SWEN30006: Software Modelling and Design
Project 1 Report

Group W13-5:
Ziqi jia 693241
Kevin Liang 864665
Tan Saint 940539

The aim of this report is to describe the changes and additional classes that were made to the initial system design of Automail; which allows the system to coordinate multiple robots to carry and deliver heavier packages. This report will include a design class diagram of the recommended system along with an explanation of the changes that were made using related design patterns and principles.

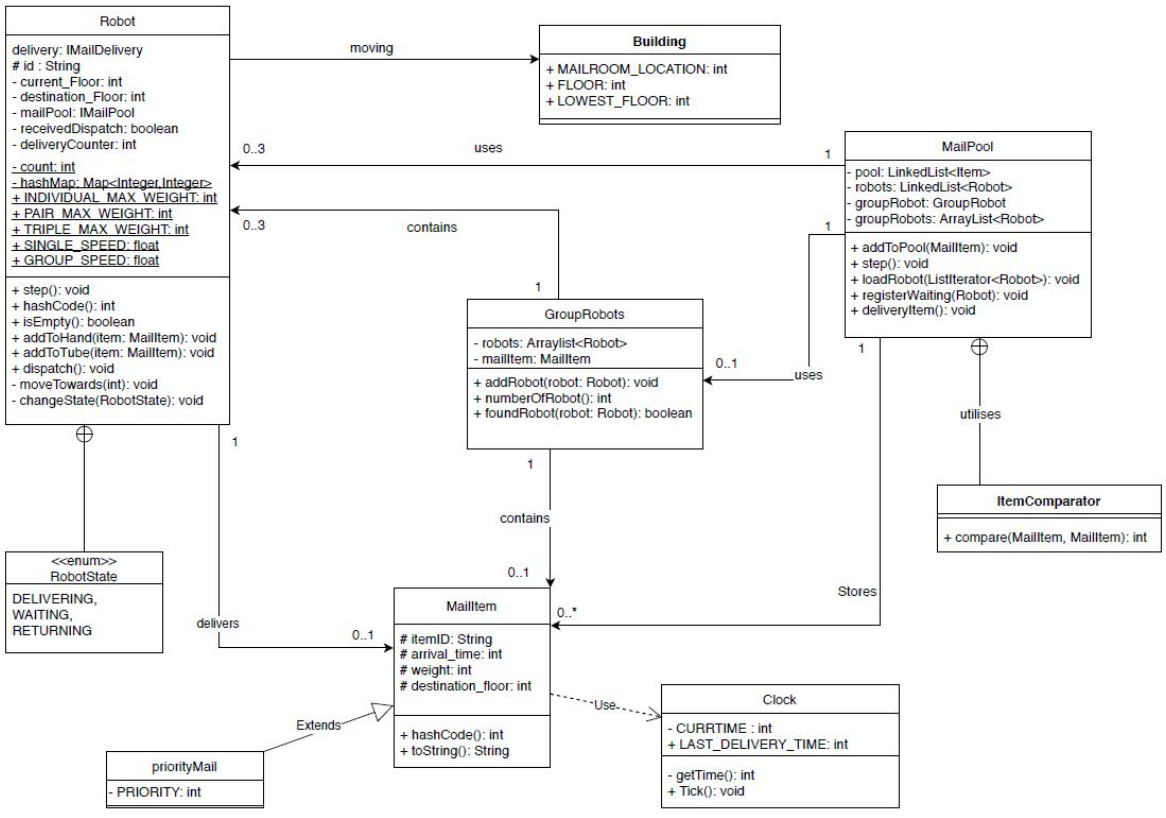

Figure 1: Automail Design Class Diagram

In order to extend the robots' functionality of delivering heavier items, a GroupRobot class was created to help better organize the robots. It is essentially an expert class that stores all the information of the robots that belongs to a group, along with the item that they are carrying. The formation of groups to deliver heavier items is done in the MailPool class, which would check the availability of the other robots if a robot encounters a package that is too heavy; and will put them into a group to send them out for delivery. In addition, the moveTowards() method in the Robot class has also been changed to facilitate the different speeds of the robots delivering an item; it will check whether the robot is in a group or not and moves the robots to their destination accordingly. The implementation of the GroupRobot class was decided to simplify and make use of the existing system design as most of the basic functionality was already implemented. Although this resulted in a higher coupling between the Robot class and the MailPool class as the MailPool class has to decide which robots/group to send out, it also increases the cohesion between the classes. Thus, making the MailPool an information expert and creator of GroupRobot .

The initial decision of the implementation for the GroupRobot class was to create a class that would automatically place available robots into a group so they could act as a single entity to deliver heavier items. This meant that whenever an item was needed to be delivered, a GroupRobot will be created (even if it is a group of 1) to deliver the item; all the MailPool had to do was to call the constructor for GroupRobot class with the MailItem that needs to be delivered and it would determine how many robots were required to deliver the item and place them into a group. As such, it helped simplify the delivery of items because instead of treating each robots individually, it would treat them as a single entity instead; which would allow for more control of the robots as the robots in the same group would have the same movement as opposed of checking and move each robots individually. Since the responsibility of creating and the delivery of the item is not implemented in other class, this design suggests a lower coupling but it would also maintain the cohesion between classes. Therefore the GroupRobot class would serve as the Information expert for creating a group as well as a Pure fabrication.

However, the implementation of the later design was found to be very complicated and difficult. This was due to the fact that the initial system design has showed to have a high coupling between the classes; as the implementation of the GroupRobot class meant that numerous changes needed to be made to the other classes. Hence we've decided to preserve the functionality of Robot in the original design, instead of changing the existing code to suit the new class.