Team Number: W05 Team 05 [Wed 05:15PM]

Design decisions for implementing cost calculation:

- By info expert, the Robot class is responsible for keeping track of activity units, service fees and a delivery counter. By extension, the information expert tells us that the Robot class should also charge the tenant upon delivery.
- Also by the info expert, a Robot should log its total deliveries and various costs it calculates, the contraindications of which would make the Robot class bloated and hard to read. For this reason, our team opted for pure fabrication, creating classes specifically to hold statistics, kept in a separate statistics package. This greatly increases cohesion as it gets rid of messy statistics tracking code that is not necessary in the Robot class. This was implemented by having a class called RobotStatistics (Robot is the creator), which holds the total deliveries and costs/fees made by a single robot, and StatisticsProvider, which calculates the overall stats required for the end of the simulation. This design choice also makes our code more cohesive, modular and separate from the original source code, making it very easy to revert or extend the system's functionality in the future. It also separates the logic of calculating statistics from the actual functionality of the Robot, further increasing cohesion and low coupling.
- Assuming over 10 failures in a row to lookup a service fee, the robot will according to spec charge according to the most recent service fee retrieved for any floor, as this information is available most easily to the robot, by information expert we've added the most recent service fee as an attribute of the robot.

Estimated charge

- By information expert, the MailItem class calculates the estimated charge because it knows its destination floor. We assumed that only one lookup would be needed in this estimation (i.e. incremented activity units by 0.1).

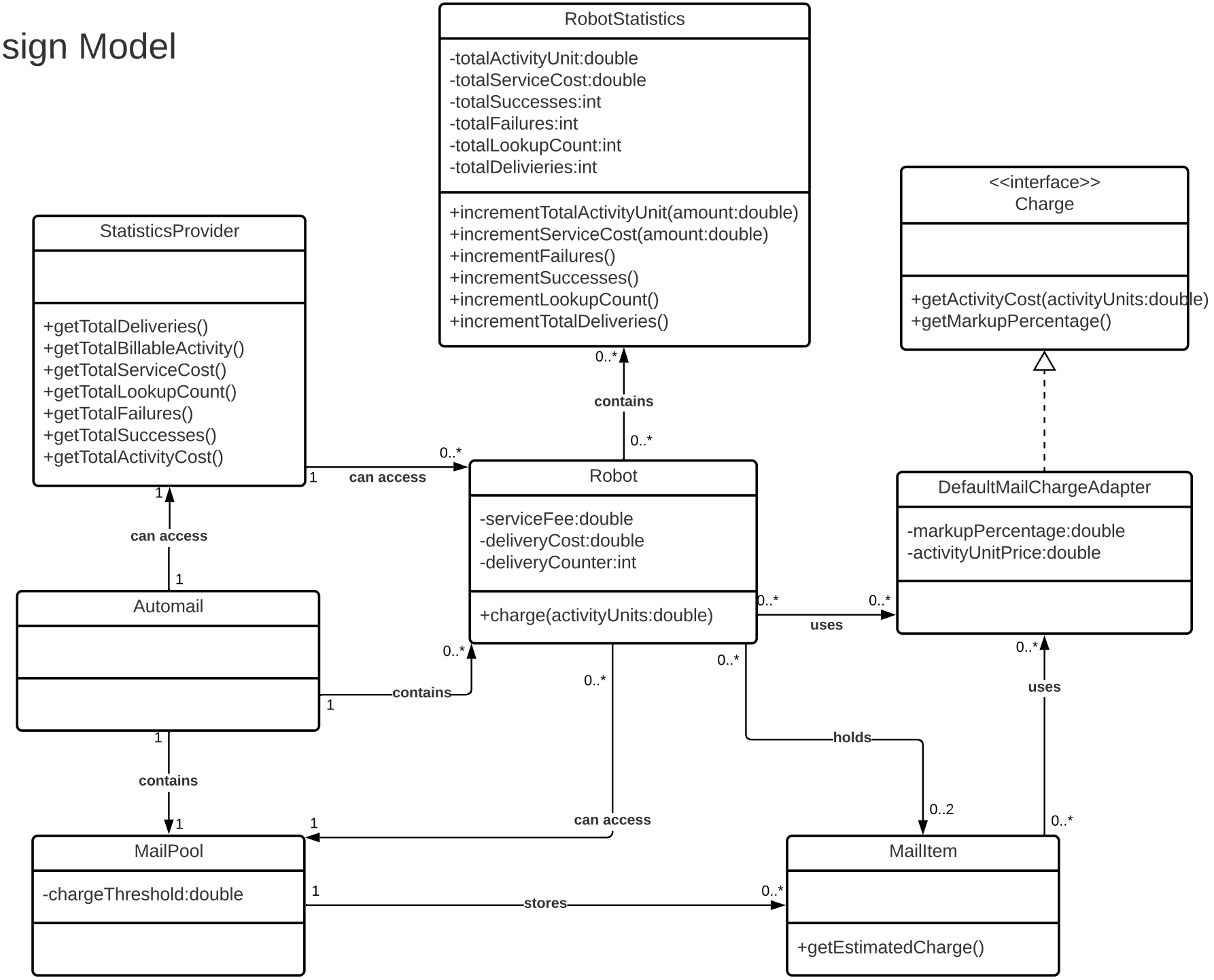
Leaving room for change

- Since the company wants *markup percentage* and *activity unit price* to be configurable in the future, we created an interface called 'Charge' which holds the methods to calculate activity cost and to get the markup percentage. Not only does this allow ease of extension by allowing for the creation of multiple charge adapters with varying values of *markup percentage* and *activity unit price*, it also allows for the calculation of charge to be a polymorphic function, which ensures protected variation for the system.

