



Open Universiteit

# From Rationalism to Empiricism in Software Testing Education Through Gamification

INTED 2024

Niels Doorn Tanja E.J. Vos Beatriz Marín

# Importance of software testing

SCHARON HARDING, ARS TECHNICA

GEAR MAR 1, 2024 1:12 PM

## A Leap Year Glitch Broke Self-Pay Gas Station Pumps Across New Zealand

It's like if the Y2K bug happened, but only for gas station pumps. And only in New Zealand.

Figure: Screenshot of an article titled "A Leap Year Glitch Broke Self-Pay Gas Station Pumps Across New Zealand" [1]

# Software Testing in CS Education

- ▶ Integrating it into Computer Science curricula is challenging [2], [3].
- ▶ Often a **rational design paradigm** is used in CS programs.
- ▶ Little research on didactic approaches is available.

# Consequences

The way we now teach software testing leads to:

- ▶ Students who use a ‘developer approach’ to testing [4].
- ▶ This approach lacks exploration and experimentation.

We need to shift the mental model of students away from this rational approach.

# RATIONALISM - EMPIRICISM



# Abductive reasoning as the base for testing

Abductive reasoning is a form of logical inference that seeks the simplest and most likely conclusion from a set of observations [5].

# Abductive reasoning as the base for testing

Abductive reasoning is a form of logical inference that seeks the simplest and most likely conclusion from a set of observations [5].

This fits very well with exploratory testing because:

**What** the behaviour of the system looks like is unknown, **how** the design process of tests should look like is unknown. The **desired situation** is unknown, and so is **the road towards it**.

# Development of a game to teach software testing



Our goals for a serious game:

- ▶ Incorporating empirical methods and critical thinking.
- ▶ Supporting different educational contexts.
- ▶ Enabling abductive reasoning.

# Gamification in CS

Some results of our literature review (including gray literature):

- ▶ Gamification is effective in CS education through: Real-world scenarios, competitive elements, immediate feedback, interactive activities, and collaboration [6].
- ▶ Gamification is applicable across various educational strategies and contexts [7]–[9].
- ▶ Innovative tools and techniques include educational chatbots and the use of serious games in secure programming [10], [11].
- ▶ Gamification for learning Scrum [12].
- ▶ Applying gamification can lead to oversimplification and decreased intrinsic motivation [6].

# CodeDefenders: game to learn mutation testing



The screenshot shows the CodeDefenders game interface. At the top, there's a navigation bar with links for 'Home', 'Multiplayer', 'Puzzles', and 'Impressum'. Below that, the title 'Game #3666' is displayed, along with player roles: 'Meles' (Melee) and 'Observer'. A toolbar at the top right includes buttons for 'End Game', 'Rewards', 'Score', 'Scoreboard', 'Timeline', 'Grade Report', 'Feedback', 'Editor Mode: default', and 'Chat'.

The main area is divided into two sections:

- Existing Mutants:** This section lists various mutants, each with a collapse/expand arrow. The mutants include:
  - All Mutants
  - Mutants outside methods
  - UFL(0x, int)
  - getTopFloor()
  - getCurrentFloor()
  - getCapacity()
  - getNumFloors()
  - isFull()
  - addRider(int)
  - goUp()
  - goDown()
  - cal()
- Class Under Test:** This section displays the source code of the UFL class.

```
1 public class UFL {  
2     //  
3     private int topFloor;  
4     private int currentFloor = 0; // default  
5     private int capacity = 10; // default  
6     private int numFloors = 8; // default  
7     //  
8     public int highestFloor() {  
9         topFloor = highestFloor;  
10    }  
11    //  
12    public int highestFloor, int maxFloors) {  
13        this(highestFloor);  
14        capacity = maxFloors;  
15    }  
16    //  
17    public int getTopFloor() {  
18        return topFloor;  
19    }  
20    //  
21    public int getCurrentFloor() {  
22        return currentFloor;  
23    }  
24    //  
25    public int getCapacity() {  
26        return capacity;  
27    }  
28    //  
29    public int getNumFloors() {  
30        return numFloors;  
31    }  
32    //  
33    public boolean isFull() {  
34        return numFloors == capacity;  
35    }  
36}
```

At the bottom of the screen, there are buttons for 'Live', 'Killed', 'Claimed Equivalent', and 'Equivalent'. A note 'Mutant/Test restrictions: Moderate' is also present. The footer contains links for 'About CodeDefenders', 'Contact Us', 'Help', and 'Imprint and Privacy Policy'.

