SOFTWARE DEVELOPMENT AND ANALYSIS OF THE WAREHOUSE PROBLEM USING A MULTI-AGENT SYSTEM IN JASON

REZART TABAKU

BACHELOR'S DEGREE

EPOKA UNIVERSITY

2016

SOFTWARE DEVELOPMENT AND ANALYSIS OF THE WAREHOUSE PROBLEM USING A MULTI-AGENT SYSTEM IN JASON

By

REZART TABAKU

Thesis submitted to the Faculty Architecture and Engineering, Epoka University,

in Fulfilment of the Requirement for the Degree of Bachelor of Science

June 2016

ABSTRACT

Faculty of Architecture and Engineering

Advisor: Igli Hakrama

A Multi-Agent System (MAS) is a system that depicts a specific environment as comprised of multiple agents that are intelligent, possibly self-learning, autonomous, perceptive by way of sensors to the changes occurring in the environment and capable of interaction with other agents present and the environment itself. MASes are used in the simulation of many problems that involve multiple entities that will be cooperating or competing for a specific resource based on their assigned roles and functionalities. One such problem is the warehouse problem.

The warehouse problem is a blanket term that mostly generalizes any logistics concerns regarding a warehouse: item allocation and distribution, supply and exit points, shelf types, placement and layering, routing of various kinds of transporters and so on. Due to these concerns present in a warehouse planning process, it is not unreasonable to simulate a warehouse environment as a MAS, where objects and actors can be translated into agents, in order to analyze and possibly optimize the different aspects concerning warehouse operations. This paper is going to outline a warehouse environment simulation that is conceptualized as a MAS, created using Java and JASON, a framework for creating agents using the belief-desire-intention (BDI) model, where emphasis is placed on agent communication and synchronization and path routing of the transporters.

Keywords: JASON, Agent, Multi-Agent System, Warehouse

ABSTRAKT

Fakulteti i Arkitekturës dhe Inxhinierisë

Udhëheqës: Igli Hakrama

Një Sistem me Shumë Agjentë (SSA) është një sistem i cili një mjedis veprimi të caktuar e paraqet të përbërë prej një sërë agjentësh inteligjentë, të pavarur për të vepruar, të aftë për të mësuar, për të perceptuar me anë të sensorëve ndryshimet e ndodhura në mjedis dhe për të ndërvepruar me agjentët e tjerë të përfshirë si dhe me vetë mjedisin. SSA përdoren në simulimin e shumë situatave të cilat përmbajnë një grup njësish me rolet dhe veprimet përkatëse që do të bashkëpunojnë apo konkurrojnë për resurset në mjedis. Një prej këtyre situatave është problemi i magazinës.

Problemi i magazinës është një përkufizim i përdorur për të përmbledhur vështirësitë e hasura në logjistikën dhe veprimtaritë e një magazine, p.sh. shpërndarjen dhe vendosjen e artikujve, pikat e furnizimit dhe daljes, llojet, vendosjet dhe shtresimet e rafteve mbajtëse, rrugët që mbajtësit do të përshkojnë dhe shumë të tjera. Në prani të këtyre elementëve, për të analizuar dhe mundësisht për të optimizuar mjedisi i magazines mund të hartohet si një SSA në të cilin objektet ose vepruesit mund të trajtohen si agjentë pjesë të mjedisit të simuluar. Ky artikull synon të paraqesë një mjedis magazine të simuluar në një SSA të krijuar duke përdorur Java dhe JASON, një platform e gatshme për krijimin e agjentëve sipas modelit bindje-dëshirë-qëllim (BDQ), ku rëndësi i është kushtuar sinkronizimit dhe komunikimit midis agjentëve dhe përshkimit të rrugëve të transportuesve të artikujve.

Fjalë kyçe: JASON, Agjent, Sistem me Shumë Agjent, Magazinë

ACKNOWLEDGEMENTS

I will give my heartfelt thanks to my family who have supported me this entire time, for always encouraging me to perform better and to never give up. I would also like to express my sincerest gratitude to the esteemed professor and my advisor Igli Hakrama, whose guidance and counseling has constantly helped me progress through and complete the thesis.

I would also like to give special thanks to all my professors throughout the years, who have gradually shaped my skillset and to all the department peers whom I went through this journey of a university together.

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Epoka University or other institutions.

Rezart Tabaku

June 2016

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

There exists a plethora of complex environments present in natural or manmade habitats, wherein a large number of entities reside and interact with each other and with the environment itself. This complexity has prompted many disciplines of science, such as biochemistry, logistics, electrical engineering, social sciences, and informatics to start analyzing these environments by simulating them at a smaller scale in an attempt to learn about them and derive meaningful conclusions from them. This paper is going to focus on and analyse one instance of these systems: the warehouse problem and the method that we will use to model the problem is the Multi-Agent System (MAS).

The warehouse problem is a blanket term that mostly generalizes any logistics concerns regarding a warehouse: item allocation and distribution, supply and exit points, shelf types, placement and layering, routing of various kinds of transporters (Bowersox, Closs & Cooper, 2002). In a typical warehouse environment several entities communicate with each other, e.g. the item transporters will be notified by the coordinators who will be conveying the destination of the incoming supply, after which the transporter will then determine the route to take and the shelf to occupy.Therefore it is not unreasonable to conceptualize the warehouse problem as a MAS. A MAS is considered a system that consists of a number of agents which interact with one another by exchanging messages through some computer network infrastructure (Woolridge, 2009). An agent is an entity capable of perceiving the environment it is locatied in and acting upon that environment (Russell and Norvig, 2003). An agent is considered intelligent and as such is able to

exhibit autonomy and self-learning behavior. By building agent behaviours, placing these agents in the environment and observing the agents' interactions with one another, accurate models of an environment can be derived.

