Design of a Serious Game on Exploratory Software Testing to Improve Student Engagement

April 2025

Niels Doorn, Tanja E. J. Vos, Beatriz Marín

Open Universiteit, NHL Stenden, Universitat Politècnica de València

niels.doorn@{ou.nl, nhlstenden.com} research.nielsdoorn.nl 0000-0002-0680-4443

nielsdoorn @niels76@mastodon.online

Outline

1. About me
2. Introduction
3. Theoretical Foundation
4. Game Design
5. Educational Integration
6. Conclusion
7. Q&A

About Me

- PhD student on Software Testing in CS education at Open Universiteit
- Team leader / Lecturer at NHL Stenden University of Applied Sciences
- Interested in software testing, education, and games

The Problems with Software Testing in CS Education

- Students often follow a rationalist testing paradigm [Doorn et al., 2021]
- This limits exploration and context awareness
- Exploratory testing based on empiricism is generally under-represented
- Students are not motivated to test their software

Our Position

- Use serious games to support sensemaking in testing
- Integrate software testing tours and Socratic questioning
- Foster reflective, inquiry-based learning

Design of a Serious Game on Exploratory Software Testing to Improve Student Engagement

Niels Doorn[1,2], Tanja E. J. Vos[1,3] and Beatriz Marín[3]
[1]*Open Universiteit, The Netherlands*
[2]*NHL Stenden University of Applied Sciences, The Netherlands*
[3]*Universitat Politècnica de València, Spain*

Keywords: Software Testing Education, Exploratory Testing, Game Based Learning.

Abstract: Teaching software testing in computer science education faces challenges due to its abstract nature and students' focus on approaches using paradigms based on rationalism. Exploratory testing, which uses a paradigm based on empiricism and employs reflective learning, is under-represented in computer science curricula. To address this gap, game-based learning presents promising approaches to enhance engagement and foster critical thinking in software testing education. This position paper presents the design of a serious game to support the teaching of exploratory software testing to improve the students' engagement. The game integrates software testing tours and uses Socratic questioning as scaffolding to promote deeper reflection-in-action, allowing students to experience hands-on learning in software testing. Using a mapping review, this study identifies the most effective gamification techniques for software testing education and principles of Socratic questioning. Based on these findings, we designed a game that focusses on exploratory testing scenarios, where players follow a tour-based test strategy on a system under test.

1 INTRODUCTION

Software testing is a crucial component of the software development life cycle and a highly valued skill in the industry. However, integrating software testing effectively into Computer Science curricula has been a challenge for educators (Garousi et al., 2020)(Scatalon et al., 2020).

Other studies investigated how students approach testing, revealing that many adopt the so-called 'developer approach' rooted in a design paradigm based on rationalism (Doorn et al., 2021)(Doorn et al., 2023). This approach focusses on algorithmic problem solving and structured planning, often leading to incomplete testing practices. This approach lacks exploration and context awareness, which is essential to gain insight into software quality. To address this, a paradigm shift based on empiricism is proposed that encourages experimentation, asking questions, and critical thinking (Doorn et al., 2021)(Doorn et al., 2023).

In this paper, we state our position that the sensemaking of students in learning testing within an empiricism-based paradigm can be effectively enhanced and supported by employing *software testing tours*, scaffolded by *Socratic questioning*, through the *serious game* we propose.

Software Testing tours (Bolton, 2009)(Kaner et al., 1993) are testing heuristics that use metaphorical "tours" to guide testers through different areas of an application, helping them detect defects, improve usability, and identify edge cases. Each tour serves as a specific approach or perspective for examining the software, such as concentrating on its features, data, configuration, or user behaviour. For instance, the Feature Tour is designed to help testers become familiar with the application's primary features, whereas the Complexity Tour delves into the most complicated portions of the system where defects are prone to occur. These tours motivate testers to think analytically, systematically alter inputs and conditions, and investigate areas that might be neglected in other testing methods. Testing tours prove to be highly effective in exploratory testing due to their focus on creativity, flexibility, and thorough evaluation of the software.

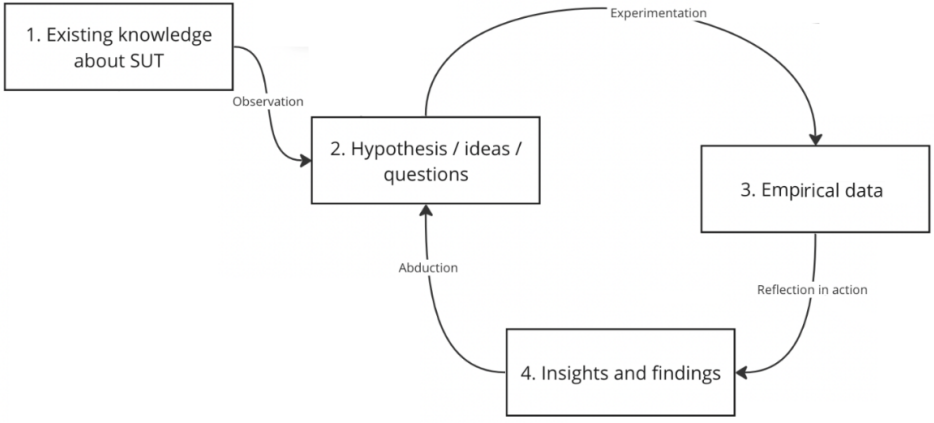
Socratic questioning (Paul and Elder, 2019) is a pedagogical method that fosters critical thinking, reflective inquiry, and problem solving by challenging assumptions and encouraging deeper analysis. In software testing, it complements empirical testing

https://orcid.org/0000-0002-0680-4443
https://orcid.org/0000-0002-6003-9113
https://orcid.org/0000-0001-8025-0023

