Autonomous Warehouse based on a Multi-Agent System

Anders Tasken
University of Trento, Italy

***Abstract*—Multi-Agent Systems, hereby referred to as MASs, are an area within the field of distributed Artificial Intelligence with great growth and promising results nowadays. With the use of several intelligent agents mutually dependent on each other, problems difficult to solve with a monolithic system can easily be resolved with the use of MASs. Strategic interaction between agents working explicitly together is implemented throughout this report in a distributed environment.**

***Index Terms*—Multi-Agent Systems, Artificial Intelligence, MAS, Prolog, Unity, Autonomous, Agent-Oriented Software Engineering.**

I. Introduction

THE use of MAS have at the present time received immense attention within the Computer Science community for it's ability to solve complex problems, and this report show how MAS can be used to create an autonomous warehouse. The environment in this task consist of several drones and robots which interact in order to sort and deliver packages between post-offices.

II. Approach

The warehouse consist of four areas distinguished by color, and are named "South", "East", "North", "West". In each area, there are several post offices characterized by an index. The MAS consist of one artifact and five agents. The artifact, the five agents and the environment is thoroughly described in the assignment [1]. See figure 1 for an overview of the warehouse environment, and figure 2 for an overview of the charging stations to the drones and robots.

The code and GitHub project for the framework of the project is available at [2]. The environment is implemented in Unity, and Prolog is used as the logic programming language to design the agents.

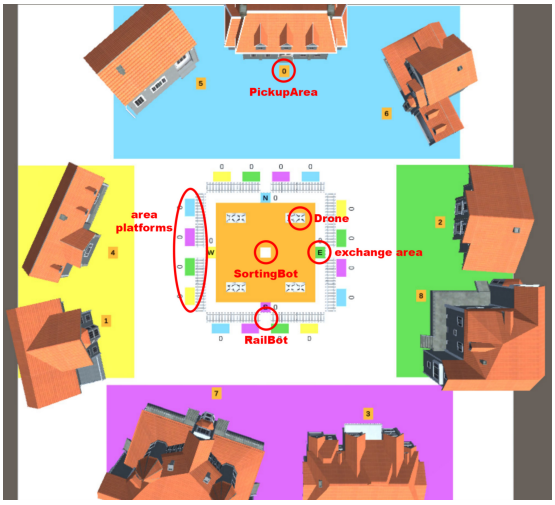

Fig. 1: Overview of the warehouse environment from [1].

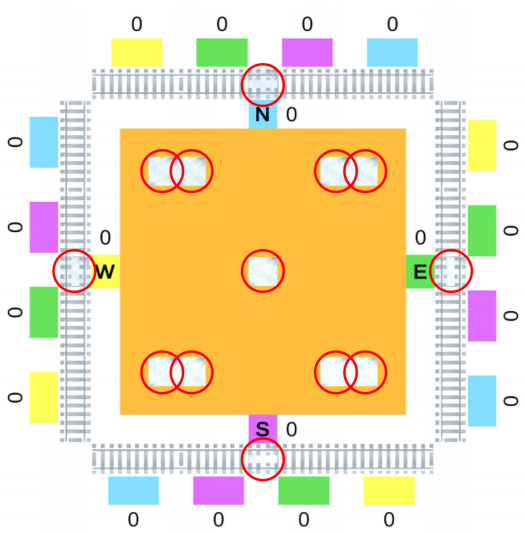

Fig. 2: Overview of the charging stations in the warehouse from [1].

III. Results

The Multi-agent System designed to complete the assignment and work as an autonomous warehouse consist, as mentioned, of one artifact and five agents. The artifact is the package to be retrieved, moved and delivered. The artifact is represented by a box, and are initiated at run-time. The agents are named PickupArea, GameManager, Drone, RailBot and

SortingBot. By designing intelligent and interactive agents, these agents are able to sort all initiated boxes to the correct post-office through the correct path. The behavior of the artifact and each agent is described in the subsections below.

A. Box

These artifacts are initiated without any beliefs or plans, but are assigned several beliefs by the GameManager agent described in subsection III-B.