MAS and other agent-based methods, such as Agent-Based Modeling (ABM) have been used extensively in the simulation of various environments in scientific research areas such as medicine (An, 2004) to simulate interactions between antibodies and viruses, transport, infrastructure and urban design (Guzy et.al, 2008; Agolli & Hakrama, 2015), economy studies (Tesfatsion, 2006; Hakrama, 2014) where agent-based computational economics are used to model and analyse financial phenomena, social studies (Smith & Conrey, 2007; Jansen & Osstrom, 2006) and so on. The purpose of this paper is to propose a model of a warehouse built as a MAS where the item transporters, item location coordinator, the supplier and the dispatcher will be translated into agents, in an attempt to merge already established application development tools with new technologies that use agent-oriented models.

LITERATURE REVIEW

According to Gu, Goetschalckx & McGinnis (2007), performance of warehousing procedures are determined based on the warehouse operations such as receiving, storage, order picking and shipping, and on the warehouse design, namely the operation strategy employed, sizing, layout of the department and the proper choice of equipment. These are the requirements that will then affect the conceptual model used for simulating the warehouse environment (Onggo, Gunal & Maden,2008). Various simulation methods have been applied in an attempt to model a warehouse environment. Discrete Event Simulations (DES) have been the most widely used approaches for modeling a warehouse using either ARENA (Detty & Yingling, 2000;Liong & Loo, 2009), or C# libraries SharpSim and WareLib (Ceynal, Gunal & Bruzzone, 2012) where the events occurring in the system are executed and resolved in sequence at discrete timestamps. Another approach, the one that this paper is going to implement, is simulating the warehouse problem in a MAS. In this case the simulation requirement is to properly define the entities that are going to comprise the simulation, their specifications and their behaviours. We will build the warehouse MAS in Jason (Bordini, Hübner & Wooldridge, 2007).

Jason is a Java-based interpreter for an extended version of AgentSpeak (Rao, 1996), a Prolog-like programming language designed for Agent-Oriented Programming. In AgentSpeak, agents are created using the Belief-Desire-Intention (BDI) model (Rao & Georgeff, 1995). Beliefs are perceptions about the environment, previous events or other agents that the agent has gained throughout its lifetime. Desires are all the agent's

possible branching decision paths, the set of all the actions the agent can take depending on the impact of other agents or the environment. Intentions are the immediate plans the agent is going to execute. BDI is a dynamic model that enables agents a high degree of learning and complex behaviours based on its experiences. Jason has been deployed in combination with two other platforms: Moise (Hübner et.al., 2010), used to create agents as part of an organization and Common Artifact infrastructure of Agents Open (CArtAgO), used to build environments (Ricci & Omicini, 2006) to form JaCaMo[24], a (Fahad, Boissier, Maret & Gravier, 2012 ;Hakrama & Frashëri, 2015). Evidently Jason is a flexible framework. Owing to that, we have used Jason in combination with Java and JavaFX to implement the MAS in effect.

Cossentino, Lodato, Lopes & Ribino (2012) have performed a simulation of a warehouse using a MAS built with Jason. In their model, the agents were assigned in a two-level hierarchy based on the duties given to the agents within the warehouse, and the warehouse environment itself was partitioned via waypoints, creating a graph of routes, in order to set the paths available to the item transporters. Our model will have a similar hierarchy of the agents, however the warehouse is modeled as a grid space instead, and the transporters travel along the grid cells.

SYSTEM ANALYSIS AND DESIGN

The major issues the warehouse problem is concerned with are inventory accuracy and location, warehouse layout and space utilization, picking optimization and redundant processes. In this section we are going to provide a complete and detailed analysis of the agents and the environment placed in the MAS designed to simulate the warehouse and address these issues. The agents are grouped in two tiers: the management tier and the transport tier. The environment will be a warehouse area of a predetermined size, where one loading area, one dispatch area and a varying configuration of storage shelves have been placed. The warehouse layout is chosen at startup of the RMI Server. The management tier is composed of:

- the coordinator agent, overseeing the item orders, item requests, shelf allotting and item locations;
- the supplier agent, storing the inbound items and issuing requests of new inbound items to the supplier transport agents;
- the dispatcher agent, the unit responsible for issuing item removal and dispatch requests to the dispatch transport agents.

There are two types of transport agents: workers and forklifts, and each transport unit is assigned one of two sets of behaviours: packer or extractor. A packer transporter will carry inbound items from the supplier to the item's assigned shelf. An extractor transporter is going to carry the outbound items from their shelves to the dispatching area per the dipatcher's agent requests. The transporters are assigned resting positions at

the initialization of the environment. The packers start closer to the supplier, and the extractors start closer to the dispatcher.

