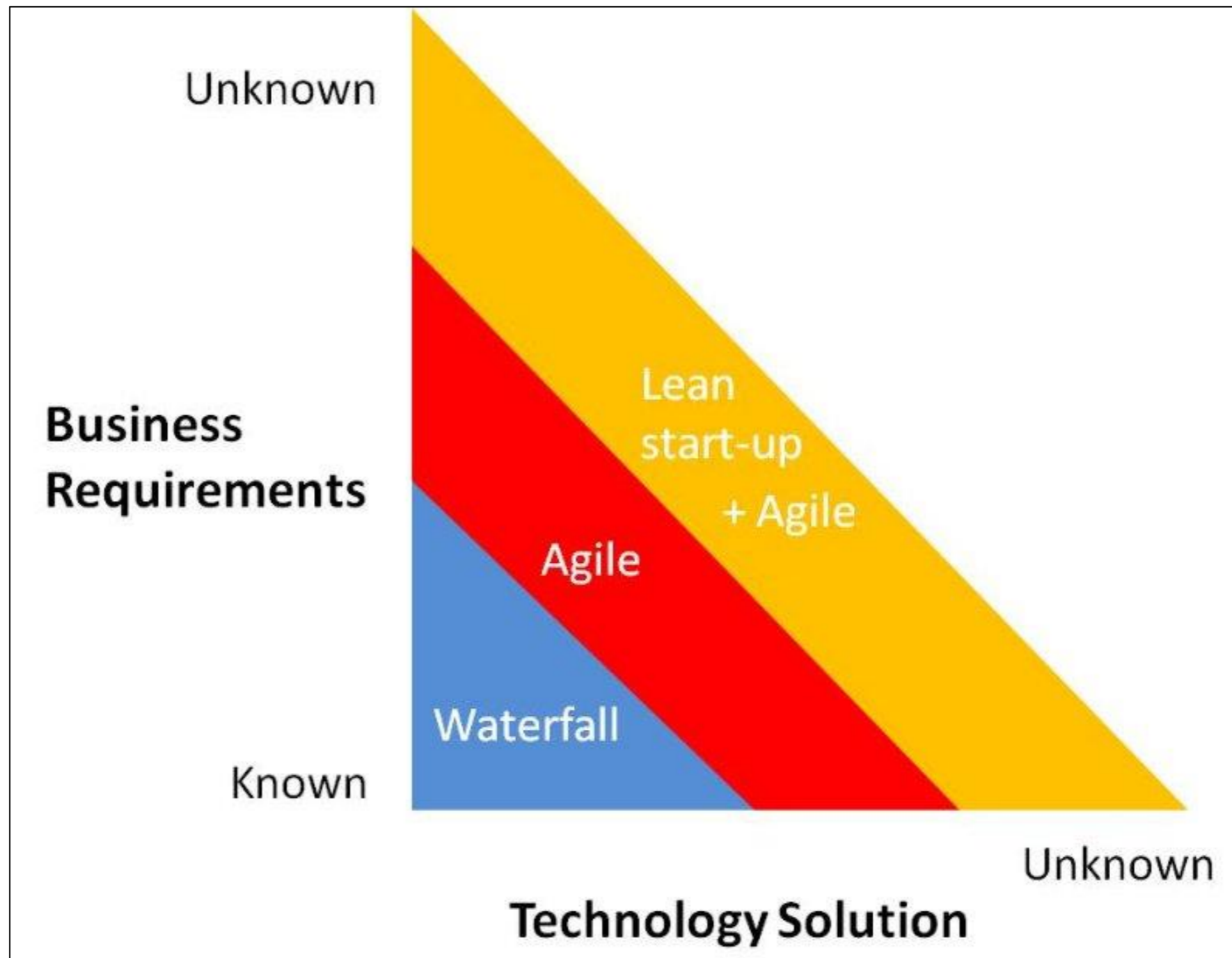


Figure 3: IBM System Science Institute Relative Cost of Fixing Defects

Defects found in testing were 15 times more costly than if they were found during the design phase and 2 times more than if found during implementation.

