Advances in Simulation for
Production and Logistics Applications
Markus Rabe (ed.)
Stuttgart, Fraunhofer IRB Verlag 2008

Quality Criteria for Multiagent-based Simulations with Conservative Synchronisation

Qualitätskriterien für multiagenten-basierte Simulation mit konservativer Synchronisation

Jan D. Gehrke, Arne Schuldt, Sven Werner
Technologie-Zentrum Informatik (TZI), Universität Bremen, Bremen (Germany)

Abstract: The concept of multiagent-based simulation introduces the agent programming paradigm to simulation. Multiagent systems ease the implementation of software systems to control complex business processes, such as supply chain management (SCM). Problem complexity is decreased by abolishing monolithic programs. Instead, decision-making is delegated to software agents as local entities. This allows coping even with processes that cannot be controlled centrally due to their inherent physical distribution. Simulation allows evaluating logistics strategies regarding their applicability in such processes. However, general agent development frameworks are not designed to consider simulation-specific issues. In particular, they provide no means for synchronisation. This paper identifies time model adequacy, causality, and reproducibility as quality criteria that must be ensured by a simulation middleware implementing synchronisation. Furthermore, a formal definition of these quality criteria for conservative synchronisation is presented.

1 Introduction

Transport logistics processes are inherently physically distributed, in supply chain management (SCM) often even over multiple continents. Several issues make it a challenging task to control such processes by software systems:

1. The high complexity due to the number of participants, as well as their distribution and their interactions.
2. The high degree of dynamics due to continuously changing demands as well as delays in the process.
3. The physical distribution that prevents relevant information from being available centrally.

These arguments illustrate why it is virtually impossible to follow the traditional way of centralised monolithic programs in order to control such processes. Even

decomposing program complexity by object orientation is often not sufficient to cope with this problem. Then, it is necessary to additionally delegate decision-making to local entities. This can be accomplished by the agent programming paradigm (Weiss 1999; Jennings 2001). In this concept, real-world entities are represented by their own autonomous software agents, thereby preserving the original structure of the system. This eases software development as follows:

1. The problem complexity is significantly reduced because every single agent only concerns the objectives of the entity it represents.
2. The decomposition into autonomous units allows flexible interaction in order to cope with dynamics.
3. It is not necessary to disclose decision-making and relevant information to a central unit. Instead, agents just interact by communication like companies that make contracts in real world.

Simulation is a powerful tool in order to evaluate logistics strategies and their application in advance. This is particularly necessary when complexity and dynamics prevent to analytically predict the outcome of a process. Multiagent-based simulation (MABS) applies the concepts of multiagent systems to simulation (Herrler and Klügl 2006). New findings from MABS can be directly transferred from simulation to real-world operation. Existing agent platforms, like JADE (Bellifemine et al. 2007), however, are generally not designed to consider simulation-specific issues. In order to apply such platforms for MABS, a middleware has to be introduced that implements synchronisation. For this purpose certain quality criteria, namely time model adequacy, causality, and reproducibility, must be considered. These criteria supplement general quality criteria (Wenzel et al. 2008) to be considered in simulation projects.

The remainder of this paper is structured as follows. Sections 2 and 3 introduce foundations of MABS as well as simulation time management. Section 4 discusses and formalises the synchronisation-related quality criteria. Subsequently, section 5 describes their implementation in a simulation middleware. Finally, section 6 provides a conclusion.

2 Multiagent-Based Simulation

Multiagent-based simulation can be categorised as distributed simulation with discrete time model using software agents as parallel logical processes in simulation. It combines simulation scalability and runtime acceleration with encapsulation of decision-making in agents (Parunak et al. 1998).

On the most abstract level, simulation structure can be either macro or micro simulation (Davidsson 2000). *Agent-based* modelling pertains to the category of micro modelling methods. In this concept the behaviours of individuals are encapsulated by logical processes that interact with each other (Davidsson et al. 2007). Instead of being globally visible, variables are evaluated and validated on the individual level.

