In: Wenzel, S. (ed.): Simulation in Produktion und Logistik 2006. Tagungsband zur 12. Fachtagung, pp. 133-142. Society for Modeling and Simulation, Kassel, 2006.

# Integration of two Approaches for Simulation of Autonomous Logistic Processes

Jan D. Gehrke [*]
Martin Lorenz
Universität Bremen

Bernd-Ludwig Wenning [†]
Markus Becker
Universität Bremen

**Abstract**

The concept of autonomous logistic processes addresses the emerging requirements in current and future logistics by applying the latest information and communication technologies. They enable autonomous systems that operate and cooperate as local representatives of logistic entities. The analysis and design of these processes is subject to simulation studies. Two simulation systems for the analysis of autonomy in logistics – an agent-based and a discrete event approach – are presented. Inspired by the time concept of discrete event simulation, a new synchronisation technique is proposed that allows for a dynamic adaptation of simulation time progression in multiagent-based simulation depending on the granularity currently needed.

## 1 Introduction

The dynamic nature of modern transport networks increases the complexity of decision-making in today's logistics. An exact or even heuristics-based solution for global optimisation becomes almost impossible. Since the distribution of planning and decision-making to autonomous components is a widely accepted promising solution to handle complex problems (Jen01), we consider it an appropriate set-up for logistic systems.

Autonomous logistic processes aim at managing logistic processes in a highly distributed way by transferring decision-making competencies to the logistic entities, e.g. in transportation, transshipping facilities, means of transport, or even single freight items, represented by autonomous software systems. These entities coordinate in dynamic, transorganisational, and even competitive environments to perform the processes depending on their respective goals and abilities.

A logistics system based on the above principles allows the transfer of more decision competence from the logistics service provider to autonomous representatives of the logistics system user. Furthermore, it creates a new approach to routing and mode-choice problems. The investigation of algorithms is supposed to include different approaches from artificial intelligence, operations research, and communication networks.

The field of *Artificial Intelligence* (AI) attempts to build such intelligent entities (RN03). In *Distributed Artificial Intelligence* (DAI) AI is broadened for cooperative problem

[*] TZI - Intelligent Systems, Am Fallturm 1, 28359 Bremen, Germany
[†] TZI - Communication Networks, Otto-Hahn-Allee NW1, 28359 Bremen, Germany

solving of multiple intelligent entities thereby providing scalability, flexibility, and effortless reusability. In this context, *Multiagent Systems* (MAS) are the state-of-the-art approach for implementing autonomous and interacting software systems. A software agent is a program that acts autonomously, communicates with other agents, and reasons about its actions with respect to goals (Wei99). In order to achieve interoperability agent communication has been standardised by FIPA (FIPA02). For autonomous logistic processes, logistic entities and services may be represented as software agents interacting with each other to coordinate the logistic process.

The *Collaborative Research Centre 637* (CRC 637) aims at providing fundamental methods and prototype solutions for the paradigm of autonomous local control in logistics. This is enabled by technologies from AI as well as communication and sensor technologies. In order to evaluate, analyse, and compare approaches for autonomy of logistic entities simulation studies are carried out. In this paper we present two simulation environments designed for different but overlapping purposes or aspects. A *Multiagent-Based Simulation* (MABS) environment is applied to survey behaviour, interaction, and performance of agents representing logistic entities. Communication behaviour and requirements of autonomous logistic entities as well as routing algorithms are evaluated with a *Discrete Event Simulation* (DES) environment.

One aim is to integrate those systems for large-scale global simulations while preserving each system's advantages as far as possible. The crucial difference between both systems is their time management. MABS is performed in an inherently parallel and distributed way. It therefore requires sophisticated synchronisation mechanisms. Based on a comparative analysis of both systems we investigate methods for MABS synchronisation regarding runtime performance, flexibility, ease of development, and simulation result accuracy.

The remainder of this paper is organised as follows: In Sect. 2 we present and compare the two simulation systems for logistic processes. Sect. 3 discussed time management in MABS and proposes an improved synchronisation mechanism with first experimental results. The paper concludes with a summary and a prospect to further research.