Waterfall Vs Agile



Waterfall Vs Agile

THE WATERFALL PROCESS



*'This project has got so big,
I'm not sure I'll be able to deliver it!'*

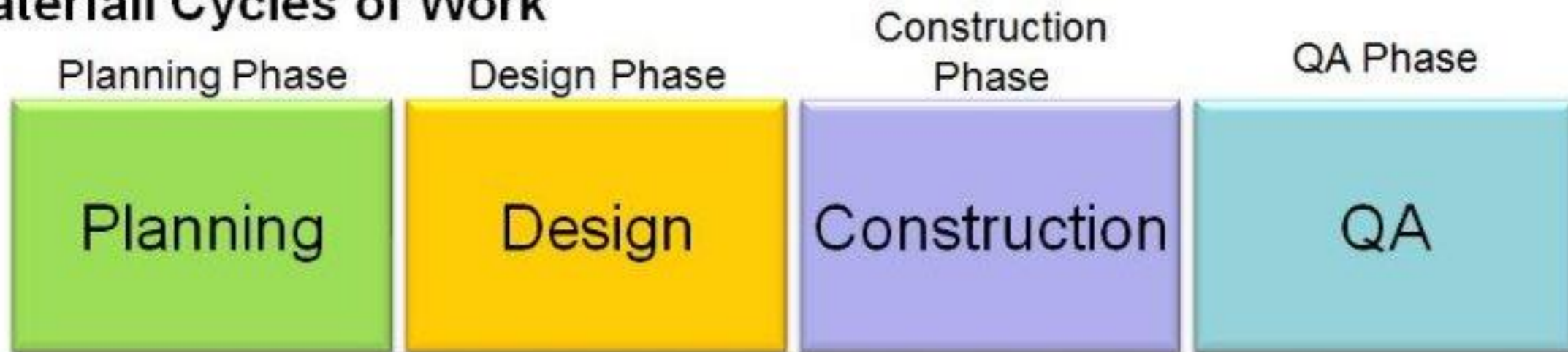
THE AGILE PROCESS



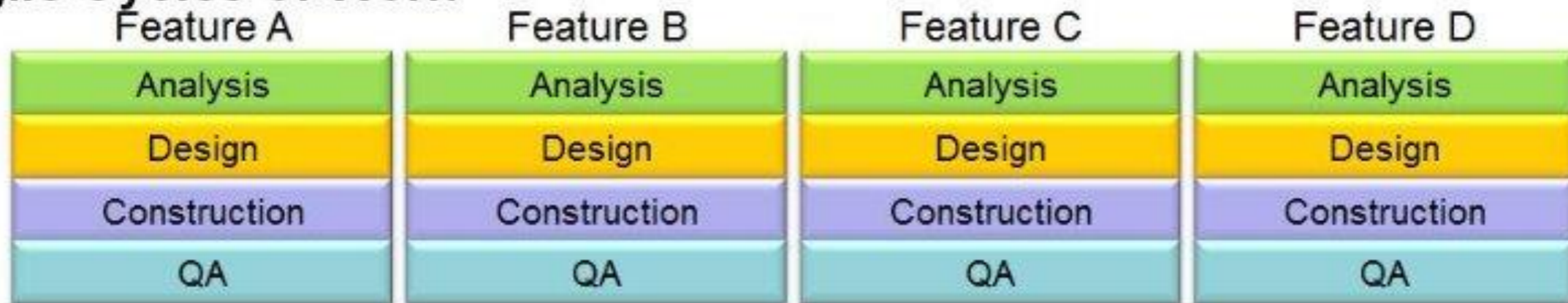
*'It's so much better delivering this
project in bite-sized sections'*

