

Software Test Report

for

Chess

Version 1.0

|  |  |  |
| --- | --- | --- |
| Daniel Maida | 11441203 | [daniel.maida@wsu.edu](mailto:daniel.maida@wsu.edu) |
| Nazar Stelmakh | 11441299 | [nazar.stelmakh@wsu.edu](mailto:nazar.stelmakh@wsu.edu) |
| Steven Call | 11451145 | [steven.call@wsu.edu](mailto:steven.call@wsu.edu) |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | Date: Nov 28, 2016 |
|  |  |
|  |  |

Contents

1- Software test Plan 1-2

2- Static v & v:

2.1- Tested items and Checklist 3

2.2- Inspection Results and Follow-up 4-5

3- Dynamic v & v:

3.1- unit Testing 6

3.2- Interface testing 7

3.3- System Testing 7

3.4- non-functional Requirements Testing 7

4- Appendix b: Group Log 8

# SOFTWARE TEST PLAN

**1.1 The Testing Process:** We will be creating a test checklist for our inspection group to follow while testing our program and variables. There will be two types of testing that will occur, static testing being done by the inspection group and Dynamic where test code is created to test specific situations among our code. The inspection group will be given the game and will run it trying to test the implementation and functionality of the game design as well as the options buttons within our program.

**1.2 Requirements traceability:** The requirements that our program is needing to meet will be based on the expectations of the game rules and standards in a common game of chess. We will have basic requirements to test such as realizing a winner, realizing when a player is in check, what moves can get them out of check, not playing pieces out of bounds (off the board), etc.

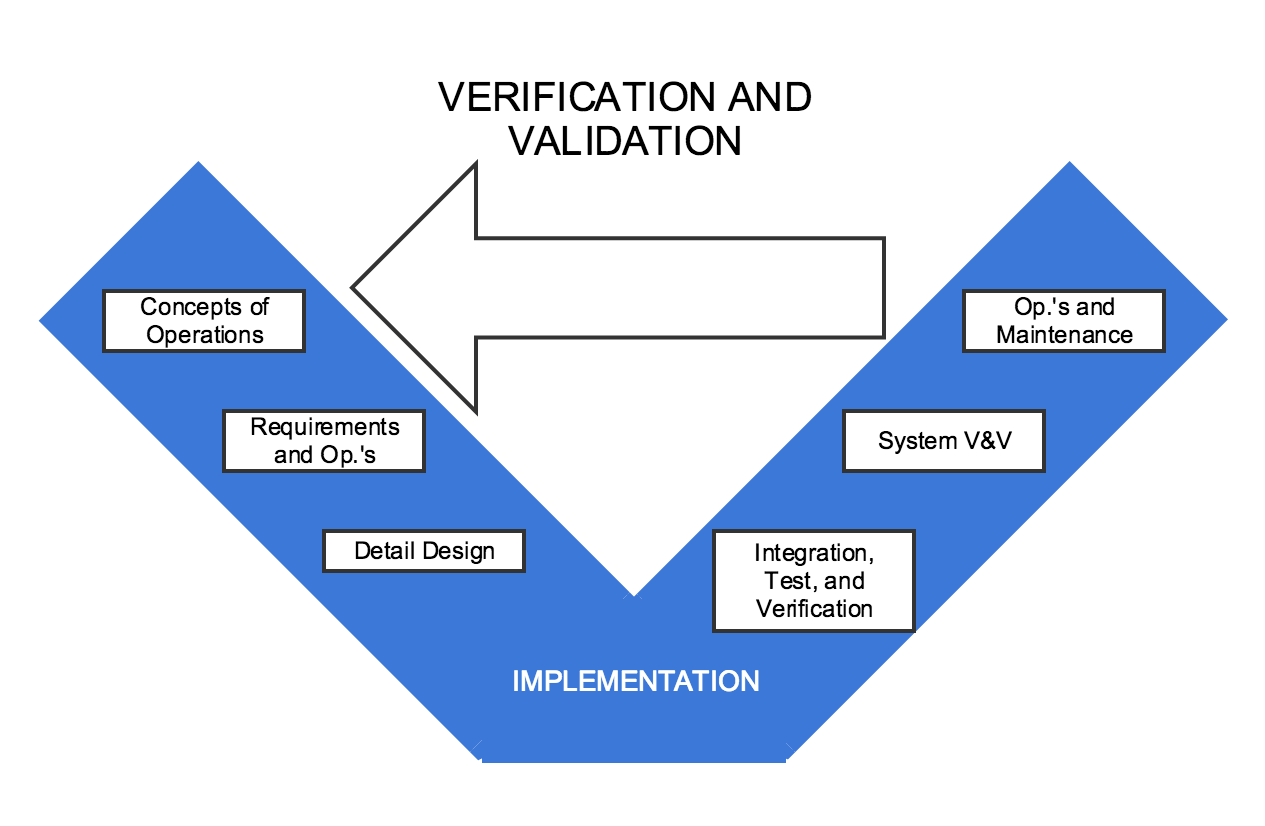
**1.3 Tested items:** The game will be tested by running the program and have the inspection group play the game. During the game, they will be trying to find exceptions and bugs that allow them to break the rules or where improper moves will break the game. Statically this will be the testing to ensure that our game design and functionality is working correctly.

