



Open Universiteit  
www.ou.nl

Philipps



Universität  
Marburg

Christoph Bockisch  
Steffen Dick

# QPED

Quality-focused Programming  
Education

Digitale Kaffeerunde



Universitat  
Oberta  
de Catalunya

TU/e

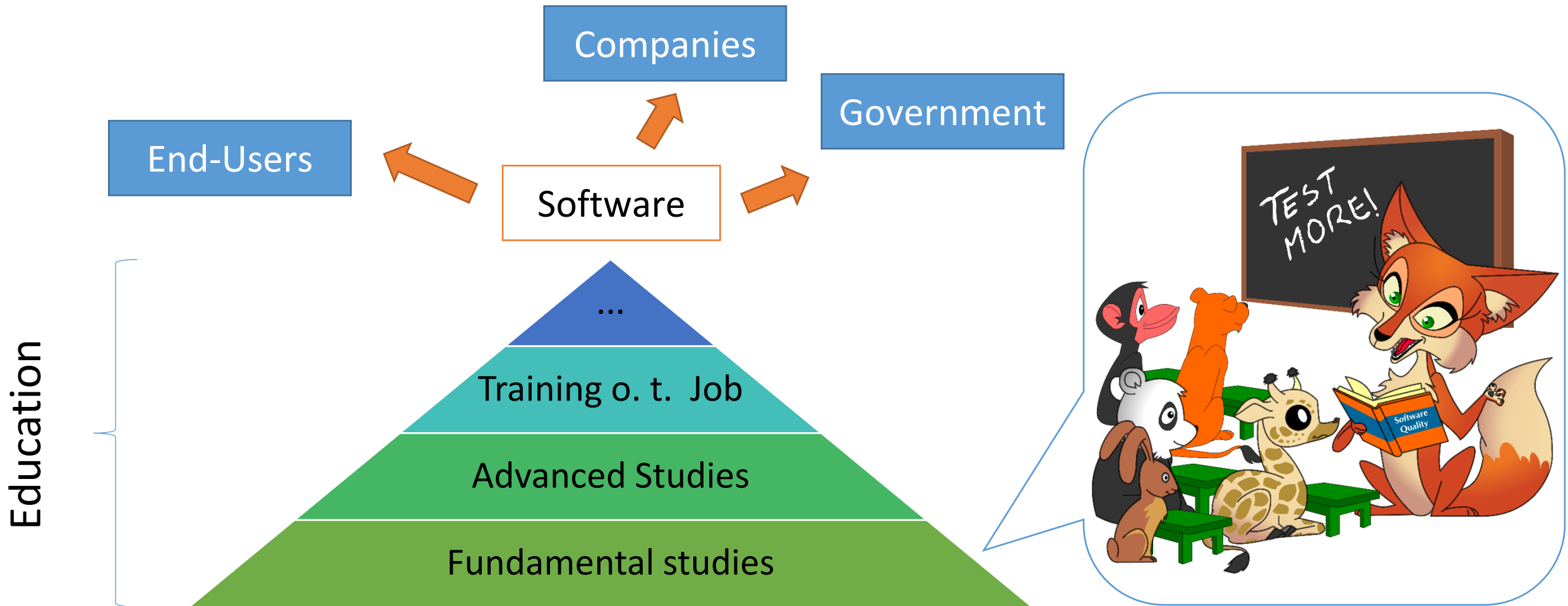
Quarterfall

Co-funded by the  
Erasmus+ Programme  
of the European Union



QPED 2020-1-NL01-KA203-064626

# Influence of Computer Science Education



# Making Students More Quality-Aware

- **MORE** testing and documentation in education
- **EARLIER** testing in education
- **HABITUAL** testing and documentation
- Make it **FUN** and set an **EXAMPLE**

But our curriculum is already full.