Waterfall Vs Agile

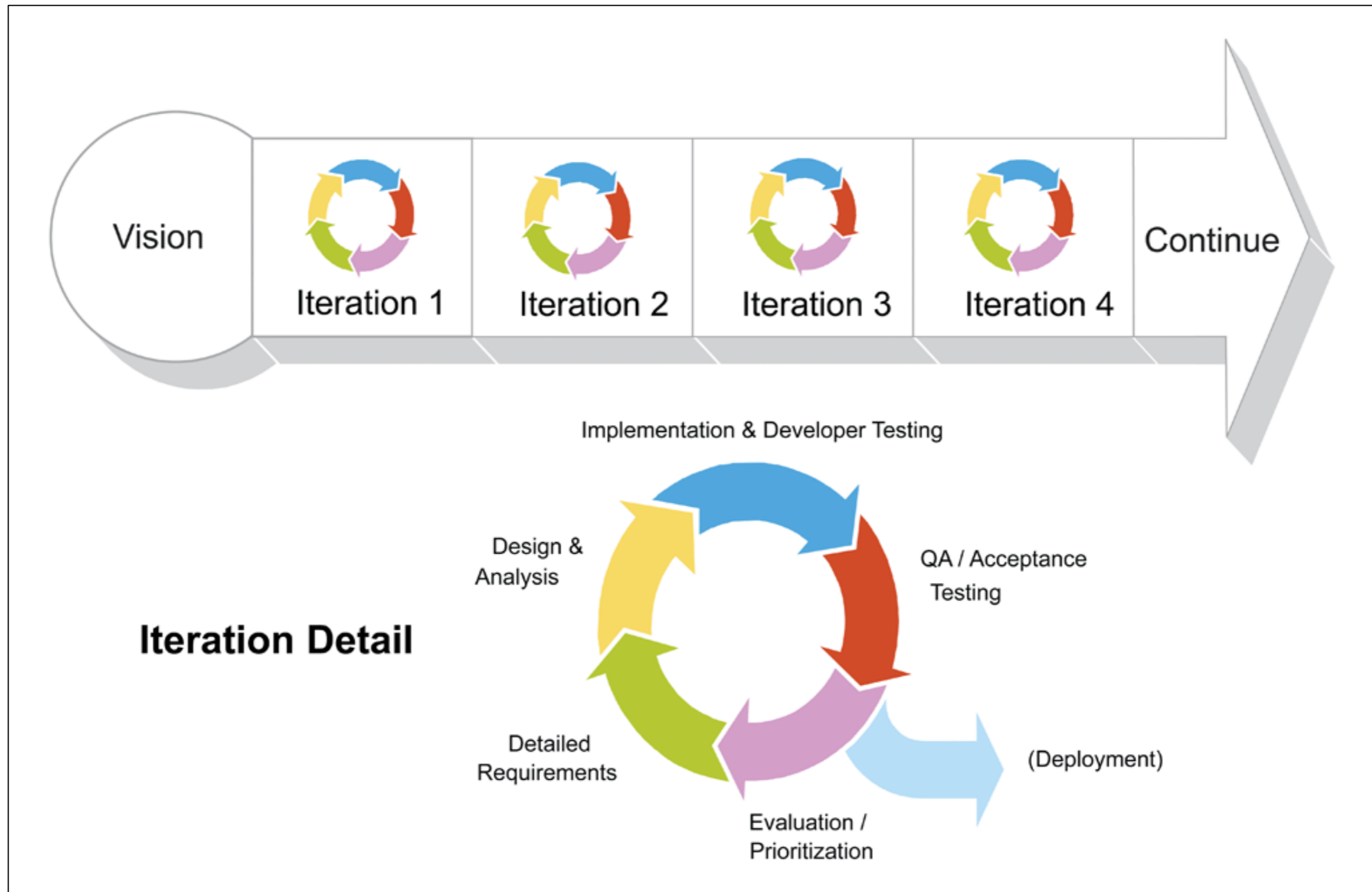
Waterfall Cycles of Work



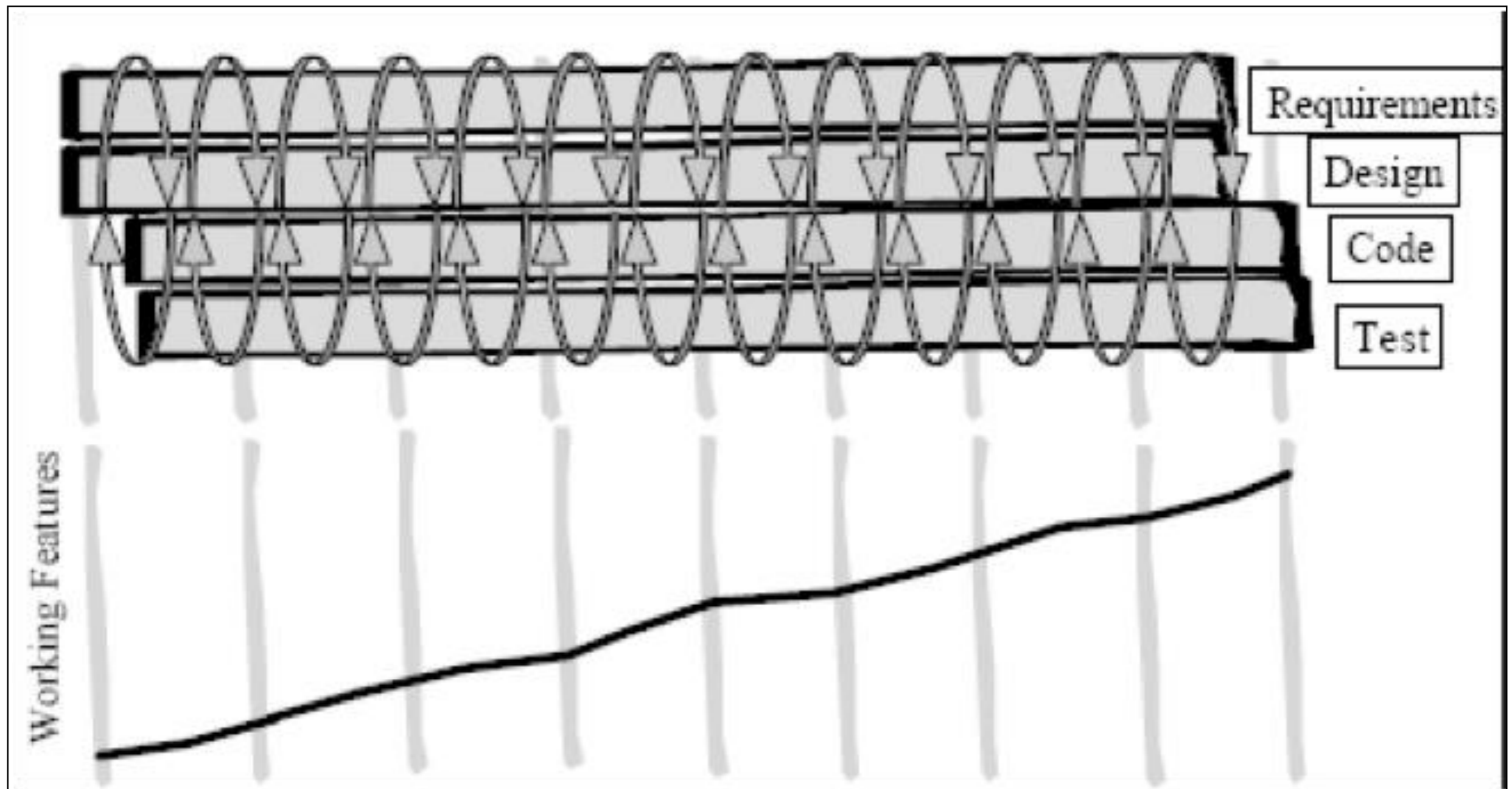
Agile Cycles of Work



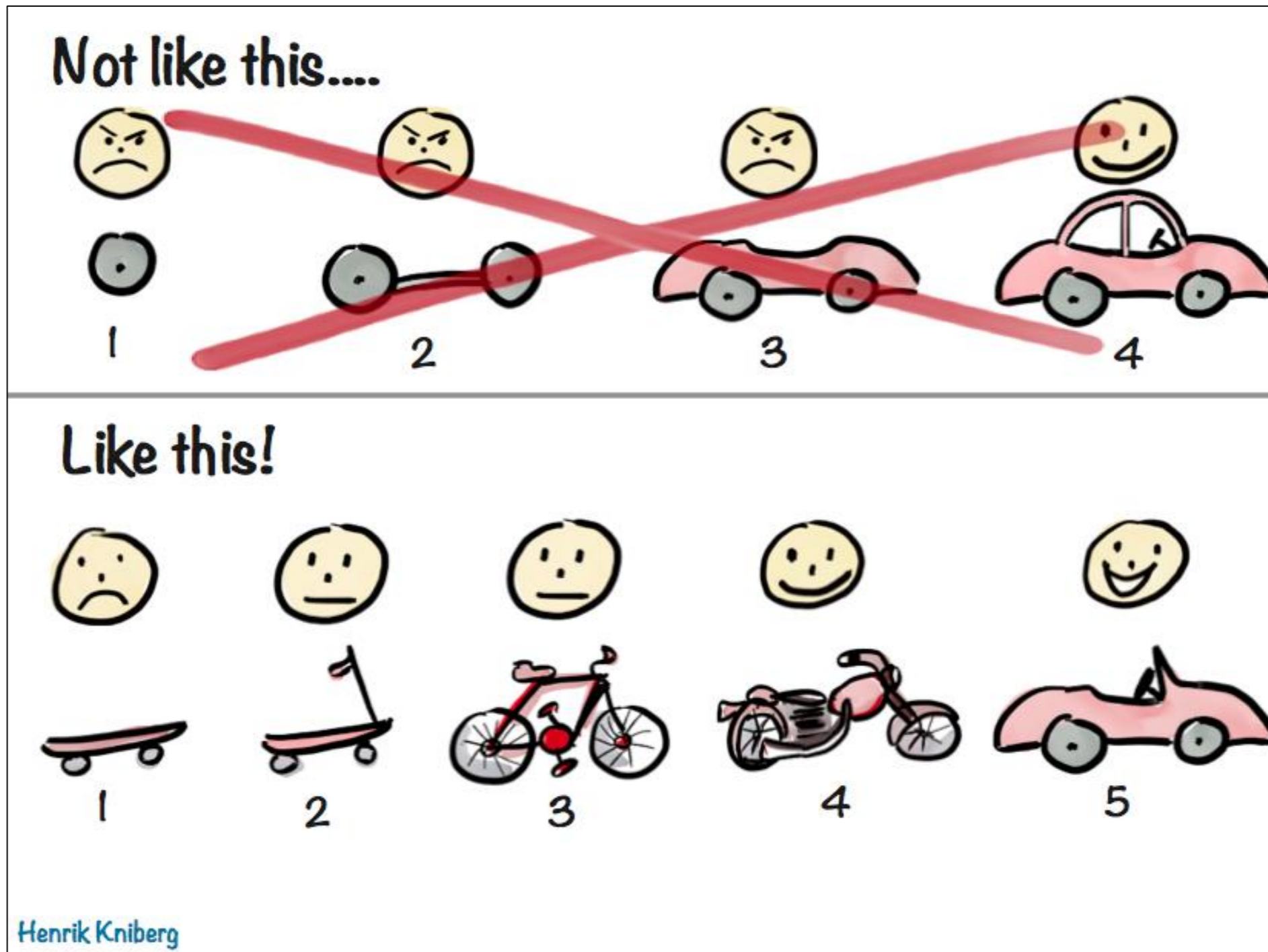
Agile – Iterative Approach



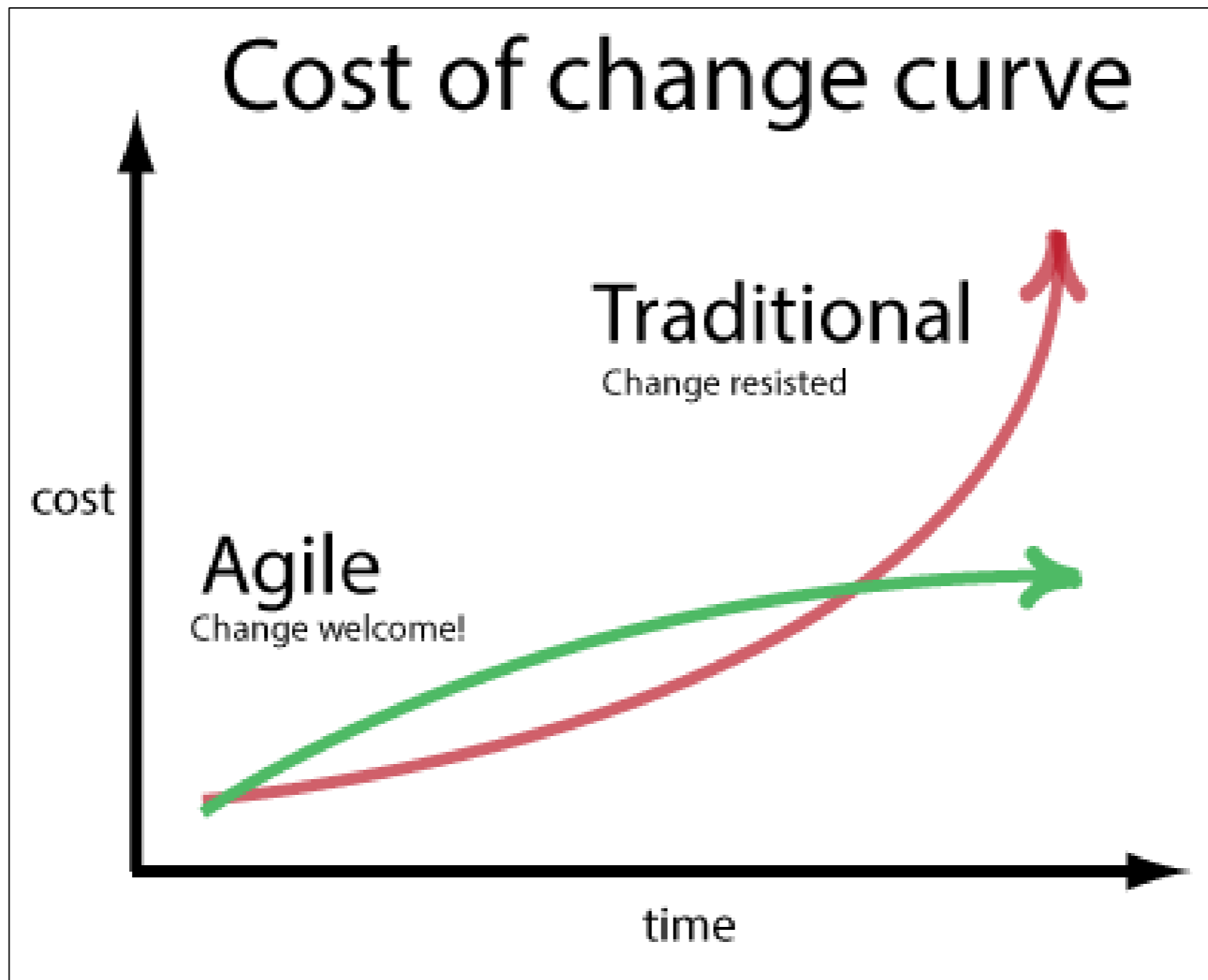
Iterative Approach - Working Features



Agile – Both Iterative and Incremental



Waterfall vs Agile – Cost of Change



Developer landscape has changed just a *little* (!) ...

- New tools have dramatically eased mundane developer tasks:
 - **Automated test tools (e.g. JUnit)**
 - System build tools (e.g. Maven, Gradle, SBT)
 - Version control (e.g. Git repositories, Github hosting service)
 - Continuous integration
- Used properly, OO languages can make software much easier to change.
- The cost curve is significantly flattened, i.e. costs don't increase dramatically with time.
- Up front modeling becomes a liability – some speculative work will certainly be wrong, especially in a business environment.

