Preface

Metaphor plays a key role in Computer Science and Engineering. New ideas and methods are infused into existing areas, creating fresh material and methodologies. An ”agent” carries out our purpose in performing an act on our behalf. The achievement of this purpose may involve differing degrees of autonomy. If we take the everyday meaning of ”agent” and marry this concept with software development, we derive the ”software agent.”

Software agents address the demand for programs that inter-operate to solve problems in an open and dynamic environment. As the number and complexity of agent-oriented programs increases, so does the need for software engineering tools and simulation systems that support their design and evaluation. New research questions whether agent-oriented techniques hold part of the answer to some urgent problems in engineering simulation systems, such as how to facilitate reuse and exchange of models and services between simulation systems.

Simulation of Multi-Agent Systems

Agent-based systems are often safety critical, and like other software systems, must be tested and evaluated before being deployed. Agents are embedded systems, and their dynamic behavior determines their efficiency and effectiveness. Therefore, simulation is an intrinsic part both during development and for testing purposes, to learn more about their behavior or investigate the implications of alternative architectures and coordination strategies.

However, work to date has largely ignored recent work in simulation methodology and systems and has instead tended to employ various ad-hoc approaches to simulation. The model of the environment the agent shall be tested in and the simulation mechanisms are typically developed and implemented from scratch. In setting up the test environment, modeling and the execution of the model are not distinguished. This hampers a reuse of test scenarios, reproducing the results of experiments, and comparison of results achieved through experimentation.

The requirements of a simulation system which has a chance to be applied by working groups that design agents has to meet a variety of requirements. Its model design should ideally support:

- a compositional, hierarchical model design
- an integration of diverse agents and agent architectures

- a comfortable interface to plug in agents' modules
- a dynamic adaptation of the model's composition and coupling structure
- a combination of continuous and discrete models

Model execution should be clearly separated from model design. Simulation techniques are required which combine a flexible model design (see above) with an efficient execution. The computational requirements of simulations of agent-based systems exceed the capabilities of single platforms. Each agent requires typically considerable computational resources, and many agents may be required to investigate the behavior of the system as a whole. Distributed, concurrent simulation techniques have to take into account that:

- to determine a lookahead in the domain of deliberative multi-agent systems is very difficult since during test runs the time needed for deliberation often varies drastically.
- rollbacks are even more storage expensive due to the flexible structures which require not only the storage of states but of entire models
- a distributed execution necessitates dynamic load balancing

Simulation systems for multi-agent systems should also be able to interact with other simulation systems. This leads us to the second focus of the discussions, i.e. employing agents to execute models and to implement flexible, state of the art simulation systems.

Software Agents for Distributed Modeling and Simulation

Distributed modeling and simulation imply a geographically distributed set of models and their components, as well as concurrent execution of model-derived code. In distributing model components, we must design model and component repositories over the web. It should be possible to search these repositories for reusable objects. Although existing mechanisms do not yet exist for distributed model repositories, certain technologies such as object oriented databases and XML (Extensible Markup Language) will help in creating appropriate vehicles for model and component representations. Distributing the simulation (i.e., model execution) is another matter and has been more widely studied in the distributed simulation research community. For both model design and execution, agent-oriented approaches create a natural fit with the problems of distributed modeling and execution.

Simulation Systems Executed by a Community of Agents

The execution of a model can be realized by a community of distributed, concurrently interacting, and moving entities. Their interaction, composition and location structure adapts itself to improve the efficiency of the simulation run. Different parts of the model can be executed by different agents specialized in different formalism, e.g. continuous and discrete models. To balance the work load processes migrate from one site to another during simulation. The flexibility of these approaches promise a scalability, which is necessary to deal with heterogeneous, large-scale applications.

Simulation Systems as a Community of Heterogeneous Agents

Simulation systems can be developed as a community of heterogeneous agents. These approaches can be rooted back to research of the late 80ties where knowledge based systems and simulation systems were combined, e.g. to select test data, evaluate and display simulation results. A multi-agent design emphasizes modularity, flexibility, and concurrency in constructing intelligent simulation systems where modules, e.g. so called intelligent "front-ends" or "back-ends", work as autonomous agents. Depending on the functionality, e.g. if searching for specific data on the web, a module's performance might even benefit from the mobility of an agent.

Simulation Systems as Agents

Due to the diversity and multitude of simulation systems and existing simulation models, the need for standardization and an improved interoperability have been recognized as pressing problems in this area. To facilitate the exchange and reuse of models, and services between simulation systems, recent research suggests exploration of agent-oriented techniques, including knowledge interchange languages and protocols for interaction and negotiating.