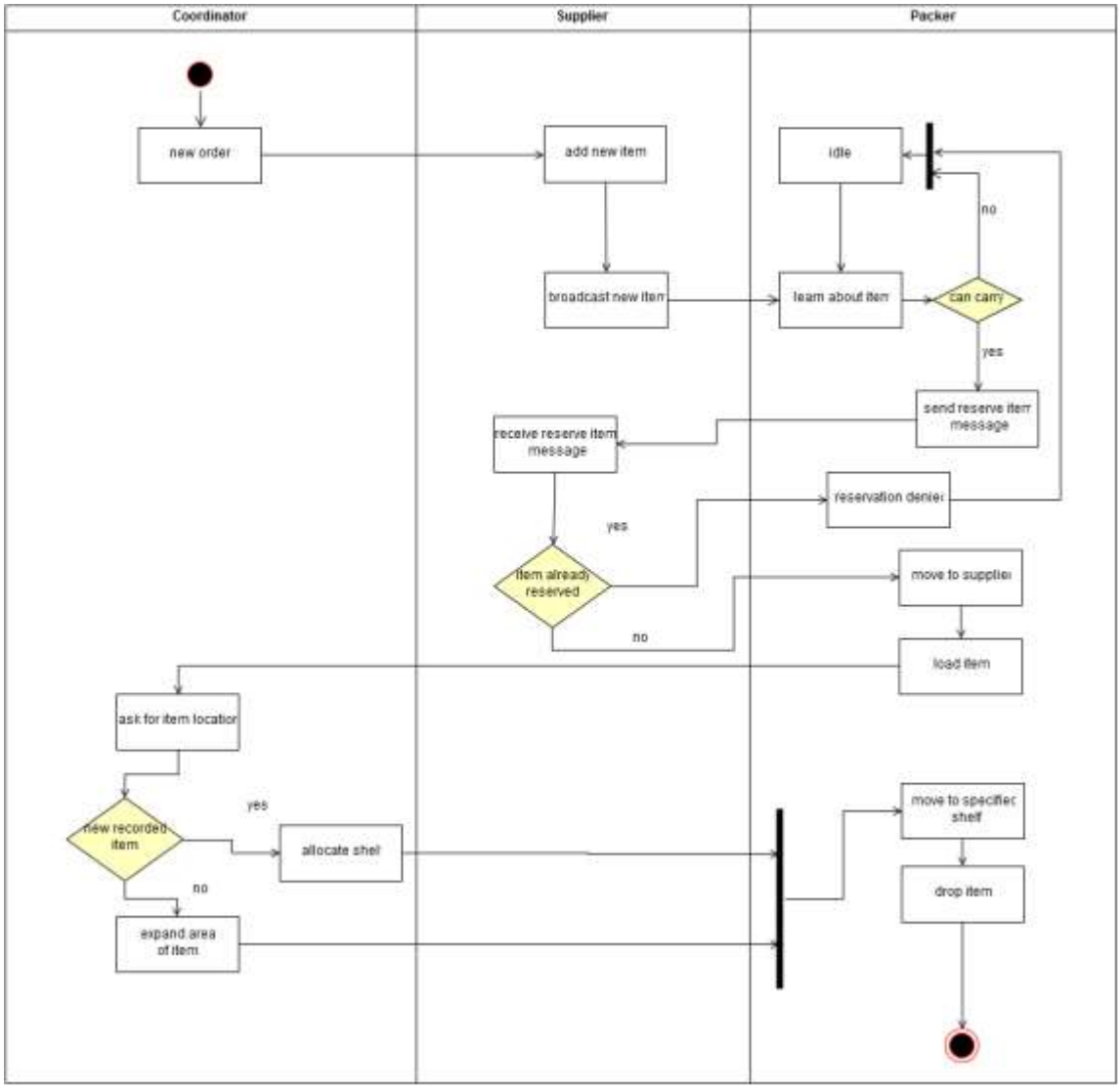

Fig. 1. Flow of operation of item supplying

The coordinator agent's role is as the central information unit in the environment. The coordinator will be placing new orders for items, after which it is going to notify the supplier regarding the new item parameters. The order for an item will be retrieved from

a file. The supplier, after enqueueing the ordered item, will then broadcast to all the packer units the new item parameters, namely the weight and name of item. The packers that are able to respond or carry the item will send a “reserve item” message in order to prevent conflicts with other packers during item loading. The supplier will approve the first packer to send the message and deny reserve requests from the rest. The responding packer will then go to the loading area, load the item and then ask the coordinator for the item’s location. The coordinator will check its beliefs for the item’s location. If this item has already been ordered before, a shelf has been allocated for the item. Instead the coordinator will try to expand the area allocated for this item.