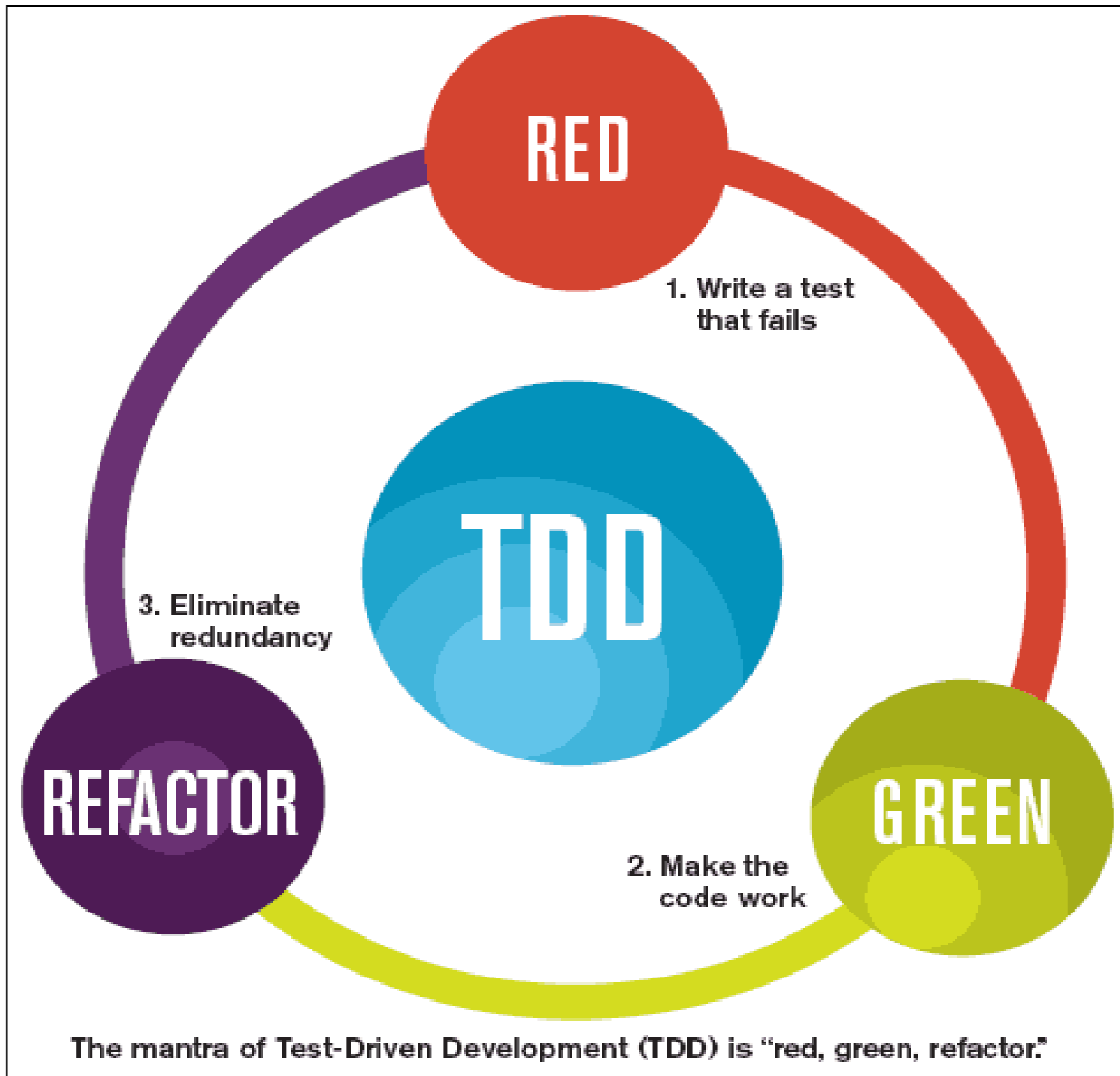
TDD

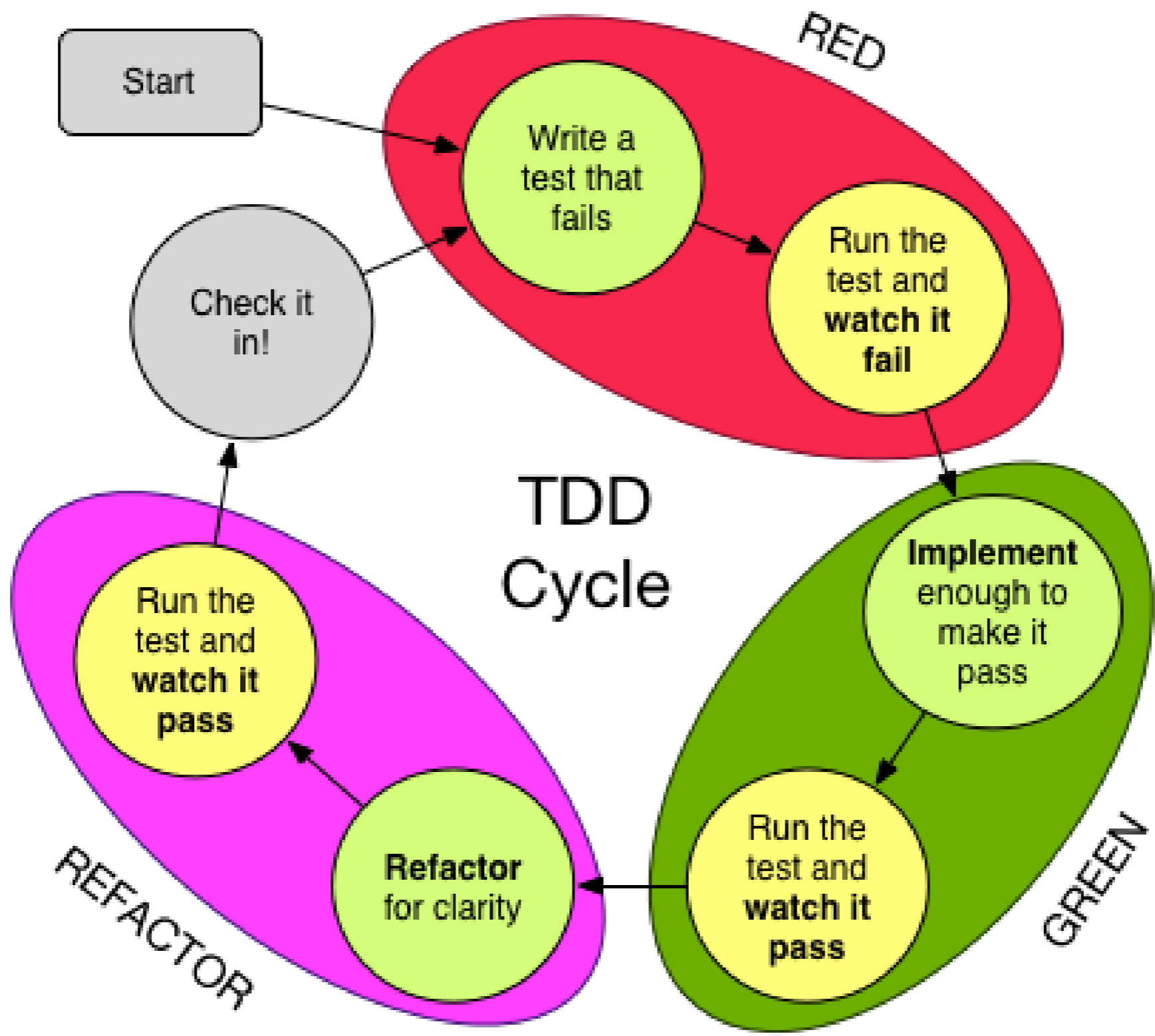
**ALL CODE IS GUILTY
UNTIL PROVEN INNOCENT**