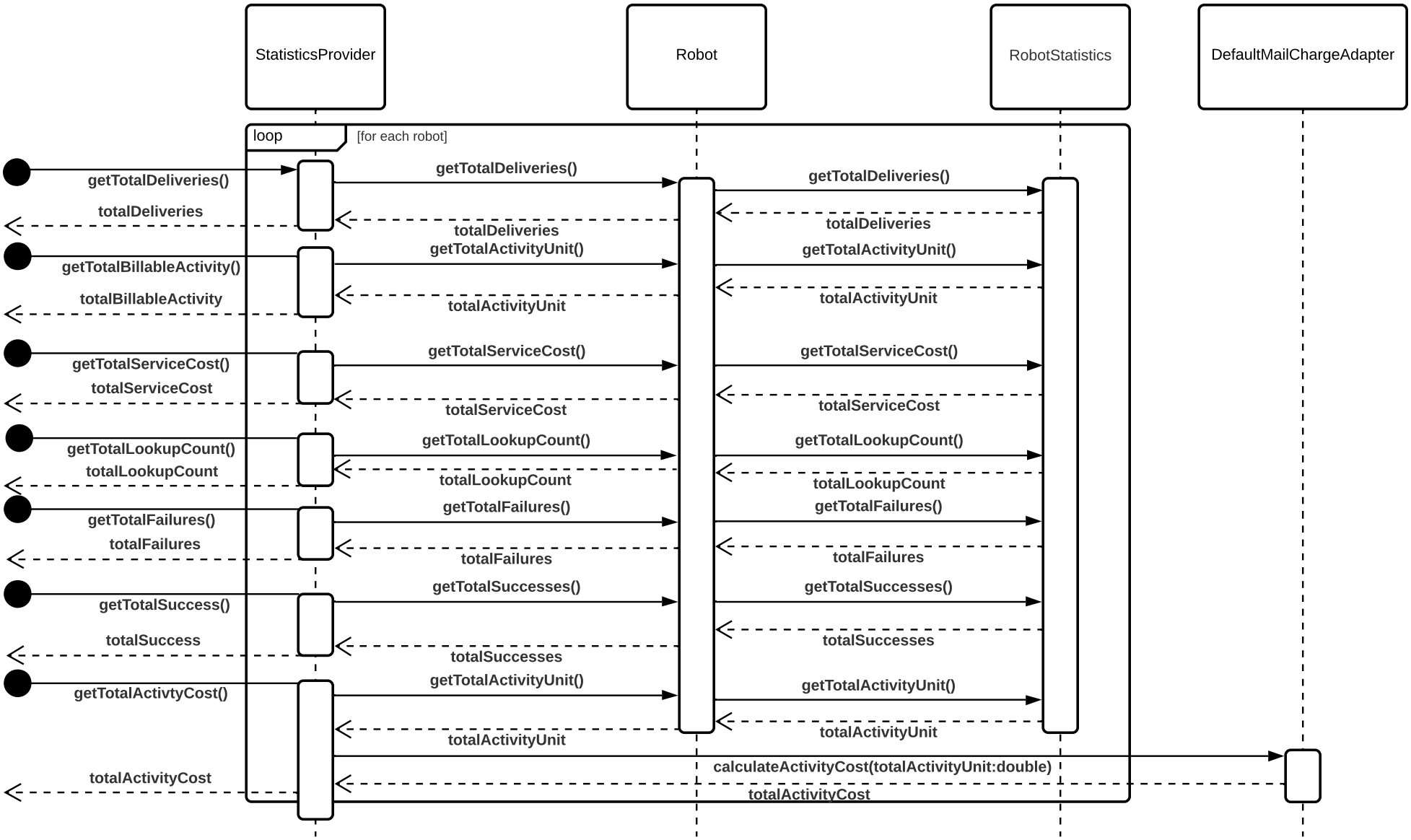
Costs not passed on to tenants

- Failed lookups are out of the tenants' control and is a fault of the BMS, so we left this out of the charge and, hence, billable activity
- A service fee lookup is necessary to calculate expected charge for determining priority mail, which shouldn't be passed onto the tenant
- The activity cost of each delivery is calculated as if the robot made a trip to only the destination of that item and returned to the mailroom. Thus it is something that can be calculated just before delivery on the spot. This is because we cannot reasonably charge tenants for extra activity units incurred from a previous delivery.

Design Model



Dynamic Sequence Diagram for delivery statistics



Dynamic Sequence Diagram for delivery charge