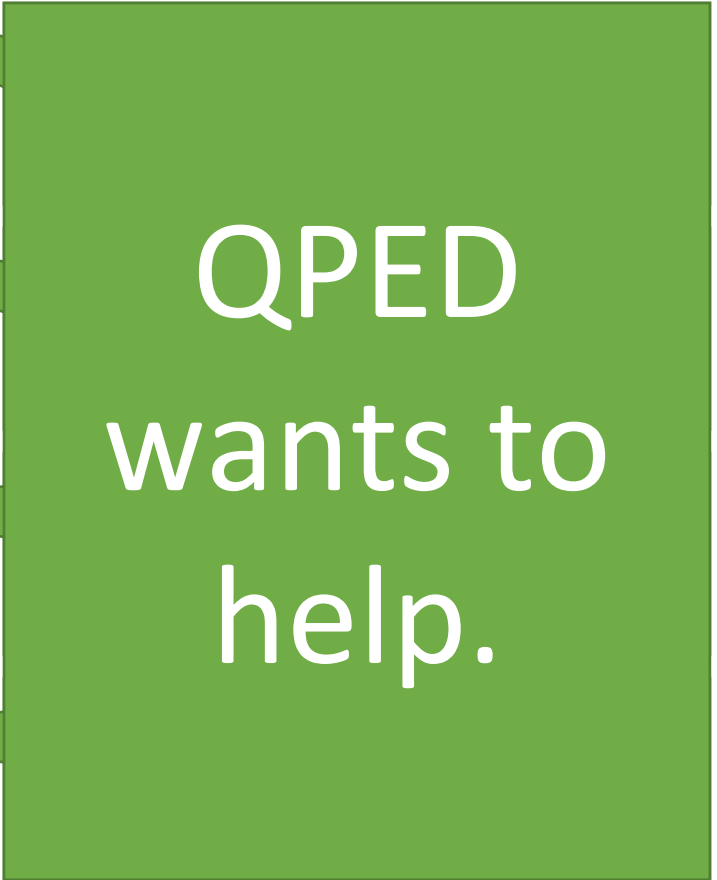
But students don't know enough in the beginning.

But we don't have the resources for constant reminders.

But we have no time to update our materials.

# Making Students More Quality-Aware

- **MORE** testing and documentation in education
- **EARLIER** testing in education
- **HABITUAL** testing and documentation
- Make it **FUN** and set an **EXAMPLE**



QPED  
wants to  
help.

# QPED Project – Multiplier Event

- Erasmus+ Strategic Partnership
- Five partners
- Germany, The Netherlands, Spain
- One traditional university, Two distant teaching universities, one SME