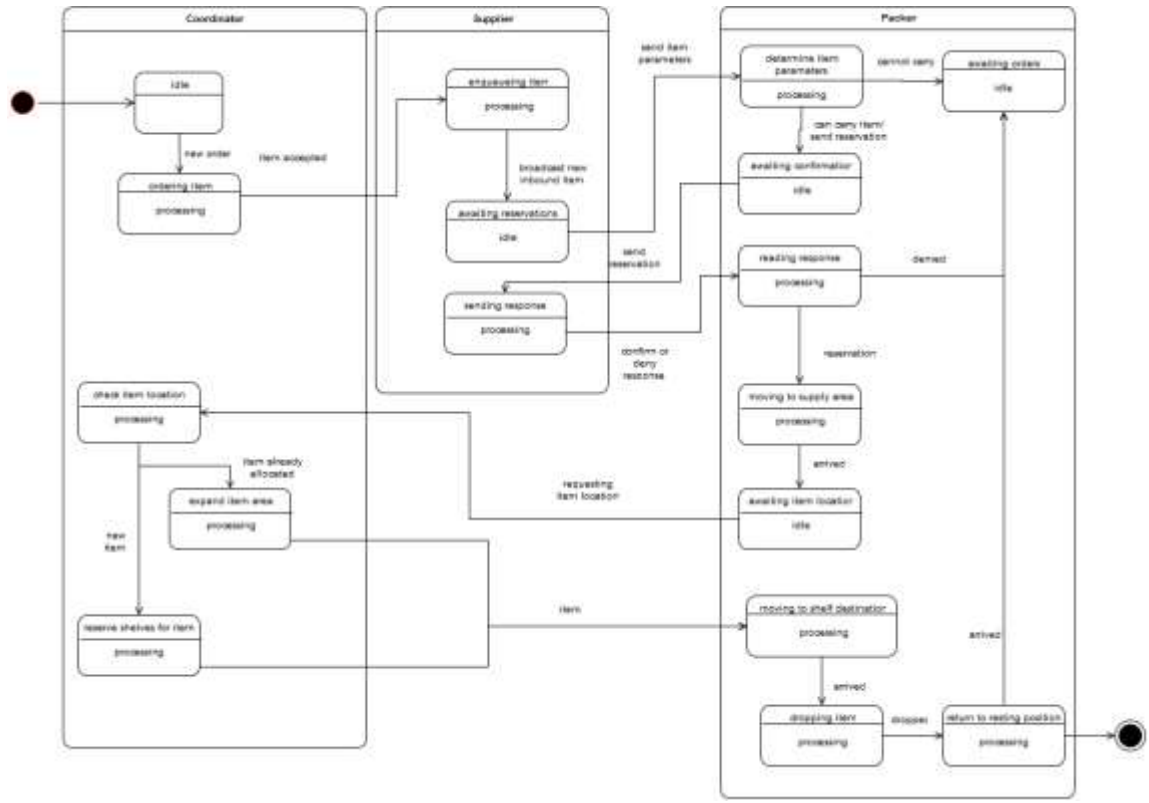

Fig. 2. The states of the agents involved in the supply

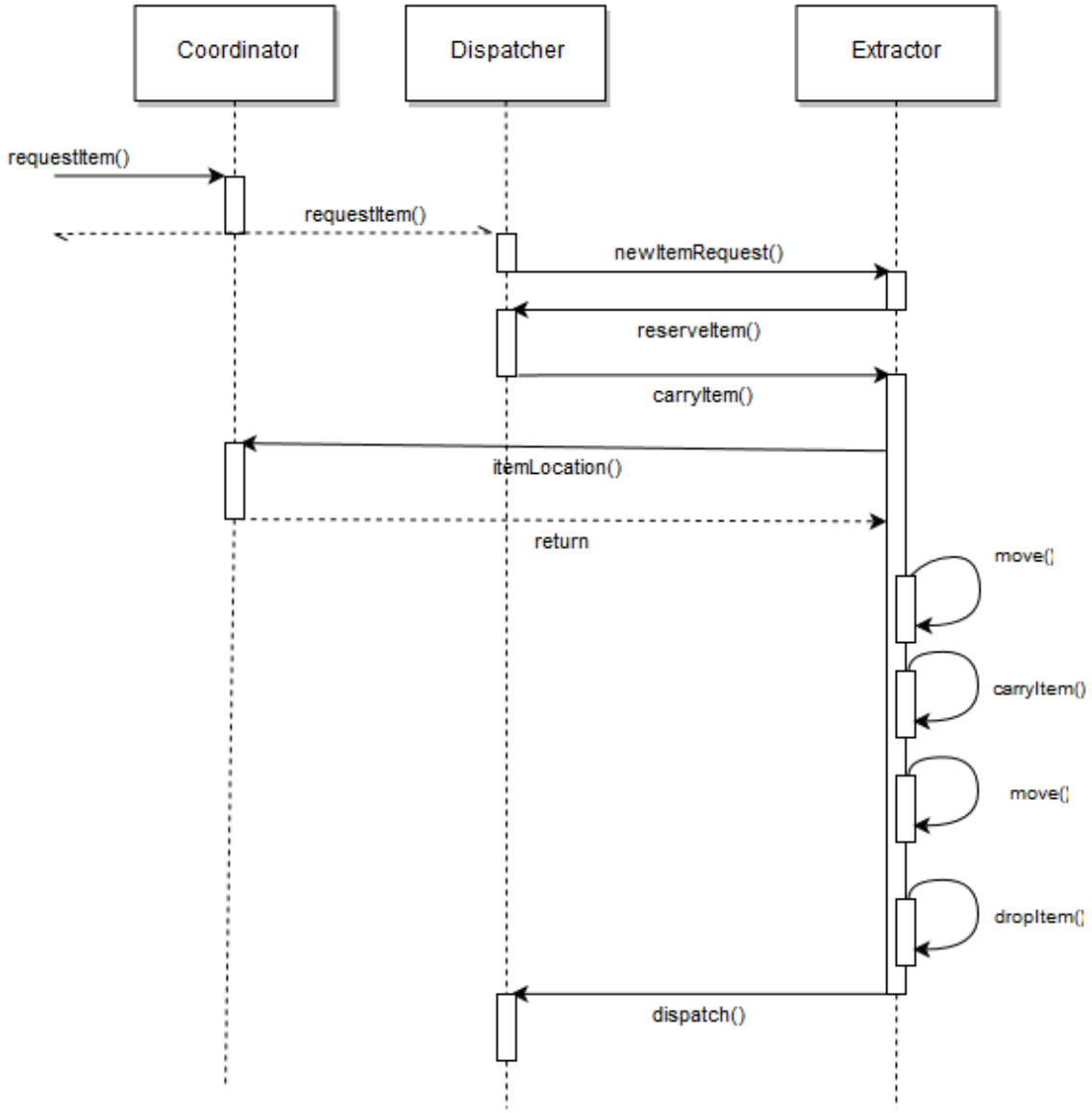

Fig. 3. Sequence of events after an item request

After trying an expansion, the coordinator will give the packer the destination shelf where the item is to be stored. The packer will then move to the shelf, drop the item by storing it in the destination shelf and then return to its resting spot, where it will be awaiting new orders. During this time, the coordinator will be accessing another file to read the items selected as outbound items. When an item is requested for an outbound

process, the coordinator will notify the dispatcher agent. The dispatcher agent will then broadcast the requested item's parameters to the extractors. The extractors will ascertain whether they can carry the item. When they do, they will send a reserve item message to the dispatcher. The dispatcher will reserve the item in first-come first-served manner, sending a confirmation message and the location of the shelf to the first extractor to send the message, and will respond with a deny message to all other requests. Then the extractor is going to travel to the shelf holding the item, load the item, travel to the unloading area of the warehouse and return in its resting position. The reservation message procedure is implemented to avoid conflicts during item retrievals. Since the requests from either the supplier or the dispatcher are broadcast to all the appropriate transporters, there is a need to synchronize the movement of the transporters. Hence the supplier or the dispatcher serves as the enabler for the transporters, deciding which of them will be selected for an item transport.

