Megamek Test Plan

Jane Valenti
Emanuel Rivera
Bruno Franchini

Outline

1.0 Introduction

The Megamek software is a video game adaptation of the BattleTech board game. The purpose of the software is to allow to two or more players to play the game online against each other. Megamek also has a one player mode where the player contends with the PC. MegaMek follows the same rules as the BattleTech board game and the game's logic reflects this.

Megamek is a large piece of software and testing it will require careful planning and prioritization. The approach that will be taken will involve the use of several tools that will analyze the code and point out packages/classes that require attention. The areas that require the most attention will be tested first and the rest of the software will be tested by priority as time and budget allows.

2.0 Features that will be tested

Following are the classes we will be testing:

2.1 High Priority Classes

Note: these are the classes that we focused the most on and spent the most time refactoring.

megamek/client/bot/BotClient.java
megamek/client/bot/GAAttack.java
megamek/common/actions/WeaponAttackAction.java
megamek/common/weapons/ArtilleryWeaponIndirectFireHandler.java
megamek/client/bot/AttackOption
megamek/client/bot/MoveOption

2.2 Lower Priority Classes

These are the classes to which changes/improvements were made, but not as much time was spent fixing.

megamek/client/ui/swing/CustomMechDialog.java
megamek/client/ui/swing/HexTileset.java
megamek/client/ui/swing/BoardView1.java
megamek/client/ui/AWT/CustomMechDialog.java
megamek/client/ui/AWT/HexTileset.java
megamek/client/ui/AWT/MegaMekGUI.java
megamek/client/ui/AWT/BoardView1.java
megamek/client/bot/GALance.java
megamek/client/bot/TestBot.java
megamek/client/bot/PhysicalCalculator.java
megamek/client/bot/ga/Chromosome.java
megamek/common/actions/PhysicalAttackAction.java
megamek/client/client/bot/CEntity
megamek/client/bat/ui/swing/botGui

3.0 Approach to Testing / Improving Software

3.1 General Approach

Each member of our group used a slightly different approach to improving the code, although we used several of the same tools. We also made sure to use GoogleCode as a way of uploading our code, but individual branches were used so that we did not “collide” when working on the code. GoogleCode also has a tool for creating “Issues” which is a tracking mechanism for keeping track of which classes have been updated, and why, when, and by whom they were updated.

3.1.1 Jane’s Approach:

Code Improvement was begun by using Metrics to analyze the entire program as a whole, to see where dependencies were and to determine which classes appeared to have the most issues. From there, the tool FindBugs was used on the three recommended packages

(MegaMek.client.bot, MegaMek.Common.Actions, and MegaMek.Common.Weapons) to locate very specific problems, such as unused/dead code, poorly executed checks (such as a badly designed oddness check or a String comparison using "=="). Finally, after fixing the issues located by FindBugs, two classes in particular were identified as being particularly badly designed (BotClient.java and GAAttack.java), as each had enormous methods (over 70 lines) that did far too much. These classes became the focus for the remainder of work, as many methods had to be extracted.

3.1.2 Bruno's Approach:

Analysis was done to three recommended packages (MegaMek.client.bot, MegaMek.Common.Actions, and MegaMek.Common.Weapons) using the Metrics tool. The classes that were identified as having the most glaringly obvious issues were improved first.

The Metrics tool indicated that the WeaponAttackActionJava's toHitIsImpossible method had 22 parameters. Metrics also indicated that the toHit method had a Cyclomatic complexity of 462, which in general terms would imply that this method is not very cohesive and possibly has a high number of errors. Lastly, Metrics indicated the toHit method also has 1188 lines of code which could mean that there is a high potential for refactoring. For these reasons, the WeaponAttackAction class has been identified as a high priority class.

