Studying students' sensemaking to improve software testing education

The problem

Software testing needs to become a first class citizen in computer science education:

- Software is everywhere however, software without failures is not.
- Software failures can have catastrophic effects.
- Software testing is one of the most effective ways to measure software quality and identify possible problems to solve.

> "Software testing is one of the most effective ways to improve software quality, unfortunately, many students currently don't learn how to test their software very well."

What are the reasons' why we haven't fully integrated software testing into the curriculum?

- Programs already created fully packed courses.
- Students and teachers do not find Software testing very sexy.
- Many courses mainly focus on programming.
- There is limited material to support teachers.

Figure 1 shows an example of a test case.

Figure 1: Comic of test case design

Why is this an interesting problem

We believe that Computer Science graduates who have better software testing skills can benefit from this by:

- Having a more thoroughly understanding of specifications.
- Delivering better software designs.
- Having a different mindset that helps them to better explore problems.

Software testing itself can be a useful tool in education:

- It provides meaningful feedback for the student.
- It challenges students to ask more questions while exploring a problem.
- It gives educators and students more insights into the quality of the produced software.

> "Not only can testing help to improve software quality, it can also be used as an instrument to provide meaningful feedback to the student and the teacher."

We think that some factors complicate this problem:

- Software testing techniques overlap with programming techniques, resulting in an underestimation of the effort needed to learn how to test.
- Software testing requires a different mindset than software programming.
- Different educational contexts require different instructional designs. For example Design Based Education, shown in Figure 2, works with external projects, whereas traditional programs, and distance learning, can benefit from a more controlled environment.
- Different schools of software testing exist which seem unable to get consensus on the right approaches. We believe testing should be both model based and exploratory.

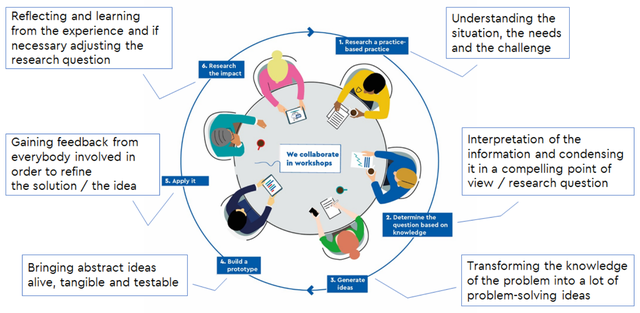

Figure 2: The phases of Design Based Education

It is an unresolved problem

There is a gap in research on software testing:

- Research shows the need for improving software testing in Computer Science Education [3, 7, 12, 4].
- Little research has been done on sensemaking of test case design of students.
- Not much research exists on the specific didactic approaches or instructional designs for this subject.
- Existing research on misconceptions, as shown in Figure 3, mostly focuses on programming and not on testing.

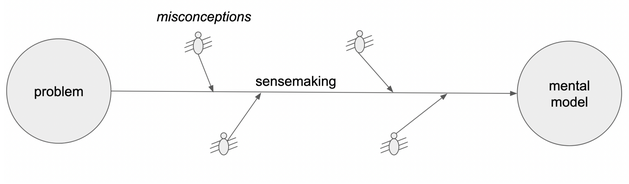

Figure 3: The effects of misconceptions on the mental model while solving a problem

Here is our idea

The goal of this research is to get students to become so so-called 'Test Infected' [5]

> "Students who are 'Test Infected' are only satisfied with their software products when they are confident they have tested their software in an effective, structured, systematic and reproducible way."

To get to this point, we need to start with improving computer science education.

Our research goals are:

- To get a better understanding of the students sensemaking while testing software.
- Identify misconceptions that occur when students design test cases.
- Develop instructional designs to create evidence based test-aware introductory programming courses.
- Study the effects of such courses in different educational contexts.

We want to use Design Based Research in Education (DBR) as an approach [1].

- DBR is an iterative approach using three phases, as shown in Figure 4:
  - **Analysis:** to explore the problem.
  - **Design:** to design an intervention or experiment.
  - **Evaluation:** to reflect on the intervention or experiment.
- This results in maturing interventions and theoretical understanding.

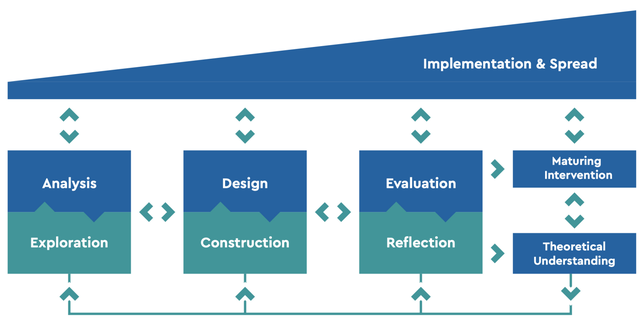

Figure 4: Generic model for conducting design-based research in education [10]

During the multiple iterations we can study different interventions around the themes of our interest. We can replicate these experiment in different educational settings to get a better understanding of the generalizability of the findings. With the findings we can refine the experiment for a next iteration. Figure 5 shows how multiple iterations combined lead to a macro iteration.

