

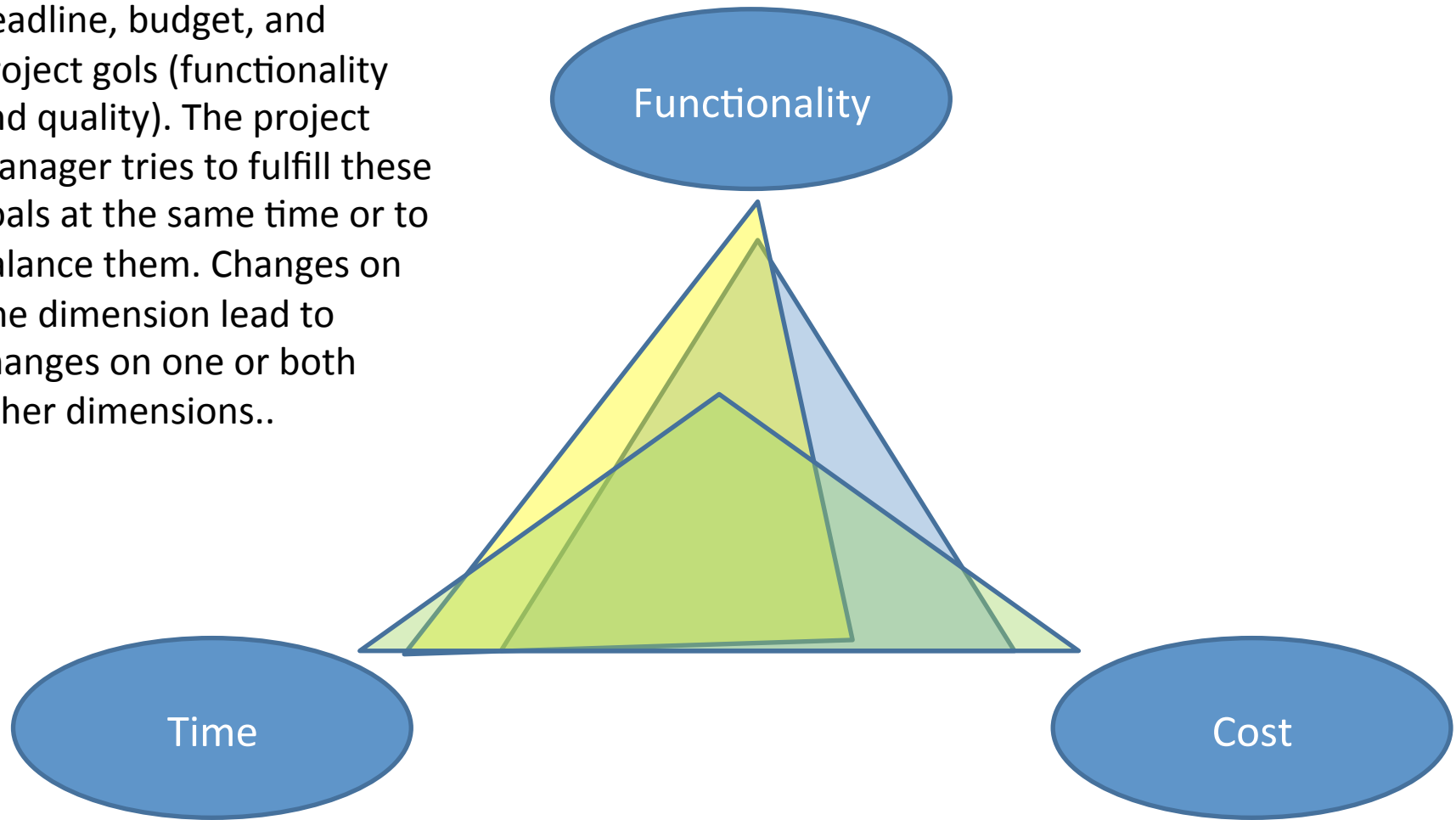
---

# Testing tutorial

