ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 7 N. 2 (2018), 5-15
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ201872515

Development of a Middleware between SUMO simulation tool and JaCaMo framework

Alexis van Haare Heijmeijer[a] and Gleifer Vaz Alves[a]

[a] Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná - Campus Ponta Grossa, Brazil, Av. Monteiro Lobato km. 4
aheijmeijer@uol.com.br, gleifer@utfpr.edu.br

KEYWORD

Smart Parking; Urban Simulation; Intelligent Agents; SUMO; JaCaMo framework; MAPS-SUMO Middleware

ABSTRACT

Smart City has an infrastructure that works with technology to reduce daily problems. The lack of parking spots is one of these problems that can be controlled by creating Smart Parking systems. The MAPS (Multi-Agent Parking System) project studies and develops multi-agent systems for smart parking using JaCaMo framework to implement agents and artifacts. This paper presents the so-called MAPS-SUMO, a middleware between the MAPS project and SUMO, a simulation tool used for urban traffic. Our middleware allows a graphical representation for the simulation executed by MAPS agents.

1. Introduction

The concept of Smart City has become a reality nowadays because of the accelerated advance of technology. Smart City is a city where its infrastructure works synchronously with technology, allowing agents to live and interact in this environment. Therefore, the agents contribute to the city become more efficient (Batty *et al.*, 2012). Smart Cities encompass several areas, such as integrated public security and video surveillance. Urban Mobility is also one of those areas and corresponds to the human needs of displacement in an environment according to its dimensions and activities practiced in it. Nevertheless, Urban Mobility is a very comprehensive area; however, one of the most essential topic is traffic. Traffic involves all infrastructure and transportation services, whether public or private, such as parking, public transportation monitoring, roads monitoring, among others.

Parking belongs to the city's traffic infrastructure and may present several problems such as the lack of parking spots and even congestion inside and outside the parking area. According to Koster *et al.* (2014), cars in search of parking spots cause 40% of New York traffic. According to the Brazilian National Traffic Department (DENATRAN), there was a 130% increase in the number of vehicles in circulation in Brazil between January 2005 and January 2016. This boom reflects directly in the search for parking spots. For example, the project presented by Deutsche Telekom in 2014 shows that drivers looking for parking spots generate 30% of the traffic of large cities. This project proposes to build an intelligent structure in the city of Pisa, Italy. It also proves that the implementation of the Smart City concept ensures a better traffic flow, reduces the emission of carbon gas and eases the search for parking spots. Related projects implement the concept of Smart Parking, which consists of using technologies to automate, facilitate and make feasible the best use of parking spots.

Revathi and Dhulipala (2012) classify different types of Smart Parking according to its characteristics: payment systems, automated parking, intelligent transport systems, parking guidance and information systems, and others. Among them, the Agent Based Guiding System (ABGS) stands out, which is the Smart Parking system used in this work. The ABGS works with intelligent agents implemented through sensors capable of gathering information from the environment and autonomously reacting over the information obtained in order to achieve an objective. In addition, the agents can communicate with each other. Wooldridge (2009) defines an agent as a computer system capable of autonomous action in an environment in order to meet its designated goals.

1.1. A Multiagent's approach for smart parking

As previously mentioned, one can use ABGS approach in order to build a model for a smart parking based on intelligent agents. With this in mind, we had established on previous works the so-called MultiAgent Parking System (MAPS) project (Castro *et al.*, 2017). The MAPS project has different branches of application, such as, implementation of JaCaMo agents, social organization, negotiation protocols, and urban simulation, among others. The major goal of our paper is to apply the Simulation of Urban Mobility (SUMO) tool in the current version of MAPS project.

The current version of MAPS projects uses the approach presented by Castro *et al.* (2017), where the authors had defined a model for spots assignment, which handles two types of agents: driver and manager. The drivers are the agents who interact in the environment – i.e., driver in search of parking spots. The manager is the central agent of the multi-agent system (MAS) responsible for allocating the spots for the drivers. The spots are the MAS resources handled by the manager agent and assigned to the driver. When there are no spots left in the parking, drivers wait in a queue. When a driver agent leaves a spot and there is a waiting queue, the manager decides which driver will receive that spot according to its trust degree value or when a waiting queue timeout is reached.