QPED Homepage



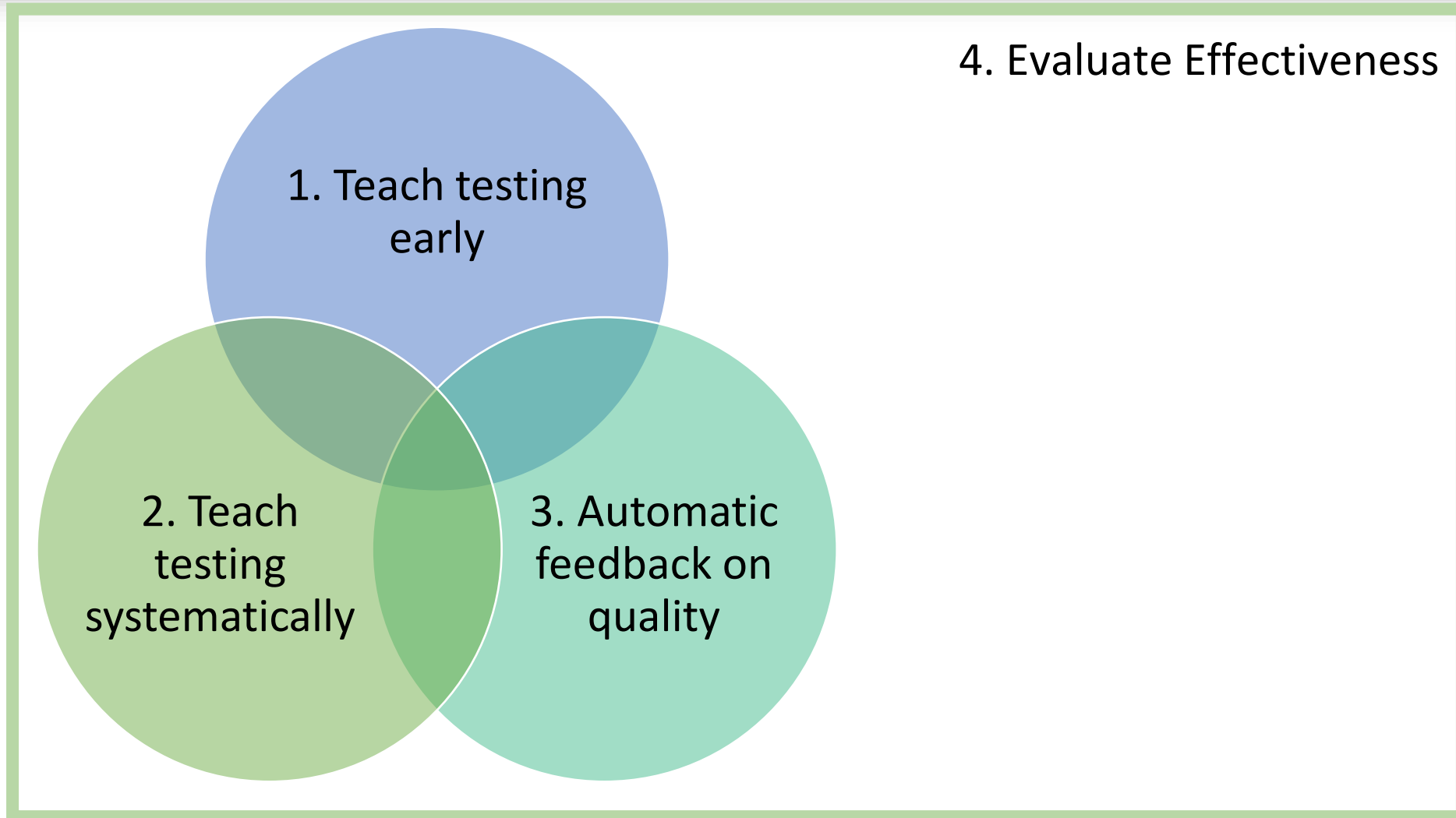
<https://qped.eu>

QPED Newsletter

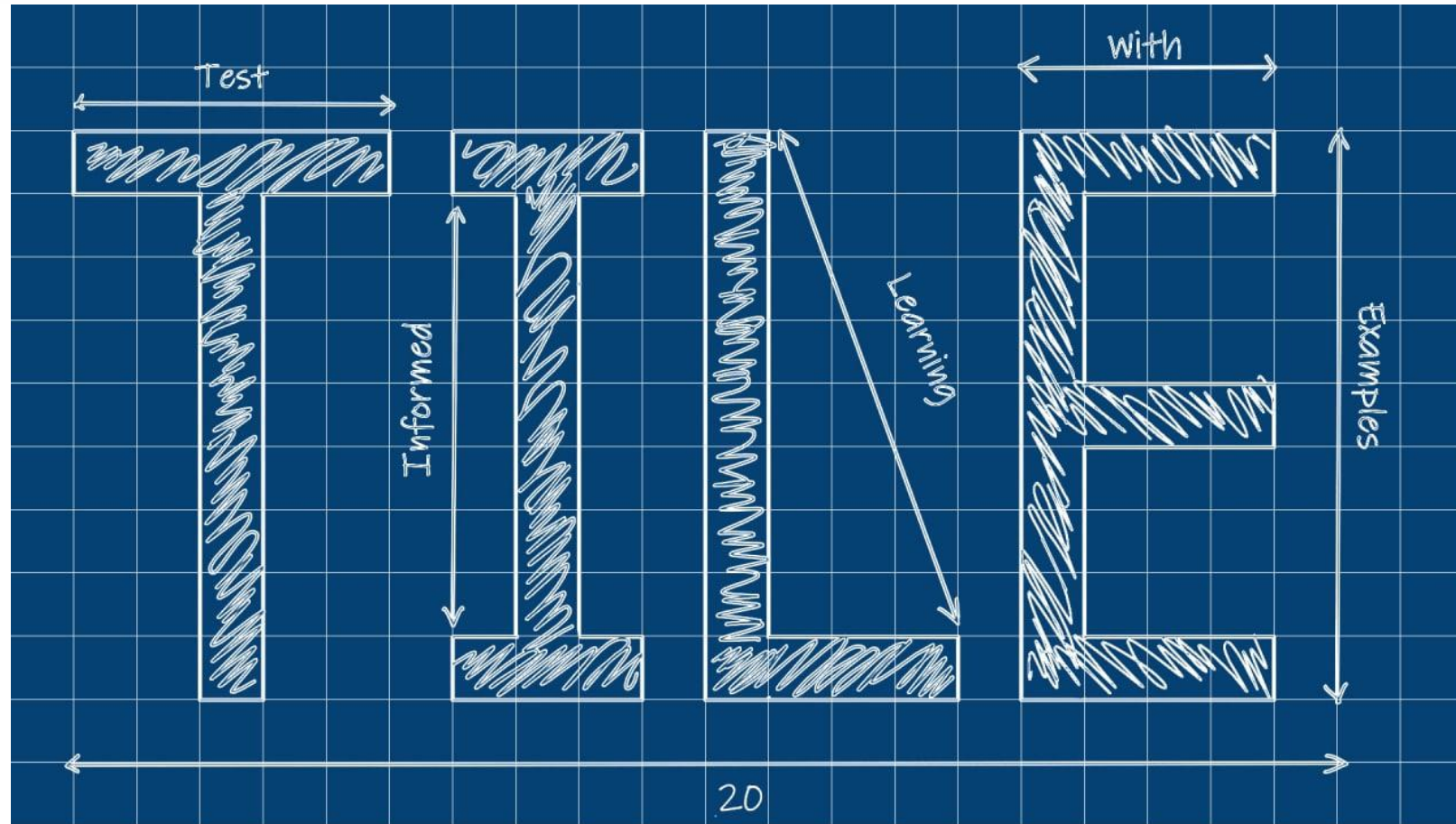


<https://listserv.dfn.de/sympa/subscribe/qped-info>

# Quality-focused Programming Education



# 1. TILE - Test Informed Learning with Examples



# TILE - Test Informed Learning with Examples

- **Early** - introduce testing from the very first program they see and write
- **Seamless** - as an inherent part of programming, and *not* as a separate activity
- **Subtle** – using clever and indirect methods





# Learning by examples

- Common way of introducing programming concepts to students:
  1. We explain the concept(s) using an **example**.
