# TOWARDS ONTOLOGY-BASED MULTIAGENT SIMULATIONS: THE PLASMA APPROACH*

Tobias Warden, Robert Porzel, Jan D. Gehrke, Otthein Herzog, Hagen Langer, Rainer Malaka
Center for Computing and Communication Technologies (TZI)
Universität Bremen
Bremen, Germany
Email: {warden,porzel,jgehrke,herzog,hlanger,malaka}@tzi.de

## KEYWORDS

Multiagent-based Simulation, Simulation Modelling, Ontology Engineering.

## ABSTRACT

In multiagent-based simulation systems the agent programming paradigm is adopted for simulation. This simulation approach offers the promise to facilitate the design and development of complex simulations, both regarding the distinct simulation actors and the simulation environment itself. We introduce the simulation middleware PlaSMA which extends the JADE agent framework with a simulation control that ensures synchronization and provides a world model based on a formal ontological description of the respective application domain. We illustrate the benefits of an ontology grounding for simulation design and discuss further gains to be expected from recent advances in ontology engineering, namely the adaption of foundational ontologies and modelling-patterns.

## INTRODUCTION

Multiagent-based simulation (MABS) has been employed successfully for system analysis and evaluation in a variety of domains ranging from simulating models of bee recruitment to simulating complex business processes, such as supply chain management. The approach lends itself in particular to the simulation of complex systems on the micro-level where individual decision makers are modeled explicitly as autonomous agents embedded in dynamic environments. In contrast to alternatives, such as equation-based modelling, these modelling approaches facilitate the design of complex systems due to task decomposition, a natural mapping from real-world actors or entities to agents, the focus on modelling of individual behaviour (Parunak et al., 1998).

Still, the concrete decision to adopt multiagent-based simulation for evaluation purposes when starting from a blank slate, is often perceived as a mixed blessing as the effort required to design particular multiagent-based simulations, especially when scaling the number of involved agents and environmental complexity, often exceeds that of comparable simulation approaches, such as Petri-nets or queuing networks (Klügl et al., 2002). Off-the-shelf agent frameworks are typically not designed to consider simulation-specific issues, such as synchronization, for which solutions exist in standard simulation approaches (Bobeanu et al., 2004). In addition, the design of the simulation environment itself in which the MAS can be placed, requires significant development resources. Thus, there is still an engineering challenge for multiagent-based modelling and simulation to be addressed.

In this work, we introduce the multiagent-based simulation system PlaSMA and focus on a description of its ontology-based simulation world model. We thereby address how the design challenge to create scalable simulations for complex domains is approached in the PlaSMA system. To that end, the system provides reusable and extendable ontological models based on an explicit logical calculus. For evaluating our approach, we show how the system has been used successfully for simulations in the domain of autonomous logistic processes, where different approaches for self-organized decision making have been examined in terms of their robustness and adaptivity, e.g. in transport processes in dynamic environments (Hülsmann and Windt, 2007). We also describe ongoing work to generalize PlaSMA and use recent advances in ontology engineering, which further facilitates simulation design, interaction and analysis.

## PLASMA SIMULATION PLATFORM

The PlaSMA system provides a distributed multiagent-based simulation and demonstration system based on the FIPA-compliant Java Agent Development Framework JADE (Bellifemine et al., 2007)[1]. Although the primary application domain is logistics, PlaSMA is, in principle, applicable for other simulation domains.

The simulation system consists of the basic components simulation control, world model, simulation agents, analysis, and user interface. Simulation control handles world model initialization, provides an interface for world model access, and performs agent lifecycle and simulation time management. Simulation control is jointly executed by a

*This research has been funded by the German Research Foundation (DFG) within the Collaborative Research Centre (CRC) 637 "Autonomous Cooperating Logistic Processes – A Paradigm Shift and its Limitations" at the Universität Bremen, Germany.