As discussed above, agents completely encapsulate their internal state (information hiding) like in *object-based simulation*. This eases simulation model development

because there exists a natural mapping between real-world entities and their simulation counterparts. Additionally, agents of different companies may interact with each other without the necessity to reveal internal structures for decision-making (Parunak et al. 1998). In classical object-based simulation the objects are passive in the sense that they are manipulated by central transition rules or programs (Herrler and Klügl 2006). Agents, by contrast, are proactive units that autonomously control their behaviours. This is important when simulating humans or autonomous systems that are capable of acting without external stimuli (Davidsson 2000). In classical parallel and distributed 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 restriction does not hold for MABS because each agent may interact with all other agents (Lees et al. 2005). Agents can even join or leave simulation during execution, e.g., depending on a stochastic simulation model. Thus, it is impossible to predict communication partners, and therewith the simulation model topology, in advance (Lees et al. 2005).

3 Simulation Time Management

When dealing with simulation it is important to distinguish different notions of time (Fujimoto 2000). *Physical time* generally refers to the time at which simulated events would happen in the real world whereas *simulation time* models the physical time within simulation. Simulation time in MABS progresses in a *discrete* way. The gold standard is *discrete event simulation* where time progression is driven by events. Finally, *wallclock time* refers to the time that is consumed by the simulation program in executing the simulation.

In distributed simulations multiple logical processes (LPs) run concurrently on different platforms or processors, which may differ regarding their computational power. Therefore, simulation time progression depends on the CPU and the computational needs of each LP. Consequently, each LP has its own *local virtual time* (LVT). In MABS, logical processes are implemented as agents that usually run as operating system threads. Thus, agents even run concurrently on a single CPU simulation platform.

As long as agents are independent of each other, concurrency does not matter. But problems may arise whenever agents interact. Consider an agent passing a message to another agent that is advanced in its local virtual time. The recipient of such a *straggler message* might have taken other decisions if it were aware of the message on time. This is denoted as the *causality problem* (Fujimoto 2000). In order to guarantee correct simulations events have to be processed in accordance with their timestamp order.

Diverging local virtual times are addressed by synchronisation which can be either *optimistic* or *conservative*. The difference between both approaches is as follows. Optimistic synchronisation generally does not restrict progression of LVT for agents. Whenever a straggler message is received the respective agent is reset to its past state at the LVT of the received message (cf. Jefferson 1985). By contrast, conservative synchronisation strictly prevents causality problems. This can be achieved

if agents commit to send no further messages before a specified point in simulation time. All events before the minimal commitment are safe to process.

Both optimistic and conservative synchronisation approaches have advantages and drawbacks. An advantage of optimistic synchronisation is that it allows a more efficient execution of simulation because fast processes do not have to wait for slower ones. Regarding time consumption, optimistic simulation is thus preferable to conservative simulation (as long as the occurrence of straggler messages is limited). But, optimistic synchronisation has potentially high requirements regarding memory (Fujimoto 2000) because all preceding states of every agent must be stored at worst. This is a particular problem in MABS because agents may need to manage extensive knowledge bases. The space complexity may be reduced by *time window*s for optimistic synchronisation (Lees et al. 2005; Pawlaszczyk and Timm 2006). Nevertheless, runtime performance may significantly decrease when state saving requires frequent and extensive I/O operations.

With conservative synchronisation time progression is potentially slower. But memory requirements are significantly lower because there is no need to store past states of agents. Human interaction or monitoring (Davidsson 2000) is an additional argument for conservative synchronisation. If the simulation requires user interaction, it is not desirable that some processes have to wait for user input while others advance (arbitrarily far) in time. Furthermore, if the simulation is linked to a visualisation component, the states of all agents have to be visualised at the same point in time. Hence, the application of conservative synchronisation is adequate in domains showing these properties although it restricts the speedup achievable by parallelism. Remember that MABS usually incorporates a great number of agents and that interaction between them cannot be predicted in advance. Obviously, it is inadequate that each agent continuously synchronises with all others. Thus, coordinated synchronisation control is required for conservative synchronisation as, for instance, provided by barrier synchronisations known from parallel computation.

