Integrating BDI Agents with the MATSim Traffic Simulation for Autonomous Mobility on Demand

Marcel Mauri, Ömer Ibrahim Erduran, Thu Pham Dieu Anh and Mirjam Minor

Department of Computer Science, Goethe University Frankfurt, Frankfurt am Main, Germany

Abstract
In this paper, we present our extension for the *BDI-ABM* interface, which provides a connection layer for BDI agents to interact with simulation platforms. We introduce a new version of the *ABM-Jadex* layer, which provides the possibility to attach *BDI* Agents developed with *Jadex*, an Agent Development Framework, to the *MATSim* traffic simulation environment. We further introduce cognitive vehicle agents that are implemented with Jadex. We tested the Jadex integration layer by debugging the code and showing examples that confirm the functionality.

Keywords
BDI Agent, BDI-ABM Framework, Traffic Simulation, Jadex, Agent Development Framework

1. Introduction

Sustainable mobility is one of the global challenges. Knowledge-based systems such as software agents can contribute to reduce the emissions of traffic [1]. Simulations of vehicle agents may facilitate better decisions in urban development, usage of multi-modal mobility, or traffic operation. There are already existing platforms like *Grab*[1] which launch mobility services with fleets of public and private vehicles. The *Grab* app connects passengers with private hire, taxi, and coach drivers. However, it takes seven minutes according to Grab's web page to be matched with an appropriate vehicle when using GrabShare which takes more bookings in one ride than one.

Cognitive software agents with their negotiation capabilities may provide a more efficient, scalable solution for transport tasks, especially for Autonomous Mobility on Demand (AMoD) scenarios. An AMoD system consists of a fleet of autonomous vehicles (AVs) that pick up passengers and transport them to their destination. Cognitive software agents have been applied in a wide range of real-world domains and scenarios for solving different challenges [2, 1, 3, 4]. The most versatile and powerful agents are the so-called BDI agents. These agents are based on the human reasoning cycle, translating into beliefs, desires, and intentions [5].

LWDA'23: Lernen, Wissen, Daten, Analysen. October 09–11, 2023, Marburg, Germany
mauri@cs.uni-frankfurt.de (M. Mauri); erduran@cs.uni-frankfurt.de (Ö. I. Erduran); minor@cs.uni-frankfurt.de (M. Minor)
http://www.wi.cs.uni-frankfurt.de (M. Minor)
0000-0002-4135-1945 (M. Mauri); 0000-0002-1586-0228 (Ö. I. Erduran); 0000-0002-6592-631X (M. Minor)

[1]https://www.grab.com, last access: 20.07.2023

Many agent development platforms that support BDI agents use a simulation as an environment. Frequently, those simulations are rather simplified and closely application-specific. In contrast to this, stand-alone simulation platforms are mostly limited to very simple agent types and do not support BDI agents. This makes it difficult to carry out more complex simulations with BDI agents.

Singh et al. [6] open an alternative strand of research. They integrate agent development platforms following the BDI agent architecture [5] with rich, agent-based simulation platforms.

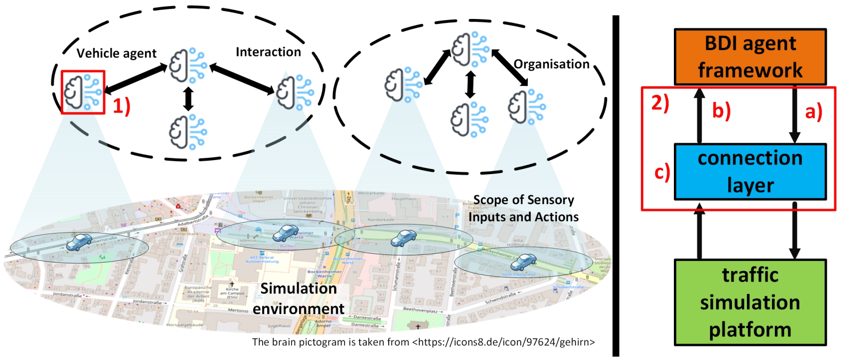

Vehicle agent
Interaction
Organisation
1)
Scope of Sensory Inputs and Actions
Simulation environment
The brain pictogram is taken from <https://icons8.de/icon/97624/gehirn>
BDI agent framework
2)
b)
a)
c)
connection layer
traffic simulation platform

Figure 1: Paper contributions (left side: following Klügl [7]).

The main scope of this paper is twofold (compare the right hand side of Fig. 1):

1. The first contribution is the design of self managing vehicle agents for AMoD applications following the BDI paradigm.
2. Second, the paper addresses the concept, design, and implementation of the synchronization between a BDI agent development framework and a traffic simulation platform building upon the results of Singh et al. [6]. The implementation of the synchronization (connection layer) consists of three main parts:
   a) The information of the vehicle agents in the BDI development platform and the traffic simulation platform needs to be exchanged correctly and at the correct time. Decisions that have been made by the BDI agents have to be sent to their counterparts in simulation platform.
   b) The results also need to be transferred back correctly. This corresponds to the sensors of the BDI agents, which sense the information in the environment (the traffic simulation platform) to update their beliefs with new information and make their own decision from the updated assumptions. This has to be achieved at the correct time so that the BDI agents could process the information to assess its goals and plans, adjust them in time and, if needed, further send the updated information to the simulation platform.
   c) The design of the cycle of the synchronization and synchronization between BDI agents in one cycle will also need to be covered to ensure that each cycle in the synchronization process is executed correctly, effectively, and efficiently.