  2. We show **example** executions to see what comes out.
  3. We let students practice the concepts with exercises about **example** problems.

No extra lecture  
time needed.

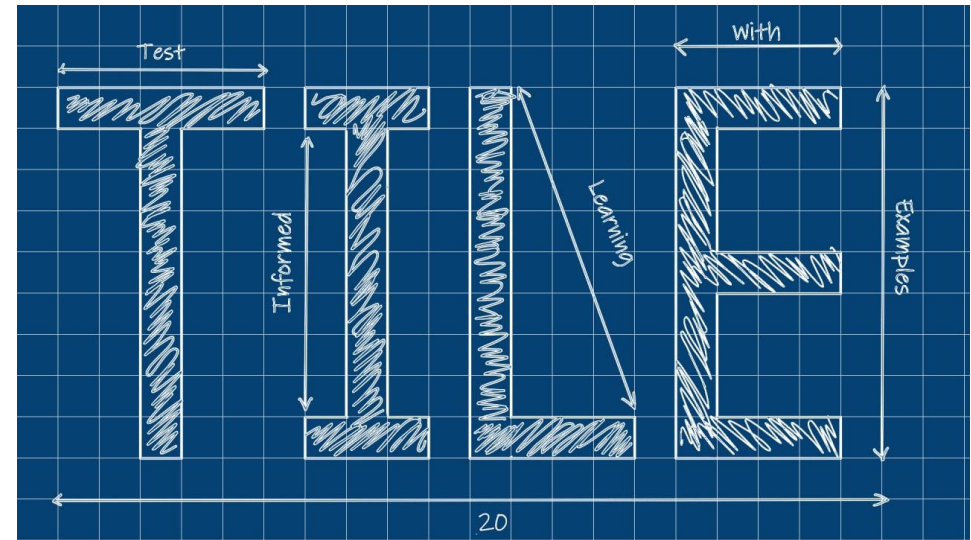
- TILE will relate these **examples** to testing

- Four types of TILES are distinguished:

1. test run examples
2. test cases examples
3. Test message TILES
4. test domain examples

# TILE - Repository

- Reusable, worked out examples
  - Over 100 TILES developed
  - Still growing
- Available at: <https://tile-repository.github.io>
- Categorized
  - TILE aspects
  - Topics
  - Technology used
  - Audience
  - Programming learning goals
  - Testing learning goals
  - Prerequisites



Reuse our teaching material.