Figure: CodeDefenders, an online game to learn mutation testing

# Testable: gamification of unit testing



Figure: Testable - gamified tool to improve unit testing teaching

# Testing Maze: adventure into functional testing

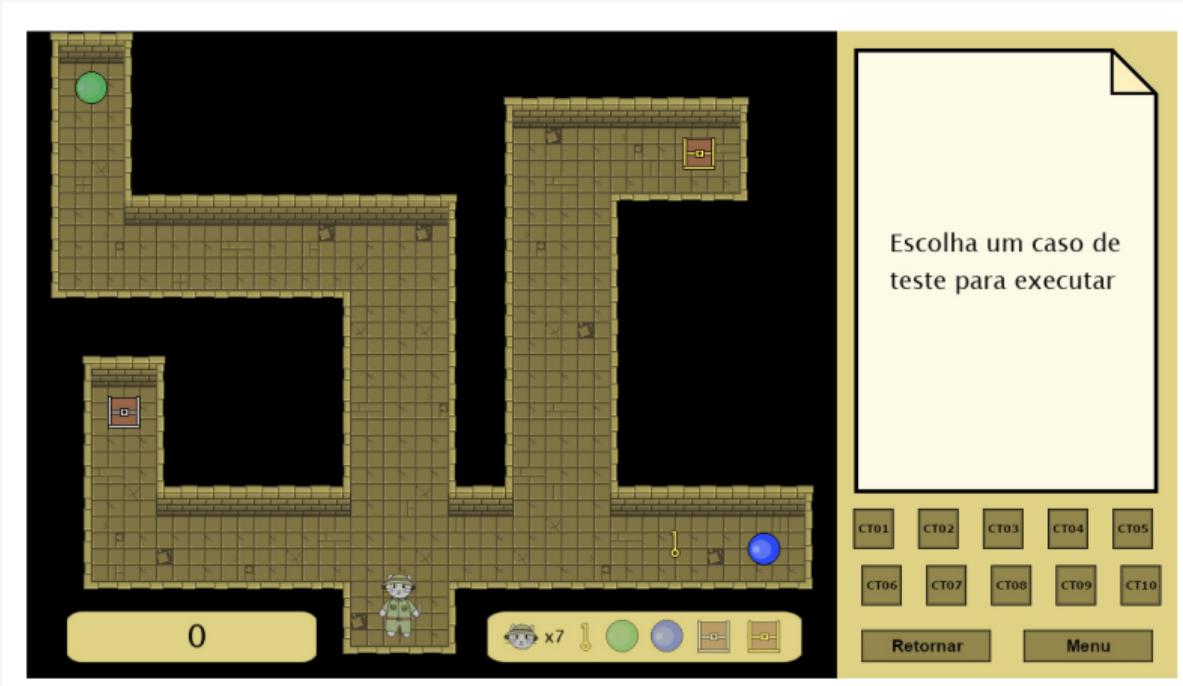


Figure: Testing Maze, an educational puzzle game for teaching functional testing concepts and test specifications containing a fantasy narrative

# TestSphere: card deck to support interaction



Figure: TestSphere, a card deck to support testers thinking and talking about testing

# Would Heu-risk it?: card deck to share experiences



Figure: 'Would Heu-risk it?' is centered around risk analysis, heuristics, patterns/anti-patterns of software testing

# No existing game that match our goals

- ▶ Most games focus on techniques.
- ▶ No games on our goals.
- ▶ We need to develop a game ourselves.

# Our game

Based on Risk Storming using TestSphere:

1. Starting with a System Under Test.
2. Identifying the most relevant quality aspects.
3. Identifying risks for these aspects, **supported by socrative questions.**
4. Mitigate these risks with techniques.
5. Form an initial testing plan.

# Socrative Questioning

Socrative questions are a form of inquiry and discussion between individuals, based on asking and answering questions to stimulate critical thinking and to illuminate ideas.