[1] *PlaSMA*, the **Pla**tform for **S**imulations with **M**ultiple **A**gents, is developed at the CRC 637 "Autonomous Cooperating Logistic Processes". PlaSMA is available at: `plasma.informatik.uni-bremen.de`

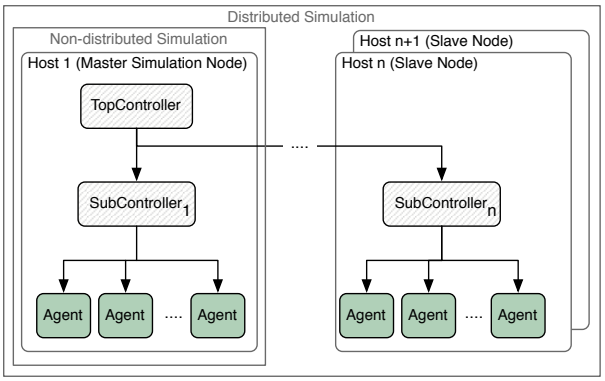


Figure 1: PlaSMA simulation control model

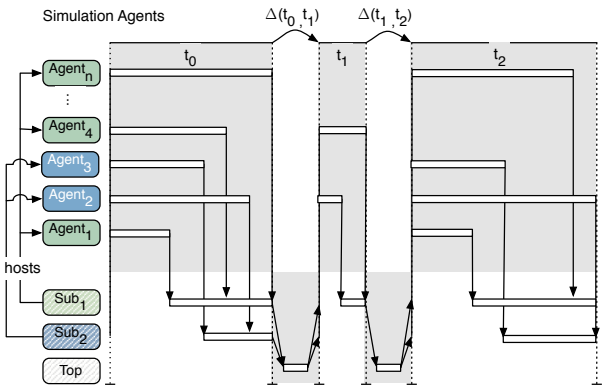


Figure 2: Course of a PlaSMA simulation.

single top-level controller and an additional sub-controller for each processor or computer in distributed settings as depicted in Figure 1. Sub-controllers handle those software agents (called *simulation agents*) that are the actors in a simulation scenario. The interaction between top-controller and sub-controllers concerns agent lifecycle management, runtime control, and time events.

**Simulation Time Management**

In general, MABS can combine distributed discrete event or time-stepped simulation with decision-making encapsulated in agents as separate and concurrent logical processes (Parunak et al., 1998). In classical simulation systems, the logical processes involved as well as interaction links have to be known in advance and must not change during simulation (Fujimoto, 2000). This is not the case in MABS as each agent may interact with all other agents (Lees et al., 2004). Agents may join or leave simulation during execution, e.g. depending on a stochastic simulation model or human intervention. This flexibility, however, complicates simulation time management.

Initially, it is necessary to distinguish different notions of time related to multiagent-based simulation. Generally, *physical time* refers to the time at which simulated events happen in the real world. *Simulation time* (or *virtual time*) models physical time in simulation. In scenarios where the advancement of *local virtual (simulation) time* or in short LVT (Jefferson, 1990) is directly coupled with the progress of each individual agent in executing its behaviours, heterogeneities in the computational demands of agents in the simulation and the distribution of agents across hosts platforms with varying computational power provoke a problematic divergence of LVTs, leading to the so-called *causality problem* (Fujimoto, 2000). In order to guarantee correct and reproducible simulations (Gehrke et al., 2008), the simulation system has thus to ensure that agents process events in accordance to their time-stamp order. This requirement is addressed by synchronization, which can be either optimistic or conservative in character. In optimistic synchronization, the progression of local virtual time for each agent is in general not restricted which allows executing simulations efficiently since fast processes do not have to wait for slower ones. Optimistic synchronization is demanding in implementation and has high requirements regarding memory. Since cascading rollbacks might require rewinding many steps back in time, preceding execution states must be retained for each agent. Conservative synchronization, by contrast, prevents causality problems by ensuring the correct order of event processing at each time by means of explicit coordination. The price to be paid is thereby a lesser speedup that achievable by parallelism.

