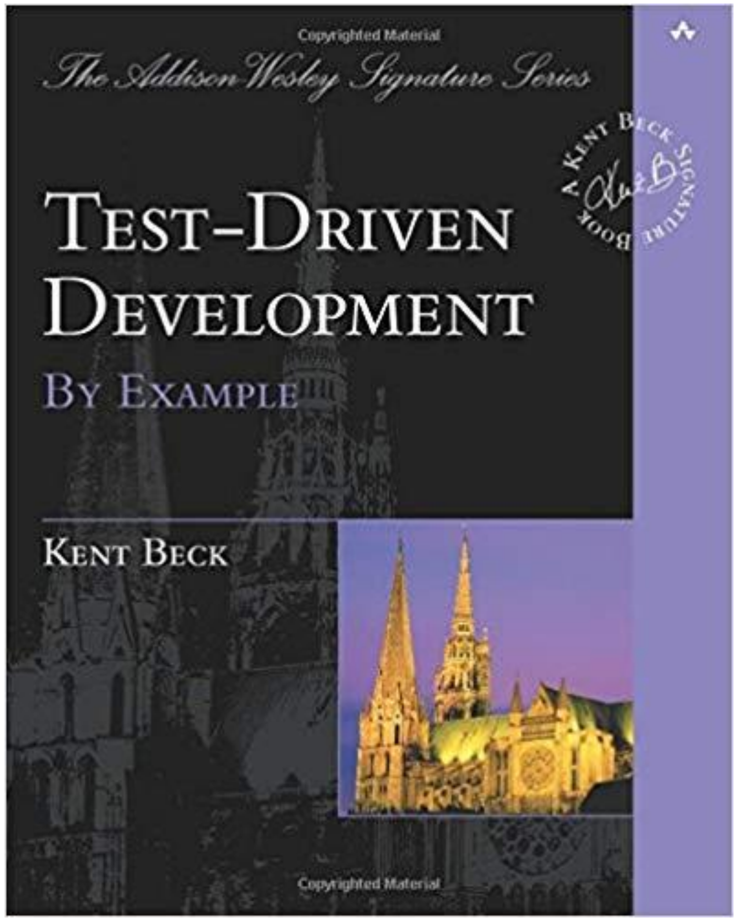
Test driven development

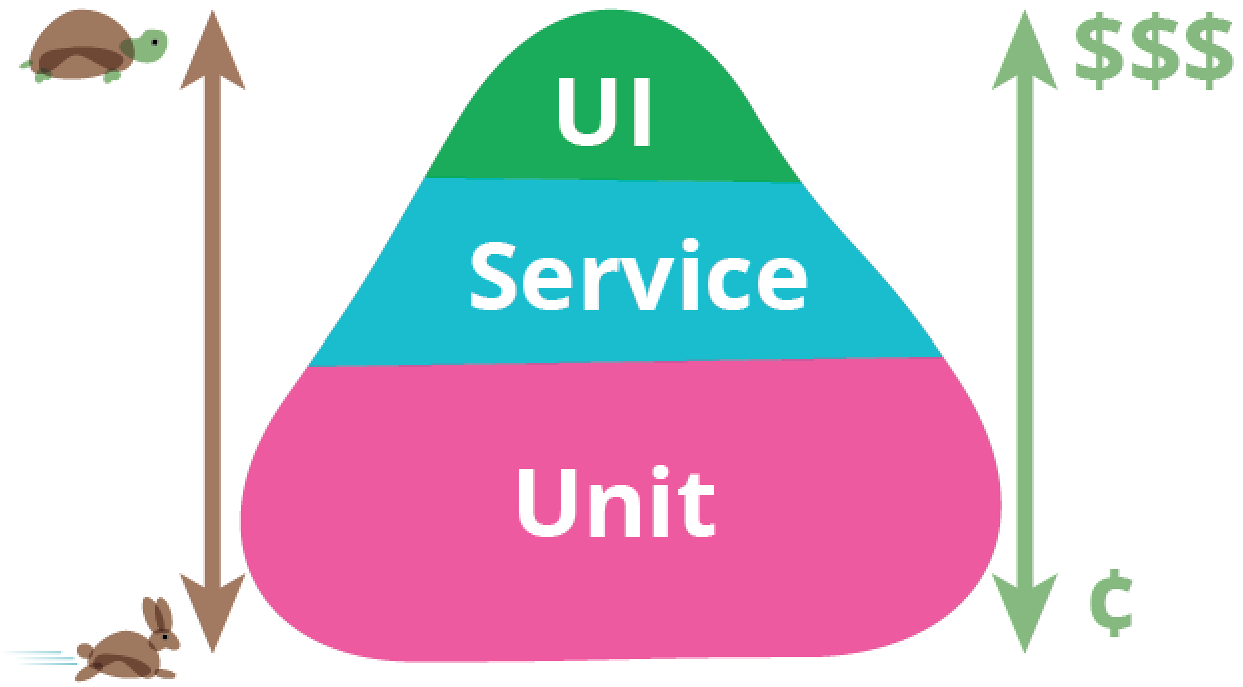
Radim Göth

Outline

- Workshop is inspired by Test-Driven Development book by Kent Beck [1]
- Outline
  - TDD background & big picture
  - What is TDD and how it is practiced
  - Unit testing
  - Get familiar with NUnit