The manager agent is responsible not only for maintaining the communication between other agents in the environment but also for assigning spots for drivers. Each driver has a trust degree value, which characterize the commitment of the driver within the parking environment. The trust degree value comprises how the driver uses the parking. For instance, whether a driver does not park in the correct assigned spot its trust degree value will decrease otherwise it will increase. Hence, it is possible to improve the environment organization and optimize its operation (Castro *et al.*, 2017).

Spots are assigned according to driver's trust degree value or when a waiting queue timeout is reached. The Figure 1 shows how the manager uses the trust degree value to decide which driver shall obtain the requested spot.

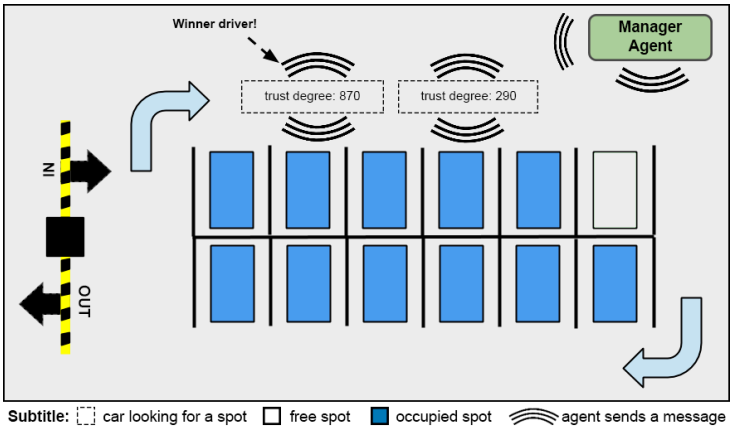

Figure 1: MAPS General representation – Adapted from (Castro et al., 2017)

1.2. Urban simulation tool

SUMO is an open source urban traffic simulation which includes a suite of applications that helps to simulate a traffic environment using different traffic flows, getting as close as possible to the real world (Behrisch *et al.* 2011). The interconnection between SUMO and MAPS seeks to fill a gap in the MAPS project with regard to the use of traffic simulators that may help to achieve better and more accurate results. The use of simulators allows the user not only to graphically visualize and analyze the environment behavior and operation but also contributes to the tests and implementation of new algorithms for allocating spots in the project.

Furthermore, the MAPS project uses JaCaMo[1] multi-agent system development as its framework, divided into three layers: agents level (Jason), environment level (Cartago) and organization level (Moise) (Boissier *et al.* 2013). The interconnection between SUMO and JaCaMo is given through an artifact, developed at MAPS environment level, and the MAPS-SUMO[2] Middleware. The MAPS-SUMO is an intermediary tool, developed for this project, which receives protocols from MAPS and forwards them to the simulation. One of the main results of our paper is to establish the interconnection between JaCaMo and SUMO, as well as to demonstrate the feasibility of a graphic representation of the parking usage by means of a simulation tool. We adopt the MAPS project as an example of use of the JaCaMo framework to show a practical application of the interconnection between MAPS and JaCaMo. As a result, we intend to create an interconnection layer between SUMO and JaCaMo in next developments, which in turn could be used for different purposes, beyond smart parking.

Specifically, our paper seeks the following outcomes: (i) a middleware protocol written in JSON; (ii) a SUMO implementation of the waiting queue from previous MAPS project versions; and (iii) the use of database for storing the trust degrees from the drivers.

The remainder of this paper is as follows. Section 2 presents the related works. Section 3 describes the SUMO simulation tool. Next, Section 4 demonstrates the MAPS-SUMO middleware. After that, Section 5 shows the obtained results. Finally, Section 6 presents the conclusion of this work.

2. Related Works

This section presents some related works in the following topics: urban mobility and traffic, smart parking and urban simulation tools.