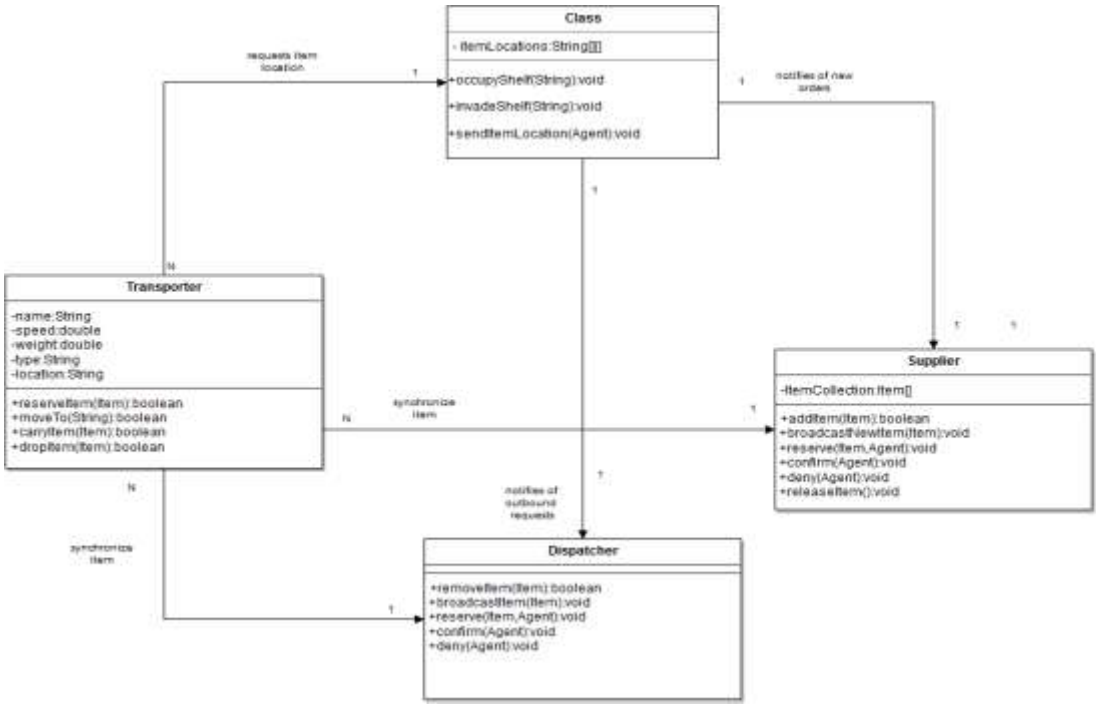

Fig. 4. The UML Diagrams of the agents

The parameters that are to be measured are the time that the items will spend while in the loading area. This is to measure the efficiency incurred by the use of the single loading area. The application is also going to measure the utilization rate of each transporter in order to calculate the cost of a single transporter and of the transporter scheme, that is whether it is effective to use the given number of transporters. These calculations will be used to determine the effectiveness of the chosen arrangement of the transporters relative to the size and layout of the warehouse.

IMPLEMENTATION

So far this paper has described the agent outline of the MAS and their purposes. In this section the application developed will be described from the programming point of view. We are going to outline the modules comprising the application, the systems built to support the user interface and the communication with the Jason platform and the classes used to generalize the objects and the agents's behaviours.

The system is composed of five communicating modules (see Fig.8):

- the agent behaviours built with AgentSpeak ,
- the environment parser,
- the RMI server interface, called FXServer
- the Model of the data,
- the View.

A. AgentSpeak

The agents' behaviours are defined in the AgentSpeak language provided by the Jason platform. These behaviours are the core of the simulation, where all the events relevant to the simulation begin. In accordance with the BDI model, the agents are initiated with a few basic plans and initial beliefs. Plans of an agent are executed only after the appropriate beliefs relevant to the intended plan or lack thereof are verified. The AgentSpeak syntax of a plan is:

+!newItem(Weight,Item)[source(Ag)]:**not** carrying **&** weightCanCarry(W)**&**Weight <= W

```
<-      +waitingResponse;
        .send(Ag,achieve,reserveItem(Item));
```

During a plan, the agent will perform actions such as execute other plans, modify their belief base or act on the environment. In the above example the plan entails adding the belief *+waitingResponse* to the agent's belief base. After that the agent will send a message to agent *Ag*, the agent whom the plan was assigned from, telling them to execute the plan *reserveItem(Item)*. Belief modification and plan execution is internally handled by Jason, however actions on the environment have to be interpreted and translated into Java functions.

```
1  // Agent coordinator in project warehouse
2
3  /* Initial beliefs and rules */
4
5  /* Initial goals */
6  !order.
7
8  /* Plans */
9
10 +!order: not no_more_items
11 <-  warehouse.nextItem(a,I);
12     warehouse.nextItemWeight(a,W);
13     .send(supplier,achieve,addItem(W,I));
14     .at("now +4 s","+!order");
15 .
16
17 +!order:  no_more_items
18 <-  .print("No more items");
19 .
20
21 -!order: true
22 <-  .at("now +4 s","+!order");
23 .
24
25 +!transport(Item)[source(Ag)]: location(Item,Loc)
26 <-  invade(Item,Loc);
27     .send(Ag,achieve,transportItem(Loc));
28 .
29
30 +!transport(Item)[source(Ag)]: not location(Item,_)
31 <- !shelf_occupy(Item,Ag);
```