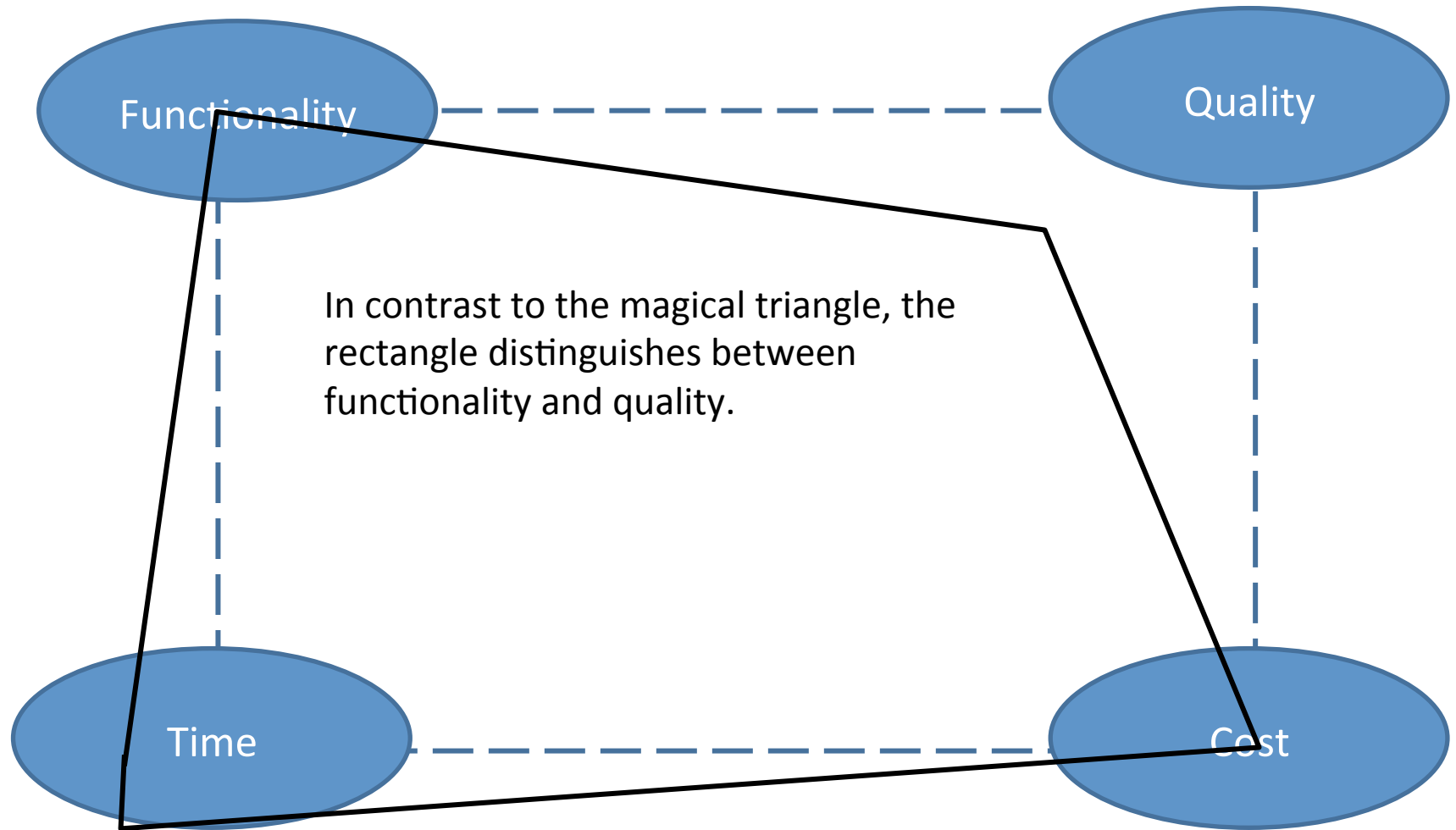
Agile Development Processes 2014  
Eric Knauss and Emil Alégroth