The entire simulation system becomes an agent. If different simulation systems shall interoperate, as the DoD initiative HLA requires, ensuring that simulation systems "understand" each other becomes crucial. Having the same semantic refers to the objects, i.e. variables, which are exchanged between simulations, but also to the temporal horizon of the simulation systems.

Conclusion

The agent metaphor very nicely supports the development of state of the art simulation systems, since it complements the main stream of current simulation research. However, agents do not solve problems by themselves. Interfaces, semantic frames, and collaboration strategies have to be defined and negotiated. Distributed model design and model execution techniques are found to be not only supportive, but also necessary for the systematic design and testing of multi-agent systems.

Dagstuhl

We began the week realizing that this was no ordinary workshop. We felt obliged to pay close attention to the interaction of all participants so as not to dictate a rigid schedule of papers and panel sessions, as would be done in a larger conference setting. Thus, all participants played an active role in driving the progress and content of the workshop.

Schloss Dagstuhl was a very pleasurable environment. We were able to combine some of our discussions with drinks in the wine cellar, walks in the woods, and trips up to the ruins on the hill. It is the environment that makes Dagstuhl an amazing location and environment. There is an unusual blend of the old with the new, and the concept of careful time management interspersed with a flexibility to deviate and collectively self-organize. The staff was most gracious and helpful. We thank them for all of their help in accommodating our needs during our stay.

Adelinde Uhrmacher
Paul Fishwick
Bernard Zeigler

Schedule of the Seminar

Monday, July 5 1999

Presentations

Welcome / Introduction *(P.Fishwick, A.Uhrmacher, B.Zeigler)*
An Introduction to HLA, DEVS/HLA and Agent Endomorphic Models *(B. Zeigler)*
The Distributed Simulation of Multi-Agent Systems *(B. Logan)*
JAMES - A Java-Based Agent Modeling Environment for Simulation *(A.Uhrmacher)*
ISSAC: An Intelligent System for Exploiting Speculative
Execution and Active Code in Large-Scale Distributed Simulations *(C. Carothers)*

Working Groups

Simulation for Multiagent System Design
Software Agents for Distributed Simulation

After Dinner Talk

Physical primitives as a basis for reconfigurable simulations,
and agent control architectures, and semantics *(P.Cohen)*

Tuesday, July 6 1999

Presentations

An Agent Architecture for Agents in Virtual Environments *(K.Fischer)*
Planning Agents in JAMES *(B. Schattenberg)*
A Framework for Modeling and Simulation of Mobile Agents *(F. Barros)*
Cognition and Emotion - Cooperative Agents
and their Internal Structure *(B. Schmidt)*
Using Agents in a Multiscale Simulation *(B.K. Szymanski)*

Working Groups . . . Continuing . . .

After Dinner Talk

Formalizing, Replicating and Publishing Simulation Models *(N. Gilbert)*

Wednesday, July 7 1999

Presentations

Creating 3D Environments for Objects and their Models *(P. Fishwick)*
Model Semantics:Automated Formalism Transformation *(H. Vangheluwe)*
Modeling Agents from a Semiotic Perspective *(C. Joslyn)*
Software Agents and Simulation: A Taxonomy *(T. Oren)*

After Dinner "Jam Session"

An Agent-Based Design of a Simulation Tool
for Complex Social Systems *(M. Möhring, E. Mentges)*
Realizing Tutoring with Agents and a Discrete Event Approach *(A. Martens)*
Using Mobile Agents for Data Acquisition in Simulation *(L. Wilson)*

Thursday, July 8 1999

Presentations

Macro - Micro - Simulation: Concepts and Applications *(R. Grützner, H. Unger)*
Using High Level Architecture in Civil Domains -
Looking Back and to the Future *(T. Schulze)*
Component-based Simulation Using Javabeans *(H. Prähofer)*
Model Reconstruction for Problem Solving by
Intelligent Demons during Dynamic Simulation *(A. Javor)*
Interacting Assemblies of Goal-Based Learning *(E. Gelenbe)*

Working Groups . . . Continuing . . .