The choice of synchronization a mechanism is an important design decision for the implementation of a MABS system. Based on the identification of important quality criteria for MABS systems presented in (Gehrke et al., 2008), a coordinated conservative synchronization approach has been adopted in PlaSMA which is handled concertedly by the simulation controller hierarchy. As shown in Figure 2, the top-controller sends time events to each sub-controller to indicate progression of simulation time. Based on this information, the sub-controllers identify the respective subset of locally managed simulation agents due to act at the next time-stamp, advance the LVT of these agents to the communicated time stamp, and finally send wake-up notifications. The agents then take over from the controllers to perform a single walk through their respective behaviour structure. Computation time will vary depending on the complexity and diversity of agent tasks and the situation at hand. At the end of each action cycle each simulation agent needs to inform its local sub-controller of its requirements with regard to continuative operation. Thereby, the agent may choose to declare explicitly a particular time point in the future. Alternatively, it can choose not to declare such a particular date and rather wait until operation is necessitated due to the reception of messages communicated by other agents or the completion of a previously commissioned action. The activity requests are consolidated by the sub-controllers such that only a single result is propagated further to the top-level controller. The latter then computes the next simulation time-stamp based on these requests. Having done so, it also computes the progress in the physical simulation world model, thus setting the stage for the next action cycle of the simulation agents.

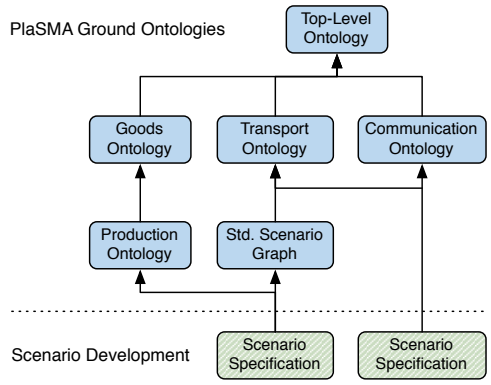


Figure 3: Modular structure of PlaSMA ontologies.

**Ontology-based Simulation World Model**

The PlaSMA implementation provides a model of the physical world which is based on a declarative, formal, and explicit model expressed as an OWL-DL ontology (Bechhofer et al., 2004). In this way, the initial process of scenario design is turned into an ontology engineering task for which standard tools and modeling principles exist. One of the main benefits of this ontology-based approach can, therefore, be regarded to lie in this *standardization* of the scenario design process.

Our implementation of the PlaSMA simulation system provides a modular set of five ontologies which specify terminological knowledge relevant in the logistics domain. These ontology modules, whose link-up is depicted schematically in Figure 3, constitute the formal basis for scenario modelling at design time and are briefly described below.

- TLO The top-level domain ontology for logistic scenarios specifies general types of physical objects, the basics of traffic infrastructure, organizational membership, ownership of physical objects, and software agents responsible for a set of physical objects or providing abstract services. All other ontologies import and extend this ontology.
- TRANS As ontology for multimodal transportation, this ontology defines location and site types, transport relations in the traffic infrastructure as well as means of transport, handling and loading equipment.
- PROD As ontology for intra- and shop floor logistics, this ontology specifies production resources, and orders with their respective properties.
- COM This communication ontology defines communication and computation devices with properties such as radio coverage or power supply type.
- GOODS The goods ontology provides a general schema to classify goods. For instance, classification may distinguish physical or non-physical good (e.g., property rights), packaged or bulk good, perishable or hazardous goods.

These ontologies are neither considered complete with regard to the diversity of logistic sub fields (i.e. the current focus is on distribution and, to a lesser degree, shop floor logistics) nor mandatory for simulations of other application domains. Therefore, the simulation designer is free to use only a custom subset of core modules, may adapt or extend these with additional concept and property assertions, or create substitute ontologies suitable to model a new domain of interest. How this can be achieved in a systematic fashion is discussed later on.

**Scenario Modelling: Infrastructure**

In most application scenarios which have been implemented in PlaSMA so far, i.e. in transport logistics (Gehrke and Wojtusiak, 2008) and spontaneous ride sharing (Xing et al., 2009), a basic traffic network is specified as an annotated directed graph. Vertices therein are non-physical locations in the modeled world. In all but synthetic scenarios, these bear a geographic grounding in the form of geo-coordinates and host stationary logistic infrastructure such as production facilities, harbors, cross-docking stations and warehouses. The transport ontology module provides several types of edges which can be classified as specializations of the base concepts LandRoute, WaterRoute and AirRoute, thus allowing for the modelling of multimodal transport relations between locations. Besides the stationary logistic resources, the scenario ontology also specifies non-stationary logistic resources, in particular means of transport and various flavors of freight objects. The modularity provided by OWL thereby allows for a separate modelling of a basic scenario graph and scenario-specific entities populating that graph, thus rendering the former reusable across scenarios.