4 MABS Quality Criteria

The multiagent-based simulation paradigm features special properties that make it distinct from normal object-based simulation or distributed discrete event simulation. In particular, agents model autonomous and proactive entities that cannot be manipulated directly by method invocation. By contrast, agents only receive messages as simulation events from other agents and decide locally when and how to handle them. This may also include ignoring incoming messages. In this survey we particularly focus on agent communication in multiagent systems following the standards of the IEEE Foundation for Intelligent Physical Agents (FIPA). This implies using the standardised FIPA agent system architecture and the Agent Communication Language (ACL). Messages are no longer just simulation-specific representations of events that change state variables but become an important part of the modelled domain because they represent the flow of information among agents.

The above characteristics of MABS imply *additional* quality criteria for simulation time management that are partially different from those known for distributed simulation systems. The applied time model and synchronisation mechanism influence

simulation results regarding *time model adequacy*, *causality*, and *reproducibility*. Time model adequacy, in this context, denotes the level of abstraction in time progression as well as duration of agent actions including communication. The synchronisation mechanism has to ensure causality constraints (also referred to as correctness) with respect to that model. Reproducibility (or repeatability) of simulation results given the same model and the same random seeds is a quality criterion that does not affect simulation result accuracy. Nevertheless, it is important for traceability and analysis of occurring effects and possible modelling errors.

4.1 Time Model Adequacy

Time model adequacy addresses the challenge of discretizing physical time to simulation time and mapping events to certain timestamps. The maximum granularity of simulation time is restricted by the maximum integer number that can be represented and the maximum virtual length of simulations permitted. Usually, all seconds from ca. 1902 to 2038 can be represented in UNIX time format; other formats also support milliseconds. At least the latter granularity should be sufficient for most applications imaginable in logistics, including flow of information.

However, very fine-grained simulation time is likely to be very harmful to simulation runtime performance. Thus, the minimal granularity needed has to be carefully determined in advance. This granularity will be referred to as Δt_{min}. It specifies the minimal progression of simulation time between events as well as the acceptable *artificial synchrony*, i.e., events within such an interval are considered simultaneous in simulation time although they are potentially not in physical time. A reasonable value for Δt_{min} depends on the modelled domain and simulation purpose. In general, one should not accept values smaller than the distance of two possible events whose order is of importance. Otherwise, simulation results may be corrupted.

Agent communication using message passing is analogue to events sent between logical processes in distributed simulation. Nevertheless, agent messages have some special properties. The timestamp of events denotes the simulation time when the respective event is intended to change some simulation variable (or behaviour) of the simulation process receiving the event. Although this also holds for MABS one has to think about the message timestamp in a different way. In a naïve perspective the timestamp of an agent message equals the simulation time it was created at. Obviously, this would presuppose that message passing can be done without time consumption. As indicated above a sequence of dependent events like an agent conversation sequence should be mapped to different timestamps for every event.

Thus, the simulation communication model should consider the transmission duration trans(m) for a message m sent at simulation time sent(m) from one agent to another. Again, an appropriate value for trans(m) depends on the modelled domain. One might look at expected Internet communication latencies for appropriate values. The timestamp of the message event is then defined by:

$$\text{received}(m) = \text{sent}(m) + \text{trans}(m)$$

The left-hand scenario in figure 1 depicts the problem of agent communication without simulation time consumption. A complex sequence of agent interactions (e.g., negotiating a contract) is handled at a single timestamp although it would con-

sume physical time. Thus, the result of this communication process happens earlier than actually possible in real world, thereby possibly corrupting simulation results.

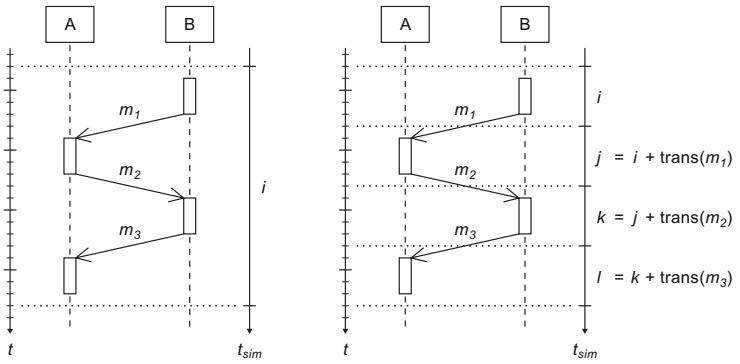

Figure 1: *Agent Communication without (left) and with (right) Time Consumption*

All agent communication should take at least a minimum amount of simulation time (fig. 1 right), i.e.