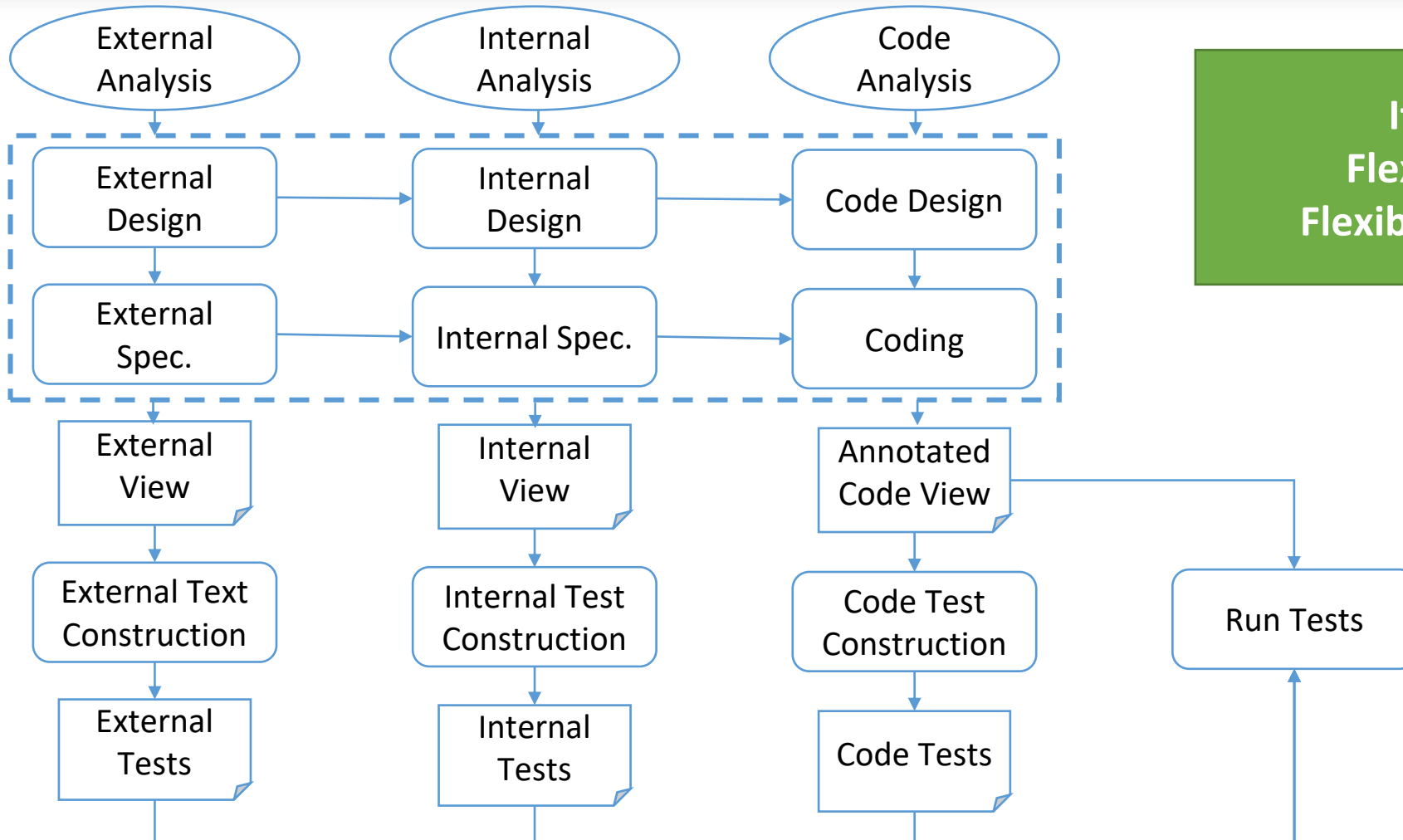
## 2. Procedural Guidance - Motivation

- Several studies demonstrate experiences:
- S. Edwards and Z. Shams, “Do student programmers all tend to write the same software tests?” in Proceedings of the 2014 conference on Innovation & technology in computer science education. ACM, 2014, pp. 171–176.
  - ~101 implementations, ~2500 failures, only 12 % bugs found
- M. Lawende, H. Passier, G. Alpár. Reproduction for Insight: Towards Better Understanding the Quality of Students Tests. In Proceedings of ITiCSE 2021. ITiCSE 2021
  - Same findings
- A. Bijlsma, N. Doorn, H. Passier, H. Pootjes, S. Stuurman. How do students test software units? ICSE 2021 – JSEET
  - Students perform only happy path tests
- Y. Kolikant, “Students’ alternative standards for correctness,” in Proceedings of the first international workshop on Computing education research. ACM, 2005, pp. 37–43
  - Attitude: “It compiles and there is console output, so we are ready!”

