Software Testing

Y1481702

March 28, 2018

Contents

1 Test Plan

1.1 Introduction

Java MASON is an agent-based..
The software fits into the category of shrinkwrap meaning it may be used by many people in a variety of ways.
In particular,
The software is freely-available and open-source, which is a special case of shrinkwrap software. While some form of automated testing may have been conducted, it cannot be taken for granted as this has not been distributed with the source code, either on the MASON website, or its GitHub repository.

1.1.1 Tools

The IntelliJ IDE was used to explore the code and develop automated tests. JUnit has been used to create automated unit tests for the software. UI tests?

Git version control has been used to manage the code for the JUnit tests. For a long term project, this would mean that automated tests could be updated alongside any code changes.

1.2 Test Coverage

As stated in the project brief, the testing only needs to cover the `sim.engine`, `sim.field` and `sim.field.grid` packages, but not their subpackages.

- `sim.engine` is responsible for the core simulation management, including the agent scheduling.
- `sim.field` provides abstract classes for the representations of space in MASON simulation models, with subpackages managing specific instances of these.
- `sim.field.grid` provides various 2D and 3D grid representations of simulation space.

The expected behaviour of the software has been determined using the extensive Java MASON documentation[1], first-party example implemtations as well as a number of open-source examples[2]. The system behaviour has been tested against the inferred requirements of the client, a private biological research institute.

In order to design appropriate test cases an understanding of all levels of the system was needed. IntelliJ's Diagram feature helped to show how different components of the system are connected. Code coverage has provided a good understanding of which parts of the [code] are regularly used while running the software. Other metrics, such as how often different source files have been changed provide an insight into which files are most subject to change.

2 Test Case Specifications

2.1 Test 1

2.2 Test 2

2.3 Test 3

2.4 Test 4

2.5 Test 5

2.6 Test 6

2.7 Test 7

2.8 Test 8

3 Test Results

3.1 Test 1

3.2 Test 2

3.3 Test 3

3.4 Test 4

3.5 Test 5

3.6 Test 6

3.7 Test 7

3.8 Test 8

4 Test Summary Report

5 References

[1] S. Luke, *Multiagent Simulation And the MASON Library*, 19th ed., George Mason University, Jun 2015. [Online]. Available: https://cs.gmu.edu/%7Eeclab/projects/mason/manual.pdf [Accessed: Mar. 28, 2018]

[2] K. Alden, J. Timmis, P. Andrews, H. Veiga-Fernandes, and M. Coles, "Pairing experimentation and computational modeling to understand the role of tissue inducer cells in the development of lymphoid organs," *Frontiers in Immunology*, vol. 3, p. 172, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fimmu.2012.00172

6 Appendix

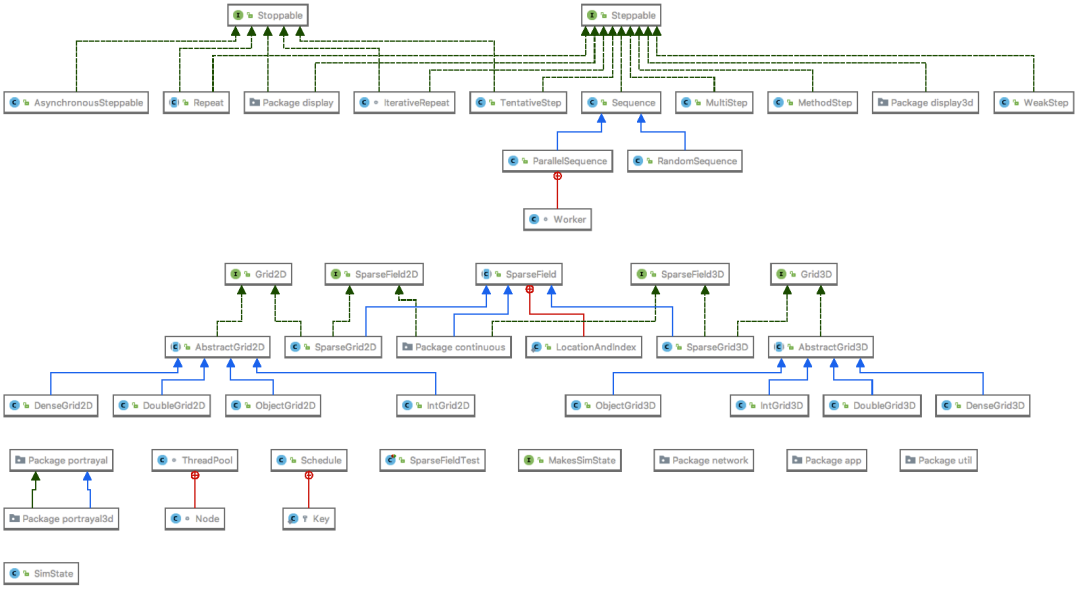

Figure 1: Reverse Engineered UML diagram of Class Hierarchy