**“Serious game to teach software testing”**

# Work Plan

* Analyze the different teaching techniques.
* Investigate novel techniques for teaching software testing.
* Develop a capsule for teaching testing, taking into account the learning needs of students and the industry's needs.
* Perform some empirical studies to measure the impact of the capsule in teaching software testing.

# Research Questions

1. How has software testing education been designed and employed over time?

*To learn how it’s been employed and justify why the capsule is needed.*

* 1. What challenges can be discerned from this history?

*Whatever challenges existed then might still exist and might also need to be addressed now.*

* 1. What solutions or propositions, if any, are there?

*Whatever solutions worked before, may also inform the capsule now.*

1. Which tools have been developed for or used in software testing education?

*To learn from what has been made before and, if needed, improve upon it.*

* 1. What key features or characteristics do these tools possess?

*To know what to look for in selecting a game-making platform or framework.*

* 1. Which pedagogical techniques or patterns do the educational parts of these tools follow?

*To assess what makes sense for the capsule to follow, what’s tried and true and how to create a positive impact in the teaching of software engineering.*

**Keywords:** Software testing, Software testing education, Serious games, Education, Software engineering, Gamification, Game-based learning, Educational games

# Requirements

## Platform Requirements

* The platform must consider the board game part and the interaction/validation component of the capsule.
  + In informal terms: a lot of code is needed to allow for the features that are at the core of the capsule, and then needs to run code on top of those for validating input;

## General Requirements

* The architecture of the capsule must allow for easy expansion.
* The capsule is intended to allow teaching: Code reading, Boolean logic, Game logic (specific to board game) and, as the main goal, White box testing;

## Game Mechanics Requirements

### Student User Perspective

* + The capsule should work as a sandbox-type game, where board games are used for teaching.
    - The teaching part comes from interaction with the game, allowing a student to perform certain programmed tasks to learn concepts.
    - The *sandbox* component requires a **free environment for interaction** – while rules should be enforced so the student can know when they’ve made a mistake, they should also be allowed to make mistakes.
    - The example game to consider is Checkers.
  + *Challenges:* students are presented with challenges to stimulate their knowledge of code reading, Boolean logic, game logic and introduce white-box testing.
    - Each challenge comprises a task or set of tasks to perform, most often based on (a) snippet(s) of code – this code is shown to the user as a form of insight into what interactions they should perform.
    - Within a code snippet, there will be a highlighted target.
      * Targets may be a specific condition, line of code, et cetera.
      * The main objective of a challenge is to ***hit*** its target.
    - Users are given opportunities to hit the target via interacting with the capsule – this means messing with board game pieces and performing game-valid moves to create the conditions required for the target to be hit, after which, the code snippet is executed to effectively test if that’s happened;
      * **Example:** if the target is a specific line of code, and that line of code is inside a specific *If* block, the user is expected to create the conditions for the *If* block’s condition to pass, thus, when the code is executed, the target is hit;
  + *Gamification:* passing challenges should be made into a gamified experience.
    - Passing challenges should grant a **score or grade** (akin to level ranks, as seen in Sonic games) that reflect how well a student performed.
      * *Not passing challenges* may also introduce “punishment” mechanics via:
        + Limited attempts (akin to “lives” in Sonic games, for example).
        + Penalizations (failing a set number of times begets a worse score).
    - Different levels of difficulty, via:
      * *Harder* challenges to pass (challenges specifically made to be… well, challenging).
      * Restrictions on interaction, such as:
        + Time (“You only have 15 seconds to pass this challenge”);
        + Game rules (“Create the necessary conditions by playing Checkers against yourself – the game rules will be enforced!”);
        + Anti-targets (“You have to hit this line of code, but *this* line of code CANNOT be executed before it!”);
        + …

### Teacher User Perspective

* Teacher users will be able to design the challenges students perform;
  + A challenge needs:
    - The game board.
    - The code snippets.
    - The targets.
  + As expected, the game board isn’t defined by the teacher, but the opportunity to create certain scenarios (for higher difficulty) would be interesting, such as:
    - Removing pieces (as if the challenge starts in the middle of a game).
    - Allowing for game-breaking moves (out of bounds, stacking pieces, etc…).
  + The origin of code snippets should come from the associated game’s code.
  + The target of a code snippet may be: a line of code, a specific logical condition, et cetera.
* Teachers should be able to check how a certain challenge has been received, via metrics (how many students have passed it, how many students have failed it, et cetera);

Made a little use case diagram below to better show the users’ flow within the capsule.

