Milestone Report

Team: Dumping to gather

Josephine Krause[1], Marc Schmidt[1] and Muzammal Hussain[1]

[1] Technische Universität Berlin, 10623 Berlin, Germany

Abstract. The following paper is a milestone report for the Application System Project B in the summer term 2018. The project is about finding a solution for the Multi-Agent Programming Contest 2018. Therefore, the contest will be presented in the first chapter. In the following second chapter the general team strategy will be explained. The implementation concept and the architecture of the system will be presented in chapter 3. The current state and the following work will be pointed out afterwards. Finally, the team structure and responsibilities will be shown.

Keywords: Multi-Agents, Decentralized Systems, Auctioning

1 Introduction

Solutions for distributed systems are becoming increasingly important [3].

The Multiple-Agent Programming Contest (MAPC) [1] is a system that could profit from a decentralized solution instead of a centralized approach. In this contest, there are supposed to be two teams of agents. The agents of each team must work cooperatively in order to win the game. The solution for the simulation should work decentralized. The simulation consists of the map of a city and is executed in steps. During a step an agent can execute one action.

The scenario of the MAPC [1] describes a water crisis in the year 2045 A.D. that needs to be solved. The main goal is to earn as much points as possible by building wells. The wells cost money which can be earned through the agents of the simulation by executing jobs.

The scenario [1] contains different roles for agents. An agent can be a drone, a motorcycle, a car or a truck. In this order, from drone to truck, the capacity and the battery of an agent are increasing while the speed is decreasing. Each agent has the following parameters: speed, load, battery, vision, skill and roads. Opposed to the others, the latter does not have a base or a maximum value. The former five parameters are initially set with the base value and can be upgraded during the simulation. Speed describes how fast an agent can move. Load states how much volume an agent can carry. The parameter vision displays how far an agent can see. Skill states how fast an agent can build a well or gather an item. The parameter roads describes the ways agents can use. Drones can fly in the air. This means that they always can take the

shortest, linear way through the air to a location while the other agents are bound to the streets.

The upgrades can be done individually for any parameter of any agent at any time. However, they cost money, they can only be bought in certain facilities and an agent needs a step to execute the upgrade [1].

The currency for the money is Massium. In order to earn money they need to execute jobs. There are three different kinds of jobs: regular jobs (further named priced jobs), auction jobs or missions. Priced jobs have a predefined reward while auction jobs have to be won in an auction. The team with the lowest bid will be assigned for the job. Even though the reward might be lower, only the assigned team can finish the job. Priced jobs can be completed by any team. The first team to complete a priced job is the team that will be paid. Accordingly, it might happen that a team does not complete a job fast enough and will not be paid at all. Hence, the auction jobs guarantee a safe income. Mission jobs on the other hand are assigned to a team. If the team does not complete a mission, it must pay a fine [1].

Generally, jobs are composed of the acquisition, assembly or transportation of items. Items must be gathered at resource nodes that must be explored beforehand. They can be distinguished in volume and in the way they are acquired. Some items must be assembled from other items. In this process the items are consumed and only the assembled item remains [1].

The assembly of items can be done in workshops. The agents can assist one another during the assembly Workshops are one type of the facilities that are randomly placed on the map. Other facilities are charging stations, shops, dumps, storages, resource nodes and wells. Agents can use charging stations to charge their batteries. Storages can be used to either deliver items in the context of a job or to store items, while dumps can be used to irrepealably destroy items. Shops can be used to sell items as well as to buy upgrades. Resource nodes are the places where items can be found [1].

Wells are generating points for the team during their lifetime. There are different types of wells differing in costs and efficiency. It is possible to dismantle the wells of the opposite team in order to decrease the points they earn [1].

In order to implement a solution for this contest, a strategy should be predefined. This will be shown in the second chapter. In the third chapter the implementation concept will be explained. The current state and the further work will be stated in the fourth chapter. Finally, the team structure will be shown in the fifth chapter.