**Scenario Modelling: Agents**

Next to this basic model of the physical environment, the PlaSMA ontologies allow for the specification of non-physical entities, such as organizations, and individual agents in their role as decision makers. Modelling of organizational contexts allows for an adequate representation of owner-, responsibility- and membership relations.

Of particular importance with regard to simulation execution is the classification of agent types adopted in PlaSMA and the association of these software agents with entities within the simulation environment. The adopted modelling approach conceptually introduces a partition of all software agents, which constitute a simulation, into distinct agent communities, namely simulation actors and environmental agents.

The former community, made up of object- and service agents, constitutes the MAS which has been deployed in the simulation environment in order to evaluate global performance, patterns of interaction among or the design of particular agents. Object agents in PlaSMA act as artificial autonomous decision makers on behalf of particular physical entities. They may either assume the role of an authoritative digital representative or conduct secondary functions. Service agents offer abstract services to fellow simulation actors such as traffic information, weather prediction or electronic market places. PlaSMA allows for an intersection of both actor groups such that a particular

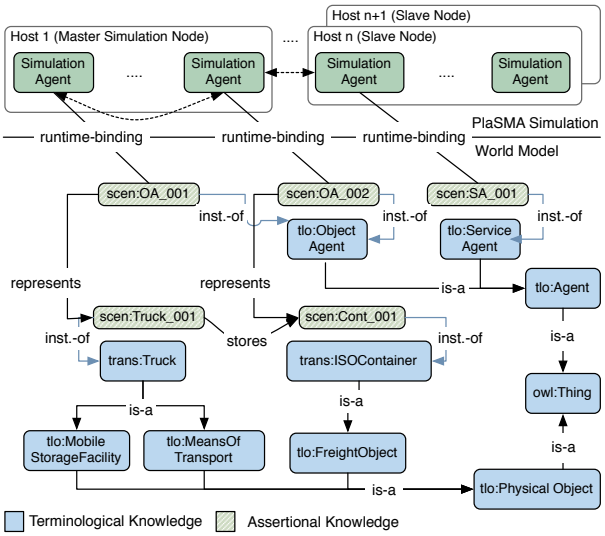


Figure 4: Interrelation of simulation agent, ontological object/service counterpart and physical objects.

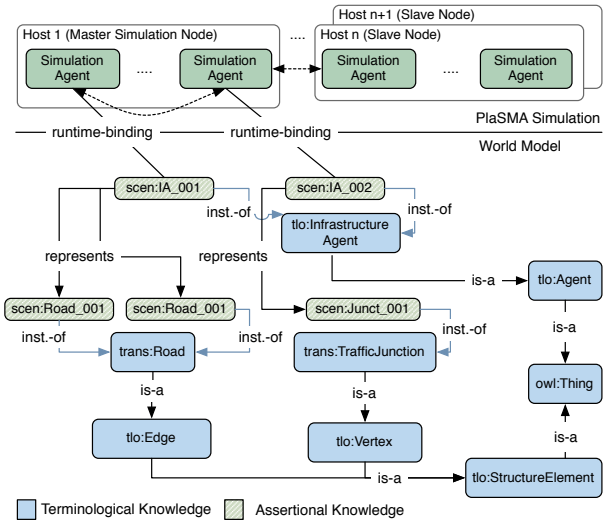


Figure 5: Interrelation of simulation agent, ontological infrastructure counterpart and structure elements.

agent may simultaneously manage physical objects and provide additional services.

The association of object agents and physical entities, whose initial state is modeled at design time, is shown in Figure 4. It may be subject to change as simulation runs unfold. Due to the explicit differentiation of agents and entities, agent modelling enables what may be called *dynamic embodiment* with a mutable 1:n relation from agent to managed entities. The modelling also provides for the additional category of environmental agents whose function can be explained by a theatre analogy. If we think of the modeled environment as set for simulation actors, the former agents as stage technicians modify the set over the course of scenario runs. They are responsible for runtime modification of both topology and characteristics of infrastructure elements within the simulation environment as shown in Figure 5. These agents are also responsible for online creation and destruction of physical entities. In contrast to object and service agents which are modeled explicitly in the PlaSMA top-level ontology, environmental agents are modeled indirectly to retain flexibility as they combine traits of object agent, e.g. when a new container is created and immediately included in a storage facility inventory, and infrastructure agents.

