|  |
| --- |
| Rose-Hulman Institute of Technology -- CSSE 376 |
| Elemental Tower Defense |
| Metrics (Milestone 3) |
|  |
| **Steven Moyes, Arda Tugay, Matthew Mercer** |
|  |

|  |
| --- |
|  |

Table of Contents

[1 Executive Summary 2](#_Toc323303248)

[2 Metrics 2](#_Toc323303249)

[2.1 Cyclomatic Complexity 2](#_Toc323303250)

[2.2 Test Cases Per Method 2](#_Toc323303251)

[2.3 Number of Lines Per Method 2](#_Toc323303252)

[2.4 Work Hours 2](#_Toc323303253)

[2.5 Launch Time 3](#_Toc323303254)

[2.6 WTFs/minute 3](#_Toc323303255)

# 1 Executive Summary

This document outlines the metrics that our group will track for the remainder of this project.

# 2 Metrics

## 2.1 Cyclomatic Complexity

1) Cyclomatic complexity measures the number of linearly independent control paths through a program’s methods.

2) Use the metric plugin from SourceForge.

3) In order to determine if a method is adequately tested, we will compare the method’s cyclomatic complexity with its number of test cases.

4) We plan to use it alongside metric 2 (Test Cases Per Method) to determine if the method is adequately tested

## 2.2 Test Cases Per Method

1) The number of independent paths through a method tested by a test case’s tests.

2) Count the number of test cases that test a given method.

3) If a method is being under-tested, we will create more test cases. This metric will be used in conjunction with metric 1.

4) To determine if our methods are adequately tested.

## 2.3 Number of Lines Per Method

1) The number of lines of code (not comments) per method.

2) Count the number of lines of code (not comments) per method. Alternatively, use the metrics plugin.

3) We will use it in our analysis by seeing which methods have a code length > 50 and refactoring them if possible. GUI methods will be exempt from this metric.

4) We plan to track this metric to determine if methods are getting too large and need refactoring.

## 2.4 Work Hours

1) The number of hours spent working on the project (either through code, document development, or graphical contributions).

2) Keep track of time spent on the project independently, and contribute to a “TimeSheet” document on the DropBox.

3) This metric demonstrates that every team member is contributing an approximately equal amount of time to the project.

4) We plan to track it to make sure that the workload is well distributed.

## 2.5 Launch Time

1) The amount of time required to load the game before it is playable.

2) Use the stopwatch class built into Java. Get the current time before the frame loads and then get the current time when the frame loads. The difference determines the load time of the game.

3) If load time is too high, we will need to make the code more efficient. We could possibly use multithreading to handle sound/image loads.

4) Since we are making a game, load times are important to determining the quality of the game.

## 2.6 WTFs/minute

1) The number of times independent team members say “WTF?” when reading through or writing code, per minute.

2) Record the number of “WTFs” said while reading or coding, per minute.

3) The WTF rate demonstrates how legible, understandable, and readable the code that is being produced is.

4) It is a quick and easy measure of code readability and clarity.

