Multiagent-based Logistics Simulation with PlaSMA

Jan D. Gehrke, Christian Ober-Blöbaum

Technologie-Zentrum Informatik
Universität Bremen
Am Fallturm 1
28359 Bremen
{jgehrke,cob}@tzi.de

Abstract: This paper gives a short overview of the multiagent-based simulation system PlaSMA. The system provides distributed discrete event simulation with conservative synchronisation. PlaSMA has been developed for scenarios within the logistics domain with simulation agents representing logistic objects like lorries or cargo as well as abstract services like route planning or transport brokering. PlaSMA is applied for evaluation and visualisation of control methods.

1 Introduction

Multiagent-based simulation (MABS) attracts increasing attention in the context of complex simulations with potentially great numbers of parallel and interacting sub-processes [Dav00]. In MABS the environment and the objects acting therein are modelled by a number of software agents handled as logical simulation processes [Lee04]. This enables distributed simulation [Fuj00] as well as natural mapping of actors to software agents with proper autonomy encapsulation. Moreover, when using FIPA-conform agents the transfer from simulation to real-world operation is eased, provided that the application includes autonomous sub-systems. In particular in logistics, new control paradigms for decentralised decision making demand for numerous autonomous sub-systems whose performance and interaction has to be tested in advance. In contrast to simulation, software verification would get especially complicated or infeasible when the number of participating instances is dynamic and validation metrics are vague and possibly influenced by stochastic events.

The PlaSMA system[1] provides a distributed multiagent-based simulation and demonstration system for logistic processes based on the FIPA-compliant Java Agent Development Framework JADE [Bell01]. PlaSMA stands for *Platform for Simulations with Multiple Agents*. The system is developed at the University of Bremen, Germany, as part of the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes - A Paradigm Shift and Its Limitations". Within this interdisciplinary project the PlaSMA system is the joint software platform for autonomous logistics applications and evaluations.

[1] Available at http://plasma.informatik.uni-bremen.de

2 System Architecture

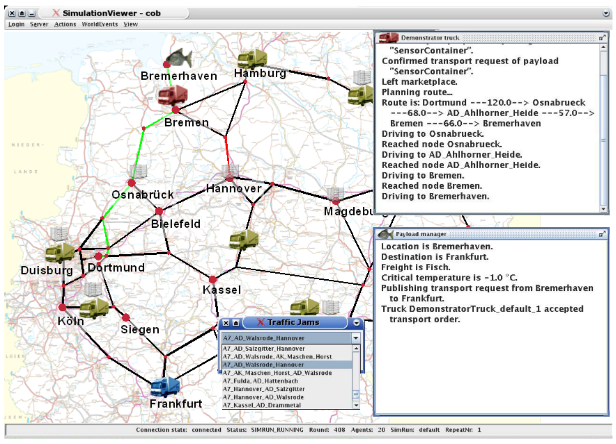
The PlaSMA system consists of the basic components *simulation control*, *world model*, *simulation agents*, *analysis*, and *user interface*. The simulation control handles world model initialisation, time management as well as agent lifecycle management. Furthermore, it provides an interface for world model access. Simulation control primarily consists of two kinds of instances: one top-level controller and a sub-controller for each processor or computer in distributed settings. Sub-controllers handle the actual software agents (called simulation agents) that are the actors in a simulation scenario. Communication between sub-controllers and simulation agents is based on local Java method invocation. The interaction between top-controller and sub-controllers concerns, e.g., agent lifecycle management, runtime control, and time events. For time management, the top-controller sends time events to all sub-controllers to indicate progression of simulation time. Sub-controllers propagate these events to all of their simulation agents (see Sect. 3.1).

The world model component represents the state of the world consisting of its environment and the physical objects acting therein. It is initialised based on a formal ontology description (see Sect. 3.2). The simulation agents represent physical objects, abstract services, or legal entities (e.g., organisations). Simulation agents are able to communicate with each other by message passing in FIPA Agent Communication Language ACL [Fip02] and may act in the environment if equipped with actuators. Scenario-specific agents may be created or adapted by the PlaSMA user by implementing an extension of the basic Java agent class *SimulationAgent*.

The analysis component consists of a relational database storing scenario performance metrics (e.g., cargo cycle times or vehicle utilisation) and an interface to log and query these metrics and additional log messages. The metrics can be observed online within the PlaSMA client. When agents provide position metrics and the scenario configuration contains additional visualisation information the simulation environment can also be tracked within the viewer (see Fig. 1). The client/viewer is connected to the server by Java Remote Method Invocation (RMI) and by JDBC access to the relational database of the analysis component. In order to conduct a simulation experiment the system user starts the PlaSMA server and the PlaSMA client subsequently. The client allows for selection of predefined scenarios available at the server. Afterwards the scenario may be started, paused, and stopped as well as visualised while running.