## 2 Simulation of Autonomous Processes

At CRC 637 two simulation environments were developed resp. adapted to model logistic scenarios and evaluate autonomous logistic processes. The systems use different simulation technologies that are considered adequate for their particular purpose. On the one hand the MABS system aims at testing mechanisms for implementing autonomous behaviour of coordinating logistic entities in a realistic scenario. The DES system on the other hand focuses on the evaluation of new routing algorithms as well as communication needs and costs arising from decentralisation of decision-making in logistics.

### 2.1 Multiagent-based Simulation

As a generic platform for simulating and evaluating logistic scenarios with logistic entities as autonomous actors, an agent-based simulation system has been developed. This system allows for a flexible mapping of logistic entities to software agents and provides logging, evaluation, introspection, visualisation, and interaction features. In general, the simulation platform is designed to support arbitrary logistic scenarios. However, the current simulation scenarios are focused on transportation logistics and road traffic, i.e., agents are chiefly representatives of trucks or their load.

#### Architecture and Control

The platform is designed upon the FIPA-compliant *Java Agent DEvelopment framework* (JADE) (BCPR03). It supports the distribution of the simulated logistic entities to multiple computers. The top-level simulation control instance is an agent called *SimulationManager*. It is responsible for the handling of simulation state transitions (e.g., starting, pausing, and stopping), the set-up of the scenario and its agents including distribution to multiple computers, and agent synchronisation. For each computer configured to participate in the simulation the SimulationManager starts a sub-controller agent, called *ContainerManager*, on the respective machine. This sub-controller is the primary contact point when sending (broadcast) control messages and events to the agents in the simulated scenario. The sub-controller forwards these messages to the agents on its computer, thereby avoiding cross-platform communication traffic.

The agent-based simulation is synchronised by sending events for discrete steps. These steps have a fixed but configurable length in simulation time. The computation time for one step varies depending on the calculations of the slowest agent.

#### Configuration and Logistic Model

The configuration of an agent-based logistics simulation on the platform incorporates different aspects, e.g., (1) the initial state of the logistics scenario, (2) the mapping of logistic entities to software agent classes, (3) the computers that host the distributed simulation, (4) the visualisation of different maps and logistic entities.

Agents may be, e.g., suppliers, customers, means of transport, transshipping facilities, traffic nodes, cargo, or services like routing, traffic information, or logistics services brokers. The initial state includes the participating and autonomous logistic entities and their respective types, and properties. Another important part of the scenario specification is the transport network which consists of a directed graph with typed and annotated nodes and edges. The scenario is specified in a formal, machine-processable way using an ontology in the W3C standard web ontology language OWL (MvH04).

Background knowledge on the logistics domain is described in several ontologies for different aspects of logistics, e.g., transportation. Ontologies not only define the initial scenario state, they provide a common formal language for the agents to represent and communicate the state of the simulated world.

### 2.2 Discrete Event Simulation

#### Development Basis and Architecture

The DES environment used for the research presented here was developed based on the *Communication Networks Class Library* (CNCL), a class library developed at RWTH

Aachen for the simulation of communication networks (GJW93). This class library provides all the basics that are needed for event-based simulation like scheduling, event handling, random generators, statistical analysis as well as other general classes. The major components of a CNCL-based simulation are the scheduler and the event handlers. The event-processing simulation objects have to be modelled as event handlers derived from a generic event handler class.

In the DES, the objects send events to the scheduler which include information about the event's destination, the time the event has to be executed and in some cases objects that are attached to the event. When the time for the event has come, the scheduler forwards it to its destination. In the object receiving the event, the event handling method executes the object's actions depending on the content of the event. So the logistic objects are modelled as object-oriented objects with event handlers. The autonomy of the logistic objects is implemented in the event handling method.