$$\forall m \; \text{trans}(m) \geq \Delta t_{min} > 0$$

The minimal time progression Δt_{min} also defines a maximally accepted imprecision of event timestamps. The potential artificial delay of messages induced by trans(m) and Δt_{min} must not exceed Δt_{min}:

$$\forall m \; \text{trans}(m) - \text{trans}'(m) < \Delta t_{min}$$

Here, trans'(m) denotes the intended message transmission duration in physical time in contrast to the simulation time duration trans(m) which is actually modelled and simulated.

4.2 Causality

As discussed in the preceding section, time model adequacy demands message transmission to consume simulation time. This constraint also has an impact on causality, the second quality criterion to be considered in MABS synchronisation mechanisms. It can be motivated by the following example (fig. 2). Consider two agents *A* and *B*. At simulation time $t_{sim} = i$, *A* passes a message *m* to *B*. The order in which *A* and *B* are executed in wallclock time *t* depends on operating system scheduling. Thus, it depends on scheduling whether *B* receives *m* at $t_{sim} = j$ or $t_{sim} = i$. In the first case (fig. 2 left), *A* is scheduled after *B* has finished execution at $t_{sim} = i$. Hence, *B* has no possibility of receiving *m* at $t_{sim} = i$ and perceives *m* not until $t_{sim} = j$. This is in accordance with the requirement that transmitting *m* must consume time. However, there also exists another case (fig. 2 right) in which *m* is delivered to *B* at $t_{sim} = i$. *A* is scheduled before *B*. Therefore, *B* could still perceive *m* at $t_{sim} = i$ after *A* has sent it at the same point in simulation time, i.e., at a point in time *m* should not yet be visible.

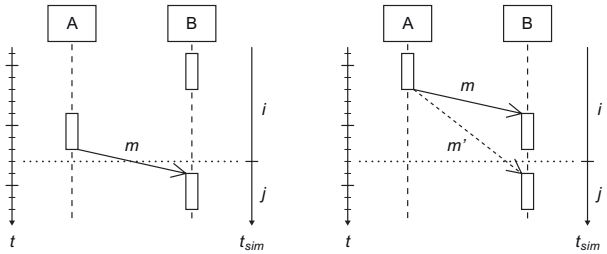

Figure 2: *Controlling Message Visibility to Ensure the Causality Constraint*

In classical parallel discrete event simulation it does not pose a major problem if messages arrive early. Whenever messages arrive early they are simply not processed until time progression arrives at the timestamp intended. By contrast, message handling is more complicated in multiagent-based simulation. This is due to the fact that agents act autonomously and that they have unrestricted access to their message inbox. Thus, simulation model developers would have to handle such messages explicitly in order to satisfy the causality constraint.

However, burdening agents and their developers with synchronisation-related issues is error-prone and thus not desirable. Instead, the simulation system should transparently handle message perception. From this follows that the simulation system has to ensure that agents cannot perceive messages before their intended arrival time (message m' in fig. 2 right):

$$\forall m \; \forall t_{sim} \; \text{Perceivable}(m, t_{sim}) \rightarrow t_{sim} \geq \text{received}(m)$$

4.3 Reproducibility

As an intermediate result, adequacy and causality constraints ensure that messages are never processed at the same simulation time they have been sent at. Thus, it does not depend on operating system scheduling at which time messages arrive. From this follows that message arrival is reproducible over multiple simulation runs. Nevertheless, this does not necessarily guarantee reproducible results. Figure 3 depicts an example. At $t_{sim} = i$, both agents A and C send a message to agent B, m_A and m_C respectively. Both messages are received by B at $t_{sim} = j$. However, there exist two possible ways in which the messages can be ordered in the inbox queue of B. If the operating system schedules A at an earlier wallclock time t than C, m_A is before m_C in the queue (fig. 3 left). Otherwise, the messages are in reverse order (fig. 3 right). From this follows that results of simulation runs would not be reproducible. In order to address this issue, a consistent message ordering $\leq_M$ must be imposed. Message ordering directly depends on operating system scheduling. Remember that an agent platform and its simulation middleware do generally not have any influence on operating system scheduling. Thus, the ordering has to be introduced at a later stage. Actually, it is sufficient to order message as soon as they are added to the message queue of the receiving agent.