**1.4 Testing schedule:** We will give the game to the inspection group and give them a few days to look over the code and mess with it to find the issues. The group will follow our checklist and keep track of issues they had so that we can record their testing results.

**1.5 Test recording procedures:** Our inspection group will be recording the results of their tests in a word document along, like that of a checklist. If issue are found the group will record the issue that they found as well as explain the scenario in which it happened and what they were trying to do while it happened. When we are running our coded test cases we will have the output of each test print whether it was successful and where the test failed in your code, this will be done using JUNIT testing.

**1.6 Hardware and software requirements:** Our program has been initialized and programmed on different computers such as mac OS X, and Windows 10. Our code is required to be accessible and deliverable to any interface or system and be functional. Our code is also required to be very user friendly to the point that the user does not need to know what is going on behind the scenes and can just use the program and not have issues with player restrictions. These will be tested within our inspection group for static testing and we will create test programs for Dynamic Testing.

**1.7 Constraints:** With this project the only constraint on who can test this program is that they have a basic understanding of the game of chess. Available moves are provided to the user but they will need to know how to win and some of the basic rules of the game to test.



# STATIC V & V

## Tested Items and Checklist

**2.1.1 Tested items:** We will be having the inspection group test each piece of the chess game as well as the game functions themselves. The items specifically that we will have tested are: Pawn, Rook, Knight, Castle, Queen, King, Undo, Reset, Change Theme, and exiting the board. These will be tested statically by playing the game and trying to do false moves or have the commands break the program.

**2.1.2 Checklist:** The checklist given will be what the inspection group will specifically be trying to target and verify that it is in working condition. While doing this, they will also be able to report any other bugs or issues they notice along the way of completing the checklist. The checklist is as follows:

1. **Static Testing:**
   1. Does the game gerate and load properly?
   2. Do all pieces only give you valid move options?
   3. Do the graphics appear legibly and clearly so that they are understandable?
   4. Does the Options drop done menu give you available options for game play?
      1. Do the options buttons correctly work within the drop down menu?
   5. Are the rules of chess implemented and create correct restrictions?
   6. Do the white pieces always initialize gameplay for the first move?
      1. Can you move the same color twice in a row?
      2. Does the game alternate players turns correctly?
   7. Does the game restrict moves when you are placed into check?
   8. When in check mate can any pieces be moved or are they locked into their location?
   9. When you change the theme of the board(in options menu), does the game play stay the same?
   10. When you change the theme of the board, does it still remember the correct person to make a move next?
   11. Does the program take too long between actions and events?
   12. Do the pieces ever have an option to go out of bounds(off the board)?
   13. Does a pawn have the option to change into any desired piece once reaching the other side of the board?
   14. Can the king and castle perform the “castling” move correctly?
   15. Can the pawns move two places forward, only on their first move?
   16. Can pieces of the same color attack each other?

# 2.2 RESULTS AND FOLLOW UP

**2.2.1 Inspection Results:**

1. Does the game gerate and load properly?  
   Yes
2. Do all pieces only give you valid move options?  
   Yes
3. Do the graphics appear legibly and clearly so that they are understandable?  
   Yes, the back ground color could offset pieces more.
4. Does the Options drop done menu give you available options for game play?  
   Yes
   * 1. Do the options buttons correctly work within the drop down menu?  
        Yes, AI isnt implemented yet so difficulty doesn’t change. Undo also works but is glitchy in some cases as if the piece didn’t undo.
5. Are the rules of chess implemented and create correct restrictions?  
   Yes
6. Do the white pieces always initialize gameplay for the first move?  
   Yes
   * 1. Can you move the same color twice in a row?  
        No, unless you use undo then move a different place.
     2. Does the game alternate players turns correctly?  
        Yes
7. Does the game restrict moves when you are placed into check?  
   Yes, can only do moves to move you out of check.
8. When in check mate can any pieces be moved or are they locked into their location?  
   They are locked showing that its game over.
9. When you change the theme of the board(in options menu), does the game play stay the same?  
   Yes, the game continues as if nothing happened.
10. When you change the theme of the board, does it still remember the correct person to make a move next?  
    Yes, the game correctly keeps playing.
11. Does the program take too long between actions and events?  
    No, the reaction time is very quick and it didn’t stall.
12. Do the pieces ever have an option to go out of bounds(off the board)?  
    No, they are never allowed to go out.
13. Does a pawn have the option to change into any desired piece once reaching the other side of the board?  
    Yes, a menu pops up.
14. Can the king and castle perform the “castling” move correctly?  
    Yes
15. Can the pawns move two places forward, only on their first move?  
    Yes
16. Can pieces of the same color attack each other?  
    No, they also cant jump over or pass through. Correctly working.

