

Simulation for Agent-Oriented Software Engineering

A.M.Uhrmacher
Department of Computer Science,
University of Rostock
D-18051 Rostock, Germany
e-mail: lin@informatik.uni-rostock.de

Keywords: multi-agent systems, software-engineering, simulation, grand challenge

Abstract

The complexity of agents, their environment, and the interaction between agents and environment suggest that experimental testing represents a major research effort in the area of multi-agent systems. However, the development of methods and tools supporting systematic experiments with agents has not found the expected attention. For testing agents, developers often resort to handcrafted solutions developed from scratch. The proposed Grand Challenge is aimed at developing modeling and simulation methods based on theoretical foundations that allow a systematic and comfortable testing of multi-agent systems and to firmly rooting simulation within the emerging area of agent-oriented software engineering. The endeavor provides plentifully technological and interdisciplinary challenges.

1 INTRODUCTION

Agents can be interpreted as software systems that are aimed at working autonomously in dynamic and uncertain environments [15]. The interrelations between agents and simulation are manifold [34]. Software agents are used to develop “state of the art” simulation systems on the one hand. On the other hand simulation provides a means for systematically analyzing the behavior of agents in dynamic virtual environments. The development of agents faces problems associated with traditional distributed concurrent systems. Additional difficulties arise from complex interactions between autonomous problem solving components [15, p.298]. The construction of agents reincarnates problems of embedded, real-time systems due to the fact that the environment of agents is typically dynamic, inaccessible, and non-deterministic. Agent-based systems are often mission critical (or even safety critical) and like other software systems, must be tested, and evaluated before being deployed. However, their autonomy and the open heterogeneous nature of the environment in which they operate make testing and evaluation more difficult than in the case of more conventional software systems. Often, the

existing formal notations are either too weak to express the structure and state of the agent and its environment, or if formalization is possible, the resulting formulations are intractable. As a result it can be difficult to verify their properties. So the only alternative is to test them empirically.

Thus, the design of multi-agent systems might become an application area for simulation tools similar to the design of manufacturing systems or network protocols. However, methods and tools are still missing that allow a comfortable experimenting with different agents and that support the development of agents as an experimental technique [38]. For testing, agent developers often resort to handcrafted solutions developed from scratch. Although the number of simulation studies which analyze the temporal behavior of multi-agent systems in dynamic environments is steadily increasing, the number of simulation systems are few; among them hardly any exploit state of the art modeling and simulation technologies.

A gap exists between the current state of the art of simulation methods and the current “*modi operandi*” in designing agents. A closer cooperation between software engineers, agent developers, and simulationists are called for. Testing of agents requires not only to bring best practice of modeling and simulation to the attendance of agent developers but also to tailor methods and technologies to the specific needs of this challenging application domain. Simulation being an inherent phase in designing agents seems still far away and the path towards this goal provides many challenges.

2 CHALLENGES EN ROUTE

The challenges of establishing modeling and simulation within the software development cycle are twofold. Technical challenges rise naturally on the path. Overcoming those is necessary but not sufficient to reach the final goal. Interdisciplinary work is not a purely technical endeavor, but closely related to the perception and acceptance of one discipline by the other. Thus, the proposed Grand Challenge reaches deep into current discussions about and efforts towards establishing modeling and simulation as a separate discipline. Addressing the later, as important as it might be,

in its entirety cannot be the purpose of this proposal and only some preliminary steps will be suggested to facilitate the process.

	1	2	3	4	5	6	7
1							
2							
3							
4							
5							
6							
7							

Figure 1. The TILEWORLD scenario in JAMES. Diamonds tiles, black circles denote holes which are attributed with a type, a depth, and with a score to be gained by filling it with the correct and incorrect type of tiles (6/2). The grey grids are obstacles

2.1 Flexibility and “Easy to Use”

Test environments for agents shall support a testing of agents in the small and in the large [13]. Testing in the small, e.g. the TILEWORLD scenario (Figure 1), is used to reveal basic limits, problems, and strengths of the proposed agent architecture, which again might re-surface in real world settings of a specific type. A number of test scenarios for testing in the small have been developed. Early test-beds supported only a specific type of test scenarios. Many of them have been inspired by the perception of agents as robots living in a two dimensional space through which agents move continuously or which, even more often, has been discretized into a grid world, e.g. MICE [21]. To support test scenarios of a different type, e.g. softbots moving through networks, more general simulation tools are required that support different application domains likewise. Providing different component libraries for different domains is one step towards the solution. A support of continuous as well as discrete simulation and an easy extensibility at the simulation and experimental level ask for a layered and component-based approach, e.g. [27, 31].

