ENPM663 RWA2 Report

Group 6

Spring 2022

Team members:
Anirudh Krishnan Komaralingam (117446432)
Jayesh Jayashankar (117450101)
Kartikeya Mishra (118106755)
Neha Saini (117556292)

Professors:
Dr. Craig Schlenoff
Dr. Zeid Kootbally

1 Introduction

This report describes the modifications made to the ARIAC competitor package of Group 6 created in RWA1. The changes are designed to address the functional requirements of the competition as well as a few of its agility challenges. The changes broadly fall into the categories of:

- Package structure updates
- Addition of sensors and cameras
- Part spawning
- Planning diagram updates
- Detection of agility challenges

2 Package Changes

The package name has been changed to `group6_rwa2` and the change has also been reflected in the package.xml file.The config folders have been created with `rwa2_trial.yaml` in the trial config folder and `group6_rwa2.yaml` in the user config folder.

3 Sensors & Parts Spawned

To handle the agility challenges as part of RWA2 and for future RWAs different sensors need to be placed at appropriate positions to track the presence of parts required to execute an order. The following are some of the sensors and parts spawned as part of rwa2:

- A total of 19 cameras have been spawned. Cameras have been above the 4 AGVs at the kitting stations, 2 above the bins, 4 for AGVs at as1 and as3, 4 for the AGVs at as2 and as4, 4 above the briefcases and 1 above the conveyor belt.
- A breakbeam sensor close to the conveyor belt.
- Parts in all the briefcases and in bins one and seven have been spawned.

4 Sequence Diagrams

4.1 Kitting

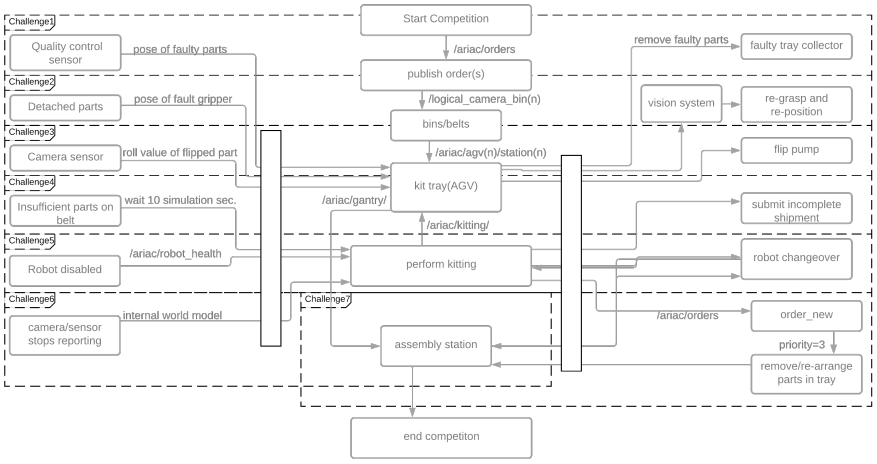

Figure 1: Kitting Sequence Diagram

4.2 Assembly

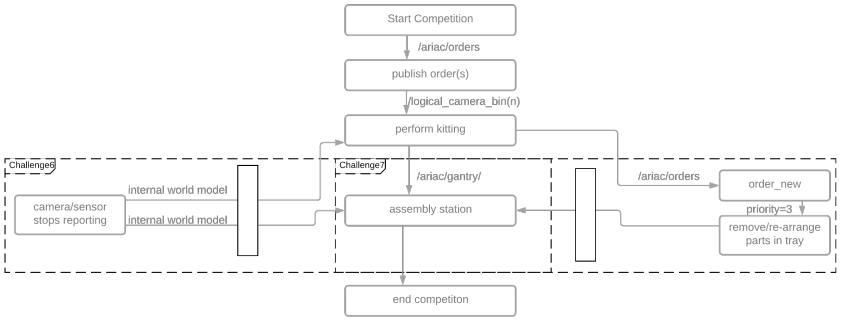

Figure 2: Assembly Sequence Diagram

5 Class Diagram

This section describes updated class structure and provides an overview of the architecture. Figure 3 illustrates the various classes and their inter-associations. Descriptions of the most significant classes follow.