The first message queue ordering criterion is, of course, the timestamp at which the respective message has been received. Whenever two messages have the same timestamp an additional ordering criterion $o(m)$ has to be considered:

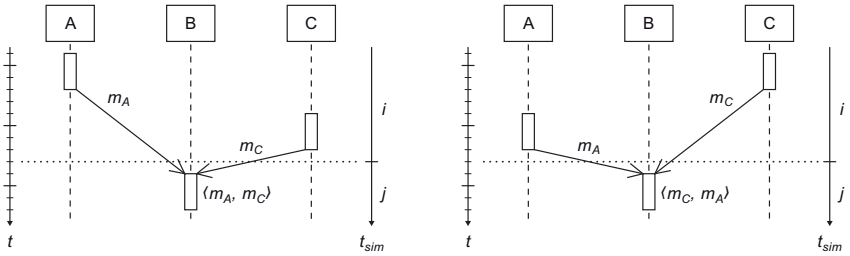

Figure 3: *Problem of Inbox Queue Order Depending on Scheduling*

$$m_i \leq_M m_j := \text{received}(m_i) = \text{received}(m_j) \wedge \text{o}(m_i) \leq \text{o}(m_j) \vee \text{received}(m_i) < \text{received}(m_j)$$

In the above formula $o(m)$ is completely generic because an appropriate ordering criterion may depend on the domain under consideration. A general attribute of messages that can be applied is the unique agent identifier of its sender. Assumed that every agent receives at most one message from each sender per time, this criterion is already sufficiently distinctive. Otherwise, additional properties (like the message content) have to be added. While this guarantees reproducibility, ordering messages this way might potentially corrupt simulation results. This has to be prevented by choosing appropriate attributes for ordering (Fujimoto 2000, pp. 84–86).

5 Implementation

The quality criteria identified have been implemented within the PlaSMA multi-agent-based simulation system (Gehrke and Ober-Blöbaum 2007) using conservative synchronisation with tree barriers. PlaSMA is based on the popular JADE agent platform that is in compliance with FIPA agent standards. PlaSMA provides a simulation control middleware for JADE that handles experiment initialisation, time management, as well as agent lifecycle management. Simulation control primarily consists of two kinds of instances: one top-level controller and a sub-controller for each processor or computer in distributed simulation settings. Each sub-controller locally handles the commitments of its respective agents concerning wake-up timestamps and transmits the minimal commitment to the top-level controller. In return, the top-level controller sends time events to the sub-controllers based on the commitments it received. Additionally, the internal message handling was adapted in order to guarantee adequacy, causality, and reproducibility of simulation. Time progression and correct message delivery is conducted transparently by the implementation. The user is not burdened with these simulation-specific issues.

PlaSMA is mainly applied for simulations with logistics entities represented by software agents. It is used in order to compare and evaluate algorithms for autonomous logistics planning and special sub-processes within this domain, e.g., coordination mechanisms (Schuldt and Werner 2007) as well as information distribution and routing algorithms (Gehrke and Wojtusiak 2008). Nevertheless, it is also applicable for other domains and thus not limited to logistic scenarios. Furthermore, PlaSMA is also part of a demonstration platform integrating real-world hardware in perishable food transport scenarios (Jedermann et al. 2007).

6 Conclusion

Multiagent systems are an adequate means to control complex logistics processes. Problem complexity is reduced by delegating decision-making to software agents as local entities. In general, the outcome of logistics strategies cannot be predicted analytically due to the underlying complexity and dynamics. Instead, multiagent-based simulation, which combines the agent programming paradigm with simulation, can be applied in order to evaluate such strategies. However, existing agent frameworks generally do not consider simulation-specific issues. Therefore, it is necessary to introduce a simulation middleware that implements synchronisation.