Friday, July 9 1999

Discussion

Issues regarding Agents for Modeling and Simulation
(P. Fishwick, B.Zeigler, A. Uhrmacher)

DSDE: A Formalism For Representing Mobile Components

Fernando Barros

Universidade de Coimbra, Dept. Eng. Informatica
Polo II, Pinhal de Marrocos, P-3030 Coimbra (P)

Many real systems are better perceived when represented as dynamic structure models. Examples of such systems include adaptive computer architectures, biological systems, manufacturing systems and communication systems. For instance, in biological systems, the synapses' network of a human's brain changes its structure in the course of its lifetime by creating and destroying synapses. One spectacular demonstration of the ability of biological systems to change their own structure is the metamorphosis of a caterpillar into a butterfly. Non-biological systems, like computer networks, can also modify their structure to adapt to component failure or to traffic increase.

The Dynamic Structure Discrete Event Systems Specification (DSDE) is a modeling formalism that supports structural changes in models. In the DSDE formalism models are built in a hierarchical and modular form due to the closure of the composition operation. One type of structural change supported by the DSDE formalism consists in the movement of a component between two models. This type of component is named here by Mobile Component, and can be fully described within the DSDE framework. Mobile Agents are a network technology that involves the movement of software units called agents, between computers through a network. Mobile Components, due to structural similarity, offer a natural way to represent Mobile Agents, and provide the ideal framework for modeling and testing agent applications.

ISSAC: An Intelligent System for Speculative and Active Code computations

Christopher D. Carothers, Boleslaw Szymanski, and Mohammed J. Zaki

Rensselaer Polytechnic Institute, Dept. of Computer Science
110 8th Street, NY 12180 Troy (USA)

Next generation distributed object systems will be required to support millions if not billions of objects. To solve the impending performance and scalability problem accompanying such large and complex systems, we propose a societal approach. Here, objects, like people, must possess the capacity to acquire and share new knowledge, to store knowledge about their current behaviors, to modify their behavior for the better, as well as be able to accept some degree of risk by exploiting opportunities for software level speculative execution, where objects synchronize only when it becomes absolutely necessary.

Specifically, our system ISSAC: an Intelligent System for Speculative and Active Code computations, enables the discovery of knowledge regarding the common case for large-scale, object-oriented systems using parallel data mining to detect run-time patterns not possible using previous compile-time or run-time systems. Using these patterns ISSAC can determine a variety of potential sources for performance optimizations, such as dependent/independent objects, data access patterns, communication patterns, code bottlenecks, abnormal execution states and resource contention, as well as crucial opportunities to exploit software level, speculative execution.

Using these patterns, we use a process called Active Code to dynamically modify the source code tree, re-compile the system and reload the newly optimized distributed object application. This allows not only the common case to be executed faster, but also the intelligent deployment of software-level, speculative execution.

To demonstrate the benefits of ISSAC, we plan to conduct a performance study using two classes of distributed applications: discrete-event simulations and irregular numerical computations.

The ultimate goal of ISSAC is to achieve robust performance of complex distributed object systems which enables them to adapt to and optimize for a dynamic critical path as well a dynamic run-time environment when executed on today's high-power adaptive "grid" computing platforms.

Physical Primitives as a Basis for Reconfigurable Simulations, and Agent Control Architectures, and Semantics

Paul Cohen

University of Massachusetts, Computer & Information Sciences
Lederle GRC, MA 01003-4610 Amherst (USA)

Recent experiments with heterogeneous agents identified semantics as the principal impediment to interoperability. The DARPA CoABS program has demonstrated interoperability of roughly 50 heterogeneous agents, but in the 1999 experiments this was achieved by human developers negotiating the semantics of the messages sent among the agents. Might agents share a semantic core and negotiate meanings for themselves? It helps to know how humans understand each other.

I am particularly interested in how semantic systems develop in children. As infants, we are sensorimotor agents, capable of pushing and grasping, mouthing and waving, and so on. The question is whether a semantic system, one that assigns meanings to symbols, can emerge from sensorimotor activities. Our work on planning for war games and robotics suggests that physical schemas are a semantic core that can be bootstrapped into more sophisticated, adult semantic systems. Physical schemas are simple plans like pushing, moving, blocking, grasping, and so on. Many physical situations can be understood in terms of these primitives, and many non-physical situations can be understood as metaphorical extensions of physical situations.

Multiagent Systems and Simulation — Thoughts and Applications

Klaus Fischer

Deutsches Forschungszentrum für Künstliche Intelligenz
Stuhlsatzenhausweg 3, D-66123 Saarbrücken (D)

The presentation general remarks on agents and multiagent systems, concentrating on characterization of agents and agent architectures. Two views on simulation using multiagent systems were presented. In the first view (MAGSY's view) the simulation world is explicitly represented by a special agent, but agents do not necessarily have to interact with this world agent. In the second view (SIF) all agents communicate using a medium which also represent the simulated world. In this latter view the agents therefore necessarily interact with the simulated world. As examples for systems designed according to MAGSY's view, the transportation scenario and two settings for flexible manufacturing were presented. To explain SIF the loading dock example was described.