As it follows, we present three different approaches that use the SUMO tool for solving urban traffic issues. Initially, the work of Kajzewicz *et al.* (2012) presents a model in SUMO, called CityMobil, which simulates an airport-like parking environment, where buses travel on a route and stop at predefined places for boarding passengers. They generate cars flows that follow some route until they park. However, vehicles repeatedly park at the same spot and there is no control of free and occupied spots. The TraCI[3] library uses an interface to control vehicles arriving at the parking, people entering the waiting queue and people boarding on the buses.

Baines and Padget (2014), who describe the development of a framework to analyze the behavior of intelligent agents, present another work that equally adopts SUMO. This work uses SUMO simulation tool for simulation and Jason programming language to control the agents. In one of the test scenarios, Jason inserts a new vehicle (agent) to SUMO and retrieves this vehicle route info to identify possible collisions and then reduces its speed or change its lane. In a second scenario, they insert vehicles to SUMO and the framework retrieves the upcoming traffic lights and its color (red, green). The agent receives an order to reduce its speed for a specified period so the vehicle arrives at traffic light after they have turn back to green, then the control of the agent returns to SUMO. The approach of Baines and Padget (2014) work is to simulate SUMO vehicles as agents controlled by Jason programming language, interconnecting these two tools through the development of a distributed framework. This framework also incorporates 3D simulation tools, Android environment and

1 Throughout the text, we have used the term JaCaMo. However, we actually used only Jason and Cartago languages.
2 MAPS-SUMO is used to name our middleware between MAPS project and SUMO simulation tool.
3 TraCI4J library (https://github.com/ergueli/TraCI4J).

debug tools. However, this work presented traffic simulation in general, where the authors have not considered parking management.

Yet, Batista Júnior and Coutinho (2013) describe a multi-agent system that acts on arterial roads, controlled by traffic lights, in order to improve crosses traffic. An agent controls each intersection. Their work uses Jason agent-oriented programming language for programming the agents and SUMO for simulating them. It also demonstrates an interconnection model between SUMO and Jason implemented throughout XTRACI, an API (Application Programming Interface) developed in Java and based on TraCI library, which is the same library used by CityMobil model.

The aforementioned works present the use of technological solutions for implementing Smart Parking as well as the use of SUMO simulation tool along with intelligent agents for traffic simulation. However, as far we know, no works presents the interconnection between SUMO and JaCaMo framework, although we have found some works that describe the interconnection between SUMO and Jason agent-oriented programming language. Moreover, the works that show the use of Jason along with SUMO focus on traffic simulation in general, opposite to this work, which focuses on parking spots management.

3. SUMO simulation tool

SUMO is a simulation tool developed by the Institute of Transportation Systems to improve the obtained results of a project and its analyzes, as well as to facilitate the testing of new algorithms developed in this area (Krajzewicz *et al.*, 2006).

SUMO provides several features for the user, like microscopic simulation; traffic light time control; importation of real environments through NETCONVERT tool and OpenStreetMaps; TraCI library, which allows to obtain values of the simulation and to manipulate them in real time, among many other possibilities.

3.1. A parking model with SUMO

We have used the CityMobil model, which simulates and airport-like parking environment, as a base model to build our restricted access parking area. Our work uses the modified CityMobil model for testing the interconnection of SUMO and MAPS. The model has 160 spots divided into 9 sectors: A, B, C, D, E, G, H and I, where sectors A and I have 10 spots each and others have 20 spots each. Figure 2 shows an overview of the parking model.