***“Good programmers write code,
great programmers write tests”***

***“Never,
in the field of programming,
have so many
owed so much
to so few”***

- Martin Fowler on the developers behind JUnit

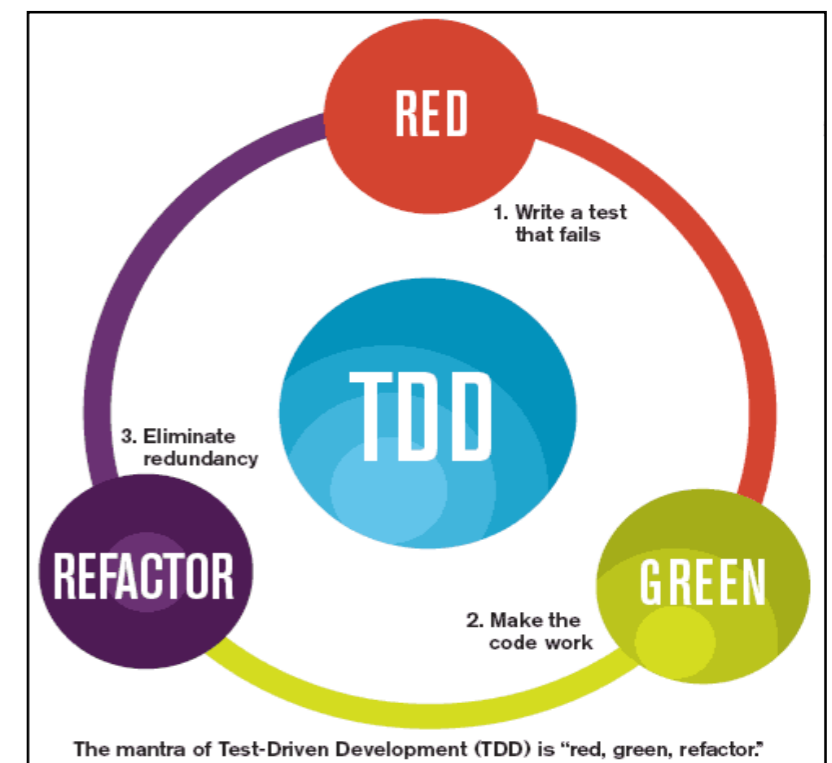




TDD – Definition

Test-driven development (TDD) refers to a style of programming in which three activities are tightly interwoven:

- coding,
- testing (in the form of writing unit tests) and
- design (in the form of refactoring).



What is Unit Testing?

- A unit test is a piece of code written by a developer that exercises a very small, specific area of functionality of the code being tested.
 - Usually a unit test exercises some particular method in a particular context
- Unit tests are performed to prove that a piece of code does what the developer thinks it should do.
- The question remains open as to whether that's the right thing to do according to the customer or end-user:
 - that is acceptance testing ([Acceptance Test Driven Development](#), [Behaviour Driven Development](#))

What is Regression Testing?

- New code and changes to old code can affect the rest of the code base.
 - ‘Affect’ sometimes means ‘break’.
- We need to rerun tests on the old code, to verify it still works – this is regression testing.
- Regression testing is required for a stable, maintainable code base.
- Unit tests retain their value over time and allows others to prove the software still works (as tested).

What does Unit Testing Accomplish ?

- *Does the **code** do what was expected?*
 - i.e. is the code fulfilling the intent of the developer?
- *Does the **code** do what was expected all the time?*
 - exceptions get thrown, disks get full, network lines drop, buffers overflow - is the code still performing as expected?
- *Can the **code** be depended upon?*
 - Need to know for certain both its strengths and its limitations.
- *Does the **test** document the developers intent?*
 - An important side-effect of unit testing is that it helps communicate the code's intended use.

TDD – General

- An iterative technique to develop software.
- Tests are written before the code itself.
- As much (or more) about design as testing.
 - Encourages design from user's point of view.
 - Encourages testing classes/units in isolation – Unit testing.
- A test framework is used so that automated testing can be done after every small change to the code.
 - This may be as often as every 5 or 10 minutes.
- Axiom:
 - 'Code that isn't tested doesn't work'
 - 'Code that isn't regression tested suffers from code rot (breaks eventually)'

TDD – General (Contd.)

- As much (or more) about documentation as testing.
 - The tests are the documentation of what the code does.
- Must be learned and practiced.
- Consequences:
 - Fewer bugs;
 - More maintainable code - loosely-coupled, highly-cohesive systems.
 - During development, the program always works—it may not do everything required, but what it does, it does right.
 - Breaks the cycle of **more pressure == fewer tests** (the fewer tests you write, the less productive you are and the less stable your code becomes).

How is Unit Testing carried out?

- **Step 1:** Decide how to test the method in question before writing the code itself
- **Step 2:** Write the test code itself, either before or concurrently with the implementation code.
- **Step 3:** Run the test itself, and probably all the other tests in that part of the system.
- *Key Feature of executing unit tests:*
 - *You need to be able to determine at a glance whether all tests are succeeding/failing. The JUnit Framework will do this for us!*

Why bother with TDD?

