Emergent Architecture Design

Blocks World for Teams

BW4T Context Project

Emergent Architecture Design

Blocks World for Teams

by

BW4T Context Project

in partial fulfillment of the requirements for the completion of

TI2805: Context Project
of the Bachelor of Computer Science and Computer Engineering

at the Delft University of Technology,

Architecture Design Team:	Daniel Swaab	4237455
	Martin Jorn Rogalla	4173635
	Jan Giesenberg	4174720
	Sander Liebens	4207750
	Sille Kamoen	1534866
	Shirley de Wit	4249259
	Tom Peeters	4176510

An electronic version of this document is available at https://github.com/MartinRogalla/BW4T/.

Contents

1

Introduction

1.1. Design Goals

For this project we were given a large codebase. This codebase had grown complex and convoluted and the task for us was to refactor it to reduce complexity, make it more manageable and give it a Plugin Architecture. On further inspection, however, it was revealed that an exact Plugin Architecture wasn't precisely what the product owner wanted, but simply a modular design that can easily be expanded or modified was desired.
To this end the goal is to have most if not all core parts of the environment and agents built using interfaces, thus allowing easy addition of new elements in the environment or agent behaviour by creating a new class utilising one of these interfaces, similarly these classes could then later be easily removed without causing parts of the system to start malfunctioning or breaking entirely.

For the refactoring of the code a number of tasks need to be completed:

- **Split up codebase into Client and Server structure** - Currently the Client and Server share the same codebase, this makes it hard to keep a proper overview as it isn't always clear at a glance which class belongs to which structure, and some are even shared entirely. We want to split these structures into individual projects to make them easier to work with.
- **Convert to Maven project** - There are currently no Unit Tests present in the codebase, therefore in addition to splitting the codebase we wish to turn these seperate projects into Maven projects including Unit Tests so we can easily verify that the codebases are still working correctly after having made changes.
- **Clean code** - There is a lot of unused code present, due to the complexity of the codebase it can be hard to tell which pieces of code are unused.
- **Fully utilize Repast** - Repast is a library that has been imported to act as the environment for the agents to act in, however due to some missing functionality and unfamiliary with Repast at the initial time of implementation Repast isn't properly utilized, many things that have been implemented manually can also be handled by Repast.
- **Improve Javadoc** - Javadoc's are sometimes lacking in information or occasionally not present at all.
- **Remove code duplication** - There are a number of instances of code duplication present in the code which can make future updates harder to manage.

2
Software Architecture Views

3 Glossary

BIBLIOGRAPHY