Another class that was identified as being a top priority for improvements was the ArtilleryWeaponIndirectFireHandler class. One reason for this is that the handle method had 289 lines of code. This is a medium to large sized method and could be a good candidate for refactoring. This class also had a maximum nested block depth of 7. This could potentially mean the culprit method is inefficient due to a high time complexity. Lastly, the CheckStyle tool indicated several parameters in a few methods within this class could benefit from a "final" modifier to improve clarity.

Other issues identified include missing documentation, dead code, and missing comments. Test cases will be created as needed to improve classes that need them the most. These standard improvements will be made to all classes that are identified as having a lower priority.

3.1.3 Manny's Approach:

Analysis was done using a combination of CheckStyle and Codepro's Metrics Tools. I focused on fixing Trailing Spaces, correcting the declaration of variables, eliminating incline conditionals, and adding javaDocs to the classes. When I corrected the declaration of variables I had to modify the other classes that were using the variables with the getters and setters I created. I added some jUnits tests to classes under the megamek using the CodePro Unit testing creation tool.

4.0 Criteria for Pass/Fail

After we have made improvements to all the potentially troubled classes, we will run analysis using tools such as Metrics and Analytix on each of these classes once again. If the analysis after making changes shows fewer problems for a given class, then that class will be considered as having passed our improvement criteria.

Another indicator of successfully improving the code will be ensuring that the program still builds. After all the necessary improvements have been made, we will make sure we can build and run Megamek successfully. We will also pay special attention to game features controlled by classes we change. We want to make sure that we do not break the game or introduce new bugs.

5.0 Test Deliverables

The following items will be turned in upon completion of software improvements:

-Test Plan
-Improved and working MegaMek source code
    -Any test cases created will be included with the source code

-Peer Evaluations

6.0 Environmental Needs

6.1 TestingTools Required

-Metrics
-Code Pro Analytix
-FindBugs
-CheckStyle
-JUnit

6.2 Environment Required For Testing/Improvement Analysis

-Eclipse
-Java Version 7
-MegaMek
-Windows 7 and Above

7.0 Responsibilities

7.1 Bruno

>Classes Modified: (bold indicates high priority)

- **megamek.common.actions.WeaponAttackAction**
- **megamek.common.weapons.ArtilleryWeaponIndirectFireHandler**
- megamek.common.actions.PhysicalAttackAction
- megamek.client.bot.PhysicalCalculator
- megamek.client.bot.TestBot.Java

>Issues Created and Code Improvement
Issues 1 through 9; Issues 13 through 18

7.2 Manny

>Classes Modified: (bold indicates high priority)

- **megamek.client.bot.AttackOption**
- megamek.client.client.bot.CEntity
- megamek.client.bot.TestBot
- **megamek.client.bot.MoveOption**
- megamek.client.bat.ui.swing.botGui

>Issues Created and Code Improvement
Issues 19 through 24

7.3 Jane

> Classes Modified: (bold indicates high priority)

- src/megamek/common/actions/WeaponAttackAction.java
- **src/megamek/client/bot/BotClient.java**
- megamek/client/bot/ga/Chromosome.java
- /src/megamek/client/ui/swing/CustomMechDialog.java
- src/megamek/client/ui/swing/HexTileset.java
- src/megamek/client/ui/AWT/CustomMechDialog.java
- /MegaMek/src/megamek/client/ui/AWT/HexTileset.java
- /MegaMek/src/megamek/client/ui/AWT/MegaMekGUI.java
- /MegaMek/src/megamek/client/ui/swing/CustomMechDialog.java
- /MegaMek/src/megamek/client/ui/swing/HexTileset.java
- MegaMek/src/megamek/client/ui/AWT/BoardView1.java
- MegaMek/src/megamek/client/ui/swing/BoardView1.java
- **/MegaMek/src/megamek/client/bot/GAAttack.java**
- /MegaMek/src/megamek/client/bot/GALance.java
- /MegaMek/src/megamek/client/bot/TestBot.java

Packages to look at

- megamek.bot.client
- megamek.common.actions
- megamek.common.weapons.

> Issues Created and Code Improvement:

Issues 10, 11, and 12;