The intended application scope of the integration is a broad range of scenarios in AMoD. Fig. 1 illustrates how the cognitive (BDI) agents in the upper layer interact with each other and the simulation platform. The spotlights indicate that a cognitive vehicle agent receives sensory inputs from its particular avatar in the simulation and makes the decisions for its actions.

We decided to use mainly open-source tools and framework solutions. For the agent development framework we choose Jadex [8]. It is a proven framework that supports BDI agents and is Java-based. A preliminary version of the VAs implemented in Jade [9] has been published in previous work [4]. The richer architectural support of Jadex in comparison to Jade led us to the decision to redesign the cognitive agents in the Jadex framework. MATSim [10] will be used as the simulation platform. MATSim is an agent-based traffic simulation platform, widely used and also based on Java.

For the connection of these two we will build upon the already existing BDI-ABM framework [6] and add a new interface for the integration of Jadex and MATSim which is not yet included. BDI-ABM has been used for training approximately 60 emergency management specialists from 20 different agencies on bushfire evacuation recently [11]. There is already an integration between an outdated version of Jadex and another simulation platform.

The remainder of this paper is structured as follows: In Section 2, the related works are presented. The idea and the design of the BDI-ABM integration in general as well as the design of Jill-MATSim integration in evacuation scenario simulation, which is the main basis for the synchronization of Jadex and MATSim, is covered in Section 3. The conceptual framework of the Jadex vehicle agent in the ride-hailing scenario is covered in Section 4. Section 5 contains the concept design and the implementation of the Jadex module. Section 6 draws a conclusion and discusses future work.

2. Related Work

The topic of integrating autonomous software agents into simulation environments has been researched extensively. Software agents have been covered in survey works focusing on the different development frameworks as well as the extensibility of the cognitive architecture [12, 13, 14, 15]. For example, *Erduran* covers the different approaches of combining the BDI agent architecture with *Machine Learning* to make the agents learn from experience [16]. The development of Software Agents is a well-researched field with a plethora of Agent development frameworks proposed by different research labs and organizations. *Silva et al.* cover the BDI agent architecture in their survey [17] and point out several research directions of extension and integration.

2.1. BDI-ABM Framework

The main contribution of this paper extends the *BDI-ABM* environment[2]. This work is grounded in several research papers, which are mentioned in this section. The concepts and fundamental approach of *BDI-ABM* is presented by *Padgham et al.* in [18]. Here, the authors present multiple layers for the integration of different agent development frameworks that especially implement

[2]https://github.com/agentsoz/bdi-abm-integration (last access: 05.07.2023)

the BDI Agent architecture, with *Agent-based Models* (ABM). In this context, ABMs provide the environment, where the agents are able to interact. Thus, research has been conducted in the application scenario of emergency evacuation and multi-modal transportation at a city-wide level [11]. One of the layers connects the Agent based simulation environment *MATSim* [19] with the agent development framework *Jill*. A Jadex layer is also considered in BDI-ABM. However, it is in an outdated version and does not support the connection to the current versions of Jadex BDI agents, called *BDIv3*. Furthermore it is developed connecting Jadex agents to the *Repast* simulation [3]. Therefore, a connection to MATSim is not provided. Furthermore, there is also no demonstration or example freely available that demonstrates the interaction of Jadex with those simulation environments.

Furthermore, *BDI-ABM* is also listed as a plugin for *MATSim* providing the connection of BDI agents to MATSim [19, 20]. One application of *BDI-ABM* is in the Emergency Evacuation Scenario (EES) [21, 6], where agents in *Jill* [4] are combined with *MATSim* by using the *BDI-ABM* framework. The integration of Agent development frameworks and traffic simulation has also been conducted by *Soares et al.* by integrating *Jade* platform and the *Sumo* traffic simulation [22]. Developing a simulation environment for Software Agents represents the same amount of challenge as developing cognitive agents. *Ricci et al.* use an *Artifact-based approach* [23, 24]. *Davoust et al.* consider an *Unmanned aerial vehicle* (UAV) scenario where the agents interact with the simulation environment [25]. Here, the focus is set on the computational performance of executing the framework.

2.2. Traffic simulation

In our considered domain of traffic simulation, we focus on AMoD settings, where the vehicle agents transport customers from a starting position to their desired destination. By using MATSim, different modes of Mobility on Demand systems can be simulated. In the current version of MATSim [5] the contribution package *DVRP* provides the necessary components for setting up a ride-sharing or ride-hailing simulation. In addition, the contribution *DRT* provides ride-pooling including vehicle agents with additional capacities. It is built on top of the *DVRP* package. Recent work that investigates scenarios on Mobility on Demand is from *Bischoff et al.*, where ride-pooling and shared taxi fleets are simulated on a city-wide scale analyzing the fleet performance [26, 27, 28]. Other mentionable work investigating ride-pooling by using MATSim is from *Zwick et al.* [29] and *Kaddoura & Schlenther* [30]. Our work differentiates from the previously mentioned works since we not only consider MATSim solely but investigate the interaction of external BDI agents with the simulation platform. Therefore, the considered MoD and Mobility as a Service (MaaS) components in MATSim are not considered in our work.