3D Behavioral Modeling

Paul A. Fishwick

Computer & Information Science Dept., University of Florida
FL 32611-6120 Gainesville (USA)

Modeling is used to build structures that serve as surrogates for other objects. As children, we learn to model at a very young age. An object such as a small toy train teaches us about the structure and behavior of an actual train. Later on in life, we use virtual world construction techniques on a computer. Virtual worlds enable us with the tools to create 3D geometric, scale-oriented models of trains. But how does the train move and operate when influenced by its engineer and the environment? We can build computer programs to facilitate behavior. We can also create models of the train's behavior and use the simulation of the models in lieu of the program.

This leads us to question the relation between the scale model (as studied in computer graphics and computational geometry) and the dynamic model (as studied in areas such as simulation). We demonstrate a methodology and environment where the same virtual world of objects is used for both types of models. There is no difference between the objects and components used to create computer graphics models and those used for computer simulation. This unification matches our intuitive definition of "model" where a 3D object serves to capture attributes of another.

Adaptive Goal Oriented Agents in Simulation Environments

Erol Gelenbe

University of Central Florida, Dept. of Computer Science
FL 32816-0362 Orlando (USA)

A recent National Academy of Sciences report [5] defines human behavior representation as a computer based model that mimics the behavior of a single human or the collective behavior of a team of humans, and decries the lack of behavior realism in simulations. The purpose of our research is to develop a scientific approach to learning and adaptation in large scale simulations. We address the key question of mathematical and computer modeling of units or simulated individuals, or manned vehicles, when these entities have the ability to learn from experience and rapidly adapt to random and time varying conditions. Many existing or projected simulation systems use Finite State Machines (FSM) to control the activity of the significant agents in the system, yet FSM are computationally too simple to represent complex adaptive behaviors. On the other hand, randomized or Stochastic FSM (SFSM), have been mathematically proven [2,3] to have full algorithmic computing power (i.e. as powerful as any computer algorithm). Thus they constitute a good paradigm for representing complex adaptive behaviors in existing simulation frameworks. In our approach, each simulated agent or entity (individual, manned or robotic vehicle, unit, etc.) is equipped with one or more decision making SFSM. We investigate and test three significant adaptation paradigms to construct "oracles" which determine the next state transitions of SFSM from current state and inputs, based on externally stated goals: (1) Adaptive SFSM, (2) Random Neural Networks with Reinforcement Learning [7], and (3) Genetic/Evolutionary Computing. Agents in our simulated world act in a structured environment containing random and time varying events and elements. The inputs to each agent are observations using sensors, and information which may be communicated among the agents. The notion of an "oracle" in mathematical automata and computer science theory is a classical concept which comes from mathematical logic [1]. The purpose of the oracles is to provide goal based adaptive control of the agents, using reward and penalty functions. We theoretically and experimentally evaluate the computing power of each of the three paradigms, examine how they can be mapped into a FSM framework, and determine equivalences and relationships between sub-classes of the paradigms. We examine how broadly defined goals can be translated into appropriate reward and penalty functions for task based adaptive control for each of the three paradigms. We will also examine the computational cost involved. In order to test these concepts, we have implemented a simulation where agents engage in an operation in a structured but dangerous urban environment, as discussed in our recent paper [8]. Each of the three adaptation paradigms (Adaptive SFSM, Random Neural Networks with Reinforcement Learning [7], Genetic/Evolutionary Computing) are evaluated in an identical setting where adaptation must lead to the agents' short-term success in real-time.

References

1. H. Rogers "Theory of Recursive Functions and Effective Computability", McGraw-Hill, New York, 1967.

2. P. Turakainen "On probabilistic automata and their generalizations", *Ann. Acad. Sci. Fenn. Ser. A1 Math.-Phys.*, No. 429, 1968.

3. E. Gelenbe "On languages defined by linear probabilistic automata", *Information and Control*, Vol. 18, February 1971.

4. E. Gelenbe "Learning in the recurrent random neural network", *Neural Computation*, Vol. 5, No. 1, 154-164, 1993.

5. R.W. Pew and A.S. Mavor (Editors) "Representing Human Behavior in Military Situations: Interim Report", National Academy Press, Washington, D.C., 1997.