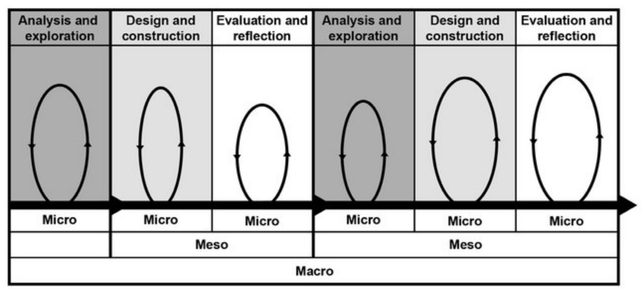

Figure 5: Micro-, meso- and macro-cycles in educational design-based research [10]

Figure 6 shows how we will do experiments in different educational settings to generalize and refine.

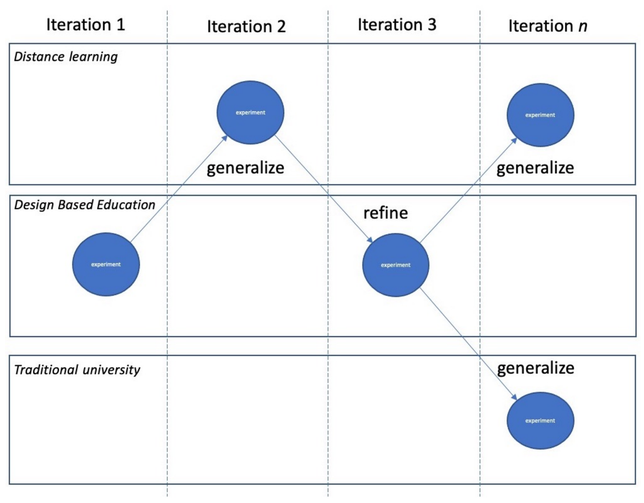

Figure 6: For each theme, we will iteratively perform experiments in the Design Based Education setting. We can repeat the experiments in different other settings. Based on the results of the experiment.

> "Using epistemic games, we can get a better understanding of the sensemaking process of students while testing software."

The following themes will be studied in the iterations:

- Getting a better understanding of sensemaking using epistemic games.
- Identifying misconceptions during test case design using lab experiments.
- Creating test awareness by using TILE.
- Evaluating the use of procedural guidance based on the results of the epistemic games.

To introduce testing in existing courses, an intervention called Test Informed Learning with Examples can be used.

> "TILE can be seamlessly integrated early on into existing introductory programming courses in a subtle way."

How our idea compares to other approaches

- Many studies focus on applying test approaches from the industry in education such as Test Driven Development (TDD) [9, 11, 2]. TDD is much more about the moment in the development process when students create tests and not about how to design good test cases [6].
- Janzen and Hossein proposed Test Based Learning (TBL) as a pedagogical approach for teaching test driven development [8]. However, it was not fully explored as the research changed focus onto test driven development.
- Other approaches focus on gamification, automation and enforcing testing, but not on the didactics of testing.

References

[1] Arthur Bakker. *Design Research in Education: A Practical Guide for Early Career Researchers*. Aug. 2018. ISBN: 9780203701010. DOI: 10.4324/9780203701010.

[2] Li-Ren Chien et al. "An evaluation of TDD training methods in a programming curriculum". In: *2008 IEEE International Symposium on IT in Medicine and Education*. IEEE. 2008, pp. 660–665.

[3] N. Doorn. "How can more students become 'test-infected': current state of affairs and possible improvements". MA thesis. Open Universiteit, 2018.

[4] Stephen H. Edwards. "Teaching software testing. automatic grading meets test-first coding". English. In: *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications - OOPSLA '03*. New York, NY: ACM Press, 2003, pp. 318–319. ISBN: 9781581137514. DOI: 10.1145/949344.949431. URL: https://doi.org/10.1145/949344.949431.

[5] Martin Fowler. *Junit test infected: Programmers love writing tests*. Feb. 2001. URL: http://junit.sourceforge.net/doc/testinfected/testing.htm.

[6] Steven Fraser et al. "Test driven development (tdd)". In: *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer. 2003, pp. 459–462.

[7] Vahid Garousi et al. "Software-testing education: A systematic literature mapping". In: *J. Syst. Software* 165 (July 2020), p. 110570. ISSN: 0164-1212. DOI: 10.1016/j.jss.2020.110570. URL: https://doi.org/10.1016/j.jss.2020.110570.

[8] David Janzen and Hossein Saiedian. "Test-driven learning in early programming courses". In: *SIGCSE Bull.* 40.1 (Feb. 2008), pp. 532–536. ISSN: 0097-8418. DOI: 10.1145/1352322.1352315. URL: https://doi.org/10.1145/1352322.1352315.

[9] Sami Kollanus and Ville Isomöttönen. "Test-driven development in education: experiences with critical viewpoints". In: *Proceedings of the 13th annual conference on Innovation and technology in computer science education*. 2008, pp. 124–127.

[10] Susan McKenney and Thomas C. Reeves. *Conducting educational design research*. English. 2nd. United Kingdom: Routledge, Sept. 2018. DOI: 10 . 4324 / 9781315105642.

[11] James Miller and Michael Smith. "A TDD approach to introducing students to embedded programming". In: *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*. 2007, pp. 33–37.

[12] Lilian Passos Scatalon et al. "Software Testing in Introductory Programming Courses: A Systematic Mapping Study". en. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. Minneapolis MN USA: ACM, Feb. 2019, pp. 421–427. ISBN: 978-1-4503-5890-3. DOI: 10.1145/3287324.3287384. URL: https://dl.acm.org/doi/10.1145/3287324.3287384 (visited on 10/31/2021).

Scan this code to learn more about the research of Niels Doorn

edu.nl/utgcw

ITiCSE 2022 DUBLIN IRELAND