# Programming is a complex activity

- Often focus on conceptual knowledge
  - Syntax: variable declaration, if-then-else, ...
  - Concepts: control structure, polymorphism, dynamic binding, inheritance, ...
- For complex activities, also procedural knowledge is required (J. Merrienboer and P. Kirschner, 2017)
- Procedural knowledge, e.g.,
  - What activities are needed to solve this problem?
  - What decisions need to be made?
  - In which order?
  - Which conceptual knowledge is needed?
  - ...

# Procedural Guidance



Iterative and incremental  
Flexible in development order  
Flexible in level: formal or informal

# Procedural Guidance - Experience

- Students don't like a first-think-then-act approach
- BUT: Students appreciate the contracts (including the specifications)
- Quality (correctness) increases significantly:
  - Number of bugs decreases (~70 % test correctly)
  - Number of bugs found by testing increases (~80% bugs detected)

# Procedure Fully Worked Out

---

## Specification based OO development: Procedural Guidance

OUNL-CS Technical Report, No 6, 2022

Harrie Passier & Lex Bijlsma & Ruurd Kuiper

With the cooperation of:

Greg Alpar

Niels Doorn

Stijn de Gouw

Cornelis Huizing

Harold Pootjes

Stefano Schivo



Co-funded by the Erasmus+ Programme of the European Union

# 3. Feedback Toolkit

- Goals
  - Support students working on assignments
  - Feedback should be: Timely, Individual
  - Support teachers
- Automatically generate feedback
  - Available 24/7 with quick response times
  - Easily scalable
  - Should be easy and flexible to configure
  - Always monitor quality criteria
    - even if they are not the current learning objective



# Feedback Toolkit

- Feedback tools

- Java

- Syntax
- Style
- Solution approach
- Test coverage
- ...

- Python

- Syntax
- Style

- Mermaid diagrams (design)

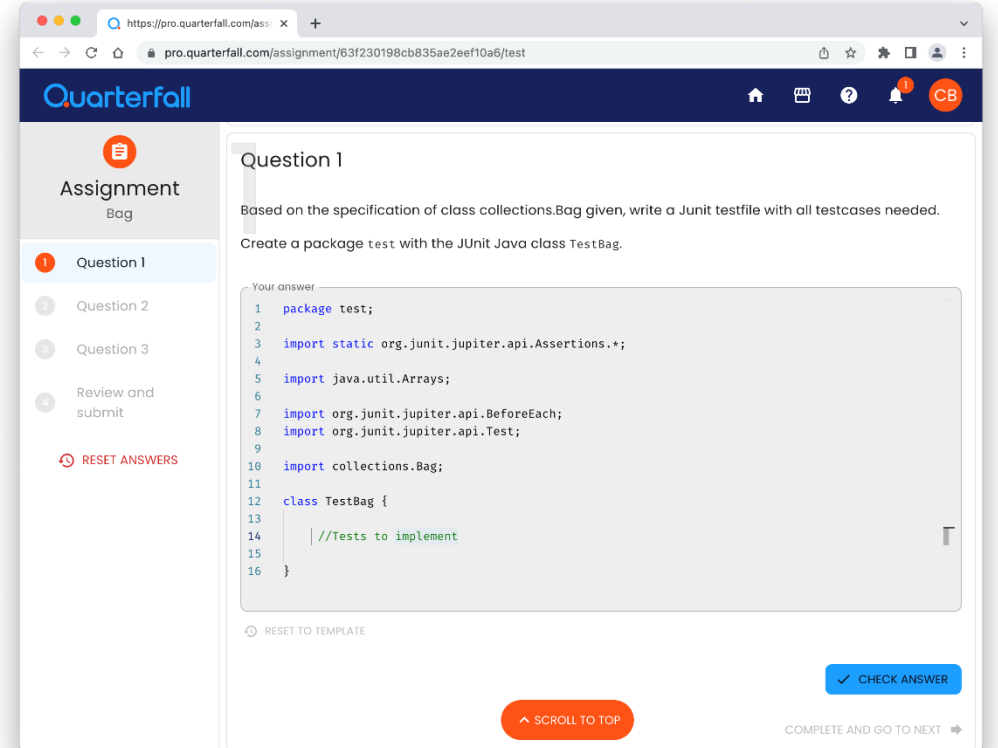
- Available as plugins for Quarterfall platform

- Also possible to use stand-alone

- Open-source

Simple, constructive,  
configurable feedback  
messages.