3. Foundations

On a high level, the *BDI-ABM* framework contains several integration layers for different *Agent Development Frameworks* (ADF) implemented in Java. Especially, the framework contains a

[3]https://repast.github.io/
[4]https://github.com/agentsoz/jill (last access: 05.07.2023)
[5]version 15.0, https://github.com/matsim-org/matsim-libs (last access: 05.07.2023)

generic layer, which represents a connection layer for different ADF and simulation environments. For each ADF and simulation environment, a specific layer is developed, which interacts via *BDI-ABM*. According to *Singh et al.* [6], the mentioned layer provides the possibility to connect other simulation environments as well. Thus, we developed a novel integration layer for the connection of Jadex and MATSim. In *BDI-ABM*, there exists an old integration layer for Jadex. However, the layer is customized for elder versions of the ADF and only works with the simulation environment *Repast* and not with MATSim. The advantage of the Jadex system over other ADF is that the contract net protocol [31] for the communication between agents has been integrated and the Jadex application is still currently further developed and researched. At the same time, MATSim is a mature and powerful traffic simulator that can be used for large-scale traffic simulations, primarily to assess the likely results of various infrastructure or road network changes.

3.1. Traffic Simulation

MATSim is an activity-based, extendable, multi-agent simulation framework implemented in Java, which is open-source [10]. MATSim is developed using the concept of Agent-Based-Modelling that is specified for transport simulation. This framework is designed for large-scale scenarios and is usually used to model a single day. With MATSim, it is possible to simulate traffic, taxi fleets, mobility as a service as well as different modes of transportation.

3.2. Interface for cognitive agents

In the conceptual framework of BDI-ABM [6], some agents in the simulation have a ”brain” in the BDI system, which is the decision-making component, and a ”body” in the ABM system which carries out actions. An agent in this integrated framework will be situated in an environment where it can perceive environmental input via percepts, and act, via actions. These activities of perceiving and acting will happen inside the ABM, where the ”body” interacts with the physical world of the domain. To be precise, as shown in Fig. 1 with the arrows of action and percept, the perceptions from ABM will be communicated to the ”brain” in BDI, and the ”brain” will use its decision-making mechanism to select the suitable action based on the input from percepts, the chosen action will be delivered back to ABM to be carried out. It is defined in the conceptual framework that a percept going into BDI from ABM does not have to be exactly identical to the percepts in ABM. The percept in BDI is a high-level percept composed of lower-level observations of the environment, which are the percepts represented in ABM. Similarly, an action going from the BDI agent to its ABM counterpart must typically be decomposed into a sequence of lower-level environment actions that the ABM agent knows how to perform. In terms of data transfer between the BDI and ABM systems, two key optimizations for this integrated framework are defined. The first one is that a single data container is passed between the systems in each simulation cycle. The data container bundles the messages for all agents and delivers them all together to the other system to simplify the synchronization between the systems. The second one is that not every percept is computed and pushed to the BDI system on every cycle. The reason is the BDI agent processes information contextually and only certain information is useful in certain situations. In case of ad-hoc information requirements, the

BDI agent can pull this percept from the ABM environment as needed via the percept queries function. From the technical point of view, the framework consists of three distinct layers. First is a generic layer, which realizes the conceptual model from the previous section. The second layer is the system layer, which provides the code necessary for linking a particular BDI or ABM system into the generic layer. With this layer, built on top of the generic layers, specific BDI systems like Jadex and Jill, as well as ABM systems (i.e. MATSim), can receive and send percepts and actions back and forth. The last layer is the application layer, which provides the application-specific code including agent behavior and reasoning. Overall, the BDI application provides action decisions to the ABM and the ABM provides observations and environmental information of interest to the BDI module.

4. BDI vehicle agents

The agent framework for AMoD we are developing consists of different types of agents. The geographical environment is divided into multiple zones, each with a responsible area agent. Vehicle agents are autonomous vehicles which are distributed in the application area. They check in and out at their area agents when they enter or leave their zone and regularly update their current location. When a customer requests a trip, it is delegated to the area agent in whose area of responsibility the starting position of the trip is located. The area agent sends the request to the vehicle agent which is located closest to the start position. The vehicle agent will then evaluate how well it is suited to fulfill the customer's request (amount of already accepted trips, battery level, ...). Depending on the outcome, it does it itself or negotiates with other vehicle agents in its area of operation to delegate it to a more suitable one. The negotiation between the vehicle agents will be realized by the Contract Net Protocol (CNP) [31]. Thus, the vehicle agents are self managed. When there are no customers, they drive to safe parking spaces, and when their battery level is low, they drive to a charging station.