TDD – Why bother with TDD/Unit Testing

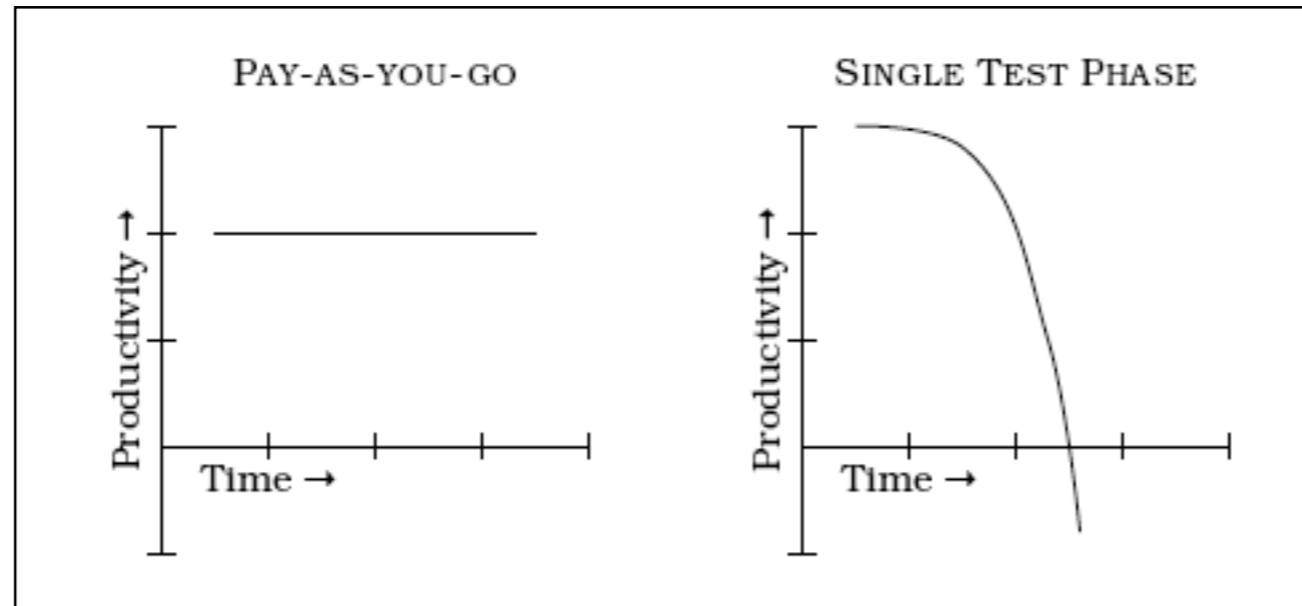
- Significant **reductions in defect rates**, at the cost of a moderate increase in initial development effort:

generally these overheads are more than offset by a reduction in effort in projects' final phases.

- Anecdotal evidence suggests that TDD leads to **improved design qualities in the code**, and more generally a higher degree of technical quality.

Excuses for not engaging in TDD

Excuse #1



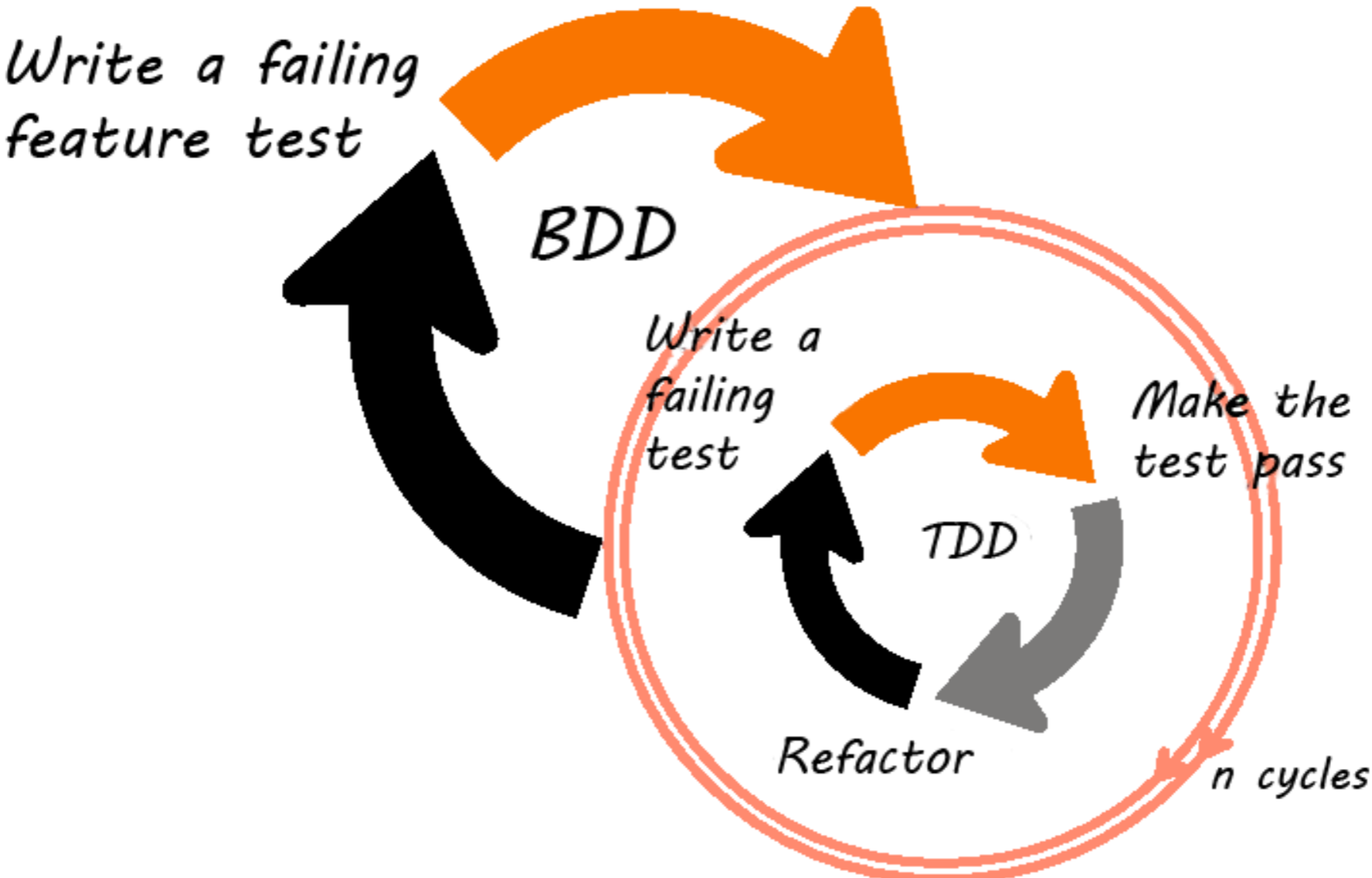
“It takes too much time to write the tests”

- The trade-off is not “test now” versus “test later”
- It's linear work now versus exponential work and complexity trying to fix and rework at the end.