# Examples of Socrative Questions used in the game<sup>®</sup>

- ▶ How does the system verify and ensure that the data processed is current and accurate?
- ▶ In what ways does the system maintain the confidentiality and integrity of personal data?
- ▶ Are there any performance benchmarks or metrics that the system is expected to meet?
- ▶ What are the disaster recovery and business continuity plans for the system?

# Wheel of socrative questions

## Wheel of Socrative questions

This app is part of a serious game on software testing. For more information, visit the [GitHub repository](#).



"Spin" the Wheel

How would a substantial shift in the data patterns impact the strategic use of the system?

# Pilot Study & Results

- ▶ We did a pilot study with four sessions with Bachelor and Master CS students of OU and NHL Stenden.
- ▶ Improvements observed in students' testing strategies.
- ▶ Students feel more secure about their tests.

# Pilot Study & Results



Figure: Students playing the game

# Future Work

- ▶ Further develop game mechanics.
- ▶ Validate and expand the socratic questions.
- ▶ Trials with students in different educational contexts.
- ▶ Publish the game (under CC license).

# Thanks for your attention

- ▶ Software Testing is important.
- ▶ We want students' to use an approach based on empiricism more often.
- ▶ Gamification can support this in multiple educational contexts.
- ▶ Abductive reasoning is the basis for didactics of software testing.
- ▶ We are developing a game with socrative questioning build in.
- ▶ We did a pilot to gain insights.
- ▶ Game mechanics need to be further developed.



# Acknowledgements

This work was funded by the ENACTEST – European innovation alliance for testing education (ERASMUS+ Project number 101055874, 2022-2025).

# References I

- [1] A. T. Sharon Harding, "A Leap Year Glitch Broke Self-Pay Gas Station Pumps Across New Zealand," *WIRED*, Mar. 2024.
- [2] V. Garousi, A. Rainer, P. Lauvås Jr and A. Arcuri, "Software-testing education: A systematic literature mapping," *Journal of Systems and Software*, vol. 165, p. 110570, 2020.
- [3] L. P. Scatalon, R. E. Garcia and E. F. Barbosa, "Teaching practices of software testing in programming education," in *Frontiers in Education Conference (FIE)*, IEEE, 2020, pp. 1–9.
- [4] N. Doorn, T. E. Vos and B. Marín, "Towards understanding students' sensemaking of test case design," *Data & Knowledge Engineering*, p. 102199, 2023.
- [5] Contributors to Wikimedia projects, *Abductive reasoning - Wikipedia*, [Online; accessed 2. Mar. 2024], Feb. 2024.
- [6] P. Rodrigues, M. Souza and E. Figueiredo, "Games and gamification in software engineering education: A survey with educators," in *Frontiers in Education Conference (FIE)*, IEEE, 2018, pp. 1–9.
- [7] M. A. Kuhail, A. ElSayary, S. Farooq and A. Alghamdi, "Exploring immersive learning experiences: A survey," *Informatics*, vol. 9, no. 4, 2022.
- [8] A. Hirsh, C. Nilhom, H. Roman, E. Forsberg and D. Sundberg, "Reviews of teaching methods – which fundamental issues are identified?" *Education Inquiry*, vol. 13, no. 1, pp. 1–20, 2022.
- [9] B. S. Tan and K. S. Chong, "Unlocking the potential of game-based learning for soft skills development: A comprehensive review," *Journal of ICT in Education*, vol. 10, no. 2, pp. 29–54, Dec. 2023.
- [10] L. N. Paschoal, L. F. Turci, T. U. Conte and S. R. S. Souza, "Towards a conversational agent to support the software testing education," in *XXXIII Brazilian Symposium on Software Engineering*, ser. SBES '19, Salvador, Brazil: ACM, 2019, pp. 57–66.
- [11] M. Maarek, L. McGregor, S. Louchart and R. McMenemy, "How could serious games support secure programming? designing a study replication and intervention," in *European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, 2019, pp. 139–148.
- [12] E. T. S. Masson, A. T. S. Calazans, I. N. Bandeira, G. R. S. Silva and E. D. Canedo, "Scrum in practice: City reconstruction as a pedagogical game challenge," in *XXII Brazilian Symposium on Software Quality*, ser. SBQS '23, Brasília, Brazil: ACM, 2023, pp. 321–331.