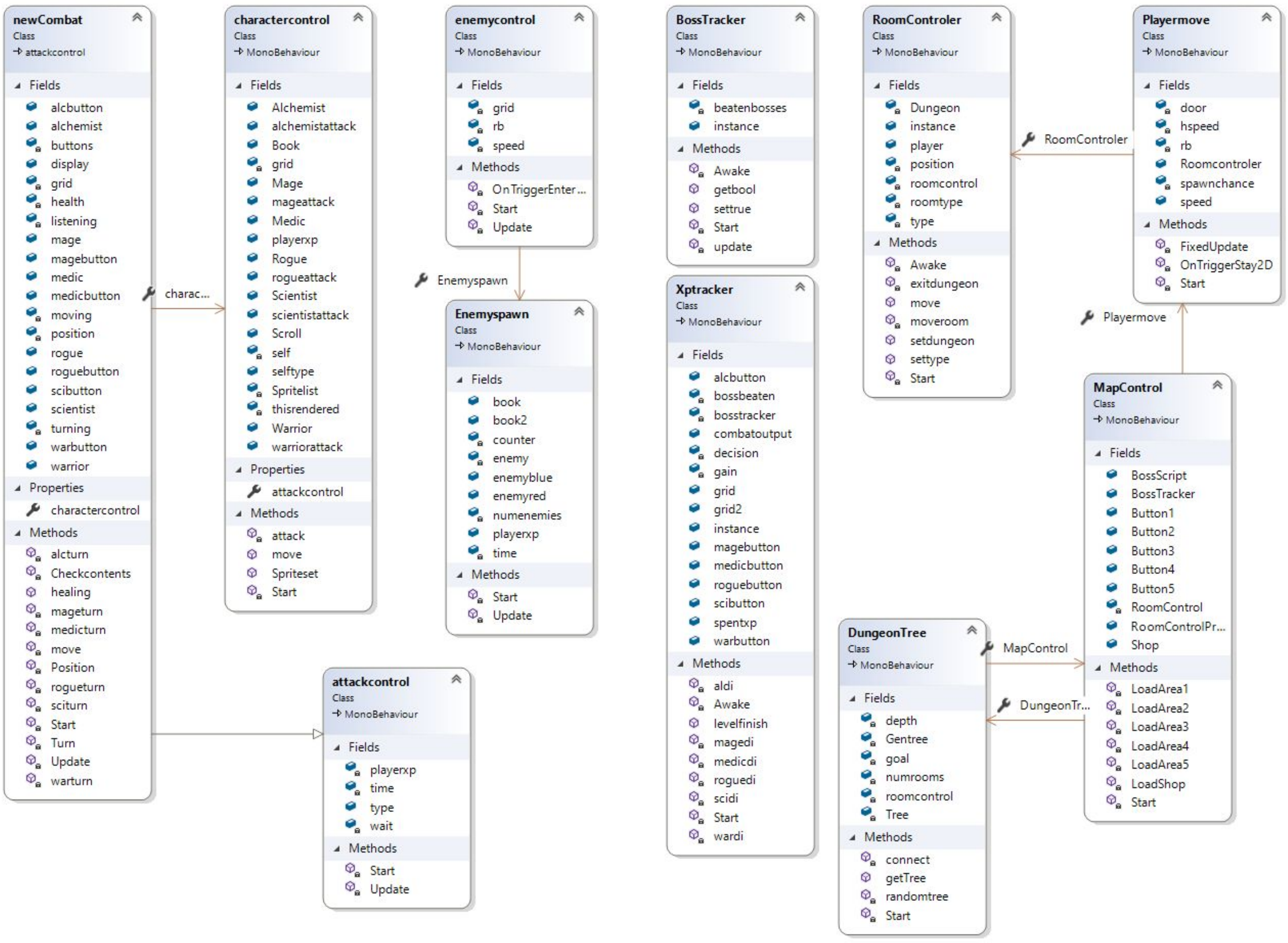
2.2 | Architecture Report

Architecture Overview

The concrete architecture where class attributes have been expanded is below.



The tool used for the above UML diagram was Visual Studio. It is capable of native UML generation, by analyzing an existing project and generating a detailed map of classes. Inheritance and association links must be manually added (as seen).

Architecture Justification

Our Architecture builds from the class model required by Unity to build our game. We had to make large changes from our initial abstract architecture outline due to the nature of Unity which meant we couldn't implement features (objects and inheritance) as we were used to. The main requirements we had to consider when creating our architecture was the three main play styles U2 U3 U4. These meant different scenes within our game had to interact with each other and let the player seamlessly pass between areas in an obvious manner for the user. We have also completely removed the character stats as the nature of our game has changed. This is due to the way Unity initialises characters making our abstract architecture hard to implement as you can't just call a new character with stats at any time in unity. Our new Enemy Spawn class creates enemies with the same set stats and our human characters come premade.

New combat situations are created by the newCombat class which interacts with the charactercontrol class and the attack control class for player interactions and information during combat. The enemy control class is used to control the actions of enemies. Enemy spawn is used to create new enemies and spawn them into the world. The MapControl class interacts with the DungeonTree and Playermove classes to allow firstly the world map to be loaded and presented to the player and then the player to be able to move around the world and into dungeons. The room controller class is used to Control rooms within a dungeon and tracks the position of the player within the room along with other things.The final two classes are the Xptracker and bossTracker classes, the boss tracker class is used to generate and control boss fights and interactions, while the Xptracker is responsible for keeping track of the players xp and level.

A major change between the actual implementation and the abstract plan is the combat loop design. We had initially intended to implement a turn-based combat system, where control is passed between player and enemy objects by checking a queue each frame, with a combat class to dictate when control is changed. Now, the combat class itself controls all elements of the combat and player-enemy interaction, and player and enemy control classes feed in any non-combat attribute updates such as speed,movement etc.

The newCombat is the largest class we implemented, this was to satisfy U1,U3,S5. We created separate controllers for each different character class with their own unique style to create different characters in terms of playability this was to satisfy U1 and so created. This class includes a cut down version of the combat loop algorithm designed in assessment 1, our new combat runs in live time not turn based and so does not follow a set algorithm for players but the monsters still follow their own turn based design within their limited script to follow a pattern for the player to understand and overcome.

MapControl is the class which controls the movement between the different Gui’s it holds the information for all the different live variables which need to be shared between areas, it will go on to hold the save and load functions in part 3. This satisfies F1,2,4 of our functional requirements this controlling class keeps checks on the other subclasses.