Testing in the large implies to simulate the real environment the agent is supposed to work in. Like

scenarios that are used for a testing in the small, scenarios for a testing in the large vary with the type of agents envisioned. Thus, problems and accordingly probable solutions appear to be similar in testing in the small and testing in the large. However, unlike in the small world scenario, the validity of environmental models becomes an issue in testing in the large. As valid models are very expensive to develop, existing models should easily be integrated into the simulation environment. Different ways to integrate pre-existing models are conceivable.

Assuming that the simulation system supports multi-formalism models [36] interfaces to modeling languages of the different simulation systems have to be defined. The integration not on modeling but on simulation level leads to the inter-operation of simulation systems re-using efforts of the High Level Architecture (HLA) initiative [6]. However, the HLA framework provides merely a syntactic framework for inter-operation. An automatic inter-operation between simulation systems requires that simulation systems “understand” the information they exchange, e.g. at the moment different network simulators shall automatically exchange packages, a consensus about protocols, their identifiers, and the transformation between them has to be reached [30]. In general, to support an automatic interoperation, ontologies of the exchanged vocabulary and the implied constraints have to be specified thus reviving the ontology problem of AI [12] in the area of modeling and simulation [20]. These efforts have to be combined with the development of coordination languages [7], which have to be tailored to the needs of agent-oriented simulation. This might lead to the development of yet another “neutral model solver protocol” [1], like e.g. Modelica [9].

Thus, we envision modeling and simulation methods and tools which support a flexible and comfortable composition of test environments for multi-agent systems by re-using models of different languages and by an interoperation with other simulation systems that takes into consideration the semantics of the exchanged information.

2.2 Dynamic Structure

Agents are characterized by interaction pattern or spheres of influence that vary over time. The ability to adapt their own interaction, composition and behavior pattern challenges the expressiveness of formalisms, and efficiency of tools [33] likewise. To “move the modeling and simulation process towards science” [23, p.27] a theoretical foundation is required. However, whereas the number of simulation systems that allow the creation and deletion of model components and interactions has been steadily increasing, work on corresponding formalisms has not been keeping pace. Among the formal approaches to discrete event simulation the DEVS formalism has been analyzed more thoroughly with respect to possibilities and implications of

integrating and expressing variable structures [4, 33] than others. As DEVS represents only one of the formal approaches to modeling and simulation, efforts have to be directed to explore and relate possibilities and implications in the context of other formalisms. E.g. the potential of mobile Petri-nets [3] or so called reference nets [18] have to be explored. Relating these efforts across different formalisms not only represents a detour into the world of multi-formalism modeling [36] but will also likely intensify the exchange between the simulation community and other Computer Science communities that have been developing and adapting their formalisms to multi-agent modeling, specification, and analysis within the last decade, e.g. [28].

2.3 Efficient Execution

Many test beds for multi-agent systems do not execute their models concurrently. They simply maintain the illusion of simultaneity on a single machine. If only a single deliberative agent is tested in a dynamic environment or the coordination strategies of a moderate number of reactive agents are tested, there is no need for a distributed, parallel execution of agents. However, to efficiently test multiple deliberative agents, each of which consumes significant storage and computation resources, a concurrent, distributed simulation is desirable.

Since the execution time of deliberative components can hardly be foretold, conservative techniques that are based on lookahead [10] are not easily applicable. In addition, multi-agent systems are characterized by dynamic patterns of composition and interaction. The costs introduced by saving states of the model, including information about added and deleted models and couplings, moving agents etc. makes state savings expensive and asks for storage saving strategies [10]. In addition, logical processes that are executing an agent or part of an agent might consume very different amount of wallclock time to advance in simulation time. A danger exists that most models of a simulation have run “optimistically” to completion, before one agent has decided on its action yet. Thus, unbounded optimistic strategies are not generally applicable, either. The use of optimistic strategies causes even more severe problems if entire agents shall be plugged into the environment. To rollback internal state changes, the internal state has to be accessible, in addition to mechanisms to suspend and resume the agent's threads. In case an agent accesses external sources, e.g. databases, rollbacks seem no longer feasible.

Often a pure discrete simulation does not suffice, and continuous and discrete simulation strategies are combined, e.g. to simulate robots. As agents move continuously through space so do their spheres of interest [19] and thus the regions they can and want to access. Strategies for a dynamic load balancing and for a dynamic re-distribution of

logical processes have to be developed to adapt the execution to the current needs.

Thus, methods have to be developed that support the efficient execution of combined, continuous, distributed models that exhibit dynamical interaction and composition structures. As no single strategy will yield the optimal solution independently of the concrete test scenario, flexible simulation tools are required that allow to replace and refine execution methods on demand.