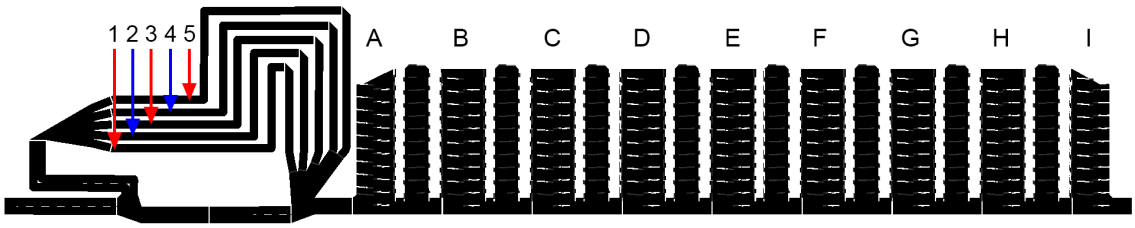

Figure 2: Overview of the parking model used in MAPS-SUMO

Moreover, we have implemented in our SUMO model five waiting queues at the left side of the model. The manager divides the drivers into the waiting queues based on the classification shown at Table 1.

Table 1: Agent's trust degree and their respective waiting queue

Trust degree	Waiting queue number
0 to 100	5
101 to 200	4
201 to 300	3
301 to 400	2
Up to 999	1

SUMO is responsible for the modeling of the environment and the definition of the spots and its routes. The MAPS project manages the environment, such as the vehicles entrance and exit control. Therefore, SUMO is only responsible for displaying the results according to instructions received from the Middleware.

4. MAPS-SUMO Middleware

The interconnection between MAPS project and SUMO simulation tool follows a client-server architecture implementation. We have developed a Middleware (MAPS-SUMO) using Java programming language that works as an intermediary tool between MAPS and SUMO. The MAPS acts as a client and the Middleware as a server. The Middleware incorporates the TraCI4J library, based on TraCI, which is responsible for communicating with SUMO.

Therefore, MAPS-SUMO acts as a server, waiting for messages sent by the MAPS projects and as a client, by handling the incoming messages and forwarding them to SUMO. In Figure 3, it is possible to observe an overview of the communication between the applications.

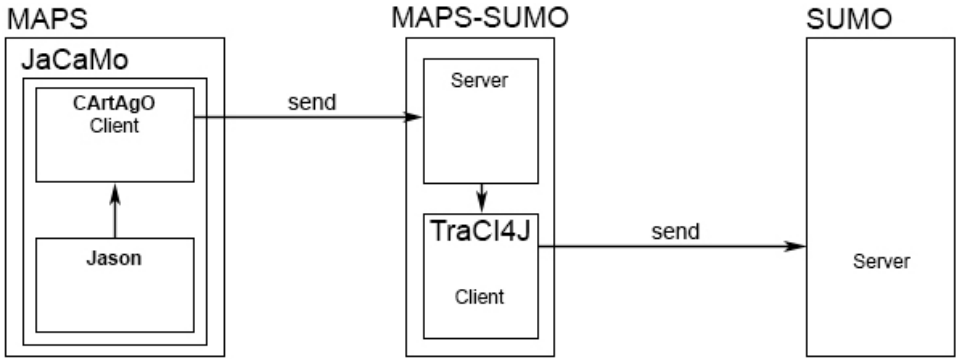

Figure 3: Overview of the interconnection between MAPS and SUMO

A JaCaMo framework artifact at Cartago's environment level does the communication between MAPS and MAPS-SUMO. The fact the artifact programming is in Java eases the communication between them. The agent manager developed in Jason accesses the artifact.

This artifact incorporate one essential operation, called sendProtocol, which establishes the communication with the simulation at execution time. The agent manager executes this operation, which controls every request made by the agents' drivers.

When the agent manager performs the aforementioned operation, it triggers a call to the artifact, which may expect as a parameter: identification of the agent that requested the operation; operation type; current parking spot identification; and driver's trust degree.

The four parameters previously defined are now wrapped into a JSON-based (JavaScript Object Notation) protocol in order to simplify the communication between the tools, JaCaMo and SUMO. The Cartago artifact defines the JSON protocol, creates a communication with MAPS-SUMO server and forward the protocol to MAPS-SUMO. Our JSON protocol is defined as follows:

```
{
        “Driver”: driverId,
        “Type”: operationType,
        “Spot”: spotId,
        “TrustDegree”: driverTrust
},
```

where,

- **driverId**: driver’s identifier;
- **operationType***:* operation type related to the context it has been generated: PS (*Parking Spot*) for allocating spot, LS (*Leaving Spot*) for leaving spot and QU (*Queue*) for inserting a driver in the waiting queue;
- **spotId***: current parking spot identification, which can be occupied or reserved to a given driver, or yet it can be empty;*
- **driverTrust***:* driver’s trust degree.