B. GameManager

The GameManager agent is a single agent responsible for the initiation of the boxes, which are supposed to be picked up, sorted and delivered by the other agents. This agent defines the start and end destination of all boxes, and trigger the PickupArea agent of the start destination to handle further distribution.

The start and end destination is added as a belief to every box, in order to make other agents able to retrieve this information. The start and end pickup area is added as a belief to the box as well, in order to sort the artifact correctly throughout the delivery process. The PickupArea is triggered by adding a desire to the agent.

C. PickupArea

Each pickup area are agents as well, with the simple role of distributing the pickup task to available drones and recall boxes that are delivered to it. This is implemented by checking if the Drone agent believe that it is busy, and giving it the desire to pick up the box if not. The box is recalled when a drone delivers a box to the destination area and give the desire to recall it.

D. Drone

The box is picked up by a random available drone with a desire handed over by the PickupArea agent. By checking the beliefs of the picked up box, the correct landing zone is found, and by several actions and co-routines available in the API the box is placed there as well. By communicating to the RailBot agent by giving it the desire to pickup the box, the artifact is then halfway to the destination.

The drone agent is responsible for the delivery of the box from a landing zone to the post office as well. This desire is achieved through communication with the RailBot agent. After each desire is fulfilled, the agent follow a co-routine to fly to it's charging station for recharging.

E. RailBot

The RailBot agents is in many ways a transporter train. These four agents can only move along its own trail, and it's main task is to transfer the box from a platform to the exchange area, or the other way around. After moving a box from a platform to the exchange area, it delegates a desire to sort the box further to the SortingBot agent. Similarly when a box is transported from the exchange area to a platform, the RailBot ask a Drone agent to continue the delivery. The RailBot return to the charging station after each delivery to recharge.

F. SortingBot

The sorting of the artifacts from one exchange area to another one is completed by the SortingBot agent. The RailBot from the starting exchange area ask the agent to transport it, and the artifact itself believes where the destination is. After transportation to the correct exchange area, a desire to move it to the landing zone is given to the RailBot of that area. Also this agent return to the charging station after it's desire is fulfilled.

G. As a whole

The MAS as a whole result as expected after an iterative development of the logic of each agent. By closely designing the beliefs, desires, actions and co-routines of each agent in a way that make both the behaviour of each intelligent agent correct and the collective handling of the task. The different agents help each other solve the task, resulting in a distributed and effective delivery service of the packages.

IV. Difficulties

The development of the Agent-oriented code was in a whole a feasible project. Most of the necessary knowledge in order to complete the assignment is learned through the Lab tasks given in the "Agent-oriented Software Engineering" course. The description of the problem, as well as the API given, is clearly conveyed. Nevertheless there are several difficulties to encounter when this system is designed.

The first problem in the development of the code was to get a clear and thorough understanding of what the functions given in the API give as output. The behaviour of the agent is hugely dependent on the correct handling of the output from these functions, and at first some misunderstandings resulted in drones flying to the wrong destinations. This issue was resolved by reading the description several times, and simply by the principle of "try and fail".

A different problem encountered was to handle the fact that several agents are busy at different times, and thus need to handle the communication in a manner that are robust. This is not easy, since the overall behaviour is dependent on the sequence of many actions and desires. Of that reason it is necessary to test the MAS in many different settings, to make sure the behaviour is as expected. It is also important to build the logic in a way that handle several different scenarios the the environment.

V. Conclusion

The Multi-agent system developed throughout this report managed to handle the task of acting as an autonomous warehouse. With the use of agent-oriented programming, the complex task is solved with a solution attainable to implement. Some difficulties related to the API and the collective behaviour where encountered and handled, resulting in a fully functioning implementation.

REFERENCES

[1] Giorgini, Paolo. Alzetta, Francesco. Software Engineering Group, University of Trento. *Agent-Oriented Software Engineering 2019/2020 Project*. May, 2020.
[2] Alzetta, Francesco. *https://github.com/ElDivinCodino/AOSEProject2019-2020*. May, 2020.
[3] Tasken, Anders. Alzetta, Francesco. *https://github.com/Anderstask1/Agent-oriented*$_{S}oftware_{E}ngineering.May$, 2020.