Jadex agents implement a BDI architecture using beliefs, goals and plans. Beliefs represent the current knowledge of the agent. Desires are goals which are desirable for the agent in general while intentions are a subset of the desired goals for which the agent has actually made a commitment. Goals in Jadex are used to implement both desires and intentions. Depending on the current state of a goal its theoretical meaning [5] may change between a desire only (inactive goal) or a desire and an intention (active goal). Jadex supports multiple types of goals for different purposes. There are perform goals that will only be executed once and maintain goals which will be triggered by a condition repeatedly. Plans describe the sequence of actions that is executed to achieve a goal.

Figure 2 shows the newly designed architecture of a vehicle agent. The design comprises of the five goals: *ManageJobs*, *BatteryLoaded*, *TripService*, *SimQuery* and *SimReceive* and their according plans. Vehicle agents have an interface to the BDI-ABM Layer and a second interface for the communication with other Jadex agents.

Vehicle agents are exchanging messages with corresponding area agents and other vehicle agents. Incoming jobs are stored inside the *DecisionTaskList*. Every entry in this list is a not yet evaluated *DecisionTask*. A *DecisionTask* contains information about a trip that has been requested by a customer (start time, start position, end position, etc.). As long as this list is not

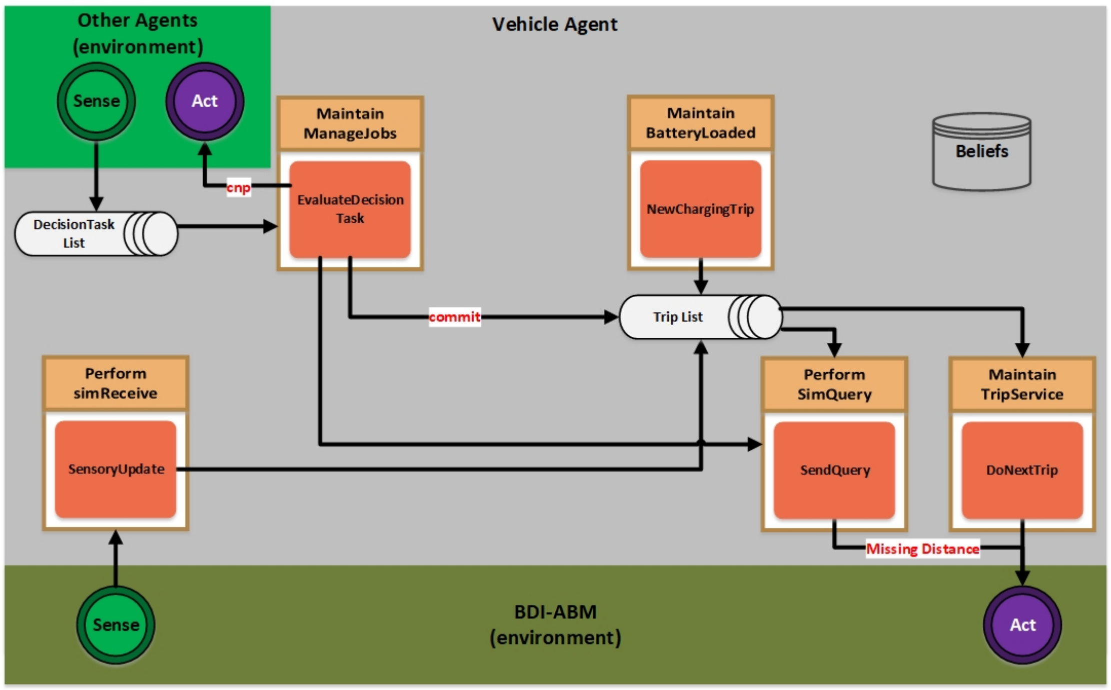


Figure 2: The components of the vehicle agent architecture

empty the *ManageJobs* goal is active. The corresponding plan *EvaluateDecisionTask* iterates through the entries and determines by their actual progress state the next needed action. We decided against a design in which every *DecisionTask*/ *Trip* will cause the generation of a separate goal/plan which handles its processing. Our approach achieves the same functionality, but is easier to handle.

For new received jobs a utility score is calculated. This score determines how well this vehicle agent is suited to perform this ride. Right now the exact utility function is ongoing work. In [4] we presented a preliminary approach. It includes information about the distance (travel times) between coordinates. Any distance that has not yet been calculated and is missing in the agents database can be requested from the BDI-ABM environment. The requests are handled by the goal *SimQuery* and its plan *SendQuery*. If the utility score is below a previously defined threshold the agent will start the CNP to delegate the trip. The call for proposals (cfp) will be sent to the other vehicle agents in that area. Cfps are treated similar to DecisionTasks and are stored inside the *DecisionTaskList*. Any further step of the CNP will also be executed by *EvaluateDecisionTask*. The recipients (contractors) then calculate how suitable they are for the job and send their proposals back to the sender (manager). When the manager has received all proposals it will send an accept/ reject to the contractors and delegate the *DecisionTask*. If the utility score of a DecisionTask is above the threshold or the agent has received an accept regarding a completed CNP the agent will commit it and create a corresponding trip. There are different subtypes of trips. Besides the customer trips that contains information about a trip that was requested by a customer there are charging trips.