6. Proceedings of the 7th Conference on Computer Generated Forces and Behavioral Representation, 1998

7. E. Gelenbe, Z.H. Mao, Y. Da Li "Function approximation with spiked random networks" *IEEE Trans. on Neural Networks*, Vol. 10, No. 1, pp. 3–9, 1999.

8. E. Gelenbe "Modeling CGF with learning stochastic finite-state machines", *Proc. of the 9th Conference on Computer Generated Forces and Behavioral Representation*, pp. 113–116, Orlando, May 1999.

Formalising, Replicating, and Publishing Simulation Models

Nigel Gilbert

University of Surrey, Dept. of Sociology
GU2 5XH Guildford (Surrey) (GB)

In this after-dinner talk, I examined the papers that have been published in the electronic journal, the Journal of Artificial Societies and Social Simulation (JASSS), http://www.soc.surrey.ac.uk/JASSS, since its first issue in January 1998. As the editor of JASSS, I have been concerned about the development of a tradition of writing articles describing social science simulations and in particular about the extent to which articles provide sufficient detail to enable readers to understand and replicate the work reported in them.

Computational modelling as a social science methodology experienced a rebirth in the 1990s. Until 1998, most research reports were published in edited proceedings. JASSS was founded as a fully refereed academic journal to provide a widely available, easily accessible outlet for the growing amount of work in the area. It follows traditional academic journal procedures, but electronic publication over the Web also allows faster turn-around of articles, the inclusion of multimedia and code listings, and distribution to readers all over the world. Access is free of charge.

To date, JASSS has published 19 refereed papers in six issues in two volumes. In addition it has published 8 unrefereed 'Forum' articles and 16 book reviews. Of the 19 papers, 18 describe computer simulations (the remaining paper is a survey of the use of genetic algorithms in social simulation). Four articles provided access to the program code of the model or in one case, a Java applet that could be run by the reader. An examination of the 18 articles showed that no two articles reported the use of the same programming language (the languages used varied from C and TurboPascal to the specification language Z, and included the use of several simulation toolkits, including Swarm, SDML, and Dynamo). Six articles primarily used mathematics to explain the models they described, usually employing difference equations. Two articles used a high level formalism (Z and a qualitative simulation language); three used process diagrams; and seven used textual descriptions.

A subjective estimate of how difficult it would be to replicate the models revealed that five articles did not provide enough detail to enable a replication to be carried out. However, it was suggested in the talk that this is not in itself a mark of an inadequate research report. The primary objective of an article in JASSS should be to provide a better understanding of some social phenomenon or process. While ideally it should also be possible to replicate the work, the most important characteristic of a paper must be that it offers social scientific conclusions that lead to generalisable findings. Nevertheless, most papers should explain the models used in detail. There remains a question about the most appropriate means of doing this. One possibility would be a universal specification language, but it is unlikely that there would be consensus about the nature of such a language. Specifications also need to reflect the objectives and theoretical context of the research, making it difficult to define a language of general applicability. The method adopted by the majority of the published articles, a fairly ad hoc mixture of textual description, process diagrams and difference equations, may be the best solution.

Macro - Micro - Simulation: Concepts and Applications

Rolf Grützner and Helen Unger

Universität Rostock, FB Informatik
Albert-Einstein-Str. 21, D-18051 Rostock (D)

The term agent is originated at the Artificial Intelligence whereas individual is defined in Artificial Life. We will consider both terms as synonymous and we use mostly individual to document that real living objects may be modelled. Individuals are used to model objects that perceive, act and reason. We divide models into two groups: micro- models and macro-models. A model which is based on equations is called macro-model. A macro-model is an indivisible whole entity, an algebraic structure, it does not take into account the fact that the modelled real world system usually consist of smaller individual components. But it may contain different levels of aggregation or of resolution. The resolution may relate to time, space, and structure. A model with a very high resolution based on individual objects may be called micro-model. A macro-model has a defined structure which may be fixed or variable. For a micro-model no explicit structure is defined.

Multi level models incorporate components at multiple levels of resolution. Multi level models appear on the one hand at the macro level if models are coupled with different resolution levels and on the other hand by the coupling of micro- and macro-models.

Simulation that run at multiple levels of resolution often encounter consistency problems because of insufficient correlation between the attributes at multiple levels of the same entity. Even if we assume that the models to be linked are valid, inconsistency can arise in the linkage. Consistency issues arise when low resolution entities interact with high resolution entities. The common solution approach is to dynamical change the resolution of a low resolution entity (or high resolution entity) to match the resolution of other encountered entities. The problem of linking simulations at different levels of resolution is so reduced to the aggregation - disaggregation problem - also called cross-resolution modelling.