Unit tests for solution  
already built-in in  
Quarterfall.



The screenshot shows a web browser window displaying the Quarterfall platform. The page title is "Question 1" and the assignment is "Bag". The question text reads: "Based on the specification of class collections.Bag given, write a JUnit testfile with all testcases needed. Create a package test with the JUnit Java class TestBag." Below the question, there is a text area for the answer containing the following Java code:

```
1 package test;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import java.util.Arrays;
6
7 import org.junit.jupiter.api.BeforeEach;
8 import org.junit.jupiter.api.Test;
9
10 import collections.Bag;
11
12 class TestBag {
13
14     //Tests to implement
15
16 }
```

At the bottom of the answer area, there are buttons for "CHECK ANSWER" and "SCROLL TO TOP".

Students view

# Individual Feedback Messages

- Feedback should be configurable, depending on
  - Level of student
  - Progress in course
  - Teaching style
  - Etc.
- Translate messages for international students

# Configuring Feedback Messages

The screenshot shows the MASS Configuration Editor. The top navigation bar includes 'MASS', 'Home', 'MASS Configurator', 'MASS Documentation', and 'MASS Checker Tutorials'. The main content area is titled 'Configuration Editor' and has three tabs: 'MAIN SETTINGS', 'STYLE CHECKER', and 'TEST COVERAGE CHECKER'. The 'TEST COVERAGE CHECKER' tab is active, showing 'Test Coverage Check Configuration' with options for 'Show Test Failures' (checked) and 'Show Full Coverage Report' (unchecked). Below this is a 'Feedback Configuration' section with a table:

Message ID	Kind of Coverage	File Name	Line Ranges	Message	Suppressed Messages
attr_0	FULLY_MISSED	collections	43	You have not	

To the right of the configuration editor is a 'Configuration Data' panel displaying a JSON configuration snippet:

```
{
  "styleSelected": true,
  "semanticSelected": false,
  "coverageSelected": true,
  "classSelected": false,
  "metricsSelected": false,
  "syntax": {
    "level": "BEGINNER"
  },
  "style": {
    "basisLevel": "INTERMEDIATE",
    "complexityLevel": "INTERMEDIATE",
    "namesLevel": "INTERMEDIATE",
    "classList": "INTERMEDIATE",
    "classLength": -1,
    "methodLength": -1,
    "cyclomaticComplexity": -1,
    "variableNamePattern": "[a-z][a-zA-Z0-9]*",
    "methodNamePattern": "[a-z][a-zA-Z0-9]*",
    "methodParameterNamePattern": "[a-z][a-zA-Z0-9]*",
    "classNamePattern": "[A-Z][a-zA-Z0-9]*"
  }
}
```

The screenshot shows the Quarterfall interface for configuring a question. The top navigation bar includes 'Quarterfall', 'Home', 'Help', 'Notifications', and 'User Profile'. The main content area is titled 'Question 1' and is a 'Code question'. The 'FEEDBACK' tab is active, showing a 'Code' editor with the following configuration:

```
14 "classLength": -1,
15 "methodLength": -1,
16 "cyclomaticComplexity": -1,
17 "fieldsCount": -1,
18 "variableNamePattern": "[a-z][a-zA-Z0-9]*",
19 "methodNamePattern": "[a-z][a-zA-Z0-9]*",
20 "methodParameterNamePattern": "[a-z][a-zA-Z0-9]*",
21 "classNamePattern": "[A-Z][a-zA-Z0-9]*"
22 },
23 "coverage": {
24   "showTestFailures": true,
25   "showFullCoverageReport": false,
26   "feedback": [
27     {
28       "showMsg": "FULLY_MISSED"
```

Below the code editor is a 'Your answer' section with a text area containing the following code:

```
1 package test;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import java.util.Arrays;
6
7 import org.junit.jupiter.api.BeforeEach;
```

Teachers view

# 4. Pilot Study

- Choose one course per partner
- Collect data from the courses:
  - without applying the QPED approach (aka Baseline)
  - after applying the QPED approach (aka Validation)
- Compare both collections of data

Can we measure  
quality-awareness?