When the battery level of a vehicle agent falls below a predefined threshold the goal Battery-Loaded is triggered. The corresponding plan NewChargingTrip will generate a chargingtrip which contains information for a drive to a charging station. Depending on the type a trip can contain one or more coordinates. While a charging trip just needs the coordinates of the charging station a customer trips needs the start position and the endposition of a trip. Trips are stored in the TripList, which contains a sorted list of all committed trips that have not yet been started. Any new created trip will be sent to a scheduler that will insert the new trips into the TripList and reschedule the entire list if needed.

When the Trip List is not empty the TripService goal is activated. The corresponding plan DoNextTrip takes the next trip from the TripList and sends a driveto command to the BDI-ABM framework which will cause the MATSim counterpart of the Agent to drive to the specified location. When a driveto command is sent the internal progress of this CurrentTrip is updated. As long as there is no feedback from the BDI-ABM framework the agent will not perfom any other driveTo operations. Any information that is sent from the BDI-ABM framework to the Jadex vehicle agent is handled by the SimReceive goal. The plan SensoryUpdate processes all incoming information from the simulation site. This information could include, for example, the result of a driveTo with the new position of the vehicle or an requested distance. After the vehicle has received the confirmation that the last drive operation on the simulation site is finished DoNextTrip can resume its work.

5. Jadex-ABM integration layer

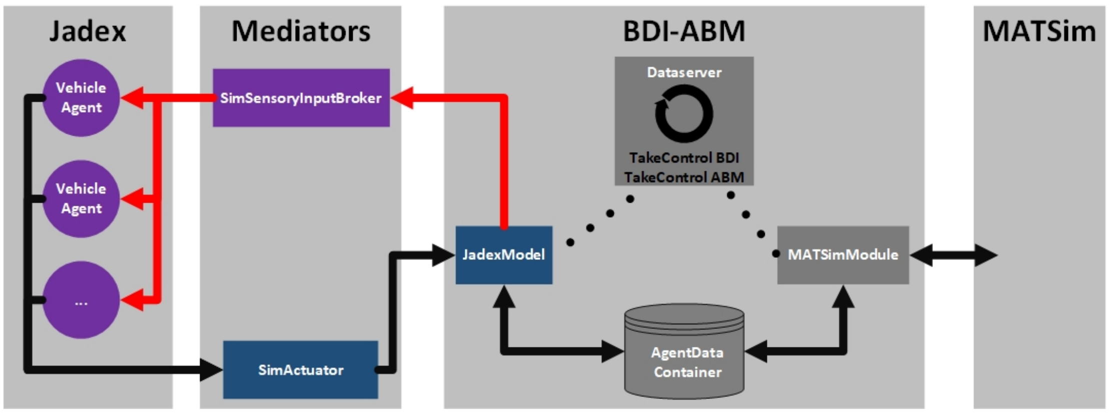


Figure 3: Simplified illustration of the connection between Jadex and matsim by BDI-ABM

The Jadex-MATSim integration framework is inspired by the already existing Jill-MATSim integration framework [20]. Fig. 3 shows the new integration layers with the existing components depicted in grey color. The BDI-ABM layer synchronises the mutual control taken by the cognitive side (Jadex agents) and the simulation side (MATSim). The *Dataserver* component controls the access to a shared memory structure called *AgentDataContainer*. The *Dataserver* grants read/write access to the cognitive side via the *TakeControl BDI* command and withdraws it via the command *TakeControl ABM*, which provides the simulation side with read/write access. Intermediate results from the reasoning cycles of the BDI side are stored in *AgentDataContainer* to to be shared with the simulation and the simulation outputs vice versa.