  - Coding katas

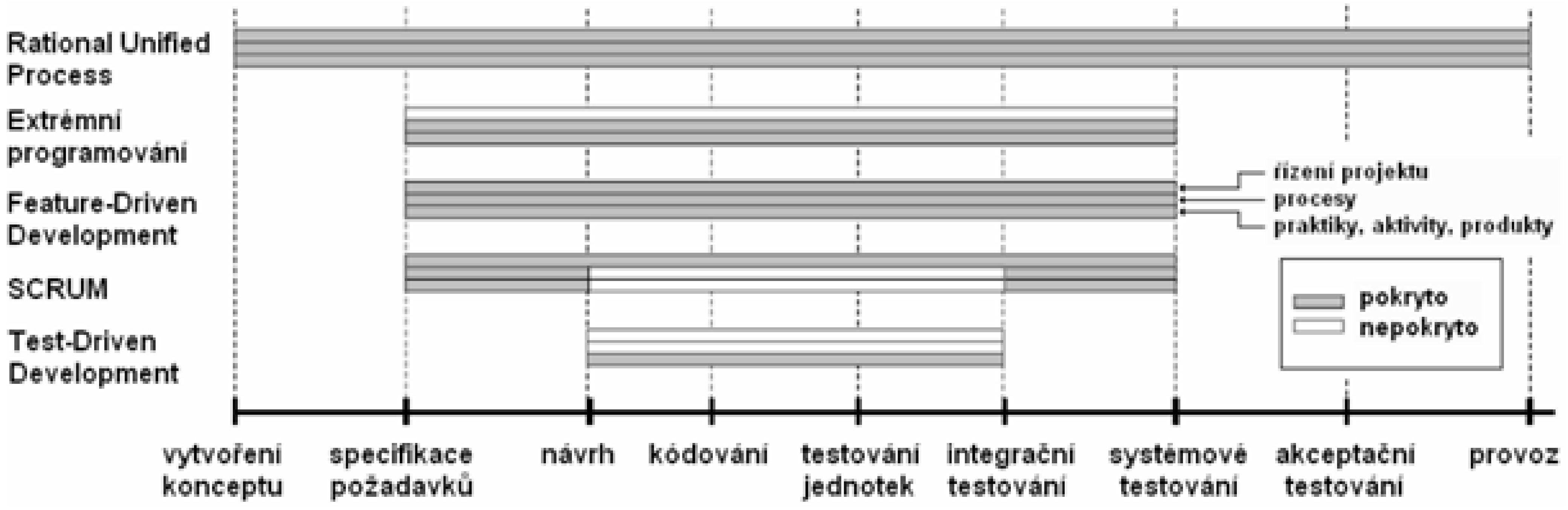


Test pyramid - unit vs integration vs system tests



Taken from [2]

Srovnání metodik z pohledu životního cyklu SW



Taken from [3]

Extreme Programming

- Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. https://en.wikipedia.org/wiki/Extreme_programming

