Software Engineering Cowards – SPRINT 2 DELIVERABLE

Brian Ashworth (Scrum Master)– RedSoxFan

John Johnson – jdj20

Edwin Mellett – WynWinz

Aaron Tamenne – atamenne

https://github.com/RedSoxFan/2174\_CS1530\_SoftwareEngineeringCowards

(15 FEB 2017)

SPRINT 2 ACCOMPLISHMENTS

This sprint a lot of the main functionality of the game was completed. This includes aspects such as clicking to move pieces on the board, the white and black teams alternating turns, and starting a new game. Some additional functionality we included this sprint that wasn’t required was ensuring that only the king can occupy the center square and any of the four corners and the ability to save a game state (but not load yet, we will do that in the next sprint).

A disagreement that arose was that one team member was very for using tabs instead of spaces, but ultimately got outvoted and had to deal with using spaces instead. This debate turned out to not matter very much since the text editors we are using can automatically convert tabs to two spaces. Other than that we continued to use the style checker to ensure that all of our code is similar in style and easy to read by everyone. Some teammates were frustrated by the style checker at first, since it enforces a slightly different style than what they were used to. They seemed to acclimate to the change relatively quickly though. We did not really have to resolve any disagreements other than deciding whether to use tabs or spaces for indentation.

For some of us this is the first time using JUnit. It was a new experience to spend longer writing tests than writing the actual code. This also led to the insight that developing quality code which is also easily testable is a challenge. As we discussed in class many of us are used to more of a “cowboy coding” style for programming assignments, so this project requires much more attention to detail.

We have not changed our development process except we are now using pull requests; we continue to communicate via a group text for major decisions. If this method proves to be too much of a burden in the future we will probably switch to something like Slack for team communication. We have not really run into any major challenges with the coding/testing yet because most of the user stories implemented thus far have not been overly complex. We decided to not designate a single person as a code reviewer and this worked well. Each person reviewed and confirmed at least one pull request. If we continue to have success spreading responsibilities such as this, this project will continue to go smoothly.

USER STORIES COMPLETED

**Story Points: 8**

As a player

I want to be able to use the mouse to click and move the pieces

So that I can play Hnefatafl

**Story Points: 2**

As a player

I want to be able to start a new game or quit the game whenever I choose

So that I have more flexibility in what I do with the game

**Story Points: 2**

As a player

I want there to be two sides: a king with his defends and attackers

So that I can play Hnefatafl against someone

**Story Points: 2**

As a player

I want there to be 12 defenders and 24 attackers

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want the board to be a grid with 11 rows and 11 columns of squares

So that the Copenhagen rules are obeyed

**Story Points: 2**

As a player

I want the initial board setup to be as follows (A:Attacker D:Defender K:King)

A A A A A

A

A D A

A D D D A

A A D D K D D A A

A D D D A

A D A

A

A A A A A

So that the Copenhagen rules are obeyed

**Story Points: 2**

As a player

I want the attacker's side to move first

So that the Copenhagen rules are obeyed

**Story Points: 2**

As a player

I want the game to progress with alternating moves between the two sides

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want to be able to move a piece on my side any number of vacant squares along a row or column

So that the Copenhagen rules are obeyed

**Story Points: 2**

As a player

I want the central square to be the throne and can only be occupied by the king

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want the king to be the only piece that can occupy any of the four corners

So that the Copenhagen rules are obeyed

**Story Points: 2**

As a player

I want any piece to be able to pass through the throne when it is empty

So that the Copenhagen rules are obeyed

**TOTAL VELOCITY: 36**

Note on our Velocity: I think we may have underestimated point values for these stories. Our velocity for this sprint was 36 whereas in an ideal world 4 developers could complete 64 points worth. We worked efficiently, but to implement these user stories with testing, good coding, and appropriate comments takes longer than we expected. We will take all of this into account when assigning point values to stories for the next sprint.

USER STORY DECISIONS

We decided on these user stories during our meeting time in class. We noted the requirements were to have at least basic functionality working by the end of this sprint. This influenced our user stories to focus on a board in the correct starting position, moving, switching turns between attacker/defender, etc. The decision on what user stories to implement was unanimous during our meeting. We also worked on another user story that cannot be listed as completed. We chose to start working on the save/load feature for games and we finished the save feature, but load has not yet been implemented. We chose this user story when one of our developers was deciding what to work on since the other 3 implemented the previously mentioned features. Save/load was the next logical user story since this will make testing easier as well. When the load feature is working, boards can be created with pieces in certain positions to test different types of captures more easily. Since we now have basic functionality working, it will be easier for individuals and our group as a whole to add new features in the sprints to come.

DEFECTS FOUND

**ID:** 1

**Title:** Master Compilation Failure

**Description:** A branch was merged into master that did not compile due to uncaught exceptions.

**Reproduction Steps:** Execute the following commands:

> git clone https://github.com/RedSoxFan/2174\_CS1530\_SoftwareEngineeringCowards/

> cd 2174\_CS1530\_SoftwareEngineeringCowards

> git checkout 2447996251a58f53544c38428fe8ee1f86c5034f

> gradle build

**Expected Results:** The project builds successfully

**Observed Results:** The project fails to compile

**Resolution:** A pull request was merged with the exceptions being caught to fix the issue.

**ID:** 2

**Title:** Save Failure Due To Missing Folder

**Description:** If the folder “saved\_games” is not manually created, throws an IOException.

**Reproductions Steps:**

1. Execute the following commands

> git clone https://github.com/RedSoxFan/2174\_CS1530\_SoftwareEngineeringCowards/

> cd 2174\_CS1530\_SoftwareEngineeringCowards

> git checkout 82deec4e8944ec3168a8c687d9278d236ba42881

> gradle run

2. Click the save button

3. Enter a file name

**Excepted Results:** The game saves under the directory “saved\_games” with the name entered

**Observed Results:** An IOException is thrown and the game is not saved

**Resolution:** A pull request was merged that created the folder that did not exist and also shows only graphical error messages on failure (ie. due to permissions or space).