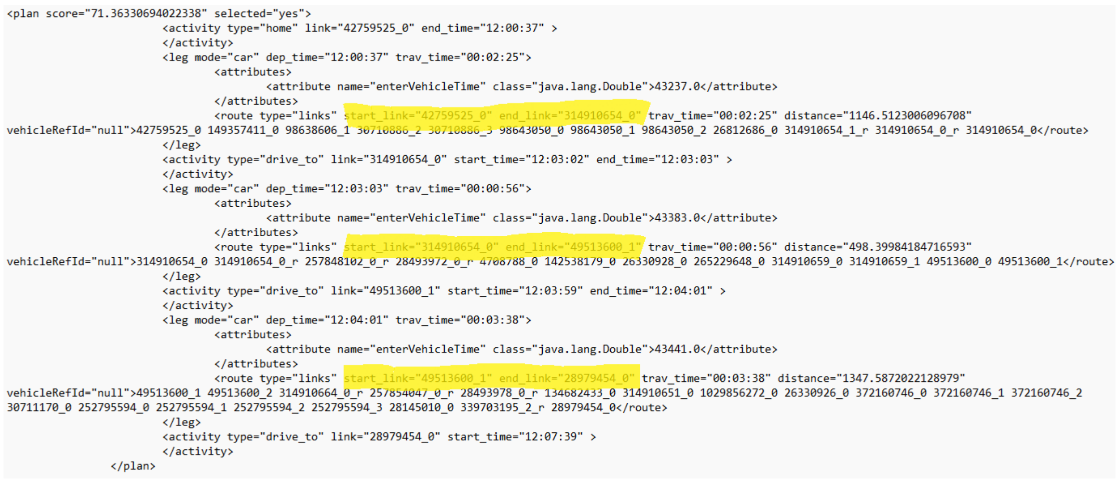
The *JadexModel* controls the incoming and outgoing data from and to Jadex. To connect the vehicle agents implemented in Jadex with the BDI-ABM layer, the *SimSensoryInputBroker* and the *SimActuator* play the role of mediators. The mediators are required because Jadex active components like the vehicle agents cannot be accessed directly by external (non-Jadex) components [32]. The *SimActuator* is used by the vehicle agents to write the actions (drive-to) into the *AgentDataContainer.* The *SimSensoryInputBroker* distributes the incoming data from the BDI-ABM (MATSim) side. The entries of the *AgentDataContainer* are directly written into the beliefs of the respective vehicle agents. Once the *SimActuator* has collected (eventual) new drive-to commands from the vehicle agents, the *JadexModel* will update the content of the *AgentDataContainer* and notify the *Dataserver* to pass control to the MATSim side again. The *MATSimModule* [18] will then translate the BDI-actions from the *AgentDataContainer* into low level actions for MATSim.

6. Discussion of results and future work

To demonstrate the functionality of our implementation, we conducted a test simulation on a street network of a university campus map. The map is extracted from *OpenStreetMap* [6]. A basic MATSim simulation consists of a road network map, a predefined population of agents, and a configuration file [33]. To check the right functionality, we investigated the output files, which are created by the simulation. We started two BDI agents which get exemplary trip requests. These are in turn executed in MATSim during the simulation and in conclusion, written into the output file. While running the framework, first the Jadex BDI agents are created and subsequently, the MATSim simulation starts. During the simulation, the BDI agents send the action requests (*drive-to*) to the mapped vehicle agents in MATSim, which in return drive on the network. In Fig. 4, the relevant output file called *events file* is shown, where the results of the commands from the BDI component are depicted. The values *start link* and *end link* indicate the start and end positions of the vehicle with *id=1.* In Fig. 5, the corresponding visualization to the mentioned events file is shown by using the tool *Via* [7]. Here, the marked route on the road network map is the route that the vehicle agent has driven. The implementation is available on GitHub [8].

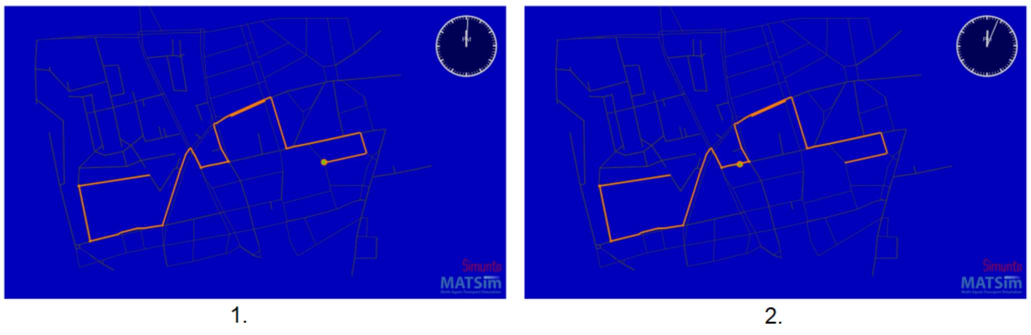
The integration of both ADF and simulation environment is considered in a larger project, where other research areas of Multi-agent systems are considered. Therefore, the project is ongoing and the implementation of cognitive agents as well as their internal architecture is under construction. Furthermore, the considered scenario is exemplary and does not yet consider other agent models or baselines as a comparison. For future work, we plan to extend the cognitive agents with Machine Learning algorithms to investigate *Neuro-symbolic Agents* as well as their