**2.2.2 Rework and Follow Up:** Going off our instructor and peer feedback from testing day as well as our inspection groups report we could revamp our program to make it more consistent with the requirements of our prospective users. We fixed our buggy undo button as well as implemented clearer and better themes to allow for an easier viewing experience. We continued to develop our program with these considerations and new requirements in mind while furthering the usability of this program.   
The next step is to add and include more features for the user so that they have a more enjoyable experience of the game play itself. Next we will create our unit tests to test the functionality of our code on a smaller level to make sure that we will not be experiencing null pointer exceptions, out of bounds exceptions, or non-declared variables. These unit tests will check our efficiency and correct rules that we implement in many different situations.

# 3 DYNAMIC V & V

## 3.1 Unit Testing

All unit tests were written in Java’s JUnit testing framework.

test\_Initialization():

The purpose of this test is test that the Controller and the game board are properly initialized.

Testing initialization.....pass

Result;

test\_PieceInitialization()

The purpose of this test case is to test the initialization of every chess piece in the game board. And that every abstract piece is correctly placed on the game board.

Result: Testing Piece initialization.....pass

test\_Bishop()

The purpose of this test is to test the Bishop’s allowable moves. Does the Bishop piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing bishop allowable moves.....pass

test\_King()

The purpose of this test is to test the King’s allowable moves. Does the King piece correcly move within the game board, and is it correcly placed in the new game board position.

Result: Testing King allowable moves.....pass

test\_Knight()

The purpose of this test is to test the Knight’s allowable moves. Does the Knight piece correcly move within the game board, and is it correcly placed in the new game board position.

Result: Testing Knight allowable moves.....pass

test\_Pawn()

The purpose of this test is to test the Pawn’s allowable moves. Does the Bishop piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing Pawn allowable moves.....pass

test\_Queen()

The purpose of this test is to test the Queen’s allowable moves. Does the Queen piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing Queen allowable moves.....pass

test\_Rook()

The purpose of this test is to test the Rook’s allowable moves. Does the Rook piece correctly move within the game board, and is it correctly placed in the new game board position.

Result: Testing Rook allowable moves.....pass

## 3.2 Interface

test\_Interface()

The purpose of this test case is to test the initialization of the game board and the controller.

test\_Buttons()

The purpose of this test case is to initialize the controller and test the initialization of the buttons in the interface.

## 3.3 System Testing

During testing day we got feedback from external testers on different aspect of our software. We solicited feedback mainly on user experience and how are software representation of chess represented the game of chess.

The main feedback back from testing was that our software was missing components of the use cases and some elements of chess. One especific use case for chess that we were missing was castling. Rework was done to implement the feature.

Another main component that was missing is the change theme function of GUI. During testing day the external testers provided feedback about the themes implementation. The external testers where not able to change the themes of the GUI and there only one theme that was available at the time. Rework was done to implement the change feature. The user now has the options to toggle between two default themes.

## 3.4 Non-functional Requirement Testing

REQ 1.0 - The A.I. engine should process a move in a reasonable amount of time:

This requirement has not yet been met due to the fact of our A.I. system not being implemented at this time. We will try to progress with this to meet the requirement before the final date.

REQ 2.0 – Every move made by the user will be processed instantaneously:

With the current program, the user can make a move using the GUI and have it instantaneously processed and carried out. There is no lag time between the user’s interaction and the actual event happening on the board.

REQ 3.0 – The game should load in less than three seconds:

Our program currently loads and does not have a stall in the process while trying to open, there is no lag time which allows for the game to load in under 3 seconds. This accomplishes this requirement even though we have extensive code it still loads instantaneously.

REQ 4.0 – Changing themes will be instantaneous:

Our button that changes themes is under the options menu, when clicked this will change the theme of the board to the second option. This action happens instantaneously when the button is clicked, there is no lag time for this command.

4 Appendix B - Group Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Member** | **Meeting/Collaborative**  **Work Time (hrs)** | **Individual Work Time (hrs)** | **Total Time (hrs)** |
| **Daniel Maida** | 4 | 1 | 5 |
| **Nazar Stelmakh** | 4 | 1 | 5 |
| **Steven Call** | 4 | 1 | 5 |