2.4 Models and the “Real Thing”

The implementation and application of dynamic test scenarios for multi-agent systems require considerable modeling efforts. In most cases, the virtual environment, the interaction between agent and virtual environment, and the agent have to be modeled. However, different strategies exist to lighten the load of modeling in experimenting with agents.

Typically, the agent is not modeled in its entirety. Already the first simulation systems for agents allowed to plug code fragments, or single modules into the modeled agent skeleton [21]. A further step towards reducing the modeling effort treats agents as external source and drain of events [26, 2]. The simulation is interpreted as a black box with a clearly specified interface. The agent software has to be changed to transform and to redirect requests and messages that are normally sent to its real environment to the simulation system. If the agent is synchronized with the simulation system in simulation time, messages are labelled with a time-stamp.

Other test beds for multi-agent systems let simulation and agents interact with each other asynchronously in wall-clock time, e.g. the soccer simulator employed in the RoboCup initiative [22]. The simulator of the environment checks frequently whether agents produced an event that the simulation engine has to take into account, otherwise the simulator proceeds with its own calculation. Agents are put under time pressure or are released of it by slowing down the execution of the simulation engine. The purpose of the simulator is to support competitions rather than a thorough testing which requires more control of the experimental frame. Recently the introduced biases have been analyzed to improve the synchronization between simulator and agents [5].

If agents and simulation system are loosely coupled it saves the user the extra effort to specify the agent in the modeling language of the simulation system. However, they require typically more effort in analyzing the interaction and actions of agents in the virtual world, as agents are not explicitly represented in the test environment and their behavior can only be analyzed based on their observable behavior within the virtual environment.

One would like to have both. On the one hand the possibility should be given to execute agents as they are and to switch arbitrarily between an execution in the real environment and the virtual test environment [35, 31]. On the other hand, agents should be an integral part of the



Figure 2. 3D view of the RoboCup rescue simulator (<http://www.robocup.org/>)

experimental setting and as such perceivable and controllable. In analyzing the behavior of agents, models focus the view on relevant aspects and changes within agents. Thus, tools should support a graceful transformation from simulation to emulation.

2.5 Schedule

Work to date in agent development has largely ignored recent developments in simulation methodology, and has instead tended to employ various ad-hoc approaches to simulation. Jennings and Wooldridge conclude, “systematic testing is the least developed area in developing multi-agent systems” [16]. Simulation seems often to be perceived as an area of straightforward techniques and handcrafted solutions rather than of multifaceted scientific research. For this to change a close interaction with agent developers and computer scientists working on software engineering methods for agents has to be installed.

In cooperation with agent developers test scenarios shall be selected. There should be no shortness of possible candidates. One group of agent developers, which is concerned with the definition of suitable test scenarios, is the international research initiative RoboCup. The soccer game is one scenario used to promote research on cooperation between autonomous agents and robots in dynamic multi-agent environments. A simulation league complements the robot league in the many national and international contests each year. Maybe inspired by the

virtual island PACIFICA [29, 11] in which the generation and execution of plans for non-combatant evacuation operations have been tested, the RoboCup initiative has defined a new challenge: the RoboCup Rescue challenge. Its simulator of a large scale urban disaster currently embraces four modules, i.e. for simulating collapsing buildings, blockages in roads, the spreading of fire, and the flow of traffic (Figure 2).

Workshops and conferences not only on simulation but particularly on agents and software engineering offer suitable platforms to present and test developed concepts and to exchange ideas.

Different working groups that address selected topics of the challenge should be established during the next year and define milestones for the next three years. Even if one system in which all developed modules are integrated seems hardly realistic a framework shall be specified, developed, and maintained to synthesize the achieved results.

To assess the success of the challenge in the context of large scale testing requires concrete time critical application projects of agents and a close cooperation between simulationists, agent developers, and software engineers. Both are currently rather rare. Whereas the latter condition, i.e. cooperation, is explicitly part of the challenge process, the challenge should implicitly help in bringing about the former condition, i.e. a sufficient number of projects where agents are applied to concrete problems, e.g. mobile networks [14]. Here, simulation will provide the means to evaluate the potential of both the technology and the paradigm and thus to overcome the dearth of quantitative results [24].