2 General Team Strategy Concept

When developing a general strategy for the MAPC one has to consider several aspects as the given simulation is quite complex. In order to give a reasonable strategy, we divided the requirements into the following categories: Exploration, Sharing, Defining roles, Massium, Auctioning, Jobs, Items, Charging, Idle time, Failure, Wells, Upgrades and Disconnection.

The category Exploration describes the requirements that need to be fulfilled in order to achieve a fast and thorough exploration. Since the agents need to know where to find the resources in order to execute jobs and hence earn money and build wells to earn points, the exploration is crucial to the success of the team [1]. To cover all areas of the given map, we prefer to use a grid exploration. Based on the vision and the speed of an agent the grid points will be placed equidistantly. An agent will explore a grid point first if the distance is under a certain threshold. If there is no grid point close enough, the agent will target the area that has the most unexplored grid points. The agents share the grid point they are aiming towards in order to ensure that the grid points will not be explored redundantly. The main exploration will end as soon as at least one resource node for all the gatherable items can be found.

The category Sharing contains the requirements that need to be fulfilled in order to guarantee a universal perception of the world [1]. The information that must be shared among the agents are those concerning the items and their allocation, the explored resources and wells as well as the position of other teams agents.

The main requirement of the category Defining roles is to assign different responsibilities to the agents based on their characteristics. Drones are fast and not bound to the streets [1]. Therefore their main task will be the exploration of the map. The trucks on the other hand have a large storage and a high skill level[1]. Hence, their main task will be the gathering of the resources.

The main goal of the MAPC is to earn points by building wells. But in order to build the wells or in order to upgrade an agent, money in the currency Massium is needed [1]. The requirement of the category Massium is consequently the handling of the money. Massium will be reserved a priori so that the ongoing investments in wells and upgrades can be covered in time.

The money can be earned by executing jobs [1]. The category Jobs covers the requirements concerning the jobs. The first requirement is a cost-benefit-analysis. The agents must balance costs with rewards. Here, the costs are the steps and the resources which are needed to execute the job. Another requirement is the consideration of the abortion of jobs. If due to a failure the job cannot be done in time, the job has to be considered impossible and therefore be aborted. The abortion of jobs should also be considered when it is more profitable to execute for example a mission job instead of an already started auction job. When a new job comes in, it has to be analyzed on profitability and feasibility. Therefore, it must be decomposed into tasks depending on the current distribution of the items.

The items are needed for the execution of the job [1]. Therefore, the trucks will hoard as many items as possible. But they will not assemble them in advance since they cannot predict what kind of assembled item might be required for a job. If an agent has reached the limit of its capacity, it will first try to store some unallocated items in a truck. If this is not possible in a reasonable amount of steps, it will try to store the items in one of the storages. If this is also not possible, the agent will try to upgrade its load capacity. If this cannot be done, for example for a lack of Massium, the agent will try to sell the items. Simply dumping the items is the last resort of an agent.

As coordination strategy for the agents an auction process was chosen. The category Auctioning describes the requirements that need to be fulfilled in order to establish a well-functioning auctioning process. The first requirement is that all agents can be auctioneer and bidder. This is necessary so it can be ensured that the agents take turns in acting as the auctioneer. This helps avoiding bottlenecks. Which agent is the auctioneer for a job is decided based on the agents ID, on the jobs ID and on the overall number of agents. The auctioneer decomposes the job into several tasks and sends an auction request to all the bidders. The bidders must plan the execution of the task and supply the auctioneer with their plans. This is important as the auctioneer must choose the agent with the most profitable plan. The auctioneer then must inform the bidder of the assignment. The bidder must send an acknowledgement of the assignment in return. While the jobs and tasks are executed, the bidders send status updates to the auctioneer so it can monitor the progress.