Key Concepts

- **Sensemaking**: constructing meaning through reflection [Odden and Russ, 2019]
- **Socratic Questioning**: challenges assumptions [Paul and Elder, 2019]
- **Software Testing Tours**: structured exploratory strategies [Bolton, 2009]

The Sensemaking Cycle



Types of Socratic Questions

Type	Purpose	Example Questions
Clarification	Understand meaning and context	*"Can you elaborate on that?"* *"What do you mean by...?"*
Probe Assumptions	Reveal underlying beliefs	*"What are you assuming?"* *"Why do you think that?"*
Probe Reasons & Evidence	Evaluate reasoning and support	*"What evidence supports this?"* *"Is this always the case?"*
Viewpoints & Perspectives	Explore different angles	*"What is an alternative?"* *"How might others see it?"*
Implications & Consequences	Examine logical outcomes	*"What are the consequences?"* *"What would happen if...?"*
Question the Question	Reflect on the question itself	*"Why is this question important?"* *"What does this ask us to consider?"*

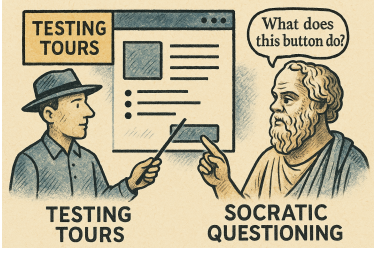
Software Testing Tours

- **Feature Tour**: Focus on specific features
- **Data Tour**: Explore data handling and storage
- **Back Alley Tour**: Investigate less obvious paths
- **Collector Tour**: Gather and analyze outputs
- **Saboteur Tour**: Test system resilience to changes

Testing Tours + Socratic Questions

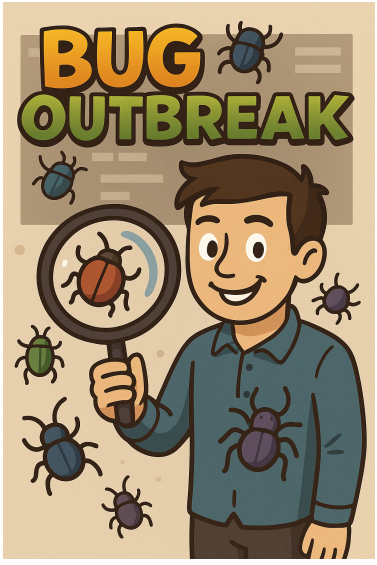
Examples:

- Feature Tour: *What is the primary purpose of this feature?*
- Data Tour: *What data is the system expected to handle?*
- Back Alley Tour: *What pathways might be overlooked?*
- Collector Tour: *Is the GUI output consistent througout the app?*
- Saboteur tour: *What are the implications of changes to authorisations?*



Game Overview

- Cooperative and competitive elements
- Assigned tours guide player actions
- Scoring system for feedback and motivation
- Risk of failure encourages thorough testing



Gameplay Scenario

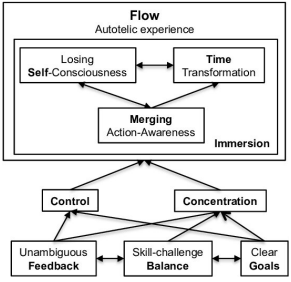
- System Under Test: can be a relevant project in the context of education, an open source project, or an example project
- 2-5 Players: Feature, Data, Back Alley Tours
- Socratic questioning lead to hypotheses and observations
- Players score points for each hypothesis and observations
- Players can lose points for incorrect assumptions
- Reflection follows collaborative analysis of the results

Embedding in Education

- Can be used in tutorials, workshops, group work
- Can be developed in digital, physical, or hybrid versions
- Supports formative, diagnostic, and self-assessment [Black and Wiliam, 1998]

Evaluation and Future Work

- Use SUS and GAMEX for lecturer feedback [Brooke, 1996, IJsselsteijn et al., 2013]
- Measure autotelic experiences [Sillaots and Jesmin, 2016]
- Plan real-world evaluations in courses



Conclusion

- Game-based learning aligns with empiricism
- Testing tours + Socratic questioning = deeper learning
- BugOutbreak is a game for teaching exploratory testing with more engagement

Ask Me Anything

Discussion Starters:

- How does this scale for large classrooms?
- Could it work in non-CS disciplines?
- What platform would be ideal for the digital version?
- How does this change the student mindset toward testing?

References I

Black, P. and Wiliam, D. (1998).
Inside the black box: Raising standards through classroom assessment.
Kings College London School of Education London.

Bolton, M. (2009).
Of testing tours and dashboards.
Accessed: 2024-10-03.

Brooke, J. (1996).
Sus: A quick and dirty usability scale.
In *Usability evaluation in industry*, pages 189–194. Taylor & Francis.

Doorn, N., Vos, T., Marín, B., Passier, H., Bijlsma, L., and Cacace, S. (2021).
Exploring students' sensemaking of test case design. an initial study.
In *21st Int. Conference on Software Quality, Reliability and Security Companion*, pages 1069–1078. IEEE.

References II

- IJsselsteijn, W., de Kort, Y., and Poels, K. (2013).
  The game experience questionnaire.
  Technical report, Technische Universiteit Eindhoven.
- Odden, T. and Russ, R. (2019).
  Defining sensemaking: Bringing clarity to a fragmented theoretical construct.
  Science Education, 103(1):187–205.
- Paul, R. and Elder, L. (2019).
  The thinker's guide to Socratic questioning.
  Rowman & Littlefield.
- Sillaots, M. and Jesmin, T. (2016).
  Multiple regression analysis: Refinement of the model of flow.
  In *10th European Conference on Games Based Learning*, pages 606–616.