3 RELATED EFFORTS

The importance of empirical testing in developing software systems cannot be sufficiently stressed. Unlike many other scientific disciplines systematic testing seems surprisingly unexplored in computer science even though testing plays an obvious and central role when selecting and assessing theories, methods, and tools, and gaining new insights [32]. The more complex programs become the more often they are objects of empirical studies “because it is no longer true that we can predict how a system will behave by looking at its code” as Paul Cohen observes in his preface to *Empirical Methods for Artificial Intelligence* [8, p.2]. If time critical embedded systems shall be tested, it seems natural to resume to simulation techniques and methods. Thereby, the spectrum stretches from specification-based testing which analyzes a model of the software in a simulated dynamic environment to testing the real code in an emulated environment. Agent-oriented Software Engineering constitutes a young but growing branch of Software Engineering [37] and research efforts are directed to structure and support an effective development of agent systems. Decomposition, abstraction, and organization are

the basic strategies in designing multi-agent systems, as Jennings and Wooldridge observe [38, 17]. That “agent systems work largely by emergent behavior and handle errors gracefully” [25] is another reason, which asks for simulation as an evaluation method employed during the agent development cycle

4 CONCLUSION

The more software systems are needed that work autonomously in open dynamic environments, the more important a systematic experimenting will become. Thus, the proposed Grand Challenge will contribute to the development of agent-oriented software systems. In addition, the Grand Challenge, application-driven and interdisciplinary by nature, will reveal inadequacies in available modeling and simulation theories and tools. This will lead to consolidating and pushing research efforts in very different simulation areas.

The goal is to develop modeling and simulation methods based on theoretical foundations that allow a systematic, comfortable, efficient and effective testing of agents and to firmly root simulation within agent-oriented software engineering. In the next 8 years to come simulation methods should play an as important and accepted role in designing agents as they do today in designing manufacturing systems. The challenges en route refer to technological areas as diverse as multi-formalism modeling, variable structure models, interoperation of simulation systems, simulation and emulation, and parallel and distributed simulation. Not to disregard are those that arise from the interdisciplinary nature of the undertaking.

REFERENCES

- [1] ESPRIT Basic Research Working Group 8467 - Simulation for the Future: New Concepts, Tools and Applications. <http://hobbes.rug.ac.be/SiE/>.
- [2] S.D. Anderson. Simulation of Multiple Time-Pressured Agents. In *Proc. of the Wintersimulation Conference, WSC'97*, Atlanta, 1997.
- [3] A. Asperti and N. Busi. Mobile Petri Nets. Technical Report UBLCS-96-10, University of Bologna, 1996.
- [4] F.J. Barros. Modeling Formalism for Dynamic Structure Systems. *ACM Transactions on Modeling and Computer Simulation*, 7(4):501--514, 1997.
- [5] M. Butler, M. Prokopenko, and T. Howard. Flexible synchronisation within robocup environment: A comparative analysis. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup 2000: Robot Soccer: World Cup IV*, number 2019 in LNAI, pages 119--128, London, 2001. Springer.
- [6] C.D. Carothers, R.M. Fujimoto, R.M. Weatherly, and A.L. Wilson. Design and Implementation of the HLA Time Management in the RTI Version F.0. In *Proc. of the Winter Simulation Conference*, 1997.
- [7] P. Ciancarini and A.L. Wolf, editors. *3th International Conference on Coordination Languages and Models: Coordination'99*, number 1594 in LNCS, London, 1999. Springer.
- [8] P. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.
- [9] H. Elmquist and S.E. Mattson. Modelica - The Next Generation Modeling Language - An International Design Effort. In *First World Congress of System Simulation*, Singapore, 1997.
- [10] R.M. Fujimoto. *Parallel and Distributed Simulation Systems*. John Wiley and Sons, 2000.
- [11] Y. Gil, M Hoffman, and A. Tate. Domain Specific Criteria to Direct and Evaluate Planning Systems. Technical Report ISI-93-365, Information Sciences Institute, University of Southern California, Marina del Rey, CA, 1994.
- [12] T.R. Gruber. A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, 5(2):199--220, 1993.
- [13] S. Hanks, M.E. Pollack, and P.R. Cohen. Benchmarks, Test Beds, Controlled Experimentation and the Design of Agent Architectures. *AAAI*, (Winter):17--42, 1993.
- [14] A.L.G. Hazyzelden, J. Bigham, M. Wooldridge, and L.G. Cuthbert. Future communication networks using software agents. In A.L.G. Hayzelden and J. Bigham, editors, *Software Agents for Future Communication Systems*, chapter 1. Springer-Verlag, 1999. ISBN: 3-540-65578-6.
- [15] N. R. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1):275--306, 1998.
- [16] N.R. Jennings and M. Wooldridge. Applications of Intelligent Agents. In N.R. Jennings and M. Wooldridge, editors, *Agent Technology : Foundations, Applications, and Markets*. Springer, 1998.
- [17] N.R. Jennings and M. Wooldridge. Agent-Oriented Software Engineering. In J. Bradshaw, editor, *Handbook of Agent Technology*. AAAI/MIT Press, to appear.
- [18] M. Köhler, D. Moldt, and H. Rölke. Modelling the structure and behavior of petri net agents. In J.M. Colom and M. Koutny, editors, *Applications and Theory of Petri Nets 2001*, number 2075 in LNCS, pages 224--241, London, 2001, Springer.