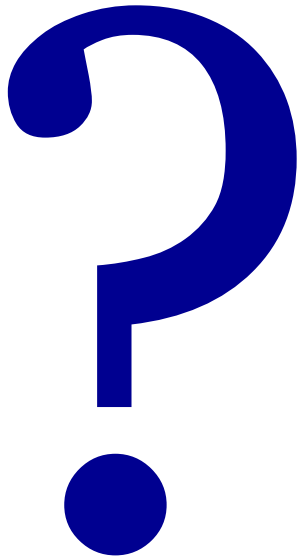
# *Magical Triangle* of Project Management

The triangle consists of deadline, budget, and project goals (functionality and quality). The project manager tries to fulfill these goals at the same time or to balance them. Changes on one dimension lead to changes on one or both other dimensions..



# Magical Rectangle of Project Management



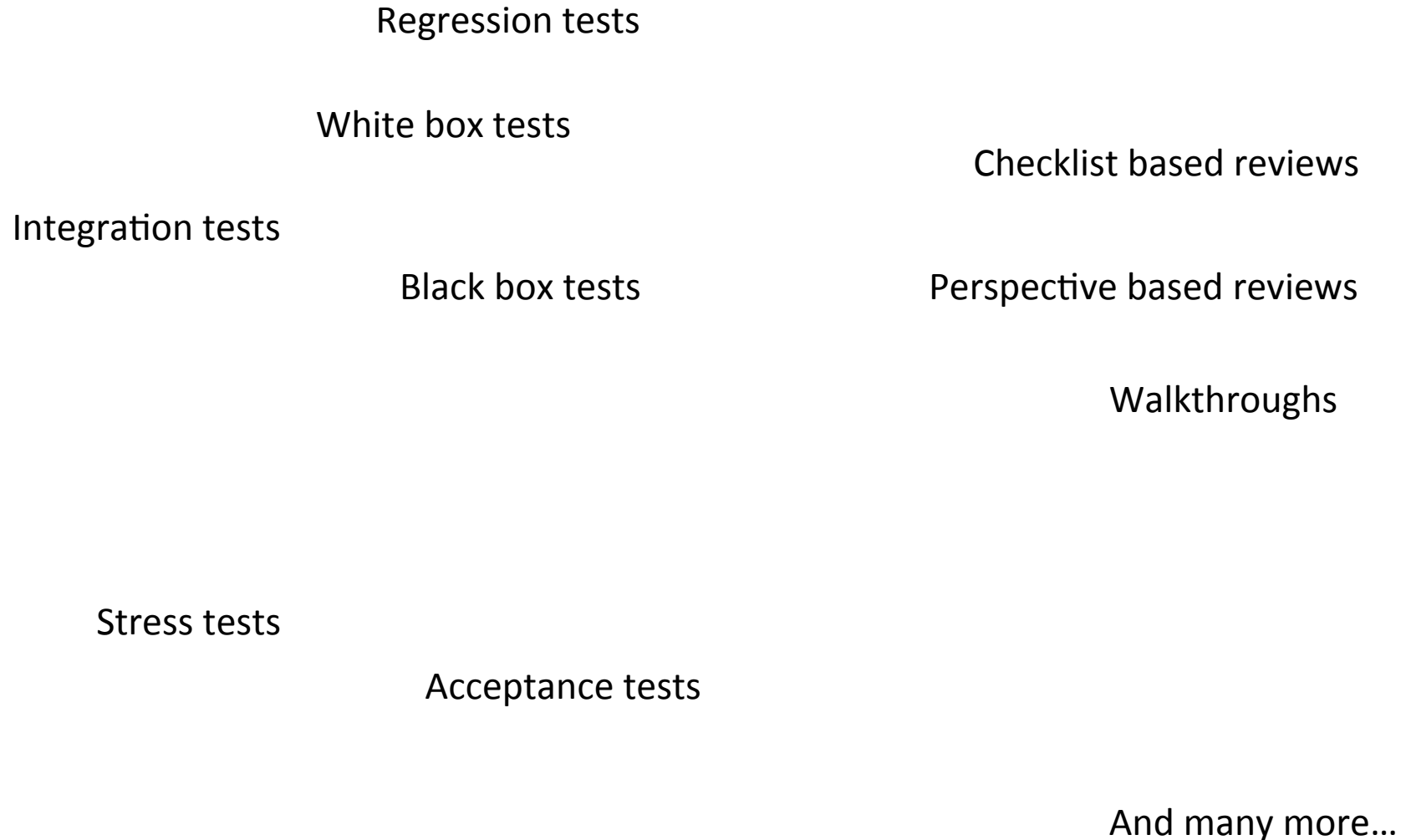


- How would you shorten time-to-market?
  - a) Reduce testing effort
  - b) Reduce functionality
  - c) Add more developers

In scope of this lecture  
(agile!): Only b) is a valid  
answer!

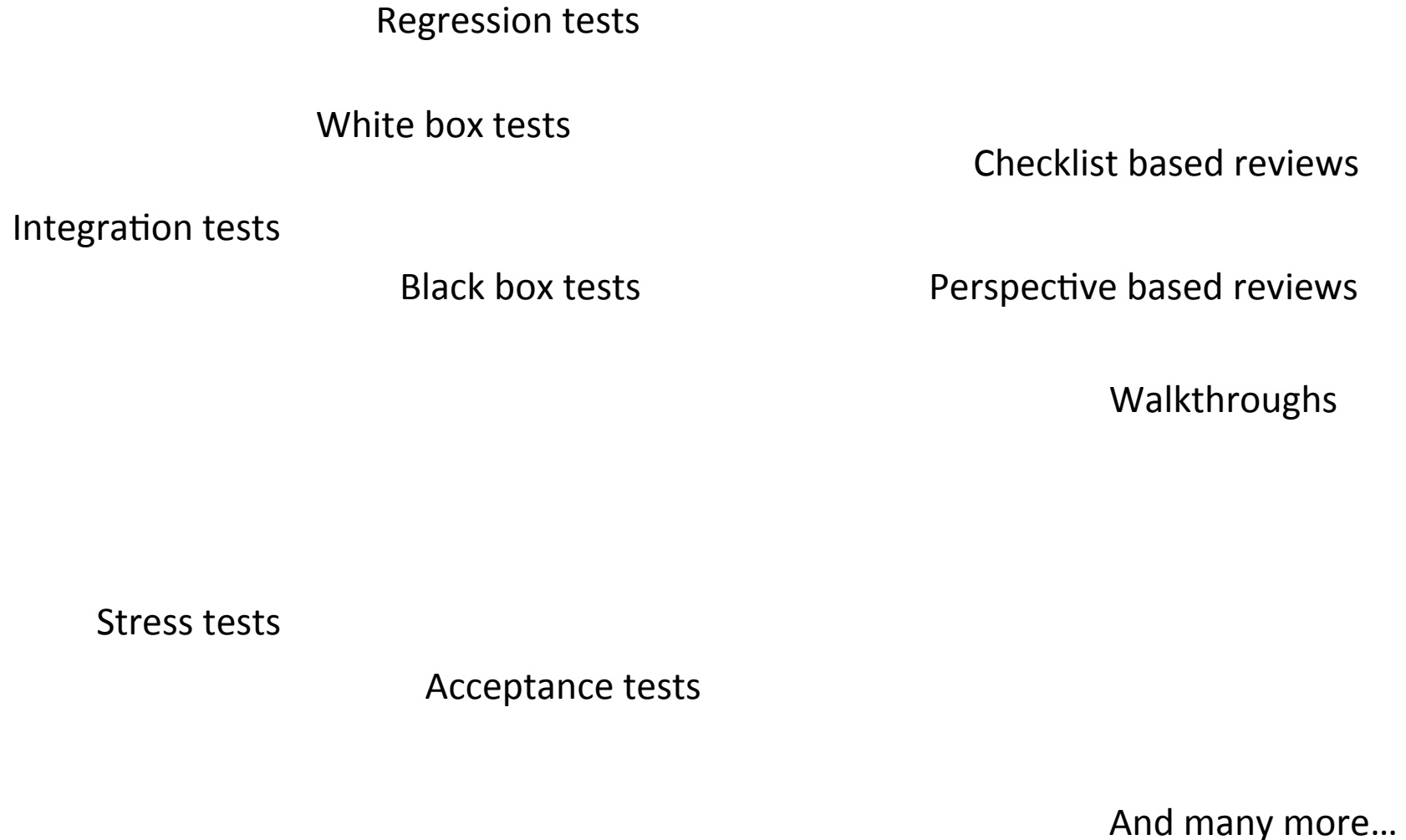
# How do you manage software quality?

