GANFLEW, Teacher View: Validation Experiment

This document serves as a script/guide to a validation experiment of GAMFLEW, a new serious game to teach white-box testing techiniques. Please follow the instructions below to properly participate in the experiment.

1. Introduction

This experiment happens at your own pace, but, as a suggestion, one should only need at maximum 60 minutes to complete it.
There are two parts to the experiment:

1. **Part 1: Playing The Game** - to allow you to understand the game mechanics, you are to complete a challenge (as a player/student) from a provided small list of suggested challenges.
2. **Part 2: Teacher View** - after playing the game, you will be ready to take the role of a teacher and follow a short tutorial on how to create your own challenges for the game. After this, there is a questionnaire.
    1. **Part 2.1: Tutorial** - a step-by-step guide on how to create a challenge and overview of other features.
    2. **[OPTIONAL] Part 2.2: Create Your Own Challenge** - you can create your own challenge.
    3. **Part 2.3: Post-Questionnaire** - a set of questions to evaluate the game and the challenge creation process.

2. Part 1: Playing The Game

The game is available at this link.

To play the game, it is required to go through authentication. For your purposes, you can simply use the following account:

- Username: `professor`
- Password: `password`

However, if you want to create your own account, you can do so by clicking on the "Register" link on the login page. After creating it, you must validate your account to be able to use it. You can do so using the account above (*Login -> Validate Administrators -> Find Your User -> Click "Validate"*).

After authentication, you will be redirected to the game's main page. Here, click on the "Single Player" button.

You will then be in the Challenges page, where all existing challenges are listed. 99 challenges are available, but, to make it easier, we selected the following challenges:

1. Challenge 1.1 (Statement coverage, Very Easy)
2. Challenge 1.2 (Decision coverage, Easy)
3. Challenge 1.3 (Modified condition/decision coverage, Hard)
4. Challenge 2.6 (Condition/decision coverage, Normal)
5. Challenge 2.8 (Condition coverage, Easy)

For the purpose of this experiment, **we only ask you to complete one of the challenges**. After this, one should be familiar enough with the game mechanics to proceed to Part 2.

How to Play

The game is built on a simple idea: players get a challenge, containing a code and specific coverage objective. The player must then "write" test cases to achieve the coverage objective.

To "write" test cases, there is a board based on the game Checkers to interact with. For each attempt, one is expected to comment on what they tried and why they think it worked.

To keep this script short, please check the "How to Play" page (Login -> How to Play) for a detailed explanation of the controls (or click here).

3. Part 2: Teacher View

After playing the game, you are ready to take the role of a Teacher. This part is divided into three sections: Tutorial, Create Your Own Challenge, and Post-Questionnaire.

Know that there is a specific "Teacher View" tab on the "How to Play" page, which may be useful before this part. It's also a quick read!

3.1 Tutorial

3.1.1 The Challenge Manager Pages

This tutorial intends to guide you through the process of creating a challenge. To start, go to the "Challenge Manager" page. For this part, we invite you to open the same challenge you beat, to see how it came to be and go through this section while checking it.

To do that, one just has to go to an existing challenge and click the "Edit" button. Obviously, to create a new challenge, one should click the "Create Challenge" button.

After opening a challenge, you will see a form with several fields. The first two are the initial board state and the code file associated with the challenge. These should be familiar to you, as they are what you are shown when opening a challenge.

While in this page you can only select existing entries for the initial pair of fields, you can create new ones by going to the "Challenge Content Creator" (click here). There, you can create new board states to initialize challenges and new code files to associate with them. They're pretty self-explanatory, only requiring a name and their content (check "How to Play" for more details). We will be using the page later in the tutorial.

After the first two fields, there is a pretty standard form: this is where everything is filled up (challenge title, its objective, the hint shown to struggling students, coverage type, difficulty). There is, also, a field for the number of conditions to consider when generating the truth table (this only for coverage types that warrant it).

After this, it's important to note the Assertions and Tests, below the group of purple callouts. The purple callouts explain how to write and understand what's written in these expressions. Giving them a quick overlook is better than explaining them here. With that, it is possible to understand how the challenge passed before is evaluated.

While both assertions and tests are simple boolean expressions, they have one thing that distinguishes them: **if a player passes a test, the assertion will have passed as well**. This means that failing an assertion means that the challenge is automatically failed - and, because **assertions are applied before every test**, it is important to write them properly. They serve as indirect shortcuts to submission evaluation, but are completely optional.

Take a moment to understand the expressions from your challenge - that is the format you will be using to create your own challenges.

NOTE: For challenges with more than one test, know that it's not required for the first submitted board state to clear the first test and so on - players can submit test cases in whichever order they please! What matters is that, when finishing evaluation, all tests were cleared at least once.

3.1.2 The Challenge Content Creator

Starting your challenge creation process, go to the "Challenge Content Creator" page.

To keep this simple, we are providing you a simple code file.