The category Charging contains the requirements that need to be fulfilled in order to ensure that the agents do not discharge completely. Even though the agents could charge using their solar collectors, this is not desirable. Charging using the solar collectors takes up more time than using a charging station and have a higher probability to fail [1]. Therefore it is important to avoid a complete discharge. If the agent is executing a job, it is required that the charging is already considered in the plan of the agent. This aids avoiding delays due to unexpected discharges. If the agent is in a state of idle time, the charging is enabled by a linear charge activator. If the agent needs to charge it will move to the nearest charging station and charge until it is full.

Idle time is considered the time an agent is not executing jobs. Hence, the agents have different priorities so the idle time is not wasted. The first priority of the trucks is to gather resources. The first priority of all the other agents is to build own wells and to dismantle the wells of the other teams. If this cannot be done, they should store unallocated items in the trucks. Their next priority is to gather resources. If they cannot do this, they should charge. Their last priority is to upgrade their speed and charging capabilities to be faster in the future job plan execution. To ensure that wells are built even if there's almost no idle time, we decided to select an agent that is responsible for building wells, if a certain amount of Massium is reached.

The category Failure states the requirements concerning the failure of an action. Due to a probability in the simulation some actions will fail[1]. If that is the case, one duration step needs to be added to the current plan of the agent. Other plan durations should be updated, too, if necessary. If the failure occurs during the execution of a job, a status update must be send to the auctioneer.

The category Upgrades deals with the requirements concerning the upgrades of the agents' capabilities. Depending on the already stated responsibilities of the agents, some upgrades are more important for an agent than other upgrades. The drones, for example, have the main task exploring. Hence, they should upgrade their vision and their speed capabilities. This increases the efficiency of the exploration. The trucks should upgrade their skill capacity before gathering items. This accelerates the resource gathering. All agents except the trucks should upgrade their speed capability and their battery when they are in an idle time or if an upgrade is crucial for the successful execution of a job.

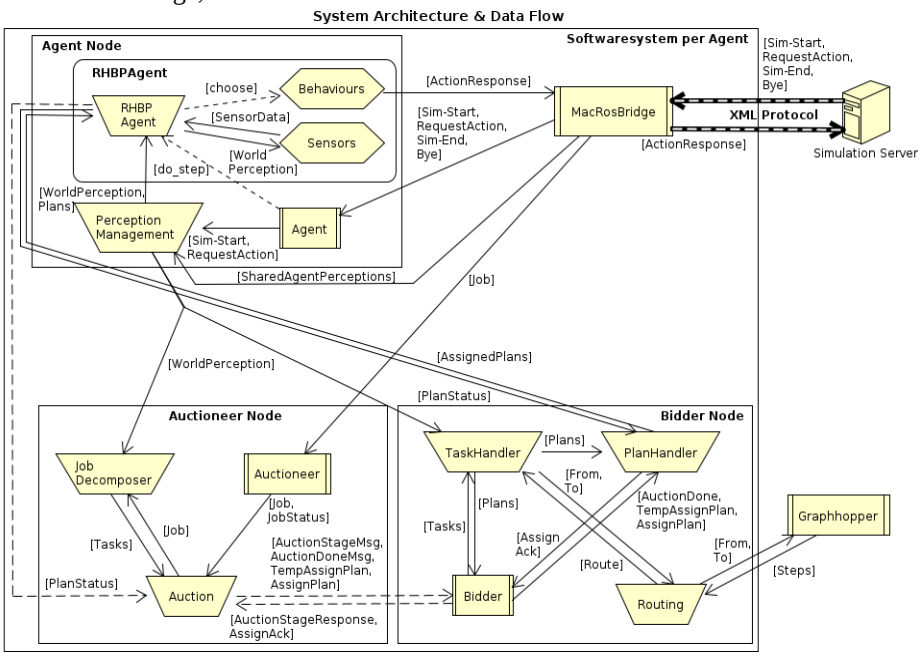
The strategy for the well building is primarily to place the wells on the borders of the map. There is less traffic at the borders of the map. Therefore the wells are not as easily detected and dismantled by other teams. Secondly, the agents should always build the well-type that will earn the most score in its lifetime.