In Figure 4, it is possible to observe an example of a communication protocol, which represents the allocation of the spot ‘spotB10’ to the agent ‘driver07’.

```
{
    "Driver": "driver07",
    "Type": "PS",
    "Spot": "spotB10",
    "TrustDegree": 350
}
```

Figure 4: An example of a communication protocol between MAPS and MAPS-SUMO

We have developed an artifact in Cartago, named Simulation, in order to assure the messages sent by the agents to MAPS-SUMO. This artifact is responsible for creating the protocols for each operation and forward them to a communication class, called TCPConnection. Diagram 1 exhibits the communication between the agent manager, the artifact and the communication class.

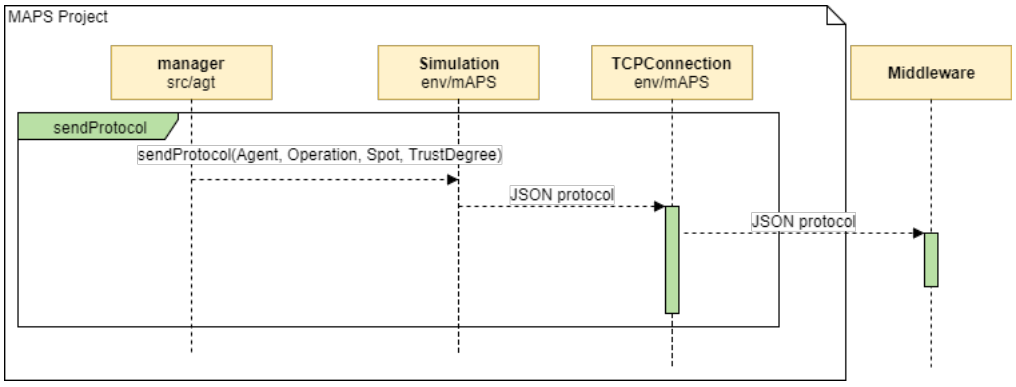

Diagram 1: Communication between the agent manager, the Simulation artifact and the communication class

The agent manager creates the artifact so the agent have access to every existent operation in the artifact and it may call any operation at any moment. The artifact is responsible for defining the communication protocol according to the parameters sent by the agent and forward it to MAPS-SUMO, which is responsible for handling the protocol and forwarding it to SUMO. In Code 1, we demonstrate the sendProtocol operation where it is possible to notice the protocol creation and its transmission to MAPS-SUMO.

```
@OPERATION
private void sendProtocol(Object driverId, String operationType, Object spotId, Object driverTrust){
        // Protocol as JSON Object
        JSONObject protocol = new JSONObject();

        // Add keys to JSON Object
        protocol.put("Driver", driverId);
        protocol.put("Type", operationType);
        protocol.put("Spot", spotId);
        protocol.put("TrustDegree", driverTrust);


        // Send protocol to connection class
        conn.connect(protocol);
}
```

Code 1: sendProtocol operation

The MAPS-SUMO is the key element of the interconnection between MAPS project and SUMO. It acts as a server handling incoming communication protocols from MAPS, related to decisions made by the agent manager. MAPS-SUMO decodes these protocols, converts them into messages and forwards them to SUMO through TraCI4J library, as shown in Diagram 2.

