2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Designing a Simulation Middleware for FIPA Multiagent Systems

Arne Schuldt, Jan D. Gehrke, and Sven Werner
Centre for Computing Technologies (TZI)
University of Bremen, Am Fallturm 1, D-28359 Bremen

Abstract

Multiagent systems ease the implementation of software systems to control complex processes. Instead of monolithic programs, decision-making is delegated to software agents as local entities. Like in software development in general, testing and evaluation play an important role also for multiagent systems. Particularly, because run-time interactions between agents and their effects cannot always be predicted at design time. Multiagent-based simulation is an adequate means to evaluate agents regarding their applicability in real-world operation. However, general agent development frameworks do not consider simulation-specific issues. Because they provide no means for synchronisation, an additional simulation middleware is required. Temporal criteria that are relevant for middleware design are defined in this paper. Furthermore, the actual implementation and example applications in logistics are presented.

1. Introduction

Testing and evaluation play an important role in the software development process. This particularly holds for multiagent systems as run-time interactions between agents and their effects cannot always be predicted at design time [7]. However, it is generally not desirable to test software systems in their actual deployment [5]. Firstly, it is quite expensive and time-consuming to test software in its real environment. Secondly, testing might compromise the integrity of actual processes which again leads to high costs. Simulation is a common alternative to avoid these problems. Multiagent-based simulation (MABS) is particularly appealing as it applies the concept of multiagent systems to simulation [5]. Agents and their behaviour can be easily transferred [9] which makes MABS a promising approach to examine them with minimal effort.

However, controlling processes in reality and simulating them exhibit a major difference. In the real world time progression is an innate feature of the environment. This does not hold for simulation because simulation time does not progress implicitly. Additionally, simulated time progression may diverge for different agents since it depends on the computational power demanded and agents are executed independently in parallel. This need for synchronisation is addressed by simulation systems. MABS systems that are designed for social simulation intend to generate new findings about reality. There is no necessity to transfer the actual agent implementation to a real-world software system. Therefore, such simulation systems implement synchronisation but do not pay attention to interoperability standards. By contrast, intercompany interaction in real world necessitates agreeing on a standard communication language. Widely spread standards have been issued by the IEEE Foundation for Intelligent Physical Agents (FIPA). Existing multiagent platforms, such as JADE [1], implement FIPA standards but do not provide means for synchronisation. Therewith, they are applicable for real-world process control but not for simulation.

A solution to this dilemma is to implement a simulation middleware for existing agent platforms. Compliance with FIPA standards is then assured by the agent platform; the simulation middleware ensures correct synchronisation. Middleware implementation requires certain quality criteria regarding an adequate and correct abstraction of time (Section 2). The contribution of this paper is twofold. Firstly, necessary temporal quality criteria are discussed and formally defined (Section 3). Secondly, these criteria are implemented in a simulation middleware for conservative synchronisation in MABS (Section 4). This middleware is currently applied in several projects to evaluate the influence of autonomous control in logistics (Section 5).

2. Multiagent-Based Simulation

Multiagent-based simulation can be categorised as distributed simulation with discrete time model using software agents as parallel logical processes. It combines simulation scalability and runtime acceleration with decision-making encapsulated in agents [9]. The agent paradigm eases simulation model development due to the natural mapping between real-world entities and their simulation counterparts.