Finally, the category Disconnection describes the requirements for the handling of a short disconnection of the agent. Even though the RHBP planner takes control after the reconnection, it might be necessary for the reconnected agent to request a world perception from other agents' perception manager. This prevents inaccurate world perceptions.

To sum it up, the general team strategy consists of defined roles and responsibilities of the agents, of strategic upgrading, charging and well building, and of a cost-benefit approach on the execution of jobs and tasks in the auctioning process. To cover unexpected problems, the handling of failures and disconnections is also part of the general team strategy.

3 Implementation Concept

In order to describe the implementation concept, first the architecture of the system must be described. Graphic 1 shows the architecture and the corresponding data flow. As shown in the graphic, the software system per agent consists of an Agent Node, the MacRosBridge, an Auctioneer Node and a Bidder Node.



Graphic 1 System Architecture and Data Flow

The Agent Node furthermore contains the Perception Manager, the Agent and a RHBP Agent. The RHBP Agent in turn contains the RHBP Agent, Behaviors and Sensors.

The Auctioneer Node contains a Job Decomposer, the Auctioneer and the Auction.

The components of the Bidder Node are the TaskHandler, the PlanHandler, the Bidder and the Routing.

Outside of the software system there are the simulation server and graphhopper, a tool that calculates route planning[2].

The simulation server communicates with the MacRosBridge, using a XML protocol. The simulation server sends the messages Sim-Start, RequestAction, Sim-End and Bye to the MacRosBridge. The message Sim-Start conveys the start of the simulation. RequestAction is a request for an action of the agent. Sim-End discloses the end of the simulation. The MacRosBridge forwards these messages to the Agent in the Agent Node. The Agent sends the messages Sim-Start and RequestAction to the Perception Management. So, the simulation server communicates with the MacRosBridge, which is forwarding the simulation and the corresponding information to the agent.

The Perception Management, that also receives the shared perceptions of the agents, can send the message WorldPerception to the RHBP Agent in the Agent Node as well as to the Job Decomposer in the Action Node an to the TaskHandler in the Bidder Node. The latter components need the world perception in order to decompose a job into tasks depending on the distribution of the resources and in order to make a plan for the task execution. The agents need a current world perception in order to find resources and wells.

The Auctioneer in the Auctioneer Node receives the message Job from the MacRosBridge. It sends the messages Job and JobStatus to the Auction. The Job Decomposer then receives the message Job from the Auction. After decomposing the job it sends back the message Tasks to the Auction. This communication needs take place so that the job can be coordinated among the agents through the auction. Accordingly, the job needs to be divided into the tasks.

The Auction in the Auctioneer Node and the Bidder in the Bidder Node exchange the following messages when an auction takes place: AuctionStageMsg, AuctionDoneMsg, TempAssignMsg and AssignMsg are being sent from the Auction to the Bidder. The messages AuctionStageResponse and AssignAck are being sent from the Bidder to the Auction. The auction is done in several stages since several tasks of a job must be auctioned. Once the auction is done, the tasks are temporary assigned by the auctioneer. The bidder must send an acknowledgement to the auctioneer, so the assignment can be set definitely.

The Bidder in the Bidder Node sends the message Tasks to the TaskHandler which in turn sends the message Plans to the Bidder and to the PlanHandler. Tasks can be assigned, temporarily or definitely, or unassigned, and are handled by the PlanHandler. The TaskHandler is creating plans in order to solve the tasks. The TaskHandler can also send a message to the Routing. This message contains the starting point and the goal. The Routing forwards this message to the Graphhopper and in turn receives the steps needed for the route from Graphhopper. The Routing then can send the