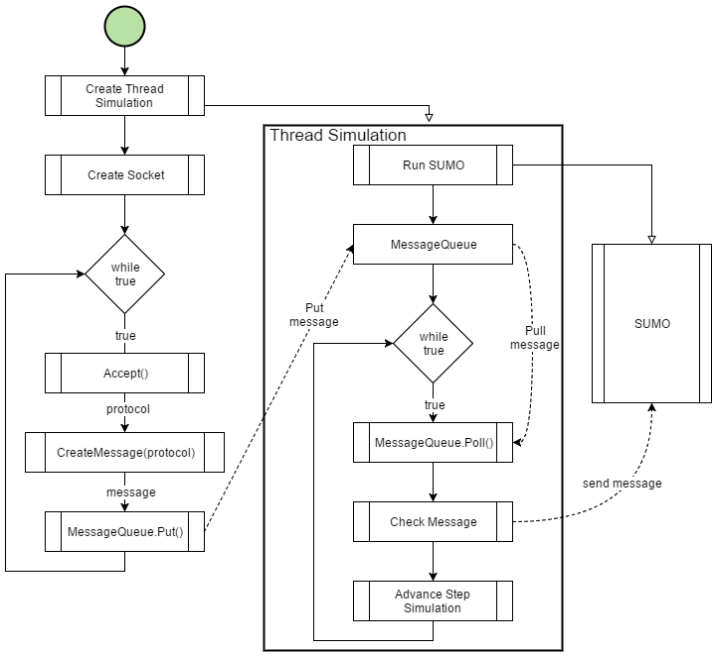

Diagram 2: Overview of MAPS-SUMO operation

Diagram 2 shows an overview of MAPS-SUMO operation. At first, MAPS-SUMO creates a thread, which runs the graphic simulation, creates a message queue and waits for incoming messages to forward to the simulation. TraCI4J controls every communication procedures with SUMO. In parallel, MAPS-SUMO creates a server responsible for receiving protocols sent by MAPS, convert them into messages and forward them to the thread.

Then, MAPS-SUMO forwards the message obtained from the protocol to the thread, where the TraCI4J library is established. The thread follows four steps: (i) receives a MAPS-SUMO message; (ii) verifies the message type; (iii) forwards the messages to simulation; and (iv) advances the simulation by step.

At last, SUMO simulation tool is responsible for receiving commands sent by MAPS-SUMO and showing graphically the parking environment operation in real time. The MAPS-SUMO is responsible for forwarding messages to SUMO through TraCI4J library. The SUMO itself acts as a server running parallel to MAPS-SUMO and waiting for commands to exhibit in the simulation.

5. Results

With the results obtained from the development of MAPS-SUMO presented in Section 4, it was possible to capture the behavior of MAPS agents when the parking spots are assigned according to the corresponding car colors, which indeed are labelled with respect to driver's trust degree, as shown in Table 2. We use this same colors classification to define the five possible waiting queue.

Table 2: Agent's trust degree and their respective colors in the simulation

Trust degree	Color
0 to 100	Red
101 to 200	Green
201 to 300	Blue
301 to 400	Yellow
Up to 999	White

In Figure 5 we may notice the use of car colors in the simulation according to drivers' trust degree, where it is possible to notice that there are 6 drivers labelled in red, 17 in green, 10 in blue, 13 in and 5 in white. We may notice that, when using colors, it is easier to visualize the results of the simulation and to understand how parking occupancy occurs in relation to the driver's trust degree values.

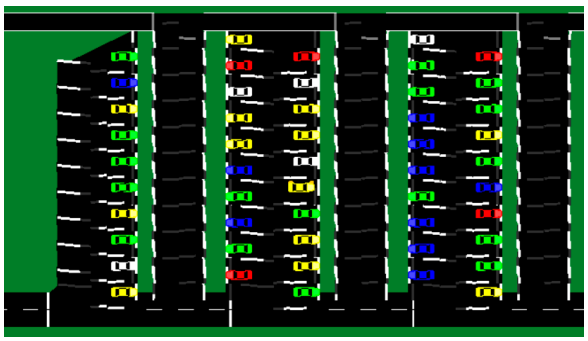

Figure 5: Vehicles' colors based on their trust degree

The connection between MAPS project and SUMO simulation tool makes the analysis of the environment behavior more practical, where it is possible to observe graphically the number of free and occupied spots. Besides that, it is also possible to follow the traffic of the vehicles in the environment, as shown in Figure 6. This figure shows that 105 out of 160 spots are occupied, corresponding to 65.62% of total occupation. In addition, 9 drivers are circulating around the parking and the sectors H and I are empty. Finally, we may notice that the waiting queues as empty as the parking lot is not full.