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.202

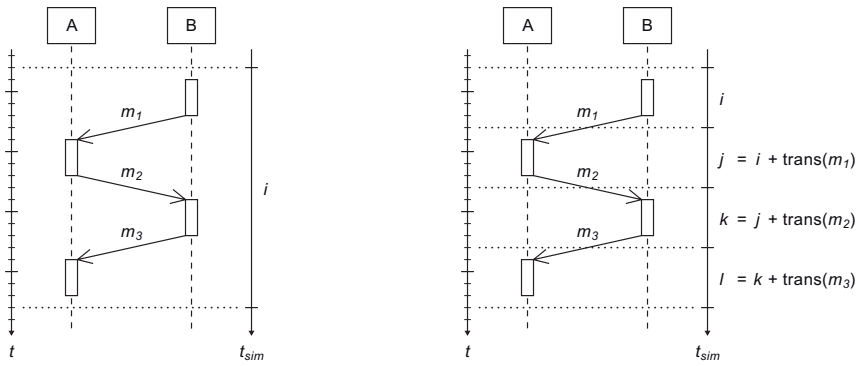

Figure 1. Agent communication without (left) and with (right) time consumption

In simulation different notions of time must be distinguished [2]. *Physical time* refers to the time at which simulated events would happen in real world. *Simulation time* models physical time within simulation. Simulation time progresses in a *discrete* way in MABS. 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 system in executing the simulation. In MABS, logical processes (LPs) are implemented as agents that usually run as operating system threads. Simulation time progression depends on the computational needs of each LP. Thus, each LP has its own *local virtual time* (LVT). As long as agents are independent from 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* [2]. In order to guarantee correct simulations events have to be processed in accordance with their time-stamp order.

Diverging local virtual times are addressed by synchronisation. *Optimistic* synchronisation generally does not restrict progression of LVT for agents. This allows an efficient execution of simulation because fast processes do not have to wait for slower ones. Whenever a straggler message is received the respective agent is reset to its past state at the LVT of the received message (cf. [6]). This, however, leads to potentially high requirements regarding space [2] because all preceding states of every agent must be stored at worst. The problem aggravates in simulations with high amounts of participating agents that possess complex knowledge. Space complexity may be reduced by *time windows* [8, 10]. 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. Time progression is therefore potentially slower. But space complexity is significantly lower because there is no need to store past agent states. Thus, we prefer conservative synchronisation for applications incorporating complex knowledge.

3. Temporal Quality Criteria for MABS

The multiagent-based simulation paradigm distinguishes from normal object-based simulation or distributed discrete event simulation. Agents cannot be manipulated directly by method invocation. Instead, they receive messages as simulation events from other agents and decide locally when and how to handle them. Messages may even be ignored. Thus, messages are no longer just simulation-specific representations of events that change state variables. Instead, they become an important part of the modelled domain because they represent the flow of information among agents. This characteristic of MABS implies 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*.

3.1. Time Model Adequacy

Time model adequacy denotes the challenge of discretising physical time to simulation time and mapping events to certain timestamps. Very fine-grained simulation time is likely to be harmful to simulation runtime performance. Thus, it is recommended to determine in advance the minimal granularity needed. 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.

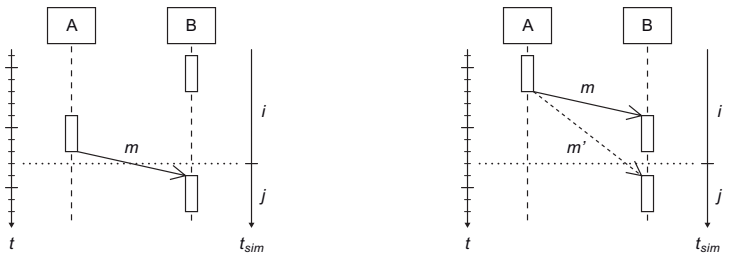

Figure 2. Controlling message visibility to ensure the causality constraint

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 $\mathrm{trans}(m)$ for a message m sent at simulation time $\mathrm{sent}(m)$:

Definition 1 (Message Timestamp) *The timestamp of message m results from the sum of its sending time and its transmission duration:*

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

Again, an appropriate value for $\mathrm{trans}(m)$ depends on the modelled domain. One might, for instance, look at expected Internet communication latencies for appropriate values.

The left-hand scenario in Fig. 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 consume physical time. The result of this communication process happens earlier than actually possible in real world, thereby possibly corrupting simulation results. As depicted right-hand in Fig. 1, all agent communication should take at least a minimum amount of simulation time:

Criterion 1 (Time Consumption) *Each message transmission must at least consume the minimal amount of time progression:*

$$\forall_m \mathrm{trans}(m) \geq \Delta t_{min} > 0$$

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

Criterion 2 (Modelling Accuracy) *The deviation of simulated transmission time and physical transmission time must not exceed the minimal amount of time progression:*

$$\forall_m \mathrm{trans}(m) - \mathrm{trans}'(m) < \Delta t_{min}$$

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

3.2. Causality

Time model adequacy demands message transmission to consume simulation time. It also impacts causality (also referred to as correctness), another temporal 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 also depends on scheduling whether B receives m at $t_{sim} = j$ or $t_{sim} = i$. In the first case, A is scheduled after B has finished execution at $t_{sim} = i$ (Fig. 2 left). 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 in which m is delivered to B at $t_{sim} = i$. A is scheduled before B (Fig. 2 right). 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.