**Scenario Modeling: Validation**

Besides advantages such as extensibility and a clear-cut ontological grounding of all simulation constituents, the modelling approach on the basis of OWL-DL adopted for PlaSMA allows to leverage inferential capabilities of dedicated ontology reasoning systems in order to ensure the logical consistency and validity of simulation models at design time. This holds both with respect to terminological knowledge which describes the simulation domain on the schema level and assertional knowledge, i.e. the particular scenario specification. Modelling flaws can thus be rectified early in development, thereby reducing the number of modelling iterations due to shortcomings only spuriously discovered at run time.

**World Model Interaction**

The ontological modelling of a simulation scenario for PlaSMA specifies the initial state of the environment and the agent to object associations. However, for the simulation to proceed, the ontological agent instances need to be bound to applicable PlaSMA agents written in Java.

Once instantiated, the simulation agents have access to their ontological specification using world model queries. In the example presented in Figure 4, the truck agent can retrieve its managed truck objects and their respective status such as information about loaded freight objects. Besides retrieval, the simulation world model also provides means for its manipulation via two related mechanisms, namely physical actions for use by simulation agents and environment events for use by environmental agents. Depending on the modelling granularity chosen for a particular scenario, the aforementioned actions may thereby correspond immediately to actions associated with physical entities, for instance the *drive* action of a truck, but also to complex actions such as *cargo transport between storage facilities*.

PlaSMA provides a growing library of reusable logistic standard actions and an API to create additional actions. Custom actions are programmatically specified in terms of a) preconditions to be met for their execution, b) concluding effects in the world model, and c), for protracted rather than point-wise actions, transitional action effects setting in at the begin of action execution. Due to the ontological foundation of the PlaSMA world model it is possible at run time to exploit inferential capabilities exposed through the world model query interface. For instance, asserted and derived classes of a particular graph edge passed to a drive action can be ascertained and matched with the action specification. Actions in PlaSMA can be conceived as expansion of the world model whose scope is respectively tailored to the domain of simulation. Thus,

they are granted unconstrained read access, allowing for context-sensitive computation of action effects. For instance, environmental events such as traffic jams or severe weather conditions can significantly prolong the execution of drive actions on affected transport relations. PlaSMA ensures that active actions are notified of world model changes. They then need to interpret these changes with regard to their own execution and, if need be, internally compute action effects incrementally.

The world model allows simulation agents to dynamically create additional agents over the whole course of a simulation run. In autonomous logistics, the runtime creation of new agents is often motivated by dynamic production or transport order inflow. For instance, an agent which manages incoming transport orders for a forwarding agency, might want to delegate the supervision of particular orders to specialized agents. These can be instantiated on demand rather than creating a fixed pool of handling agents upon simulation start.

For further development of the PlaSMA simulation world model, the following extensions are considered. Firstly, there is a need for an automated detection and subsequent resolution of conflicts that arise due to simultaneous or temporally overlapping computation mutually exclusive actions. Secondly, the world model query interface so far does not constrain the retrieval of world model information such that it is up to the agent programmer to implement an adequate scope visibility. In order to disburden the programmer, object agents require an explicit perception which accommodates their dynamic embodiment as they assume control of one or multiple logistic entities at a time. In this case, these agents should be constrained to perceive the environment based on sensors attached to represented objects. Additional challenges to be met by object agents then comprise the locality and incompleteness of perception as well as noise which can be introduced by the sensor models.

### Scenario Visualization and Control

PlaSMA comprises a visualization client which allows runtime control and progress monitoring of simulation experiments. To initiate a simulation run, the client is connected to the main PlaSMA simulation node. The experimenter may choose from previously deployed scenarios whose runtime parameters such as simulation start and duration, number of successive runs, or logging granularity can be customized. Once a scenario is loaded, the client leverages the NASA World Wind[2] mapping engine to render the annotated directed graph which represents the multimodal transport network as well as the physical logistic entities thereon onto a virtual globe. It also provides a tree representation of both agents as simulation actors and physical objects. In particular, the tree view continuously depicts the associations between software agents in their roles as autonomous decision makers and managed logistic entities, either from the agent perspective (i.e. objects managed by respective agents) or the entity perspective (i.e. agents acting either authoritatively or in secondary roles for respective objects). The selection of entities or agents in the simulation delivers insight to their associated world model state which is useful when tracing entities during simulation runs. The PlaSMA client offers support for a distributed monitoring of simulation runs. Multiple client instances may connect to the same PlaSMA server. This allows for monitoring of joint experiments from different locations via multiple viewports.