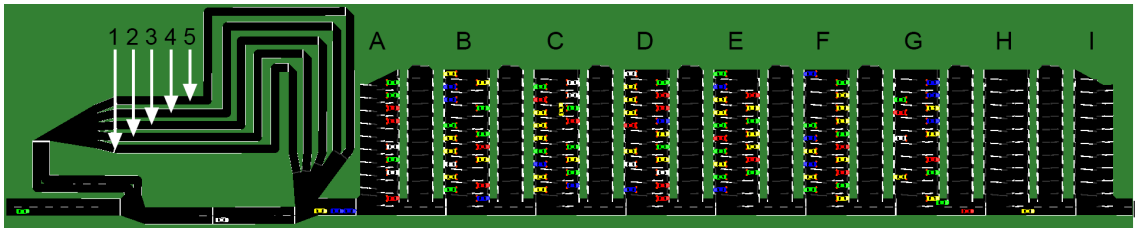

Figure 6: Parking spots simulation

Figure 7 exhibits an example where the waiting queues are in use. In this restricted scenario, drivers are allowed to park only at sectors A and B. We may notice that the parking lot is full so the manager forwards drivers to their respective waiting queue, according to Table 1. For example, queue 1 (white cars) has only 1 driver, while queue 4 (green cars) has 17 drivers. We may also observe that drivers with highest trust degree receives a spot first – some of the drivers parked before the parking got full – consequently, queue 1 has the smallest number of cars waiting.

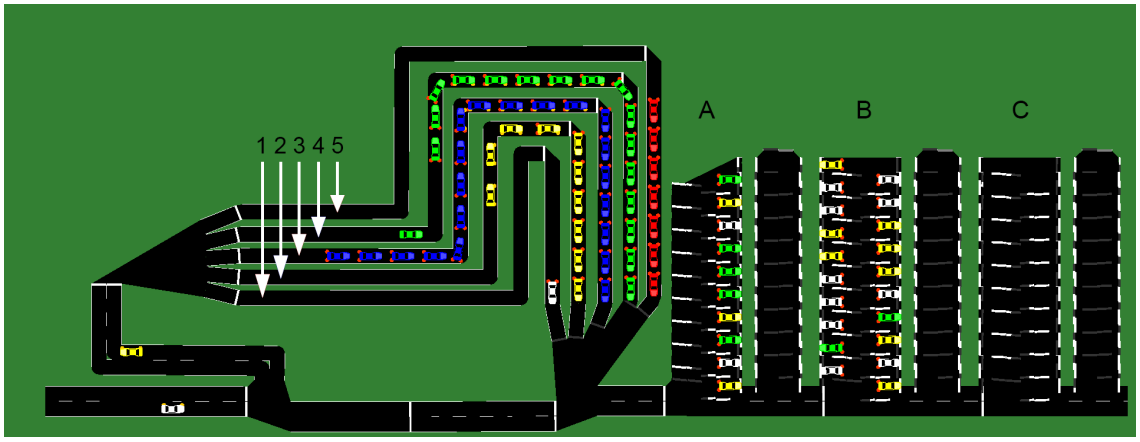

Figure 7: Parking spots simulation

In order to ensure the communication between MAPS and SUMO, the middleware became responsible for receiving MAPS protocols, transform them into messages and transmit them to SUMO. Despite of this communication procedure, we concluded that there were no losses of protocols and/or messages, since we saved them in three moments: (i) creation / transmission of the protocol by MAPS; (ii) protocol reception and message transmission by MAPS-SUMO; and (iii) receipt of the message by SUMO.