Fig. 5. The AgentSpeak code of the coordinator

B. The environment parser

The parser is a Java class that is associated with the Jason environment. This class is used to initialize environment and agent data and retrieve said data for usage in the Java workspace. The parser's main function is to interpret the actions called by the agents on the environment and modify the belief base of the relevant agents after each event. The parser creates the FXServer object necessary to communicate with the RMI Server. Through this object the parser is going to issue requests on behalf of the agents and is going to retrieve the server's response.

```
1  MAS warehouse {
2
3      infrastructure: Centralised
4
5      environment:
6          warehouse.WareEnv("warehouse.mas2j",30,30)
7
8      agents:
9          coordinator;
10         dispatcher;
11         forkExit ;
12         humanExit ;
13         supplier;
14         human ;
15         forklift ;
16
17
18     aslSourcePath:
19         "src/asl";
20 }
```

Fig. 6. System initialization

C. RMI Server

In order to allow communication of the Jason module with the JavaFX visual representation tools, it was necessary to use Remote Invocation Method (RMI) classes in

a client-server approach, where the WarehouseEnvironment class acts as the client and the Model and View classes act as the server. RMI is an API developed by Oracle[27] in which a client sends is able to call objects remotely from a host or server.The motivation behind using RMI is to have an always-active host capable of performing operations on the JavaFX Scene built for the simulation. Any JavaFX Application subclass creates separate threads in the JVM, threads that are isolated and immutable, that is the graphical elements cannot be modified if they are targeted outside of the Application. With RMI, this drawback is circumvented. The environment parser issues requests to the FXServer, detailing the actions the movement of the agents, the inbound or outbound item parameters adn the shelf allocation modifications issued by the coordinator. The server houses the JavaFX Application, thus being the class with entry privilege on the JavaFX elements.

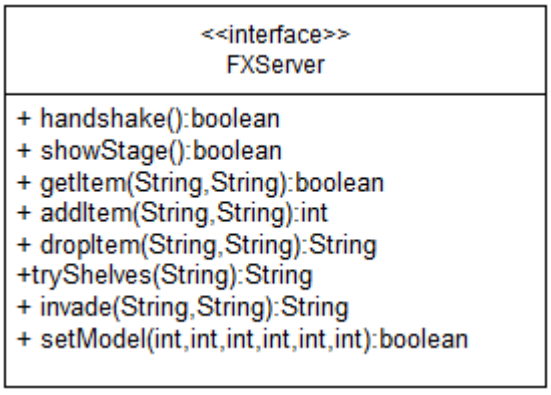

Fig. 7. The Interface of the RMI server

Connection to a RMI server is established via ports. Hence before attempting a connection a port must be bound to the server instance name:

```
Registry reg = LocateRegistry.createRegistry(5099);
reg.rebind("fx", new Server());
```

Then the parser is going to connect to this server by searching for the specified name in the appropriate port:

```
server = (FXServer) Naming.lookup("rmi://localhost:5099/fx");
```

Then communication can proceed.

D. The Model

The Model is responsible for the data management concerned in the simulation. The model keeps environment general data such as size, number of agents, items, shelves, loading and unloading points. The Model builds data structures for the agent movements, items, item transportation and allocation, shelves and their item lists. The Model classes perform all the operations on the data requested through the FXServer. These ongoing events are then shown by the View.

E. The View

The View module creates a visual representation of the warehouse simulation and the events occurring within. The simulation interface is built using the JavaFX drawing tools for the images, shapes and the animations. JavaFX is a robust framework for building interfaces, hence why it was chosen as the interface drawing framework. The View

initially creates the static warehouse look based on the model data, i.e. it draws the agents, the grid view of the warehouse, the shelves as instructed by the FXServer at the start of the simulation. Afterwards, as the simulation continues and data in the model are modified, the agents move in the simulation or items are transported in or out the warehouse, the changes are drawn and animated.