Use the "Code File" tab to create a code file with the following code (you write on the right text area, and see the appearance on the left):

```
// This is a simple function to move a piece in a checkers-like game.
// start: object with initial (x, y) coordinates of the piece that was moved.
// destination: object with (x, y) coordinates to where piece was moved
function makeAMove(start, destination) {
    if (Math.abs(start.x - destination.x) === 1 && Math.abs(start.y -
destination.y) === 1) {
        return true;
    } else {
        return false;
    }
}
```

Pretty simple stuff, right? We recommend you to give the code file the name *"Code File 0: Make A Move"*. However, you can really give it any name, as long as you can identify it.

Now, for the board state, you can really make any sort of board state you wish, as long as you submit it and give it an identifiable name for you to select in the "Challenge Creator". When making the existing challenges, we tried to make initial board states that either created little work for the players, or, somehow, made the challenge more interesting.

If you want a suggestion: just add one piece (red or blue) to the board, as that is all that is needed for the challenge!

After creating these, head on to the "Challenge Manager" page to create a brand new challenge.

3.1.3 Creating a Challenge

The idea is to create a challenge for one of the return lines. The first thing to do is selecting the code file and board state you created in the previous step.

For this challenge, we will be using the `return true;` line. As such, fill the form with the following information:

- Challenge Title: "Challenge 0.0: A Simple Move" **[IF THERE IS ALREADY A CHALLENGE 0.0, USE 0.1, AND SO ON - NEW CODE FILES WILL BE LISTED AT THE BOTTOM!]**

- Objective: "Statement coverage of line 6."
- Coverage Type: "Statement"
- Difficulty: "Very Easy"
- Score: 100 points (but you can give it any score you wish)
- Hint: "If the difference between the respective X and Y in start and destination is the same, what does that mean for the piece's movement? Think geometrically. Remember that only the last movement is considered."

After this, it's time for assertions and tests We will make one of each.

From the `if` statement right above it, one can see the function returns True if the piece is moved by 1 spot diagonally. As such, the test should check if the condition yields True. Using the "language" expressed in the purple callouts above the expression maker, we are able to see any logged interaction with the board - including movements. For the purpose here, we really just need something like the following:

```
Math.abs(board.log[board.log.length - 1].start.x - board.log[board.log.length - 1].destination.x) == 1 && Math.abs(board.log[board.log.length - 1].start.y - board.log[board.log.length - 1].destination.y) == 1
```

This is pretty standard: we are just checking if the last movement (as stated in the provided hint) is a diagonal movement of 1. We use `Math.abs()` to make sure the result is always positive. But we can make this shorter by using the provided abbreviations:

```
Math.abs(last_start.x - last_destination.x) == 1 && Math.abs(last_start.y - last_destination.y) == 1
```

This is the exact same expression logically, but it is now more readable.

We are missing the assertion, now. The assertion should be one that is always True when the test expression is True. In this case, we can use the following:

```
board.log.length > 0
```

With this, we are making sure that there is at least one interaction logged in the board. Without any interactions, there can't be movements, meaning it's impossible for the test to pass. This is a very simple assertion, but it also prevents false positives!

With both of these in respective boxes ("A" for Assertion, "T" for Test), the Challenge Creator should already have updated everything needed. This means the abstract language will be "translated" into runnable JavaScript code considering what the Teacher has put into it. Pretty neat, right?

So all that is left is passing the challenge. Because there is nothing like trying what we did to make sure it works as intended!

At this point, it is a bit repetitive, but by moving a piece one spot diagonally and submitting, the challenge will pass.

This unlocks the "Submit Challenge" button, and when you press it, the challenge is submitted! From this point on, the challenge is available for all players to play.

Go back to Single Player mode and play it yourself! After submitting an attempt, move on to the next step.

3.1.4 Check Solutions

After submitting the challenge, you can check the solutions submitted by players.

To do so, go to the "Check User Submissions" page. Here, select your challenge and your submission, which should be easy to track down (it uses your username).

3.1.5 Check Leaderboard

If you want to see how players are doing in your challenge, you can check the leaderboard. This leaderboard keeps all user stats, which is not the case for the leaderboard players have access to.

Go to the "Leaderboard" page and look for your own username!

3.2 [OPTIONAL] Create Your Own Challenge(s)

If you wish to create your own challenge, you can do so by following the steps above. You can create any challenge you wish, but we recommend you create a challenge that is simple and easy to understand, as the game is meant to be a teaching tool.

While there is a learning curve to using the challenge manager, the game is prepared with a lot of callouts and help to guide newcomers in the process. We believe that, with minimum JavaScript experience, it's very easy to create challenges.

We remind that the experiment should take at maximum 60 minutes, so if you wish to create an original challenge, please be mindful of the time! If you do create multiple challenges and or find something that is not working correctly in your exploration, please let us know in the post-questionnaire.

3.3 Post-Questionnaire

After playing the game and creating a challenge, please fill out the post-questionnaire. It is available on this link.

4. Conclusion

Thank you so much for participating in this experiment. Your feedback is very important to us, as it will help us improve the game and the challenge creation process.

For any further contacts, please reach out to us!