In this paper conservative synchronisation is applied due to domain-specific performance considerations (i.e., agents with extensive and dynamic knowledge bases). The paper identifies quality criteria to be considered on the message exchange layer as the central medium for agent interaction. Namely, these criteria are time model adequacy, causality, and reproducibility. The paper presents how to guarantee them by defining particular rules for message transfer and processing.

Additional quality criteria for synchronisation in multiagent-based simulation are performance and usability. Runtime performance of conservative synchronisation for MABS strongly depends on the agents employed and their tasks. Recent results indicate that even simulation models with more than 2,000 agents can be executed in reasonable time on a dual core computer (Schuldt and Werner 2007). The approach presented in this paper delegates most tasks to the simulation middleware, thereby disburdening the agent programmer. Nevertheless, agents (and thus their programmers) must still explicitly commit until when they will not send further messages. An open research question concerns how to determine this commitment automatically. The ultimate objective in this context is to arrive at a uniform agent design for operation and simulation.

7 Acknowledgement

The authors should like to thank Tobias Warden for his valuable remarks that helped improve the paper. This research is funded by the German Research Foundation (DFG) within the Collaborative Research Centre 637 “Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations” (SFB 637) at the University of Bremen, Germany.

References

Bellifemine, F.; Caire, G.; Greenwood, D. (2007) Developing multi-agent systems with JADE. John Wiley & Sons, Chichester, UK

Davidsson, P. (2000) Multi agent based simulation: Beyond social simulation. In MABS 2000. Springer-Verlag, Boston, MA, USA, pp. 97–107

Davidsson, P.; Holmgren, J.; Kyhlbäck, H.; Mengistu, D.; Persson, M. (2007) Applications of agent based simulation. In MABS 2006. Springer, Hakodate, Japan, pp. 15–27

Fujimoto, R. M. (2000) Parallel and distributed simulation systems. John Wiley & Sons, New York, NY, USA

Gehrke, J. D.; Ober-Blöbaum, C. (2007). Multiagent-based logistics simulation with PlaSMA. In GI 2007. Gesellschaft für Informatik, Bremen, Germany, pp. 416–419

Gehrke, J. D.; Wojtusiak, J. (2008) Traffic prediction for agent route planning. In ICCS 2008. Springer, Kraków, Poland, pp. 692–701

Herrler, R.; Klügl, F. (2006) Simulation. In Kirn, S.; Herzog, O.; Lockemann, P.; Spaniol, O. (eds.): Multiagent engineering: Theory and applications in enterprises. Springer, Heidelberg, Germany

Jedermann, R.; Gehrke, J. D.; Becker, M.; Behrens, C.; Morales Kluge, E.; Herzog, O.; Lang, W. (2007). Transport scenario for the intelligent container. In Hülsmann, M.; Windt, K. (eds.): Understanding autonomous cooperation and control in logistics. Springer, Heidelberg, Germany

Jefferson, D. R. (1985) Virtual time. ACM Transactions on Programming Languages and Systems 7 (1985) 3, pp. 404–425

Jennings, N. R. (2001) An agent-based approach for building complex software systems. Communications of the ACM 44 (2001) 4, pp. 35–41

Lees, M.; Logan, B.; Minson, R.; Oguara, T.; Theodoropoulos, G. (2005) Distributed simulation of MAS. In MABS 2004. Springer, New York, NY, USA, pp. 25–36

Parunak, H. V. D.; Savit, R.; Riolo, R. L. (1998). Agent-based modeling vs. equation-based modeling: A case study and users' guide. In MABS 1998. Springer, Paris, France, pp. 10–25

Pawlaszczyk, D.; Timm, I. J. (2006). A hybrid time management approach to agent-based simulation. In KI 2006. Springer, Bremen, Germany, pp. 374–388

Schuldt, A.; Werner, S. (2007) Towards autonomous logistics: Conceptual, spatial, and temporal criteria for container cooperation. In LDIC 2007. Springer, Bremen, Germany, pp. 313–320

Weiss, G. (1999) Multiagent systems. A modern approach to distributed artificial intelligence. MIT Press, Cambridge, MA, USA

Wenzel, S.; Weiß, M.; Collisi-Böhmer, S.; Pitsch, H.; Rose, O. (2008) Qualitätskriterien für die Simulation in Produktion und Logistik. Springer, Heidelberg, Germany