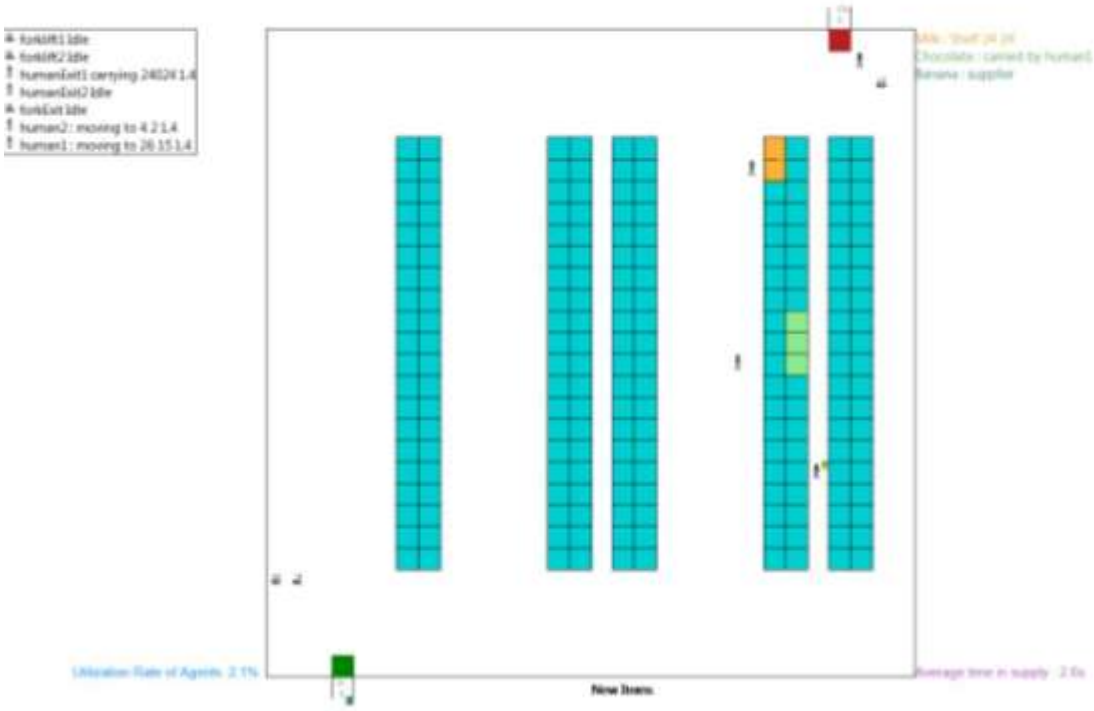

Fig. 8. The view of the simulation

CONCLUSION AND FUTURE WORK

We have created a MAS that simulates the logistics of a warehouse environment. The system was developed in the Jason platform in conjuction with JavaFX. The agents' behaviours were built in the AgentSpeak language interpreted by Jason. Their actions are translated into Java by the EnvironmentParser class and inputted in the Model through the RMI Server. After processing, the Model then provides the data to the View to provide a graphical representation of the events occurring in the simulation.

The simulation was completed using one loading area and a scripted order of items, with varying quantities and types of transporters. It yielded adequate results regarding warehouse operations, specifically that a large number of transporters does not produce efficiency in the exiting operations, instead it increases costs of deployment and maintenance. Also the sole loading area causes a bottleneck. It was observed that the transporters would remain idle for extended periods of time near the loading area to wait while the other transporters were loading their targeted items. This bottleneck led to major overheads in queueing at the supply point and further decreases the efficiency of any arrangement of the transporters.

Further analysis is required regarding the layout of the warehouse. In this simulation two different layouts of the warehouse were tested. Different shelf placement would produce varying results regarding the transporter's efficiency relating their movements. Another aspect that requires consideration is the number of loading areas and dispatch points. In the future we have planned to extend the functionalities of the application to provide

manual modification to the layout of the warehouse, such as sizing the warehouse, manual positioning and sizing of the shelves, the agents, the load and dispatch areas, setting the transporter parameters such as speed, routing algorithm, dynamically selecting the items intended for storage and dispatch and so on.

REFERENCES

[1]Wooldridge, Michael. *An introduction to multiagent systems*. John Wiley & Sons, 2009 p.5(2009)

[2]Russell, S., Norvig, P., & Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, *25*, 27.chpt. 2

[3]Bowersox, D. J., Closs, D. J., & Cooper, M. B. (2002). *Supply chain logistics management* (Vol. 2). New York, NY: McGraw-Hill.

[4]Gu J., Goetschalckx M. & McGinnis L.F. (2007). Research on warehouse operation: A comprehensive review.European Journal of Operational Research 177: 1-21.

[5]Onggo, B. S. S., Gunal, M. M., & Maden, W. (2008). The Conceptual model of Distribution Warehouse Simulation. *Lancaster: Lancaster University Management School*.

[6]Java Platform http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm

[7]R. H. Bordini, J. F. Hübner, and M. J. Wooldridge, Programming multiagent systems in AgentSpeak using Jason. Wiley-Interscience, 2007.

[8]Jason a Java-based interpreter for an extended version of agentSpeak http://jason.sourceforge.net/wp/

[9]A. Rao, "AgentSpeak (L): BDI agents speak out in a logical computable language," Agents Breaking Away, pp. 42–55, 1996.

[10]A. Rao and M. Georgeff, “Bdi agents: From theory to practice,” in Proceedings of the first international conference on multi-agent systems (ICMAS-95). San Francisco, 1995, pp. 312–319.

[11]Cossentino, M., Lodato, C., Lopes, S., & Ribino, P. (2011, September). Multi agent simulation for decision making in warehouse management. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on* (pp. 611-618). IEEE.

[12]Detty, R. B., & Yingling, J. C. (2000). Quantifying benefits of conversion to lean manufacturing with discrete event simulation: a case study. *International Journal of Production Research*, *38*(2), 429-445.

[13]Ceylan, A., Gunal, M. M., & Bruzzone, A. G. (2012). A new approach to simulation modeling of unit-load warehouses. In *Proceedings of the 2012 Symposium on Emerging Applications of M&S in Industry and Academia Symposium* (p. 2). Society for Computer Simulation International.

[14]Liong, C. Y., & Loo, C. S. (2009). A simulation study of warehouse loading and unloading systems using Arena. *Journal of Quality Measurement and Analysis*, *5*(2), 45-56.

[15]Sommerville I. (2011). *Software Engineering Ninth Edition. Pearson Education.*

[16] An, G. (2004). In silico experiments of existing and hypothetical cytokine-directed clinical trials using agent-based modeling*. *Critical care medicine*, *32*(10), 2050-2060.

[17] Tesfatsion, L. (2006). Agent-based computational economics: A constructive approach to economic theory. *Handbook of computational economics*, *2*, 831-880.