Work on extending the client's scope of operation is currently underway. Therein, we focus on supporting intuitive collaborative interaction with larger, and therefore more complex, simulations. Within this approach we investigate multi-touch surface computing environments as a means to provide an intuitive interaction framework which allows human agents a) to manipulate the physical simulation environment directly via environment events and b) to allow for online human-agent interaction. The latter thereby renders possible the involvement of human actors as an additional category of decision makers in what could then be considered a *participative* simulation.

## APPLICATION AND EVALUATION

Although actively used as a joint experiment platform by several research groups within the Collaborative Research Centre (CRC) 637 for four years, PlaSMA is still in a prototype stage of development. It is applied for comparison and evaluation of algorithms for logistics planning and special sub-processes therein, such as coordination mechanisms of logistics objects, information distribution, environment adaptation and prediction (Gehrke and Wojtusiak, 2008) as well as routing and cargo clustering algorithms (Schuldt and Werner, 2007). Furthermore, PlaSMA is part of the *Intelligent Container* demonstration platform[3] integrating simulation with real-world hardware in perishable food transport scenarios. In the context of adaptive route planning, PlaSMA has been integrated with the AQ21 machine learning system for prediction of expected traffic and speed on potential routes (Gehrke and Wojtusiak, 2008). PlaSMA is currently integrated with the learnable evolution model (LEM), a library for non-Darwinian evolutionary computation that employs machine learning to guide evolutionary processes (Wojtusiak, 2009). Complexity of simulation surveys ranges from very few agents to large agent communities (20,000).

## EXTENDING PLASMA SCOPE AND USABILITY

While the successful application and evaluation so far has adduced initial evidence that the PlaSMA system facilitates the compilation of simulations for multiple scenarios in our application domains, there is still potential for development which we seek to explore in ongoing research.

[2]Web Site: `http://worldwind.arc.nasa.gov/java/` (visited: 2010/02/25)

[3]Web Site: `http://www.intelligentcontainer.com` (visited: 2010/02/25)

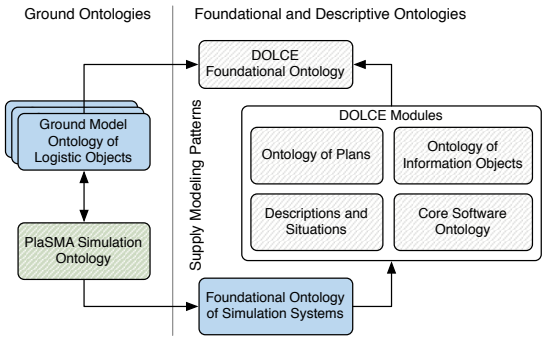


Figure 6: Framework for modelling multiagent-based simulation systems, grounded on DOLCE.

**Reaching beyond Simulations** With regard to the medium-term goal to propagate multiagent-based implementation of logistic decision and control systems from the lab into real-world production systems, MABS is considered a suitable means to test multiagent applications for compliance with specifications at hand. Although multiagent-based applications are initially deployed in a simulation test bed in early stages of their product lifecycle, agent developers should be put in a position where they can focus on the production use case. We propose to augment multiagent-based simulation environments such that simulation-specific portions of the agent code bases are no longer required (Gehrke and Schuldt, 2009). This renders possible a *uniform agent design* suitable for both simulation and operation. The characteristics of the agents' target environment, either real or simulated, is kept transparent from the point of view of the agents. Working towards the uniform agent design ideal, the PlaSMA system has been extended to handle implicit simulation time synchronization (Schuldt et al., 2008).

**Ontology Support for Design, Interaction & Analysis** The field of ontology engineering was motivated in general by the promise of yielding scalable, portable and reusable domain models. Initial results reached by the research community, however, fell short of achieving these promises and turned out to yield more pain than gain. As a consequence, the ontology engineering *process* has been revised and put on more rigorous modelling principles such as the employment of foundational models, such as DOLCE (Masolo et al., 2003) or SUMO (Niles and Pease, 2001), the explication of the corresponding ontological commitments as well as the application of design- and content patterns (Gangemi and Presutti, 2008).