The simulation covers the interrelated communication and transport problem. This means it is neither a simulation of just transport nor a simulation of just communication, but an interconnected simulation, where each autonomous logistic object acts on its own and its communication need is recorded. The integration of communication investigations into the simulation is because the autonomous cooperation is not only to be investigated with respect to the impact on the logistic performance, but also with respect to the communication system requirements. By monitoring the communication traffic, it is possible to measure the amount of data that is communicated and therefore the bandwidth that is required. Furthermore, the robustness of the autonomous cooperation concepts under limited availability of communication networks can be investigated by interrupting communication flows.

### Logistic Model

The DES model is based on a scenario description which contains transport related components as well as communication components. The transport related components are instances of the following types:

**Fixed components** form the geographic layout and the transport needs in the scenario. Locations where the direction can be changed and/or load can be transshipped are called *vertices*. Their complexity can range from a road junction to a fully equipped transshipment centre. They can be extended by sources that generate transport demand (packages and their orders). The scenario description specifies details for the generation such as generation rates and destinations, the latter implicitly defining the respective vertices as sinks. The vertices are connected by *edges*. Edges are considered to be directed, this means there have to be two edges between two vertices if travelling is allowed in both directions.

**Mobile components** are those components that move from vertex to vertex, i.e. the vehicles and the *packages* they carry. Vehicles are only allowed to move on edges of a suitable type, i.e. a road vehicle cannot travel on rails. Packages are the goods to be transported. A package is considered to be indivisible, i.e., if a load is divisible, it consists of more than one package.

Furthermore, the scenario contains the following communication relevant objects: *CommunicationUnits* are the devices enabling the logistic objects to communicate. A logistic object may have multiple CommunicationUnits for different communication networks. They are managed by a *CommunicationUnitManager* (one per logistic object) which selects the appropriate CommunicationUnit according to current requirements. Outside of the logistic objects, one *MetaCommunicationUnitManager* performs the actual communication and enables tracking of the amount of communicated data.

Most of the logistic and communication components mentioned here have a couple of attributes, e.g. a vehicle has a capacity, a maximum speed and others. The scenario description is given in XML formatted input files. At the initialisation of a simulation, these input files are parsed and all objects contained in the scenario are created and initialised with the respective attributes.

The output observed in the DES is the packet travel time, utilisation of the vehicles, storage usage in vertices, etc. in the format of histogram and probability density function. Furthermore the routes taken by packages and vehicles are recorded to track their ways through the logistic network. The total amount of data communicated, amount of data per vehicle, per communication system etc. is output with respect to the communication.

## 2.3 Comparative Analysis

The described systems share the goal of simulating aspects of autonomous logistic processes. However, they differ in a variety of important aspects because they were designed for different purposes and application areas. In the following, the two simulation environments are briefly juxtaposed with a focus on time management. A more detailed overall comparison can be found in (BWG+06).

In (BWG+06) it was discussed that the strength of DES is reproducibility of simulation results (given a common random seed) and runtime performance. MABS enables distributed simulations and is more flexible w.r.t. changes in scenarios and agent behaviours. Concerning statistical evaluation, both approaches provide similar tooling.

The multiagent simulation system provides a flexible test bed to analyse and compare different algorithms for autonomy in complex environments. Its major advantage is that it perfectly meets the notion of autonomous logistic entities that directly interact with each other. Thus it is supposed to be easily transferable for real-world applications.

But problems in run-time performance are apparent in comparison to DES. In similar medium-size transportation scenarios with 25 nodes, 250 trucks, and 1000 hours simulated model time the DES system was clearly superior to the MABS system regarding run-time performance (>300 times faster). Besides the efforts for text parsing in agent message passing, the stepwise simulation process is one of the major performance problems. This synchronisation mechanism causes a considerable administration overhead in large-scale simulations and thus neutralises the advantage of parallelism.