The state-of-the-art in multilevel interaction (cross resolution modelling) can be described as being moderately successful in that linkages have been effective to some extent in each project but a general technique and associated theory are lacking. The used techniques are:

- full disaggregation,
- partial disaggregation,
- pseudo disaggregation,
- cross-level aggregation,
- unify method, and
- the concept of virtual individuals and virtual compartments.

Virtual compartments and virtual individuals are concepts suggested by J.Ortmann, a PhD student of our research team. Some projects in the field of micro-macro simulation and of micro-simulation now are represented. For example the source traffic and back traffic of a suburban settlement should be investigated. The cars are modelled as individuals characterised by the type of the driver and by a daily plan of activities of this individual.

A model of an adaptive individual which starts to navigate in an unknown city and which learns to move better and better in the city by perceiving its environment. That is a problem of learning in geographical spaces and in variable environments.

A further task is the investigation of the behaviour of persons which have to choose the transport mode, that means the vehicles for a city trip (the private car or a public transport mode) by an individual oriented model with adaptive individuals. A macro-model component models the traffic density which influences the future behaviour of persons.

Another problem may be the modelling of an ecological food chain, containing four populations. Three populations are modelled as differential equation system and one of these populations is modelled as individual oriented model. A typical micro-macro model.

Model Reconstruction for Problem Solving by Intelligent Demons During Dynamic Simulation

Andras Javor

Technical University of Budapest, Dept. of Information Management
Muegyetem rkpt 3 H-1111 Budapest (H)

The methodology using static and mobile agents in form of intelligent demons [1,2] for changing model structure and parameters during the process of dynamic simulation as well as Knowledge Attributed Tokens of high level Knowledge Attributed Petri Nets [3] are introduced and their implementation in an AI controlled simulation system is outlined.

Examples where the agents have been applied already for determining optimal solutions in the fields of urban traffic and air pollution control as well as for optimizing flexible manufacturing systems are presented [4].

As a new type of application the multifaceted [6] problem of the development of regions with special emphasis on traffic conditions is dealt with. In this example the agents are also given the task to find a model describing the real system adequately [5].

Possibilities for the application of demons as building blocks of the models in form of interacting agents is also dealt with.

References

1. Javor, A.: Demons in Simulation: A Novel Approach Systems Analysis, Modelling, Simulation 7(1990)5. 331-338.
2. Javor, A.: Demons in Simulation: A Novel Approach in A. Sydow (ed.), Computational Systems Analysis, Topics and Trends, Elsevier, Amsterdam, 1992, 355-370.
3. Javor, A.: Knowledge Attributed Petri Nets Systems Analysis, Modelling, Simulation, 13(1993)1/2, 5-12.
4. Javor, A., Szucs, G.: Intelligent Demons with Hill Climbing Strategy for Optimizing Simulation Models Summer Computer Simulation Conference, Reno, Neveda, July 19-22, 1998. 99-104.
5. Javor, A., Szucs, G.: AI Controlled Simulation of the Complex Development of Regions Summer Computer Simulation Conference, Chicago, Illinois, July 11-15, 1999. 385-390.
6. Zeigler, B.P.: Multifacetted Modelling and Discrete Event Simulation Academic Press Inc., 1984.

Agent Modeling from a Semiotic Perspective

Cliff Joslyn

Los Alamos National Laboratory
CIC-3, MS B265, NM 87545 Los Alamos (USA)

We present a perspective on agent-based modeling and simulation based on semiotics, or the science and signs and symbols. After reviewing the current state of the use of the "agent" concept in modeling and simulation, we describe the semiotic approach, based on a generalized control theory architecture and attention to the perceptual (measurement) and action modalities of the agents. Consequences of the semiotic approach include the relativism of knowledge (as interpreted symbols) to particular agents, and control as decentralized constraint on decision-making which has many sources, including the virtual environment, semiotic communication structures, and shared knowledge among agents. Finally, we discuss the ideas concerning application to the modeling of socio-technical organizations with hierarchical command structures.

The Distributed Simulation of Multi-Agent Systems

Brian Logan

University of Birmingham, School of Computer Science
Edgbaston, B15 2TT Birmingham (GB)

Agent-based systems are increasingly being applied in a wide range of areas including telecommunications, business process modelling, the control of mobile robots and computer games. Such systems are typically extremely complex and it is often useful to be able to simulate an agent-based system to learn more about its behaviour or to investigate the implications of alternative architectures. However the computational requirements of simulations of many agent-based systems far exceeds the capabilities of conventional sequential von Neumann computer systems. One solution to this problem is to exploit the high degree of inherent parallelism in agent-based systems to allow distributed simulation.