---



# How do you manage software quality?

---



# How do you manage software quality?

## Test-driven development

Integration tests

Regression tests

White box tests

Black box tests

Stress tests

Acceptance tests

Onsite customer

Checklist based reviews

Perspective based reviews

Walkthroughs

Pair programming



# TestFirst

- If testing is good, then testing more often / always is even better
  - We want to embrace change – Regression testing
- Idea: Write test early, even before implementation
  1. Write test
  2. Let test fail
    - Do we really test non-existing functionality?
  3. Implementing, until test is green
    - As *simple as possible*!
  4. Refactoring



# Principle of TestFirst: a Dialogue

Task: Java method `len(int)` returns number of digits of an int.

Test starts

„`len(5)` should be 1!“

```
assertEquals(1,  
    len(5));
```

JUnit

COMPILER-ERROR!

What is the meaning of  
“len”?

Program: That is easy:

```
public int len(int zahl) { return 1; }
```

JUnit: ok. Testcase  
fulfilled.

Test: Just you wait!

„`len(321)` should be 3!“

```
assertEquals(3,  
    len(321));
```

JUnit: Error!  
1 instead of 3

Program: No problem ...

```
if zahl<10 then return 1 else return 3
```

JUnit: ok.

Test: I don't believe this!

“`len(12345678)` should be 8!“

```
assertEquals(8, len(12345678));
```

JUnit: Error!  
3 instead of 8

Program: ... ok, I see a pattern here:  
for (i=...

# Test-Driven Development

Testcases and automatic regression tests for every class in product

- 10 The automated tests are the design. The on-site customer makes the acceptance tests.
- 8 After doing design and prototypes, we create a few testcases
- 6 As soon as the code is done, we create thorough unit tests, only after that goes the code to the test team.
- 4 We have heard about JUnit. Never tried it though.
- 2 Our system test phase always runs out of time: There are many errors!
- 0 We do not test explicitly. Sometimes a customer tells us when there is a problem.

c.f.: Krebs, William (2002):  
Turning the Knobs: A Coaching  
Pattern for XP through Agile  
Metrics. Springer, Lecture Notes  
on Computer Science 2418

# Roman Numbers Kata

- Idea of Katas: Practice by repetition.
  - Here: Test-First

The Kata says you should write a function to convert from normal numbers to Roman Numerals: eg

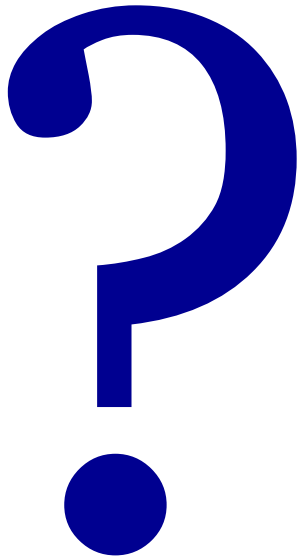
1 --> I

10 --> X

7 --> VII

etc. For a full description of how it works, take a look at [\[http://www.novaroma.org/via\\_romana/numbers.html\]](http://www.novaroma.org/via_romana/numbers.html).