3 Simulation Model

The PlaSMA simulator employs a discrete-event-like conservative synchronisation and time model. That is, simulation time progresses in discrete steps of heterogeneous length. The simulation actors are parallel logical simulation processes, i.e., software agents representing physical entities, abstract services, or parts of the environment. These agents govern virtual time progression by requesting their next wake-up time at their simulation sub-controller respectively. The simulation agents communicate by sending messages in FIPA agent communication language ACL [Fip02].



```
"SensorContainer".
Confirmed transport request of payload
  "SensorContainer".
Left marketplace.
Planning route...
Route is: Dortmund ---120.0--> Osnabrueck
  ---68.0--> AD_Ahlhorner_Heide ---57.0-->
  Bremen ---66.0--> Bremerhaven
Driving to Osnabrueck.
Reached node Osnabrueck.
Driving to AD_Ahlhorner_Heide.
Reached node AD_Ahlhorner_Heide.
Driving to Bremen.
Reached node Bremen.
Driving to Bremerhaven.
```

```
Location is Bremerhaven.
Destination is Frankfurt.
Freight is Fisch.
Critical temperature is -1.0 °C.
Publishing transport request from Bremerhaven
  to Frankfurt.
Truck DemonstratorTruck_default_1 accepted
  transport order.
```

```
A7_AD_Walsrode_Hannover
A7_AD_Salzgitter_Hannover
A7_AD_Walsrode_AK_Maschen_Horst
A7_AD_Walsrode_Hannover
A7_AK_Maschen_Horst_AD_Walsrode
A7_Fulda_AD_Hattenbach
A7_Hannover_AD_Salzgitter
A7_Hannover_AD_Walsrode
A7_Kassel_AD_Drammetal
```

```
Connection state: connected  Status: SIMRUN_RUNNING  Round: 408  Agents: 20  SimRun: default  RepeatNr: 1
```

Figure 1: PlaSMA client and user interface with visualisation example.

3.1 Time and Synchronisation Model

Time management, i.e., the synchronisation of simulation time of parallel simulation processes, is a crucial issue in distributed simulation [Fuj00, Lee04, Paw06]. In multi-agent-based simulation each agent as a logical simulation process may have its own local virtual time (LVT). Synchronisation is needed when agents interact to avoid causality problems. While optimistic synchronisation uses rollback mechanisms that allow for handling of events with time stamps that differ from the agents LVT, conservative synchronisation prohibits such events.

PlaSMA applies conservative synchronisation using simulation controllers. Although an agent-to-agent synchronisation would be possible in general, the dynamic setting in PlaSMA with new agents that may enter the simulation in a stochastic manner demands for more coordinated handling. That is, each simulation agent may request a simulation time to be activated next at its sub-controller and there is one common simulation time that progresses depending on these requests. When simulation time progresses all agents are activated that either requested for activation at this time or received some event or message. The latter ensures in-time handling of (unexpected) incoming events.

Simulation time is stated as UNIX time stamp with millisecond granularity. To avoid very small synchronisation steps the user may define a minimum synchronisation distance, e.g., 10 seconds, in scenario configuration. Because PlaSMA is also used as demonstration platform with online visualisation, progress of simulation time needs to be delayed here. Therefore PlaSMA provides two parameters that specify (a) the maximal ratio of simulation time to computation time, e.g., at most 100 times faster than real time, and (b) a maximum synchronisation distance. The latter forces all simulation agents' activation within the given simulation time interval in order to ensure steady logging of performance metrics and position data needed in visualisation. The configuration also allows to select the simulation date time at scenario start.

3.2 World Model

Simulated physical objects, e.g., vehicles or transport containers, are positioned in a directed graph of typed and annotated nodes and edges representing the multi-modal transport infrastructure. The scenario infrastructure and the objects or agents therein are specified by an OWL-DL ontology file [Bec04], i.e., a logical domain description. Each scenario ontology file has to import a basic logistics domain ontology and optionally specific sub-ontologies, e.g., for transportation, that are predefined but may adapted by the system user as well.

3 Application and Integration

Due to the world model and its corresponding ontologies PlaSMA is focused on the logistics domain. Nevertheless the general system is easily adaptable for other domains. Currently PlaSMA is applied for comparison and evaluation of algorithms for autonomous logistics planning and special sub-processes therein, e.g., coordination mechanisms of logistics objects, information distribution, and routing algorithms. Furthermore PlaSMA is part of the "Intelligent Container" platform [Jed07] integrating simulation with real-world hardware in perishable food transport scenarios.

Acknowledgement. This research was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes".

References

[Bec04] Bechhofer, S.; van Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D. L.; Patel-Schneider, P. F.; Stein, L. A.: OWL Web Ontology Language Reference. World Wide Web Consortium, Sept. 2004. Internet: http://www.w3.org/TR/owl-ref/.

[Bell01] Bellifemine, F.; Poggi, A.; Rimassa, G.: Developing multi-agent systems with a FIPA-compliant agent framework. Software – Practice & Experience, 31 (2):103–128, 2001.

[Dav00] Davidsson, P.: Multi Agent Based Simulation: Beyond Social Simulation. In Moss, S.; Davidsson, P. (eds.): Proc. Workshop on Multiagent-Based Simulation (MABS 2000), LNAI, Vol. 1979, pp. 97-107. Springer, 2000.

[Fip02] FIPA ACL Message Structure Specification. Document Nr.: SC00061G. Internet: http://www.fipa.org/specs/fipa00061/. 2002.

[Fuj00] Fujimoto, R.: Parallel and Distributed Simulation Systems. John Wiley & Sons, 2000.

[Jed07] Jedermann, R.; Gehrke, J. D.; Becker, M.; Behrens, C.; Morales-Kluge, E.; Herzog, O. and Lang, W.: Transport Scenario for the Intelligent Container. In: Hülsmann, M.; Windt, K. (eds.): Understanding Autonomous Cooperation in Logistics, pp. 393-404. Springer, 2007.

[Lee04] Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G.: Distributed Simulation of MAS. In Davidsson, P. et al. (eds.): Proc. Workshop on Multiagent-Based Simulation (MABS 2004), LNAI, Vol. 3415, pp. 25-36. Springer, 2005.

[Paw06] Pawlaszczyk, D.; Timm, I. J.: A Hybrid Time Management Approach to Agent-based Simulation. In: Proc. 29th Annual German Conference on Artificial Intelligence (KI 2006), LNCS, Vol. 4314, pp. 374-388. Springer, 2007.