In this talk, I briefly outline some of the problems we have encountered in attempting to apply conventional distributed simulation techniques to the simulation of multi-agent systems and sketch a new approach to the efficient distribution of agents and their environments across multiple processors.

Realizing Tutoring with Agents and a Discrete Event Approach

Alke Martens

Universität Ulm, Abt. für Künstliche Intelligenz
D-89081 Ulm (D)

Most of the work in the area of simulation and education is dedicated to the process of teaching students the essentials of dynamic systems by simulation. Thereby, the system to be taught becomes the subject to be modeled and simulated. However, the process of tutoring can be perceived itself as a dynamic process and can be treated as such. To support a flexible and intelligent tutoring process we distinguish between model and simulation level. Based on JAMES - a Java Based Agent Modeling Environment for Simulation [2] we employ a state-based, modular and hierarchical agent-oriented model design. The tutoring process is structured into different situations each of which consists of information and decision tasks. The navigation options are constructed by coupling tasks.

The user is represented as an agent which moves through the tutoring process by coupling to and uncoupling from tasks. Depending on the user's interaction and experience level it steers the tutoring processes and might even adapt the structure. Thus, agents and their changing interaction structure becomes central [1].

The system is designed to work on the Internet and is aimed at supporting the interaction of multiple users in a distributed setting.

References

1. Martens A., Uhrmacher A.M., Modeling Tutoring as a Dynamic Process - A Discrete Event Simulation Approach. In: Proc. European Simulation Multiconference ESM'99, 111-119, SCS, Ghent, 1999.

2. Uhrmacher A.M., Tyschler P., Tyschler D. Modeling and Simulating Mobile Agents. Future Generation Computer Systems, Special Issue on Web-Based Simulation, to appear.

An Agent-Based Design of a Simulation Tool for Complex Social Systems

Michael Möhring and Elke Mentges

Universität Koblenz - Landau, FB Informatik
Rheinau 1, D-56075 Koblenz (D)

There is a variety of different requirements and qualifications of users applying simulation as a research method in the social sciences. Experience gained with users of the simulation system MIMOSE and discussions in interdisciplinary workshops suggest that developing a unified modelling and simulation language for the social sciences is an unrealistic aim. This is why our project called MASSIF (an abbreviation of its german name Multi-Agenten-Simulation in der sozialwissenschaftlich-interdisziplinären Forschung) goes a different way. Instead of aiming at one unique modelling language we are working on a framework applying agent-based modelling as the approach that seems to be the most promising one to cope with the complexity of social structures and processes. The underlying concept suggests diverse abstraction levels for the description of multi-agent systems.

The design of the framework takes into account that its potential users can not be regarded as a homogeneous group but differ very much in their experience and capabilites using simulation tools. Hence, it offers different user layers for constructing multi-agent models with increasing abstraction level. The layers are horizontally independent, i.e. models can be completely defined on any layer without taking care of the others. Prefabricated building blocks adjusted to the specific needs and capabilities support the user constructing executable models. On the other hand the layers are vertically permeable: The set of building blocks can be enlarged using the tools and building blocks of the layer below.

To improve comprehensibility and transperancy we suggest to divide the process of modelling multi-agent systems into modelling agents and modelling interactions and relations between them: On the first rather low-level layer, relations and interactions between agents are modelled as communicative acts, i.e. agents interact with each other by exchanging messages. Describing communication between agents implicitly, i.e. as an internal process inside each communicating agent, is quite difficult and intransparent. Dialogues between agents often build patterns where certain message sequences are expected and other messages are expected to follow. This, as a rule, makes planning capabilities and an internal model of the communication partner(s) necessary. To cause such dialogues to arise spontaneously is restricted to societies of rather complex agents. A more transparent and pragmatic way is modelling communication explicitly: Typical patterns of communication are defined as message sequences and agreed as protocols between agents. This facilitates the description of even rather complex interaction structures. It is not restricted to high-level agent types - also quite simple agents are able to engage in meaningful conversation, simply by following the known protocol. Modelling agents is supported by a generic agent model. Its architecture is strongly related to the concept of INTERRAP, a hybrid, multi-layered agent architecture. The structure is modular, so several different types of agents can be assembled using the concept as a basis reaching from simple agents interacting by reactive patterns of behavior to social agents provided with complex planning and decision capabilities.