## 3 Time Management in MABS

Time management is a crucial issue in parallel simulation - in parallel discrete event simulation as well as in multiagent-based simulation - as discussed in (Fuj00; LLT04; PT06). While the presented DES system is strictly sequential, MABS based on agents running as single processes is inherently parallel. Although this enables distributed simulations and thus the advantage of scalability and increased performance, synchronisation needs arise that again may significantly decrease performance. In parallel simulations each partial simulation is a separate logical process (e.g., an agent in MABS) with its own simulation time, called Local Virtual Time (LVT). Whenever two or more agents interact by communication or actions in a common (simulated) environment, they need to synchronise their LVTs, e.g., to prevent the physical impossibility of two trucks overlapping in space-time (without crashing). That is, every agent has to process events (percepts and messages) in increasing event timestamp order, called local causality constraint (LCC).

The MABS environment presented in Sect. 2.1 prevents LCC violations by discrete time steps for equal LVTs triggered as events by a simulation controller. All agents perform their actions within these steps, report when they have finished, and wait for the next time step event afterwards. This is considered the easiest way to achieve agent synchronisation with respect to implementation efforts.

On the other hand, discrete steps of homogeneous length create a number of problems. First, there is a considerable synchronisation overhead due to the explicit report of each agent that has finished its time interval followed by the event of the next step that has to be sent to all agents afterwards. Secondly, the granularity of time steps influences the accuracy of the simulation results. With longer synchronisation intervals accuracy decreases, e.g., because agent interactions wear on several time steps but would have ended within a split second in real world. On the other hand, when scaling down all time steps for accuracy, nothing might have happened in the meantime, thereby slowing down the simulation.

### 3.1 Dynamic Adaptation of Time Progression

As a result of the comparative analysis (Sect. 2.3, (BWG+06)) and the above discussion of synchronisation issues we extended the MABS system by a dynamic step size adaptation. In this approach each agent determines the point in simulation time when it needs to deliberate next. Within an agent multiple behaviours determine this next time respectively. The minimum time is decisive. Alternatively an agent or agent behaviour may omit this time statement to indicate that it is passively waiting until something happens.

The desired next deliberation time is communicated to the simulation sub-controller (i.e., *ContainerManager*) as an additional parameter to the signal denoting that the agent has finished its deliberation. In a hierarchical process of social choice each container manager determines the minimum requested time. This time is reported to the parent controller,

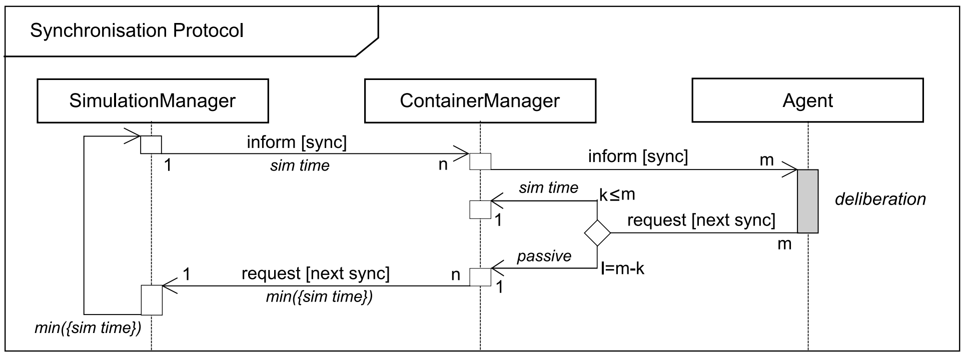

Figure 1: The simulation synchronisation process as simplified agent interaction diagram.

i.e., the simulation manager as top-controller. The overall minimum time is chosen as the next point in simulation time for deliberation of all agents. This is reported as a time event to the container managers and subsequently to each agent in simulation thereby starting an agent's new deliberation cycle. The process is depicted in Fig. 1.

In our simulation model agent deliberation is considered to take no or a minimum amount of simulation time, i.e., an agent does not have to take into account time passing by while reasoning but may reason as long as needed in actual computation time. Agents may communicate within their deliberation cycle and choose their actions at cycle end. The action results are processed in the next cycle. The next requested simulation time is greatly influenced by the actions to be performed (or *scheduled*). Agent communications are a special case discussed in Sect. 3.3.