6. Conclusion

Our main goal is to establish a connection between JaCaMo framework – focusing on Jason and Cartago – and SUMO simulation tool using the MAPS project as an application of JaCaMo. A middleware is responsible to connect both tools throughout a protocol written in JSON. This protocol defines the agent responsible for the action and its beliefs, such as spot and trust degree. Moreover, we present a SUMO implementation of the waiting queues from previous MAPS project versions, where drivers are divided into selected groups based on their trust degree values. Furthermore, this work introduces the use of database for storing drivers' trust degrees in order to control every agent access to the environment. Thus, it is also possible to improve or decrease the agent's trust degree value based on how compromised he was at the parking environment.

Therefore, we notice the connection of MAPS project and SUMO simulation tool can bring the following outcomes:

I. a better analysis of MAPS agent's behavior, considering the trust degree of each driver;

I. an easier comprehension of the flow traffic inside the parking environment;

II. the understanding of the use and impact of trust degree for each group of drivers based on the corresponding labelled colors;

The interconnection between MAPS and SUMO contributes for future extensions of the MAPS project, such as implementing new algorithms for handling the waiting queues and developing real-world environments. Furthermore, the JSON-based protocol may also make it possible to establish a general-purpose interconnection between JaCaMo framework and SUMO.

7. References

Baines, V., and Padget, J., 2014. A situational awareness approach to intelligent vehicle agents. In SUMO2014 – Modeling Mobility with Open Data. 24:1-17.

Batista Júnior, A. A., and Coutinho, L. R., 2013. Incorporating explicit coordination mechanisms by agents to obtain green waves. In Advanced Methods and Technologies for Agent and Multi-Agent Systems. 252:137-145. doi: 10.3233/978-1-61499-254-7-137.

Batty, M, Axhausen, K., Fosca, G., Pozdnoukhov, A., Bazzani, A., Wachowicz, M., Ouzounis, G., and Portugali, Y., 2012. Smart Cities of the Futures. The European Physical Journal: Special Topics. 2014:481-518. Springer. doi: 10.1140/epjst/e2012-01703-3.

Behrisch, M., Bieker, L., Erdmann, J., E Krajzewicz, D., 2011. SUMO – Simulation of Urban Mobility. In SIMUL 2011 – The Third International Conference on Advances in System Simulation, pages 55-60. Iaria. ISBN: 978-1-61208-169-4.

Boissier, O., Bordini, R., Hübner, J., Ricci, A., and Santi, A., 2013. Multi-Agent oriented programming with JaCaMo. In Science of Computer Programming. 78(6):747-761. Elsevier. doi:10.1016/j.scico.2011.10.004.

Castro, L., Alves, G., and Borges, A., 2017. Using trust degree for agents in order to assign spots in a Smart Parking. In ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal. 6(2):45-55. doi: 10.14201/ADCAIJ2017624555.

Denatran. Departamento Nacional de Trânsito, Available at <http://www.denatran.gov.br/frota2016.htm>. Last accessed on November 2017.

Deutsche Telekom, 2014. Deutsche Telekom and Pisa start Smart City project. In Mobile World Congress 2014.

Koster, A., Koch, F., and Bazzan, A., 2014. Incentivising Crowdsourced Parking Solutions. In Citizen in Sensor Networks. Springer International Publishing. 8313:36-43. doi: 10.1007/978-3-319-04178-0_4.

Krajzewicz, D., Bonert, M., and Wagner, P., 2006. The open source traffic simulation package SUMO. In RoboCup 2006 Infrastructure Simulation Competition.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L., 2012. Recent development and applications of SUMO – Simulation of Urban Mobility. In International Journal on Advances in Systems and Measurements, 5(3&4):128-138. ISSN 1942-261x.

Revathi, G., and Dhulipala, V., 2012. Smart parking Systems and Sensors: A Survey. In Computing, Communication and Applications (ICCCA), pages 1-5. doi:10.1109/ICCCA.2012.6179195.

Wooldridge, M., 2009. An Introduction to MultiAgent Systems. Wiley Publishing, 2nd edition. ISBN 0470519460, 9780470519462.