Excuse #2



Excuse #2 (contd.)



Excuse #3



“It takes too long to run the tests”

Excuse #3



“It takes too long to run the tests”

- Separate out the longer-running tests from the short ones.
- Only run the long tests once a day, or once every few days as appropriate, and run the shorter tests constantly.
- **Your code isn't finished until you have verified it works!**

Excuse #4

“It's not developers job to test his/her code”

Excuse #4

“It's not developers job to test his/her code”

- Integral part of developer job is to create working code.



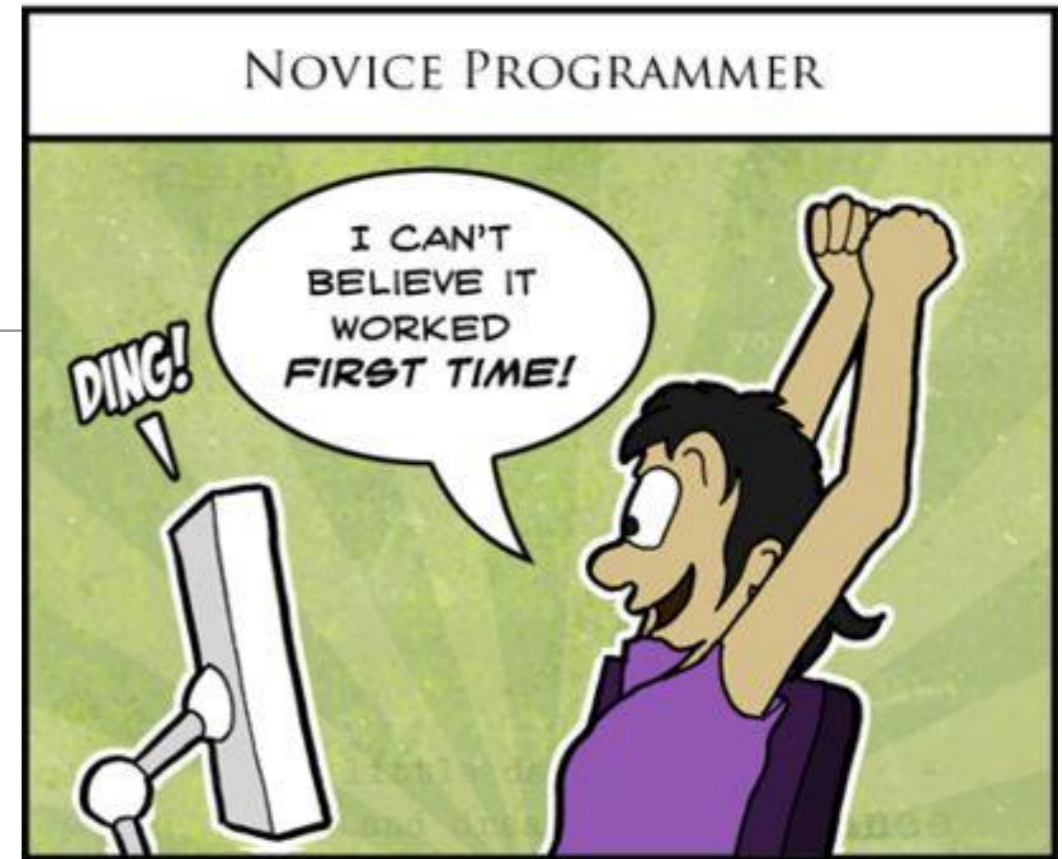
Excuse #5

“But it compiles!”

Excuse #5

“But it compiles!”

- A compiler's blessing is a pretty shallow compliment.



Excuse #6

“We refactor our code so frequently, that the time we invest in tests just isn't worth it - they are going to change and be irrelevant anyhow”

Excuse #6

“We refactor our code so frequently, that the time we invest in tests just isn't worth it - they are going to change and be irrelevant anyhow”

- How can you be certain you didn't break anything when refactoring your code?
- Regression testing is one of the number one reasons for doing TDD...good regression tests will, almost immediately, show up un-intended side effects of your code change.
 - A good rule is...NEVER refactor without tests!

Excuse #7

“We are such talented programmers, we don’t need tests”



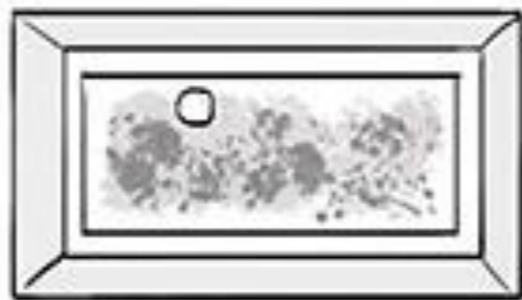
Excuse #7

“We are such talented programmers, we don’t need tests”

- *Everyone has bugs in their code...we are human after all!*
- *Ok, even if you are a “bug-free coder”, what about Regression testing in the future by you and other programmers?*



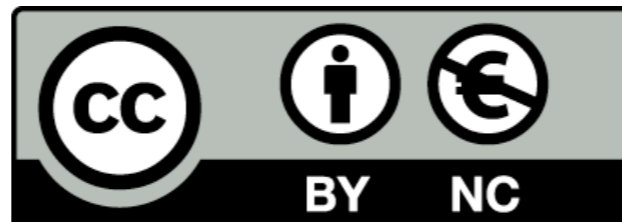
Can we engage in TDD too much?



I understand that you're committed to Test Driven Development, but don't you think that testing the tests that test the tests that test your public methods and properties is a bit much?



ROELBOB 2016



Except where otherwise noted, this content is licensed under a [Creative Commons Attribution-NonCommercial 3.0 License](http://creativecommons.org/licenses/by-nc/3.0/).

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