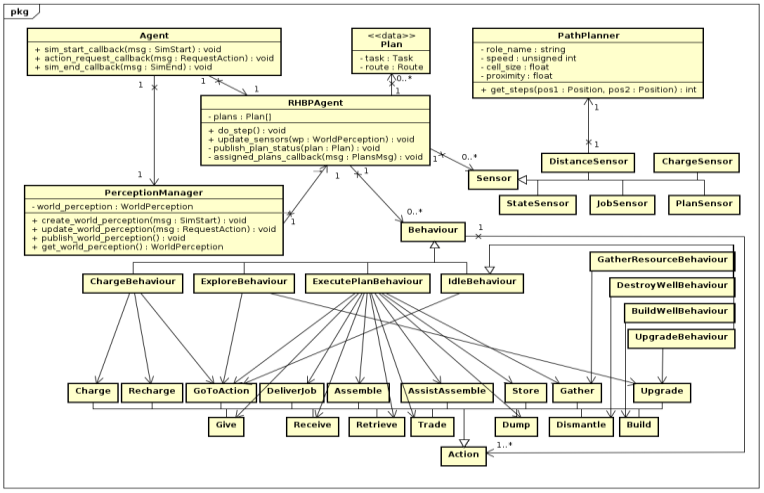
Route to the TaskHandler. These messages must be exchange in order to enable the bidder to make plans for the tasks. Since the tasks also contain the ways the agent has to take, for example to deliver an assembled item to a storage, the routing is crucial to the planning.

The PlanHandler in the Bidder Node exchanges the following messages with the Bidder: AuctionDone, TempAssignPlan and AssignPlan are being sent from the Bidder to the PlanHandler. The Bidder receives the message AssignAck from the PlanHandler. These messages are sent to ensure that the plans provided for an auction are actually are scheduled once the auction is completed and the task is assigned to this bidder and acknowledged by it.

The PlanHandler sends the message AssignedPlans to the RHBP Agent. The RHBP Agent receives data from the Sensors. Depending on the plans and the world perception it then chooses a behavior.

The component Behaviors sends the message ActionResponse to the MacRosBridge which in turn forwards it to the Simulation Server using the XML protocol.

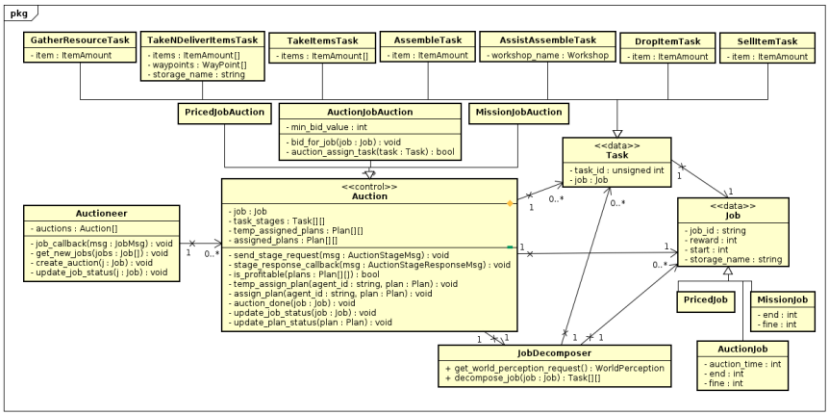
The Graphics 2, 3 and 4 display the different class diagrams. Graphic 2 shows the class diagram of the Agent. As shown in the diagram each agent has one RHBPAgent and one Perception Manager. The RHBPAgent has a Plan, several Sensors and several Behaviors. The Sensors can be StateSensor, JobSensor, PlanSensor, ChargeSensor or the DistanceSensor. The Behaviors can be ChargeBehaviour, ExploreBehaviour, ExecutePlanBehaviour and IdleBehaviour. The different types of Behaviour can consist of several Actions as shown in the diagram.