- [19] B. Logan and G. Theodoropolous. The Distributed Simulation of Multi-Agent Systems. *Proceedings of the IEEE*, 89(2):174--185, 2001.
- [20] R. Meersman. New Frontiers in Modeling Technolgy: The Promise of Ontologies. In E.J.H. Kerckhoffs and M. Snorek, editors, *Proc. European Multi-Simulation Conference*, pages 3--6, San Diego, 2001. SCS.
- [21] T. Montgomery and E. Durfee. Using MICE to Study Intelligent Dynamic Coordination. In *Conference on Tools for Artificial Intelligence*, pages 438--444, Washington, DC, 1990. IEEE.
- [22] I. Noda. Soccer Server: A simulator for Robo Cup. In *JSAI AI-Symposium 95: Special Session on RoboCup*, 1995.
- [23] E.H. Page. *Simulation Modeling Methodology: Principles and Etiology of Decision Support*. PhD thesis, Virgina Polytechnic Institute and State University, 1994.
- [24] T. Papaioannou. *On the Structuring of Distributed Systems: The Argument for Mobility*. PhD thesis, Loughbourough University, 2000.
- [25] C. Petrie. Agent-oriented software engineering. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, number 1957 in LNCS, pages 59--76, London, 2001. Springer.
- [26] M.E. Pollack. Planning in Dynamic Environments: The DIPART System. In A. Tate, editor, *Advanced Planning Technology*. AAAI, 1996.
- [27] H. Praehofer, J. Sametingar, and A. Stritzinger. Discrete Event Simulation Using the JavaBeans Component Model. In A.G. Bruzzone, A. Uhrmacher, and E.H. Page, editors, *1999 International Conference on Web-Based Modeling and Simulation*, volume 31, pages 107--112, 1999.
- [28] J.L. Rash, C.A. Rouff, W. Truszkowski, D. Gordon, and M.G. Hinchey, editors. *Formal Approaches to Agent-Based Systems*, volume 1871 of LNAI, London, 2001. Springer.
- [29] G.A. Reece, A. Tate, D.I. Brown, and M. Hoffman. The PRECiS Environment. Technical Report ARPA-RL/CPE, Artificial Intelligence Applications Institute, University of Edinburgh, Scotland, 1993.
- [30] G. Riley, M. Ammar, R. Fujimoto, K. Perumalla, and D. Xu. Distributed network simulations using the dynamic simulation backplane. In *Proceedings of the International Conference on Distributed Computing Systems - ICDCS-2001*, 2001.
- [31] H. Sarjoughian, B.P. Zeigler, and S.B. Hall. A Layered Modeling and Simulation Architecture for Agent-Based System Development. *Proceedings of the IEEE*, 89(2):201--213, 2001.
- [32] W.F. Tichy. Should computer scientists experiment more? 16 reasons to avoid experimentation. *IEEE Computer*, 31(5):32--40, 1998.
- [33] A.M. Uhrmacher. Dynamic Structures in Modeling and Simulation - a Reflective Approach. *ACM Transactions on Modeling and Simulation*, to appear.
- [34] A.M. Uhrmacher, P.A. Fishwick, and B.P. Zeigler, editors. *Special Issue: Agents and Simulation: Exploiting the Metaphor*, volume 89 of *Proceedings of the IEEE*, 2001.
- [35] A.M. Uhrmacher and B. Kullick. Plug and Test Software Agents in Virtual Environments. In *Winter Simulation Conference - WSC'2000*, Orlando, FL, December 2000.
- [36] H.L. Vangheluwe and G.C. Vansteenkiste. A Multi-Paradigm Modelling and Simulation Methodology: Formalisms and Languages. Technical report, ESPRIT Basic Research Working Group 8467, Working Paper, 1996.
- [37] M. Wooldridge and P. Ciancarini. Agent-oriented software engineering: The state of the art. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, number 1957 in LNCS, pages 1--28, London, 2001. Springer.
- [38] M. Wooldridge and N.R. Jennings. Software Engineering with Agents: Pitfalls and Prاتفalls. *IEEE Internet Computing*, May/June 1999.