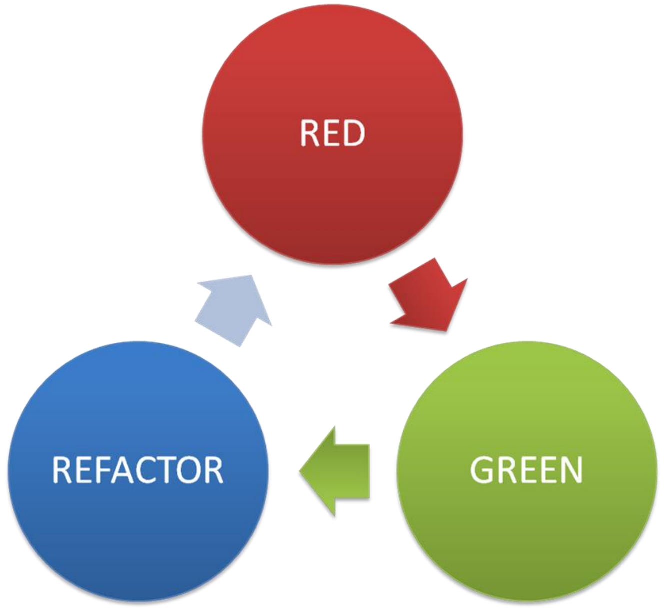
Extreme programming

- Communication
- Simplicity
- Feedback
- Courage
- Respect

- Pair programming
- TDD
- Collective code ownership
- Continuous integration
- Acceptance tests

TDD

- Kent Beck – reinvented TDD, invented XP
- Software development process
- Pair programming
- Rules:
  - Write failing test
  - Write simplest implementation to pass the test
  - Refactor your code
  - Repeat



TDD some thoughts

- Write test first should make the application design better. If you need to vandalize design of you application just to make it testable, you’re probably doing it wrong.
- Principles of TDD can be used on any level (unit, integration, system)
- However, in real world, it could be hard to implement system test before the system itself.
- TDD was invented in a time when computers were slow and integration and system tests runs for a days.

Coding katas

- Is a way to
  - Exercise TDD
  - Exercise SOLID design
  - Exercise refactoring
  - Exercise test design
  - Learn new language

Naming conventions

- Project naming
  - <ProjectUnderTest>.Tests

- Class naming
  - <ClassUnderTest>Tests

- Test method naming
  - Given_When_Then
  - <methodUnderTest>_Given_Then
  - Pragmatic approach ;)

Unit test structure

- Arrange
- Act
- Assert

- One Assert per test (ideal situation)
- More asserts per test
  - Could be refactored (e.g. custom assert method)
  - https://www.amazon.com/xUnit-Test-Patterns-Refactoring-Code/dp/0131495054

Leap year

- 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.

- 2001 is a typical common year
- 1996 is a typical leap year
- 1900 is an atypical common year
- 2000 is an atypical leap year

What is Kent Beck saying about small steps?

“Remember, TDD is not about taking teeny-tiny steps, its’ about being able to take teeny-tiny steps. Would I code day-to-day with steps this small? No. But when things get the least bit weird, I’m glad I can.”

Fizz Buzz kata

- Fizz Buzz is a mathematical game which is played with a group of people. Each person says a number in sequence, but when the number is a multiple of 3, they have to say "Fizz", when it is a multiple of 5 they have to say "Buzz", and if it is a multiple of both 3 and 5, "FizzBuzz". If someone makes a mistake and it is noticed, they are out.

- A typical game might start like: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, etc.

Pangram kata

- Determine if a sentence is a pangram. A pangram (Greek: παν γράμμα, pan gramma, "every letter") is a sentence using every letter of the alphabet at least once. The best known English pangram is:

  The quick brown fox jumps over the lazy dog.

- The alphabet used consists of ASCII letters a to z, inclusive, and is case insensitive. Input will not contain non-ASCII symbols.

Test behavior, not implementation

- Implementation details should be hidden from the tests. If you need to change tests often because of changes that could be considered as implementation details, you are testing implementation, not behavior.
- It is ok to use tests to validate some implementation (e.g. complex LINQ expression) and delete it afterwards.
- Making methods *public* in order to test them is not a good idea.
- Making them *internal* as a shortcut is the same. It often shows poor decomposition of a system and violation of Single Responsibility Principle.

Ron Jeffries summary of TDD

The goal is clean code that works. [1]

Divide and conquer, baby. First, we’ll solve the “that works” part of the problem. Then we’ll solve the “clean code” part. [1]

Events

- http://codingdojo.cz/
- http://globalday.coderetreat.org/

Bibliography

[1] Beck, K., Test-Driven Development: By example, 2003

[2] Fowler, M., Test Pyramid, online
https://martinfowler.com/bliki/TestPyramid.html

[3] Ráček, J, Analýza a návrh systémů, studijní materiály FI MU, 2010