The dynamic-step-size approach adapts the idea of discrete event simulation with time progressing in heterogeneous steps. At the same time it preserves the ability of pro-active agent behaviour by scheduling their deliberation events instead of just simulation events, e.g., a truck arrival at some destination. The mechanism is relatively easy to implement by not scheduling general events but synchronisation points integrated in the existing step size control. Furthermore, communication efforts are distributed and reduced locally by means of sub-controllers that manage groups of agents.

## 3.2 Optimistic Synchronisation

The presented time management prevents violation of local causality constraints by conservative synchronisation. That means that all agents are steadily in sync. In optimistic synchronisation approaches possible causality violations are caught by rollbacks if an incoming event has a timestamp in the past with respect to LVT. Rollbacks require undo-operations for every action performed and message sent after the event's timestamp. This again may cause a chain reaction in other logical processes (i.e., agents). In order to avoid extensive rollbacks there are also hybrid approaches that constrain the degree of optimism by a maximum time window for LVT deviation (LLT04; PT06). As an additional problem, indirect interactions in a common environment are not imperatively obvious to recognise, particularly in complex models like large-scale logistic scenarios.

Thus, optimistic synchronisation is more challenging in implementation. The adaptive conservative approach outlined in this paper is a compromise between ease of development, simulation accuracy, run-time performance, and flexibility in agent development. Anyhow, we are planning to test optimistic synchronisation as introduced by Pawlaszczyk & Timm (PT06).

## 3.3 Communication and Local Deliberation Termination

As stated in Sect. 3.1, deliberation and also agent communication is not considered to consume simulation time. But again, parallel execution of simulation entities complicates implementation. In order to faithfully terminate a deliberation cycle for synchronisation, an agent needs to know whether a conversation counterpart will respond within the current time step. Two issues have to be regarded here: (1) Explicit knowledge is needed on whether a response will be received within the same time step depending on the interaction protocol and the kind of message sent therein. (2) Due to concurrent agent execution the opponent might already have finished its deliberation cycle.

The second issue reveals a general problem in step-oriented synchronisation: the heterogeneity of the agents' end of deliberation cycles. One agent might have finished while another is still reasoning. In this case, should an agent be able to react on new incoming percepts/messages although its deliberation cycle already ended? If it is not allowed to do so this might have consequences on accuracy of simulation results. For instance, a truck agent might have finished deliberation when a freight item agent sends a transportation request. If the truck handles this request in his next deliberation cycle the freight item has to wait for response for the length of step size in simulation time which may be a significant delay.

This problem can be addressed in two ways: (1) one could allow for handling of incoming agent messages between two agent deliberation cycles. (2) Step size could be adapted when waiting for responses in order to minimize delays. The approach of permitting agent communication although an addressed agent has already finished his current deliberation cycle turns out to be complicated. If the agent handles these incoming messages they would have to be interpreted as if they were received in advance, i.e., before choosing the next actions and the next simulation time for deliberation. Thus, the agent could have to undo his actions and correct his next requested synchronisation point. The latter again is difficult with respect to concurrency because the agent needs to be certain that the synchronisation point has not been finished globally by the simulation top-controller in the meantime. Thus, we favoured the second approach of adapting step size to a minimum when engaged in communication processes. This has the disadvantage of increased synchronisation costs but ensures simulation accuracy.

## 3.4 First Results

The dynamic progression of simulation time by means of hierarchical social choice of simulation time for next synchronisation and deliberation has been implemented in our MABS system. Up to now, the gain in run-time performance was rather low. This is due to the dependencies of each agent's implementation. If agents do not select next deliberation and synchronisation points that allow for larger steps in simulation time the synchronisation costs do not decrease significantly. But we could observe significant improvements in simulation result accuracy. Agent negotiations that could wear on several minutes (sometimes an hour!) in simulation time now are handled within a split second. Furthermore agents are able to react on events instantaneously. Logistic scenarios that got unstable w.r.t. in-time delivery and amount of freight items waiting are now managed smoothly.