In classical parallel discrete event simulation it does not pose a major problem if messages arrive early. Early messages are simply not processed until local virtual time arrives at the timestamp intended. In MABS, message handling is more complicated because agents act autonomously and have unrestricted access to their message inbox. Thus,

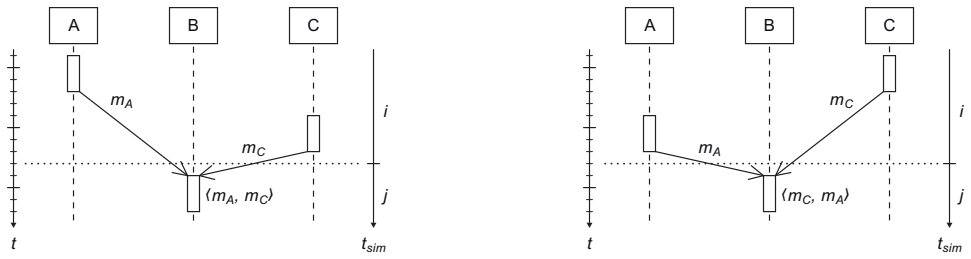

Figure 3. Reproducibility problem of inbox queue order depending on scheduling

simulation model developers would have to handle such messages explicitly in order to satisfy the causality constraint. But 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:

Criterion 3 (Causality) *Messages must not be perceivable to agents before their intended arrival time:*

$$\forall_m \forall_{t_{min}} \, \text{Perceivable}(m, t_{sim}) \rightarrow t_{sim} \geq \text{received}(m)$$

The right-hand side of Fig. 2 illustrates how the perception of message m is delayed to m'.

3.3. Reproducibility

Reproducibility (or repeatability) of simulation results given the same model and the same random seeds is another quality criterion. It does not affect simulation result accuracy. But it is important for traceability and analysis of occurring effects and possible modelling errors. Adequacy and causality constraints ensure that operating system scheduling does not influence at which time messages arrive. But this does not 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 at $t_{sim} = j$. But the inbox queue of B can be ordered in two different ways. 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). Because message ordering directly depends on operating system scheduling, results of simulation runs would not be reproducible. Thus, a consistent message ordering $\leq_M$ must be imposed.

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 time-stamp an additional ordering criterion $\text{o}(m)$ has to be considered:

Criterion 4 (Reproducibility) *Received messages m_i and m_j must be ordered by their arrival time and an additional distinctive ordering criterion:*

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

An appropriate ordering criterion depends on the domain under consideration. Thus, $\text{o}(m)$ is completely generic. A message attribute that can be applied is the unique sender identifier. Assumed that every agent receives at most one message from each sender per time, this criterion is sufficiently distinctive. Otherwise, additional properties (like the message content) have to be added. While this guarantees reproducibility, ordering messages this way potentially causes a bias in simulation results. This has to be prevented by appropriate attributes for ordering [2, pp. 84–86].

4. Implementation

Users should not be burdened with satisfying the temporal quality criteria themselves. The temporal criteria hold for all agents and should thus not be considered by each agent itself. In particular, implementing the same criteria individually for each agent is error-prone. The additional effort for testing agents from operation in simulation should be as small as possible. Therefore, the simulation middleware must implement the temporal quality criteria rather than each individual agent. The important question is to which extent the underlying platform has to be modified. Such modifications are most probably not completely avoidable. Nevertheless, it is generally desirable to make only as few modifications as possible. If the middleware directly changes the underlying source code, it will be necessary to modify also all future versions of the platform.