Graphic 2 Class Diagram of the Agent

Graphic 3 displays the class diagram of the Auctioneer. An Auctioneer can have several auctions. An auction for a job can be a type of the following auctions: PricedJobAuction, AuctionJobAuction or a MissionJobAuction. Those Jobs can be PricedJob, AuctionJob or a MissionJob. Each Auction has one Job. Furthermore, each Auction can have several Tasks. There are several types of Tasks such as an Assem-

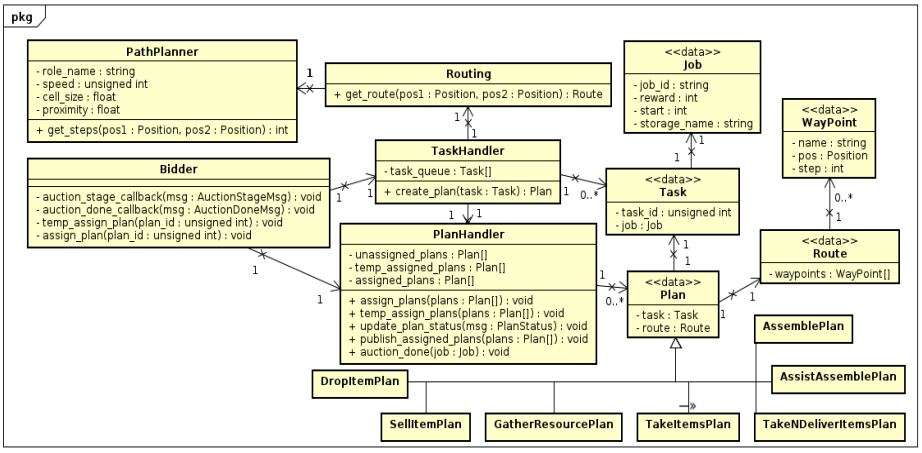
bleTask or a GatherResourceTask. The Tasks cover all possible tasks a job could be decomposed into. Each Task belongs to a Job.



Graphic 3 Class Diagram of the Auctioneer

The class diagram of the Bidder is displayed in Graphic 4. Each Bidder has a TaskHandler and a PlanHandler. The former is used to figure out how an agent can perform a particular task, while the latter is used to manage the temporary assigned, assigned and unassigned tasks, and share them with the agent node for the plan execution. The TaskHandler can have several Tasks. Each Task is related to a Job. Furthermore the TaskHandler has a Routing that is related to a PathPlanner. The PlanHandler on the other hand uses several Plans which are related to Tasks. There are different types of Plans like the DropItemPlan or the AssistAssemblePlan. The plans are covering every step that might be necessary for the execution of a task, so the plan can be send to the auctioneer who in turn needs to evaluate all the plans it receives.