[6]https://openstreetmap.org
[7]https://simunto.com/via/
[8]https://github.com/WI-user/LWDA23-submission



```
<plan score="71.36330694022338" selected="yes">
                        <activity type="home" link="42759525_0" end_time="12:00:37" >
                        </activity>
                        <leg mode="car" dep_time="12:00:37" trav_time="00:02:25">
                                <attributes>
                                        <attribute name="enterVehicleTime" class="java.lang.Double">43237.0</attribute>
                                </attributes>
                                <route type="links" start_link="42759525_0" end_link="314910654_0" trav_time="00:02:25" distance="1146.5123006096708"
vehicleRefId="null">42759525_0 149357411_0 98638606_1 30710886_2 30710886_3 98643050_0 98643050_1 98643050_2 26812686_0 314910654_1_r 314910654_0_r 314910654_0</route>
                        </leg>
                        <activity type="drive_to" link="314910654_0" start_time="12:03:02" end_time="12:03:03" >
                        </activity>
                        <leg mode="car" dep_time="12:03:03" trav_time="00:00:56">
                                <attributes>
                                        <attribute name="enterVehicleTime" class="java.lang.Double">43383.0</attribute>
                                </attributes>
                                <route type="links" start_link="314910654_0" end_link="49513600_1" trav_time="00:00:56" distance="498.39984184716593"
vehicleRefId="null">314910654_0 314910654_0_r 257848102_0_r 28493972_0_r 4708788_0 142538179_0 26330928_0 265229648_0 314910659_0 314910659_1 49513600_0 49513600_1</route>
                        </leg>
                        <activity type="drive_to" link="49513600_1" start_time="12:03:59" end_time="12:04:01" >
                        </activity>
                        <leg mode="car" dep_time="12:04:01" trav_time="00:03:38">
                                <attributes>
                                        <attribute name="enterVehicleTime" class="java.lang.Double">43441.0</attribute>
                                </attributes>
                                <route type="links" start_link="49513600_1" end_link="28979454_0" trav_time="00:03:38" distance="1347.5872022128979"
vehicleRefId="null">49513600_1 49513600_2 314910664_0_r 257854047_0_r 28493978_0_r 134682433_0 314910651_0 1029856272_0 26330926_0 372160746_0 372160746_1 372160746_2
30711170_0 252795594_0 252795594_1 252795594_2 252795594_3 28145010_0 339703195_2_r 28979454_0</route>
                        </leg>
                        <activity type="drive_to" link="28979454_0" start_time="12:07:39" >
                        </activity>
                </plan>
```

Figure 4: Extract of the MATSim Output Events File



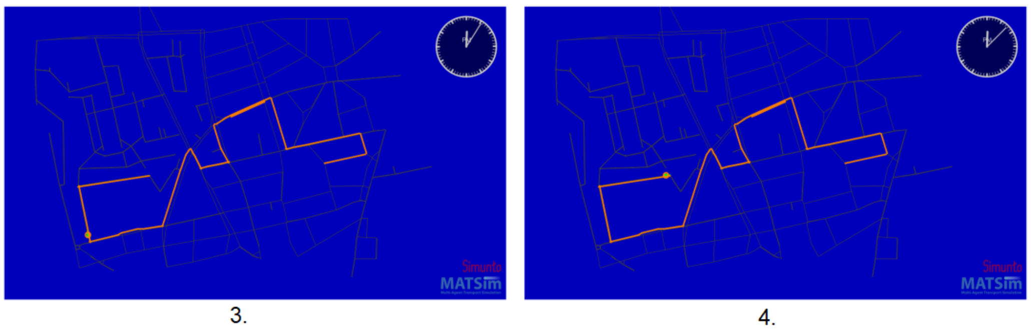


Figure 5: Visualization of the MATSim Output Events File using *Via*.

explainability. Furthermore, we will design an experimental setup and run ride-hailing scenarios with different configurations. Other application scenarios like ride-pooling and waste collection by a fleet of trucks will be investigated using the presented framework in this paper. We think that our knowledge-based approach with its reasoning-simulation integration contributes some foundational methods to achieve more sustainable solutions for mobility in the future.

References

[1] A. L. Bazzan, F. Klügl, A review on agent-based technology for traffic and transportation, The Knowledge Engineering Review 29 (2014) 375–403.

[2] Ö. I. Erduran, M. Minor, L. Hedrich, A. Tarraf, F. Ruehl, H. Schroth, Multi-agent Learning for Energy-Aware Placement of Autonomous Vehicles, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), IEEE, Boca Raton, FL, USA, 2019, pp. 1671–1678.

[3] A. Malas, S. E. Falou, M. E. Falou, M. Itmi, A. Cardon, Solving on-demand transport problem through negotiation, in: Proceedings of the Summer Computer Simulation Conference, 2016, pp. 1–7.

[4] Ö. I. Erduran, M. Mauri, M. Minor, Negotiation in ride-hailing between cooperating BDI agents, in: Proceedings of the 14th International Conference on Agents and Artificial Intelligence, volume Volume X, Scitepress, Online Streaming, 2022, pp. 425 –432.

[5] M. Georgeff, B. Pell, M. Pollack, M. Tambe, M. Wooldridge, The Belief-Desire-Intention Model of Agency, in: Intelligent Agents V: Agents Theories, Architectures, and Languages, volume 1555, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999, pp. 1–10.

[6] D. Singh, L. Padgham, B. Logan, Integrating BDI Agents with Agent-Based Simulation Platforms, Autonomous Agents and Multi-Agent Systems 30 (2016) 1050–1071.

[7] F. Klügl, Multiagentensysteme, in: Handbuch der Künstlichen Intelligenz, 6. auflage ed., De Gruyter Oldenbourg, Berlin/Boston, 2021, pp. 755–781.

[8] A. Pokahr, L. Braubach, K. Jander, The Jadex Project: Programming Model, in: Multiagent Systems and Applications: Volume 1:Practice and Experience, Springer, Berlin, Heidelberg, 2013, pp. 21–53.

