Defiant Worlds

Configuration Management Plan

# Identification Scheme

Through the project, there will be several different aspects which will need to be managed. As well as the actual product development, all design documentation, including diagrams and models, will need to be managed and controlled. Also all assets required for development will have to be managed.

**ID Scheme**

Defiant Worlds

- DFW

Design Documentation - DDT

Art Assets

- AAS

Core Product - CPD

Outcome Requirement Models - ORM

Scoping Models - SPM

State Transition Models - STM

Domain Class Model - DOM

Use Case Models - UCM

CRC Cards

- CRC

Design Class Model - DCM

Sequence Models - SQM

# Responsibilities

As there is limited time for project completion it has been assigned such that each developer will have a responsibility to manage the changes and the versions of the various manageable items. Using the version control and configuration management (CM) tools we have decided to use, developers will be able to control the CM procedures.

Also communication will be key and as such all developers will be contactable should confusion around version control arise.

During routine team meetings, the entities submitted to the group central repository will be discussed and managed, as well as configuration assurance audited as part of a team.

# Version Management Policies

For Version control and management, the group will work on pre-determined areas of the project that they have been assigned to. This ensures that the overlap involved will be minimal and so will cut down on merging errors. Any new errors, with priority, will be mentioned in the title of the most recent version of the outcome, and all known issues will be mentioned and described within the main.cpp file.

After a set amount of features are released or a significant number of issues are corrected, a new baseline will be constructed. From this we will be able to re-assess the requirements of the new baseline and to prioritise features or issue fixes.

After the initial development phase, we will then use CM for defect management and run the solution through pre-determined tests. Upon a test failure, a note will be made within the repository and then necessary changes can be made.

# Version Management Tools

The main version control tool we will use is Git, which is a web based repository hosting service. Through this we will be able to manage the source code for the project, as well as have a hosting repository for the rest of the manageable files and perform all required CM procedures.

The Git repository will be used by all team members. When starting development sessions, the procedure will be that ‘fetches’ and ‘pulls’ will be done by all developers, to ensure that there is the lowest chance of merging issues when it comes to ‘pushing’ the changes to the central repository.

Throughout the development session, the procedure will also be to contact the other developers, should a sudden issue arise. This will be done to ensure that if the central repository is compromised, that the local copies from developers can be used as a restore point.

At the end of each development session, a final fetch and pull should be done to ensure that conflicting edits do not occur or are resolved before committing changes.

# Configuration Database

The configuration Management database will be the central location in which faults and changes will be made. Within our group, all current faults will be documented within the main page of the solution so that developers can be made aware of issues when starting a development session. One of the main benefit GitHub, the web based repository, is that it tracks all changes between repository versions. This will be what is used to track repository changes.