For some ideas on how to continue, look at <https://github.com/pedrovg/RomanNumerals-Kata>





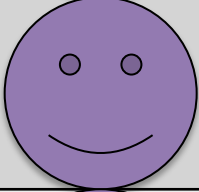
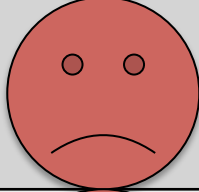
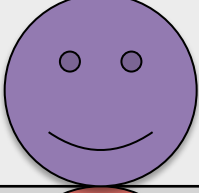

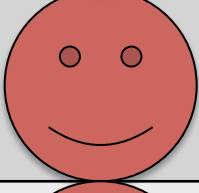
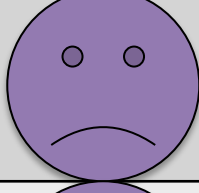
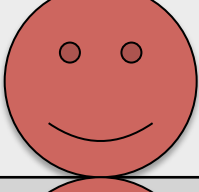
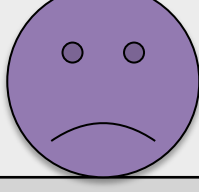

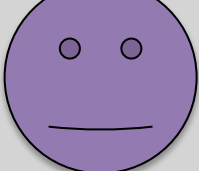
- Are those tests Blackbox or Glassbox?
- Traditionally, programmers and testers are supposed to be different persons.
  - Why?
  - Does that not kill the testfirst idea?

# Assume that you are a quality agent

---

- What is your goal?
  - Systematically manage quality
  - Make sure that system works as specified
- What are your competencies?
  - Methods and practices of QM
  - Delay delivery to customer?
- What are your responsibilities?
  - Sign of that software was developed according to state of practice in quality management

# You are a Quality Agent / Agile Coach

Daily builds		
Face-to-face communication over written documents		
Iterative requirements		
Long code- / feature-freezing		
Exact specification		
QM strategy	 	



Inspired by / based on Original Software: The reality of software testing in an agile Environment, Whitepaper

# TESTING IN AGILE ENVIRONMENTS

# “You only need to unit test”

---

- Investigative testing?
  - Goal of developer: Show that code works
  - Goal of tester: Show that code does not work



# “You can reuse unit tests to build a regression test suite”

---

## **Unit test**

- Prove that code will do what is expected

## **Regression test**

- Ensure that no unexpected effects result from changes

# “Unit tests remove the need for manual testing”

---

- Manual testing is a repetitive task; it's expensive, boring and error-prone.
- Though manual testing is a time-consuming (and therefore expensive) way to find errors, the costs of not finding them are often much higher.

# “We no longer need testers”

---

- Quantity of productive code = quantity of test code
- Need to do regression tests
- Need to ensure a systematic approach
- Need to coach developers

# “User acceptance testing is no longer necessary”

---

- Seeing the product leads to new requirements
  - Expectations change / are not met
- Agile offers feedback cycles to capture this effect early
- Still need to sign of

# “Developers have adequate testing skills”

**From story to unit test** – “For each requirement, how would I test that?”

**Integration testing** – “Which tests do I need to run to ensure the new code works seamlessly with the surrounding code”

**System testing** – “Does the functionality supported by the new code dovetail with functionality elsewhere in this system, or in other systems within the process flow? ”

**Regression testing** – “How often do I need to run a regression test to ensure there are no unforeseen impacts of the new code?”

**Acceptance testing** – “While TDD (in collaboration with business users) should ensure that a specific function performs correctly, is the cumulative impact of changes still acceptable to the business users?”

# “The unit tests form 100% of our design specification”

---

- That might be a lot of stuff
- Is test code always a good choice to document that amount of information?
- As size of project is increased, the execution time of tests is increased as well
  - Need to partition the project and/or the tests
  - Test and Execution Management!

# Conclusion

- QA can play an important role in agile projects
- Who else is better placed to
  - Bridge the gap between users and developers,
  - Understand what is required,
  - Understand how it can be achieved
  - Understand how it can be assured prior to deployment?
- To allow this, QA's need to be experts in
  - Quality management
  - Agile development
  - Requirements engineering

