Software Engineering Cowards – SPRINT 2 DELIVERABLE

Brian Ashworth – RedSoxFan

John Johnson (Scrum Master) – jdj20

Edwin Mellett – WynWinz

Aaron Tamenne – atamenne

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

(1 MAR 2017)

SPRINT 3 ACCOMPLISHMENTS

USER STORIES COMPLETED

**Story Points: 8**

As a player

I want to be able to save and load my Hnefatafl games

So that I can take a break in the middle of a game to resume later

**Story Points: 8**

As a player

I want the attackers to win if there is an attacker's piece on all four adjacent sides of the king

So that the Copenhagen rules are obeyed

**Story Points: 8**

As a player

I want the king and his defender's side to win if the king occupies any of the four corners

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want the game to end in a draw if there is 50 consecutive moves without a capture

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want a side to win if the opposing side moves a piece back and forth three times

So that the Copenhagen rules are obeyed

**Story Points: 8**

As a player

I want any piece other than the king to be captured if there is an opposing piece and/or hostile square on each side of that piece along a column or row

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want the throne to always be hostile to attackers

So that the Copenhagen rules are obeyed

**Story Points: 4**

As a player

I want the four corners to be hostile to any piece

So that the Copenhagen rules are obeyed

**TOTAL VELOCITY: 48**

USER STORY DECISIONS

We decided to complete these user stories based on current requirements for the sprint. We were tasked to focus first on completing checking moves are legal, allowing players to lose/win, and special squares being visually distinct. We already handled legal moves and special squares being distinct in the last sprint. Since we had these finished we had time to focus on other user stories. This includes checking for draws and determining if a side loses by moving a piece back and forth three times in a row. We have been working quickly, so we decided to split stories and accomplish extra during the past two sprints. This is why we decided to implement save/load functionality as well. It seemed like the next logical step and may be due for the next sprint, so we will already have it done.

DEFECTS FOUND

**ID:** 1

**Title:** King Capture Failure

**Description:** When the king is captured it remains on the board.

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

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

> cd 2174\_CS1530\_SoftwareEngineeringCowards

> git checkout 2447996251a58f53544c38428fe8ee1f86c5034f

> gradle build

> gradle run

> Move pieces until king is surrounded by attackers

**Expected Results:** Attackers win, the game is over, and the king is removed from the board

**Observed Results:** Attackers win, the game is over, but the king is still on the board

**Resolution:** Add code to update the state of the king’s square from KING to EMPTY.

**ID:** 2

**Title:** Save Game Failure

**Description:** A game can still be saved after the game is over.

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

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

> cd 2174\_CS1530\_SoftwareEngineeringCowards

> git checkout 2447996251a58f53544c38428fe8ee1f86c5034f

> gradle build

> gradle run

> Move pieces until game is over

> Save game

**Expected Results:** User notified they cannot save a game once it is over.

**Observed Results:** Save game feature works when a game is over.

**Resolution:** Added code so user is notified “Cannot save a board in a game over state.”