[9] F. Bellifemine, G. Caire, D. Greenwood, Developing multi-agent systems with JADE, reprint. ed., Wiley series in agent technology, Chichester, 2008.

[10] K. W Axhausen, A. Horni, K. Nagel, The multi-agent transport simulation MATSim, Ubiquity Press, 2016.

[11] D. Singh, P. Ashton, E. Kuligowski, G. Pawan, Bushfire evacuation decision support system use in incident management training, Australian Journal of Emergency Management 37 (2022).

[12] R. C. Cardoso, A. Ferrando, A Review of Agent-Based Programming for Multi-Agent Systems, Computers 10 (2021) 16.

[13] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, A. Santi, Multi-agent oriented programming with JaCaMo, Science of Computer Programming 78 (2013) 747–761.

[14] K. Kravari, N. Bassiliades, A Survey of Agent Platforms, Journal of Artificial Societies and Social Simulation 18 (2015) 11.

[15] A. Dorri, S. S. Kanhere, R. Jurdak, Multi-Agent Systems: A Survey, IEEE Access 6 (2018) 28573–28593.

[16] Ö. I. Erduran, Machine Learning for Cognitive BDI Agents: A Compact Survey, in: ICAART (1), 2023, pp. 257–268.

[17] L. d. Silva, F. Meneguzzi, B. Logan, BDI Agent Architectures: A Survey, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence Organization, Yokohama, Japan, 2020, pp.

4914–4921.

[18] L. Padgham, K. Nagel, D. Singh, Q. Chen, Integrating BDI Agents into a MATSim Simulation, ECAI (2014).

[19] L. Padgham, D. Singh, Making MATSim Agents Smarter with the Belief-Desire-Intention Framework, in: ETH Zürich, A. Horni, K. Nagel, TU Berlin (Eds.), The Multi-Agent Transport Simulation MATSim, Ubiquity Press, 2016, pp. 201–210.

[20] D. Singh, L. Padgham, K. Nagel, Using MATSim as a Component in Dynamic Agent-Based Micro-Simulations, in: Engineering Multi-Agent Systems, volume 12058, Springer International Publishing, Cham, 2020, pp. 85–105.

[21] D. Singh, L. Padgham, Emergency Evacuation Simulator (EES) - a Tool for Planning Community Evacuations in Australia, in: Proceedings of the Twenty-Sixth IJCAI, Melbourne, Australia, 2017, pp. 5249–5251.

[22] G. Soares, Z. Kokkinogenis, J. L. Macedo, R. J. F. Rossetti, Agent-Based Traffic Simulation Using SUMO and JADE: An Integrated Platform for Artificial Transportation Systems, in: M. Behrisch, D. Krajzewicz, M. Weber (Eds.), Simulation of Urban Mobility, volume 8594, Springer Berlin Heidelberg, 2014, pp. 44–61.

[23] A. Ricci, M. Piunti, M. Viroli, Environment programming in multi-agent systems: an artifact-based perspective, Autonomous Agents and Multi-Agent Systems 23 (2011) 158–192.

[24] A. Ricci, A. Croatti, R. H. Bordini, J. F. Hübner, O. Boissier, Exploiting Simulation for MAS Programming and Engineering—The JaCaMo-sim Platform, EMAS (2020) 19.

[25] A. Davoust, P. Gavigan, C. Ruiz-Martin, G. Trabes, B. Esfandiari, G. Wainer, J. James, An Architecture for Integrating BDI Agents with a Simulation Environment, in: L. A. Dennis, R. H. Bordini, Y. Lespérance (Eds.), Engineering Multi-Agent Systems, volume 12058, Springer International Publishing, 2020, pp. 67–84.

[26] J. Bischoff, I. Kaddoura, M. Maciejewski, K. Nagel, Simulation-based optimization of service areas for pooled ride-hailing operators, Procedia Computer Science 130 (2018) 816–823.

[27] J. Bischoff, M. Maciejewski, K. Nagel, City-wide shared taxis: A simulation study in Berlin, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE, Yokohama, 2017, pp. 275–280.

[28] J. Bischoff, M. Maciejewski, Proactive empty vehicle rebalancing for Demand Responsive Transport services, Procedia Computer Science 170 (2020) 739–744.

[29] F. Zwick, N. Kuehnel, R. Moeckel, K. W. Axhausen, Agent-based simulation of city-wide autonomous ride-pooling and the impact on traffic noise, Transportation Research Part D: Transport and Environment 90 (2021).

[30] I. Kaddoura, T. Schlenther, The impact of trip density on the fleet size and pooling rate of ride-hailing services: A simulation study, Procedia Computer Science 184 (2021) 674–679.

[31] Smith, The contract net protocol: High-level communication and control in a distributed problem solver, IEEE Transactions on Computers C-29 (1980) 1104–1113.

[32] A. Pokahr, Aktive Komponenten: Ein integrierter Entwicklungsansatz für verteilte Systeme, Ph.D. thesis, Hamburg University, 2017.

[33] K. W. Axhausen, ETH Zürich, The Multi-Agent Transport Simulation MATSim, Ubiquity Press, 2016.