3

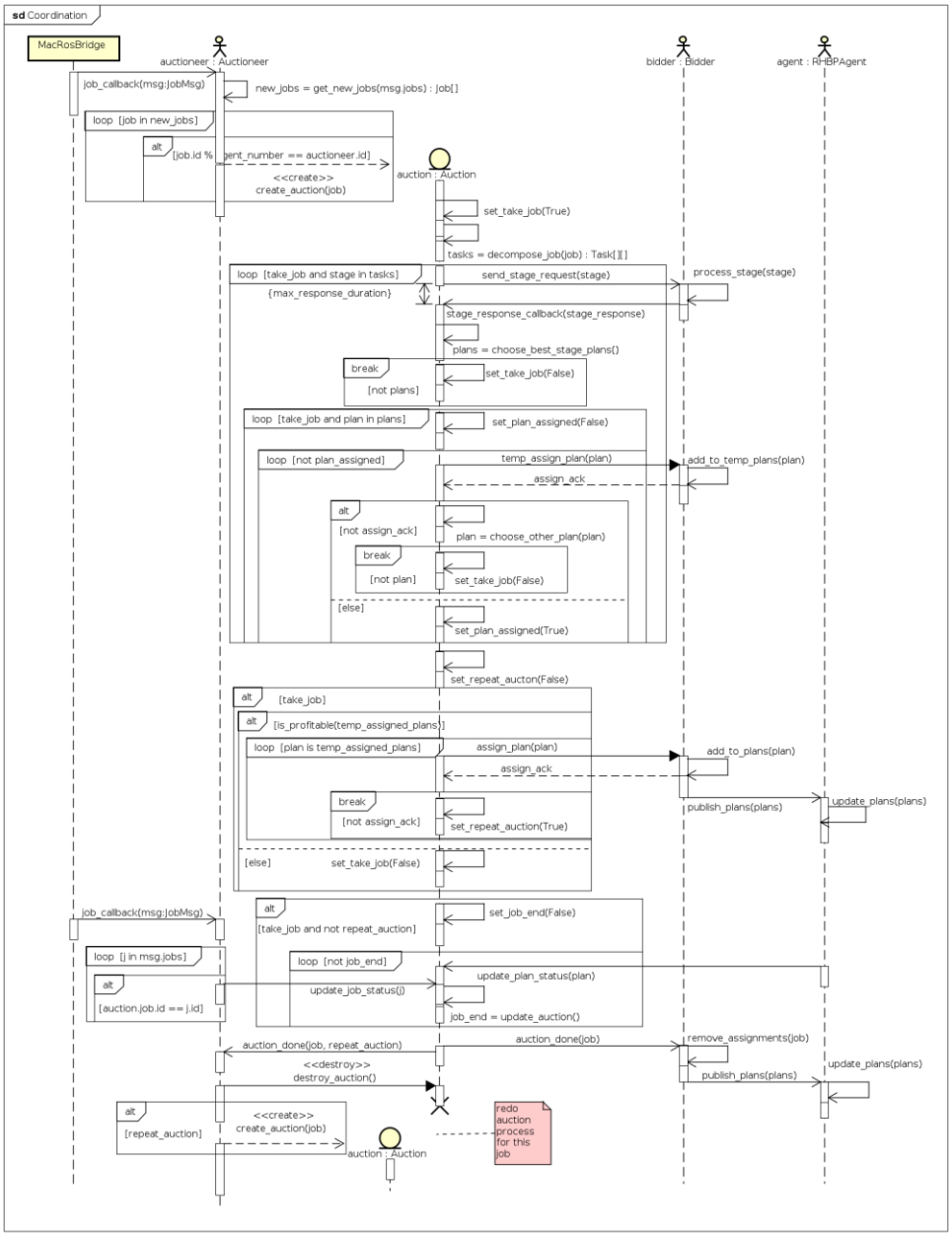


Graphic 4 Class Diagram of the Bidder

Graphic 5 shows the sequence diagram of the auction. The auction is initiated by the MacRosBridge which informs the auctioneer about the job. The auctioneer then starts an auction and decomposes the job into tasks. The tasks than are being auctioneered in several stages. It is important to use several stages for some tasks are depending on former temporary assigned tasks. A chronological hierarchy is necessary. The bidder, informed by the auctioneer about the auction, provides plans for the execution of a task. The auctioneer then chooses the best plan among the provided plans for each task and informs the bidder about their assignments. The bidder then has to acknowledge the assignment. If the acknowledgement is not received by the auctioneer, the auction must be repeated. This is necessary in order to ensure the completion of the jobs. An auction is aborted, when it is not profitable or not feasible in time.

This procedure equates to the Contract Net Protocol (CNP) as developed by Reid G. Smith[3]. The CNP is a high-level protocol for the nodes of distributed systems, such as the presented MAPC simulation. Using the communication between nodes the CNP enables cooperative task execution. In the CNP, the manager sends out a task announcement containing the task description and a time frame. The bidders then send the manager a bid for this specific task announcement. Successful bidders are informed about the successful auction. While the contract between manager and bidder lasts, the bidder is the contractor. While a task is done and when a task is completed, the contractor sends a report to the manager.