[18] Hakrama, I. (2014). Artificial Economy and the usage of ACE, 5th International Conference on*Information Systems and Technology Innovations: projecting trends to a New Economy*, Tirane Albania, 2014.

[19] Guzy, M. R., Smith, C. L., Bolte, J. P., Hulse, D. W., & Gregory, S. V. (2008). Policy research using agent-based modeling to assess future impacts of urban expansion into farmlands and forests. *Ecology and Society*, *13*(1), 37.

[20] Janssen, M. A., & Ostrom, E. (2006). Empirically based, agent-based models. *Ecology and Society*, *11*(2), 37.

[21] Smith, E. R., & Conrey, F. R. (2007). Agent-based modeling: A new approach for theory building in social psychology. *Personality and social psychology review*, *11*(1), 87-104.

[22] Ricci, A., Viroli, M., & Omicini, A. (2006). CArtAgO: A framework for prototyping artifact-based environments in MAS. In *Environments for Multi-Agent Systems III* (pp. 67-86). Springer Berlin Heidelberg.

[23] Hübner, J.F., Boissier, O, Kitio, R. &Ricci, A. (2010). Moise Organisation Management Infrastructure based on Agent & Artifact model Artifact and Agent infrastructure . In *Journal of Autonomous Agents and Multi-Agent Systems*, *20*(3), 369-400

[24] JaCaMo: Project Multi-Agent Programming Framework http://jacamo.sourceforge.net/

[25] Hakrama, I., Frashëri , N. (2015). Modeling an Artificial Economy with JaCaMo, In *Proceedings of the 10th AnnualSouth-East European Doctoral Student Conference* (p.374-382). DSC

[26] Fahad, M., Boissier, O., Maret, P., & Gravier, C. (2012, April). Smart places: multi-agent based virtual community management system. In *Proceedings of the 4th International Workshop on Web Intelligence & Communities* (p. 2). ACM.

[27] Remote Method Invocation

http://www.oracle.com/technetwork/articles/javaee/index-jsp-136424.html

APPENDIX

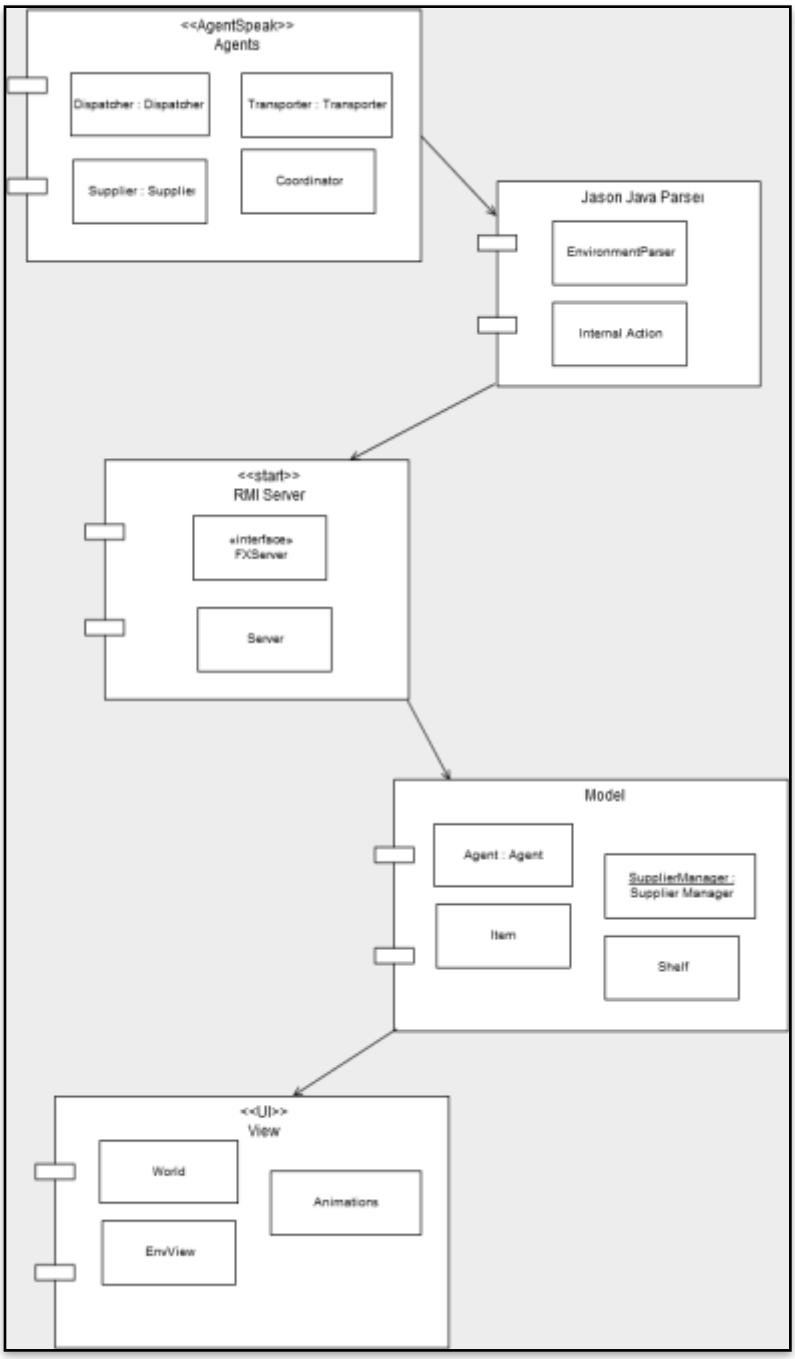

Fig. 9. Components of the application and classes within them