# Instruments for Assessing Quality Awareness

- **Rubric:** to assess assignments by using a common instrument which helps us to compare results
- **Questionnaire:** to gather students' perceptions about how they apply quality issues while programming.
- **Diagnostic test:** to assess students' awareness in an objective way.

# Rubric

4-point Likert scale

Feature	Negative Examples	1 - Fully Failed	2 - Partly Failed	3 - Partly Satisfied	4 - Fully Satisfied	Positive Examples
Readability	<input type="checkbox"/> Wrong indentation <input type="checkbox"/> Broken lines of code <input checked="" type="checkbox"/> Parentheses wrongly placed <input type="checkbox"/> Poor naming <input checked="" type="checkbox"/> Bad comments	<input type="radio"/> 1	<input type="radio"/> 2	<input checked="" type="radio"/> 3	<input type="radio"/> 4	<input checked="" type="checkbox"/> Correct indentation <input checked="" type="checkbox"/> Considered lines of code <input type="checkbox"/> Correctly placed parentheses <input checked="" type="checkbox"/> Good names <input type="checkbox"/> Useful comments
DRY principle	<input type="checkbox"/> Repeated code <input checked="" type="checkbox"/> Magic numbers	<input type="radio"/> 1	<input checked="" type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4	<input type="checkbox"/> Helper functions are used <input type="checkbox"/> Use of constants
Correctness	<input type="checkbox"/> Wrong file format <input type="checkbox"/> Not compiling/running <input type="checkbox"/> Specifications not met	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input checked="" type="radio"/> 4	<input checked="" type="checkbox"/> Functions properly <input checked="" type="checkbox"/> Correct results

**Achieved Points**

**Calculated Points**

**Maximum Points**

Choose a format for exporting (default is JSON):

10+ items related to quality issues

Pro and Con examples

Can generate grade and response based on selection.

Pilot Study: collect rubric data

# Questionnaire

## Assessment of Basic Self-efficacy at SQ

### 4. Evaluate the following statements:

(1 = strongly disagree; 2 = disagree; 3 = neither agree nor disagree; 4 = agree; 5 = strongly agree)

Concept	1	2	3	4	5
a. I can create to create correct programs for a given task.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
b. I can find errors and correct them easily.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
c. I can test my code effectively to check for errors.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
d. I can create code that can be easily readable by others.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
e. I am capable of reusing previous code for new projects.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

## Perception of Utility of SQ

### 5. Evaluate the following statements:

(1 = strongly disagree; 2 = disagree; 3 = neither agree nor disagree; 4 = agree; 5 = strongly agree)

Concept	1	2	3	4	5
a. I understand what software quality means.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
b. Using testing enhances the quality of programs.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
c. Testing is as important as coding.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
d. Readability is as important as the correctness of the program.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

23 items

5-point Likert scale

# Diagnostic test

We want to create a code that calculates the factorial of a number in the range of 0 and 12, both included.

```
public int factorial (int number) {
    if (number<0 || number>=12){
        System.out.println("Error: number out of range");
        return -1;
    }else if (number==0){
        return 1;
    }else{
        return number * factorial(number-1);
    }
}
```

Indicate three test cases that we must to create in order to check the correctness of the program

The following code has some semantic style errors.

```
public int specialFunction(String word, int times) {
    int length;
    length = (int) (word.length() * times);

    if(word.length() * times % 2 == 0){
        System.out.println("New length: "+length);
        return true;
    }else if(word.length() * times % 2 != 0){
        System.out.println("New length: "+length);
        return false;
    }
}
```

- Which are they?
- Refactor the code:



# Resources

- Available from the our web site (<https://qped.eu>)
  - TILE repository
    - Open-source Git repository
  - Procedural Guidance
    - Fully worked out documentation
  - Feedback tools for Java and Python
    - Tutorials
    - Documentation
    - Open-source Git repository
  - Rubric tool
    - Open-source Git repository
    - Available as Github Page

# QPED Project – Thank you

QPED Homepage



<https://qped.eu>

visit our demo  
on the  
exhibition area

QPED Newsletter



[https://listserv.dfn.de/sympa/  
subscribe/qped-info](https://listserv.dfn.de/sympa/subscribe/qped-info)