As discussed above, we employ simulations as a central method for examining the capabilities and limits of autonomous logistic processes. In order to enhance the PlaSMA ontologies described above, efforts are underway to put the corresponding models on firmer ontological grounds. For this purpose, we employ the DOLCE foundational ontology with several additional modules, namely *Descriptions and Situations* (Gangemi and Mika, 2003), the *Ontology of Information Objects*, the *Ontology of Plans* and the *Core Software Ontology* (Oberle et al., 2005). We currently develop a foundational ontology that models simulations themselves. It will facilitate the import of other ontological models that employ the same foundational top level and, consequently, further the adoption of PlaSMA for simulations in new application domains. Also, the subsequent use of domain models, initially developed specifically for use in PlaSMA, is rendered possible in new contexts. Next to enabling portability and re-usability, one can employ the resulting framework to explicate context-dependent reifications of the simulated entities and their actions, e.g. a logistic object, such as a truck, can be ontologically described as a *MeansOfTransportation*, a *FreightObject*, or a *TrafficObstacle* depending on the context at hand. In the end the resulting framework will provide a foundational domain-independent ontology of simulations - modelling the constituents of simulations *per se* - together with their respective docking stations for domain ontologies, which model the environment and entities, simulated or real.

## CONCLUSION AND OUTLOOK

In this paper, we have introduced the multiagent-based simulation system PlaSMA with a particular focus on its ontology-based world model. We have described the benefits of the ontological grounding for simulation design and execution. We have shown, how the adoption of the PlaSMA system to new domains requires - in principle - only the creation and integration of a set of new domain ontologies. We hope to ease this engineering effort by employing a standardized and well-used foundational ontology and dedicated simulation-specific design patterns. Encouraged by successful adoption of PlaSMA for experimentation and demonstration in autonomous logistics, in the future, we will explore the potential advantages of using a foundational simulation ontology beyond increasing the scalability, portability and re-usability of the domain simulation models. We anticipate this generalization to affect also the interaction with agent-based simulations for the human stakeholders. Specifically, in the design phase scenario engineering will be facilitated by exploitation of reusable ontological patterns. At runtime, complex simulations could be rendered (more) accessible for human-computer interaction. Finally, in the analysis phase, an easier and well-grounded evaluation of recorded simulation runs will become feasible. We see usability as a central challenges for MABS of domains where we find a mixture of human and artificial decision makers, as one needs to guarantee that the human stakeholders involved will be able both to understand what is happening within such systems and control a running system.

## REFERENCES

Bechhofer, S., van Hamelen, F., Hendler, J., et al. (2004). OWL Web Ontology Language Reference, W3C Recommendation. Technical report, W3C.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Develop-*

*ing Multi-agent Systems with JADE*. Wiley Series in Agent Technologies. Wiley Inter-Science.

Bobeanu, C.-V., Kerckhoffs, E. J. H., and Landeghem, H. V. (2004). Modeling of Discrete Event Systems: A Holistic and Incremental Approach using Petri Nets. *ACM Trans. Model. Comput. Simul.*, 14(4):389–423.

Fujimoto, R. (2000). *Parallel and Distributed Simulation Systems*. Wiley & Sons, New York, NY, USA.

Gangemi, A. and Mika, P. (2003). Understanding the Semantic Web through Descriptions and Situations. In *Proceedings of the ODBASE Conference*, pages 689–706. Springer.

Gangemi, A. and Presutti, V. (2008). *Handbook of Ontologies (2nd edition)*, chapter Ontology Design Patterns, pages 221–244. Springer.

Gehrke, J. D. and Schuldt, A. (2009). Incorporating Knowledge about Interaction for Uniform Agent Design for Simulation and Operation. In *8th Int. Conference on Autonomous Agents and Multiagent Systems*, pages 1175–1176.

Gehrke, J. D., Schuldt, A., and Werner, S. (2008). Quality Criteria for Multiagent-Based Simulations with Conservative Synchronisation. In *13th ASIM Dedicated Conference on Simulation in Production and Logistics*, pages 545–554, Berlin, Germany. Fraunhofer IRB Verlag.