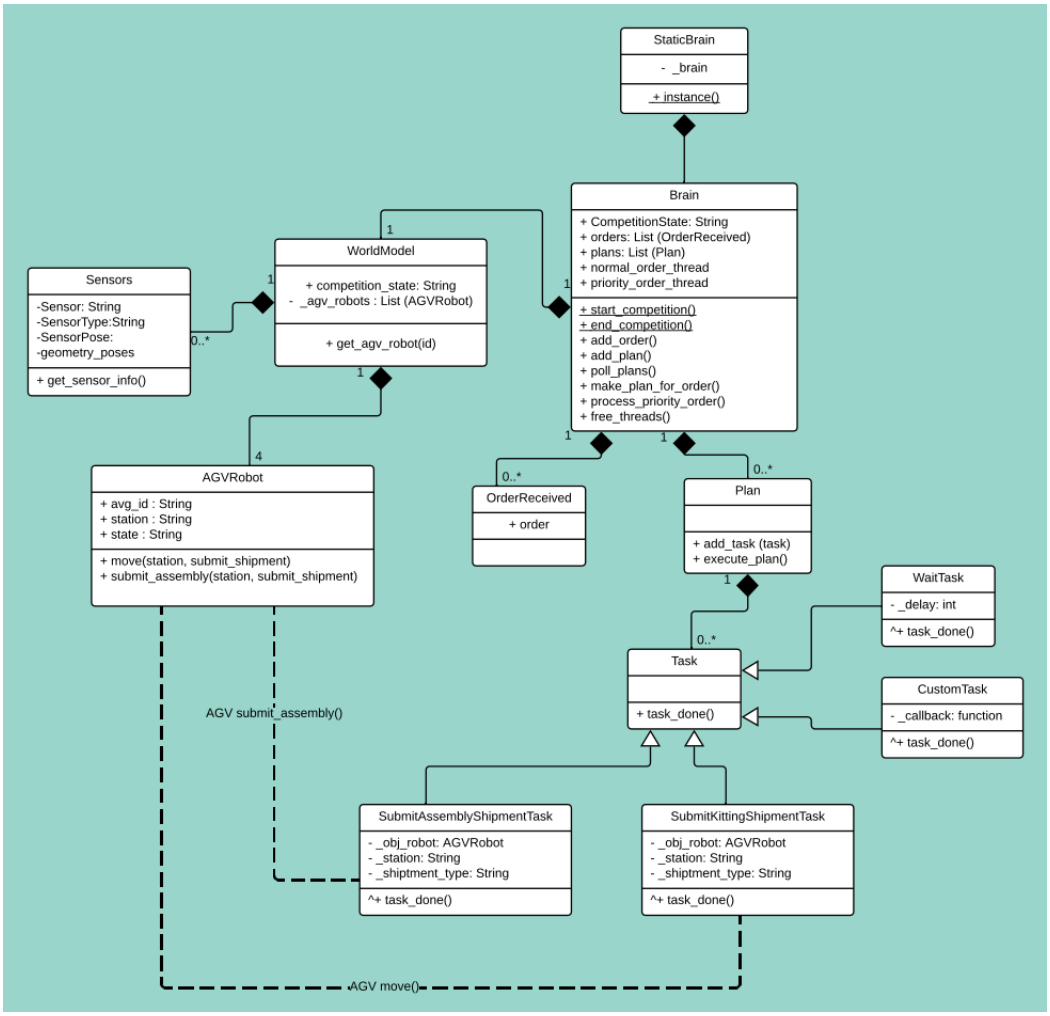

Figure 3: UML Diagram

5.1 Brain

The Brain class orchestrates the actions that are taken to complete the competition. This class contains methods to start and stop the competition, crate plan threads from orders, track and execute plans and re-prioritise plans when new orders are encountered. A static instance of this class is shared to all modules that use it.

5.2 World Model

The World Model class contains representations of the physical objects in the environment and capabilities to facilitate accessing and searching of these objects. At present, this includes the AGV robots, their associated methods and parts used for assembly and kitting, their poses and methods.

5.3 Orders and Plans

Orders are received through the GEAR interface on the */ariac/orders* topic and are decomposed into Plans that comprise of uninterruptible executable base units called Tasks. Plans are executed as individual threads, Plans are paused when higher priority orders are accepted. Orders received after the initial one are considered higher priority for now.

5.4 Tasks

Tasks are actions such as submitting a kitting shipment or waiting for a predetermined time that are executed as part of a Plan to fulfill an order. Tasks and not interruptible. When a Plan is to be paused, the currently executing task is completed if there one after which the higher priority order is serviced.

5.5 Summary of Changes

1. Addition of distinct tasks.
2. Reorganization of orders and plans.
3. Removal of shipment classes.
4. Threading of Plan execution.

6 Agility Challenges

6.1 Faulty Part

Faulty parts are detected by a variant of the Logical Camera called the Quality Control Sensor. Faulty parts are detected by subscribing to the Quality Control Sensor topic and checking if the value of the type field is non empty.

6.2 In-Process Order Change

When an existing order is being deployed on any agv and a new order is triggered. Then the new order is considered to be a priority order. The old thread is passes and begins a new thread to process the new order. Once the new order is process the old thread resumes to complete the earlier order.

6.3 Sensor Blackout

A thread is created to decrease the value of the a counter which has been created to verify the scenario of a sensor blackout. Whenever the sensors are publishing something the counter value is reset. The counter is set to 200 and decreases with a frequency of 0.01s. When the sensors are not publishing anything the count reaches 0 and sensor blackout is triggered.

6.4 Flipped Part

A part is considered to be flipped if it has a roll value of pi.

6.5 Insufficient Products

Using the logical cameras above the bins we store the part's ID and position in the world model. Once a part spawns for an order we check if the part already exists in the world. If the part is not found in the stipulated amount of time (20s) it is declared as insufficient.