The temporal quality criteria identified in Section 3 have been implemented within the PlaSMA[1] multiagent-based simulation system [3] 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 middleware for JADE that handles experiment initialisation, time management including message passing, as well as agent lifecycle management. Simulation control primarily consists of two kinds of

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

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. Internal message handling has been adapted to guarantee adequacy, causality, and reproducibility of simulation.

5. Application

The economic importance of global logistics processes has increased in recent decades. Today, former linear supply chains have evolved into networks with complex interrelationships between their participants. Each supplier has many customers and vice versa. A great many natural and legal persons participate in keeping these globally distributed processes running. The challenges in controlling such processes are the complexity and the high degree of dynamics. Furthermore, the physical distribution prevents relevant information from being available centrally.

These issues make it virtually impossible, to effectively apply centralised control to supply networks. Autonomous logistics is a new paradigm that aims at making complex logistics processes controllable [12]. This approach delegates decision-making to local entities, e. g., to shipping containers which then plan and schedule their way through the logistics network themselves. This increases robustness because unexpected changes can be immediately handled on the local level. Autonomous logistics can be implemented by intelligent agents that act on behalf of the objects represented. The PlaSMA simulation system is mainly used to evaluate approaches for autonomous logistics, e. g., coordination mechanisms [11] and information distribution and routing algorithms [4]. But PlaSMA is also applicable for other domains and thus not limited to logistic scenarios.

6. Conclusion

In general, the behaviour of multiagent systems cannot be predicted analytically due to the underlying complexity and dynamics. Instead, multiagent-based simulation, which combines the agent paradigm with simulation, can be applied for evaluation. A simulation middleware must be introduced because existing agent platforms generally do not consider simulation-specific issues. Such a middleware must satisfy temporal quality criteria on the message exchange layer. These criteria consider time model adequacy, causality, and reproducibility. For this purpose, this paper defines particular rules for message transfer and processing. The approach presented in this paper already 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.

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

[1] F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. John Wiley & Sons, Chichester, UK, 2007.

[2] R. Fujimoto. *Parallel and Distributed Simulation Systems*. John Wiley & Sons, New York, NY, USA, 2000.

[3] J. D. Gehrke and C. Ober-Blöbaum. Multiagent-based Logistics Simulation with PlaSMA. In *GI 2007*, pages 416–419, Bremen, Germany, 2007.

[4] J. D. Gehrke and J. Wojtusiak. Traffic Prediction for Agent Route Planning. In *ICCS 2008*, pages 692–701, Kraków, Poland, 2008. Springer-Verlag.

[5] R. Herrler and F. Klügl. Simulation. In S. Kirn, O. Herzog, P. Lockemann, and O. Spaniol, editors, *Multiagent Engineering: Theory and Applications in Enterprises*, pages 575–596. Springer-Verlag, Heidelberg, Germany, 2006.

[6] D. R. Jefferson. Virtual Time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425, 1985.

[7] N. R. Jennings. An Agent-Based Approach for Building Complex Software Systems. *Comm. ACM*, 44(4):35–41, 2001.

[8] M. Lees, B. Logan, R. Minson, T. Oguara, and G. Theodoropoulos. Distributed Simulation of MAS. In *MABS 2004*, pages 25–36, New York, NY, USA, 2005. Springer-Verlag.

[9] H. V. D. Parunak, R. Savit, and R. L. Riolo. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users’ Guide. In *MABS 1998*, pages 10–25, Paris, France, 1998. Springer-Verlag.

[10] D. Pawlaszczyk and I. J. Timm. A Hybrid Time Management Approach to Agent-Based Simulation. In *KI 2006*, pages 374–388, Bremen, Germany, 2006. Springer-Verlag.

[11] A. Schuldt and S. Werner. Distributed Clustering of Autonomous Shipping Containers by Concept, Location, and Time. In *MATES 2007*, pages 121–132, Leipzig, Germany, 2007. Springer-Verlag.

[12] K. Windt and M. Hülsmann. Changing Paradigms in Logistics. In M. Hülsmann and K. Windt, editors, *Understanding Autonomous Cooperation and Control in Logistics*, pages 1–16. Springer-Verlag, Heidelberg, Germany, 2007.