The CNP is similar to the proposed auctioning process. The manager equates the auctioneer who sends out tasks that the bidders can bid on. The auctioneer also informs all the successful bidders about their assignments and awaits their acknowledgement. The bidders in the proposed auctioning process also inform the auctioneer about the progress of a task execution. Even though the auctioning process was designed by the authors specifically to ensure the coordination of the agents in the MAPC, in the end it equates the CNP. Therefore the CNP is stated as the chosen coordination algorithm.



Graphic 5 Sequence Diagram Auction

4 Current State

Currently, the following things are already implemented. The access to Graphhopper has been established. The exploration has been improved so that the grid points are already in use. Furthermore, a manager for perception handling has been implemented. It ensures the saving and updating of the facilities, the explored resource nodes and the information of all agents. It also contains the information of all available items and their allocation as well as the Massium allocation.

Additionally, the charging has been improved and the distance is now measured in simulation steps instead of meters. Moreover, the auctioneer node and the bidder node have already been setup, requests can be broadcasted and corresponding jobs can be selected by an agent.

The next requirements that need to be fulfilled are the decomposition of the jobs, as it is crucial to the auctioning process, the gathering of resources and the implementation of the TaskHandler and the routing. The TaskHandler is needed to solve tasks by creating plans. The routing on the other hand is needed in order to create plans. These requirements have the highest priority 1. They are necessary for a rudimentary functionality of the system.

Subsequently the roles and their priorities as described above, the finalized auctioning process, including the cost-benefit-analysis, as well as the implementation of the storage and capacity management must be done next. This includes the storage of items in the trucks. Additionally, the assembling of items must be implemented. Also, the management of the Massium must be established. These requirements are graded with the priority 2 for they are needed to enable the achievement of the overall goal of the contest.

The management of the upgrades must be implemented next. Furthermore, the building and the dismantling of the wells will be implemented. Those requirements have the priority 3. Even though the wells are generating the points that are needed to achieve victory in the contest, the priority for job-related requirements is higher. Without the money earned in the jobs, the wells cannot be built.

Finally, the handling of failure and reconnections as well as further improvements in the exploration and charging must be implemented. Those requirements have the lowest priority level 4, because they are merely improvements and are not absolutely crucial for the overall success. If unexpected complications occur, these requirements could be aborted.

5 Team Structure

As the teams were asked to choose a team name, this group now operates under the name “Dumping to gather”. The phrase itself is a reference to the Multi-Agent Progamming Contest. Sometimes an agent has to dump an item that is currently not needed, in order to gather another item that is more valuable. So, the agent is dumping one thing to gather another. Besides the reference to the simulation itself, the meaning of dumping one thing to gather another also applies on the group project. Sometimes

one has to give up on one approach to work on a better one. This implies the necessity of an open mind and flexibility in such projects.
The name was chosen in this specific wording because it is an annomination, a play of words that sound alike. The phrase “Dumping to gather” sounds like “Dumping together” insinuating collaboration. This collaboration not only reflects the team work of the agents, that need to work together in order to win, but also is a hint on the combined effort of this group.
The team consists of three members: Josephine Krause, Marc Schmidt and Muzammal Hussain. In the context of the project the following responsibilities were assigned to the group members:

Josephine Krause will be the group coordinator. She will be responsible for the monitoring of the progress and of the issues concerning the programming and documentation tasks. Preparing the weekly progress will also be her task from now on.

Marc Schmidt is the Head of Programming. He has the authority over decisions concerning the programming. He will also be responsible for the monitoring of the continual improvement of the code.

Muzammal Husain will be the Scientist. It is his responsibility to conduct necessary research and present his results to the group so every decision can be made in a reasonable way.

Even though the team members have their specific responsibilities, the main tasks of the project, the programming and the preparation of the milestone and the final report and presentation, still are done by the entire group.

References

1. MASSim Scenario Documentation, https://github.com/agentcontest/massim/blob/master/docs/scenario.md#roles, last accessed 2018/06/17.
2. Graphhopper Homepage, https://www.graphhopper.com/, last accessed 2018/06/17.
3. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Trans. on Computers C-29(12):1104-1113.