Gehrke, J. D. and Wojtusiak, J. (2008). Traffic Prediction for Agent Route Planning. In *International Conference on Computational Science 2008 (vol. 3)*, volume 5103 of *LNCS*, pages 692–701, Poland, Kraków. Springer.

Hülsmann, M. and Windt, K., editors (2007). *Understanding Autonomous Cooperation & Control in Logistics: The Impact on Management, Information and Communication and Material Flow*. Springer, Berlin, Germany.

Jefferson, D. R. (1990). Virtual Time II: Storage Management in Conservative and Optimistic Systems. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, pages 75–89, Quebec, Canada. ACM Press.

Klügl, F., Oechslein, C., Puppe, F., and Dornhaus, A. (2002). Multi-Agent Modelling in Comparison to Standard Modelling. In *Artificial Intelligence, Simulation and Planning in High Autonomy Systems (AIS 2002)*, pages 105–110. SCS Publishing House.

Lees, M., Logan, R., Minson, T., Oguara, T., and Theodoropolus, G. (2004). Distributed Simulation of MAS. In *Multi-Agent Based Simulation, Joint Workshop*, volume 3415 of *LNCS*, pages 25–36. Springer.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Otramari, A. (2003). D18: Ontology Library (final). Project deliverable, WonderWeb: Ontology Infrastructure fo the Semantic Web.

Niles, I. and Pease, A. (2001). Towards a Standard Upper Ontology. In *Proceedings of the Int. Conference on Formal Ontology in Information Systems*, pages 2–9.

Oberle, D., Lamparter, S., Eberhart, A., and Staab, S. (2005). Semantic Management of Web Services. In *Service-Oriented Computing - ICSOC 2005*, volume 3826 of *LNCS*, pages 514–519. Springer.

Parunak, H. V. D., Savit, R., and Riolo, R. L. (1998). Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In *Multi-Agent Systems and Agent-Based Simulation, First International Workshop*, volume 1534 of *LNCS*, pages 10–25, Paris, France. Springer.

Schuldt, A., Gehrke, J. D., and Werner, S. (2008). Designing a Simulation Middleware for FIPA Multiagent Systems. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 109–113, Sydney, Australia. IEEE Computer Society Press.

Schuldt, A. and Werner, S. (2007). Distributed Clustering of Autonomous Shipping Containers by Concept, Location, and Time. In *5th German Conference on Multiagent System Technologies*, volume 4687 of *LNAI*, pages 121–132, Leipzig, Germany. Springer.

Wojtusiak, J. (2009). The LEM3 System for Multitype Evolutionary Optimization. *Computing and Informatics (28)*, pages 225–236.

Xing, X., Warden, T., Nicolai, T., et al. (2009). SMIZE: A Spontaneous Ride-Sharing System for Individual Urban Transit. In *Multiagent System Technologies: 7th German Conference*, pages 165–176, Hamburg, Germany. Springer.

## AUTHOR BIOGRAPHIES

**Tobias Warden** is a computer scientist and joined the artificial intelligence group at the University of Bremen as a research assistant in 2008. His research interests span distributed knowledge management and collaborative multi-agent learning.

**Robert Porzel** is a senior researcher at the Digital Media Research Group at the University of Bremen. His research encompasses knowledge representation, contextual computing, and natural language processing.

**Jan D. Gehrke** is a computer scientist and joined the artificial intelligence group at the University of Bremen as a research assistant in 2005. His research focuses on intelligent agents in logistics as well as knowledge representation and management in MAS.

**Otthein Herzog** is a professor emeritus. From 1993 to 2009, he held the chair on Artificial Intelligence in the Department of Mathematics and Computer Science at the University of Bremen. Dr. Herzog continues to contribute to the interdisciplinary activities of the CRC 637 which he represented as speaker till 2009.

**Hagen Langer** is a senior researcher at the Artificial Intelligence Research Group of the University of Bremen. Besides knowledge representation and reasoning, his research focus is on natural language processing.

**Rainer Malaka** holds the chair on Digital Media in the Department of Mathematics and Computer Science at the University of Bremen. He directs the TZI – Center for Computing and Communication Technologies.