The run-time performance could be improved by enhancing the message passing in the synchronisation process. Currently synchronisation messages from and to simulation agents are FIPA messages that require text parsing. We are going to replace plain text by serialized Java objects. As another approach one could replace message passing by a shared memory for synchronisation control. The change to dynamic step size also involved a change in the basic agent design that permits four times as much agents to be simulated on one computer by reducing the number of operating system threads needed for one agent.

## 4 Conclusions and Further Research

In this paper two simulation systems for the analysis of autonomous logistic processes were presented. The multiagent based simulation system was enhanced by integrating time concepts available in the discrete-event simulator.

The MABS gains with respect to simulation accuracy due to the introduction of non-fixed time steps. At the same time it keeps the ability to concurrently execute parallel actions. This results in more comparability between both simulators. Furthermore, the analysis of processes in different time scales, e.g., transport (hours and more) and communication (minutes and less), has been enabled. Further research in the simulation of autonomous logistic processes will comparatively analyse algorithms such as the *Distributed Logistic Routing Protocol* (DLRP) (SRRF06), which are currently implemented in both simulators.

One major drawback of parallel distributed simulation is its potential breach of causality. The synchronisation mechanisms needed to prevent this have been identified as the main performance handicap in MABS. Optimistic synchronisation as a means to speed up scenarios with slightly interconnected components will be integrated in the MABS system. Another step to improve synchronisation is to introduce spheres of influence to avoid overall synchronisation between all components.

## Acknowledgements

This research was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 “Autonomous Cooperating Logistic Processes”. We would like to thank our student research assistant Christian Ober-Blöbaum for his contributions in implementing the MABS platform.

## References

[BCPR03] Fabio Bellifemine, Giovanni Caire, Agostino Poggi, and Giovanni Rimassa. *Jade - A white paper*. Internet: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf, 2003.

[BWG+06] Markus Becker, Bernd-Ludwig Wenning, Carmelita Görg, Jan D. Gehrke, Martin Lorenz, and Otthein Herzog. *Agent-based and discrete event simulation of autonomous logistic processes*. In: W. Borutzky, A. Orsoni, and R. Zobel, editors, Proceedings of the 20th European Conference on Modelling and Simulation, Bonn, Sankt Augustin, Germany, S. 566–571, 2006.

[FIPA02] Foundation for Intelligent Physical Agents. *FIPA standard status specifications*. Internet: http://www.fipa.org/repository/standardspecs.html, 2002.

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

[GJW93] C. Görg, M. Junius, and B. Walke. *Hand-on-Tools Praktikum*. RWTH Aachen, Germany, October 1993. Available at http://www.comnets.rwth-aachen.de/doc/cncl.html.

[Jen01] Nicholas R. Jennings. *An agent-based approach for building complex software systems*. Comms. of the ACM, 44(4): 35–41, 2001.

[LLT04] Michael Lees, Brian Logan, and Georgios Theodoropoulos. *Time windows in multi-agent distributed simulation*. In: Paul Davidsson, Brian Logan, and Keiki Takadama, editors, Multi-Agent and Multi-Agent-Based Simulation: Joint Workshop MABS 2004, number 3415 in Lecture Notes in Artificial Intelligence, S. 25–36. Springer, 2004.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. *OWL web ontology language overview*. Internet: http://www.w3.org/TR/2004/REC-owl-features-20040210/, February 2004.

[PT06] Dirk Pawlaszczyk and Ingo J. Timm. *A hybrid time management approach to agent-based simulation*. In: Proceedings of the 29th annual German Conference on Artificial Intelligence (KI 2006), June 2006.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2003.

[SRRF06] Bernd Scholz-Reiter, Henning Rekersbrink, and Michael Freitag. *Kooperierende Routingprotokolle zur Selbststeuerung von Transportprozessen*. Industrie Management, 3, 2006.

[Wei99] Gerhard Weiß, editor. *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts, 1999.