The second layer abstracts from the quite technical and low-level message level. It provides pre-defined basic interaction types to be used as building blocks modelling interaction structures between actors in a multi-agent system. These types are composed as sequences of messages. The set of pre-defined interaction types offered on this layer is expandable. Modelling agents is supported by pre-definied, adaptable agent types differing in structure and (inter-)action capabilities. To cover a broad spectrum, the range offered contains rather simple reactive agents up to complex social agents. Users can generate instances of these types and initialize their knowledge or rule bases and their goal and planning components respectively. All the types are grounded on the generic hybrid agent model and can therefore easily be stepwise enlarged. This ensures reuse of already defined agent models as their components can be refined.

The third layer offers a further step of abstraction to support modelling of multi-agent systems. It corresponds to the way people conceive and organize their understanding of the world around them. Actors are not perceived as isolated entities but as related to other entities. They interact having certain expectations and a restricted perspective on each other. Their perception and knowledge about the other's properties and behavior is always selective. Thus, they play certain roles for each other in their relations and interactions. Roles and role models are rather new concepts in object-oriented software engineering. Thinking in roles can also support modelling of actors in multi-agent models. It is a promising way as it makes it easier to break down complex models to delimited submodels. For this purpose pre-defined role patterns determining goals, resonsibilities, tasks and expertise of agents are offered as building blocks to users of the third layer of the framework. These patterns can be merged so that submodels can be combined to an overall model. As in real world phenomena agents own several roles in parallel, each one at a given time. So role patterns have to be coupled together. Role patterns have to be instantiated and adapted. Those properties and abilities ascribed to a role are defined by the chosen role pattern. They can be enlarged and complemented by the user with other characteristics of the modelled agent.

Further, we apply agent-based approaches designing the system architecture of the framework to facilitate reuse of simulation models and interoperability of simulation systems: The overall architecture of the MASSIF system is designed as a system of software components. It connects both components that will be designed and implemented within our project and already existing applications. They interact and cooperate to exchange information and services with each other by communicating in an agent communication language like KQML or FIPA ACL. Thus, in the sense of agent-based software engineering, the framework can be viewed as a multi-agent system itself.

Sequence and content of communication between system components are defined by interaction protocols. A special software wrapping enables both internal and external components and applications connected to the framework to engage in meaningful communication - no matter of the programming language they are implemented in. All parts of the system are managed and coordinated by specialized system agents: A so-called manager agent receives and transmits meta-level information about all the agents connected to the system by request. Router agents are responsible for transport and delivery of messages. Their main function is to unburden agents from the technical level of communication.

As future work within the project we plan - besides the implementation of a prototype - to apply the introduced concept of several hierarchically ordered user layers to an experimental frame as the second part of the framework. This will be done to make tools available for analysing models and describing experiments that are adapted to the specific requirements of different user groups.

Simulation, Artificial Intelligence and Agents: A Taxonomy

Tuncer Ören

Marmara Research Center, Informatics Institute
P.O.B. 21, TR-41470 Gebze-Kocaeli (TR)

The aim is to present a unified framework for different types of synergy of simulation, artificial intelligence and agents to appreciate the unity of the field and to systematize the exploration of the possibilities for research and applications. Six groups of possibilities can be discussed in two general categories:

AI-directed simulation, i.e., use of AI in simulation

- Cognitive simulation (i.e., simulation of systems with cognitive abilities). It is also be called simulation of intelligent systems. This field covers natural systems (humans, animals) and engineered systems such as software systems and computer-embedded systems.
- AI-based simulation (or cognizant simulation) where AI is used for the generation of model behavior and in associated machine learning. Some possibilities include knowledge-based simulation, qualitative simulation and knowledge-based systems used for non-experiential knowledge generation and which can be nested with simulation systems.
- AI-supported simulation (or cognizant simulation environments) where AI is used in user/system interfaces. Cognitive (also called intelligent) front-end interfaces can provide advanced help, just-in-time learning ability, assistance, guidance and solicited or unsolicited advice to the user. Furthermore, they can also provide abilities such as perception, speech, deictic (gesture) inputs and haptic (touch) inputs. Cognitive back-end interfaces can be used in communicating primary and secondary outputs to the users. Primary outputs include unprocessed and processed behavior, performance measures, results of evaluations and advice on the problem. Auxiliary outputs include automated documentations (including virtual gauges) and explanations. Cognitive back-end interfaces can also be used in virtual and augmented reality applications.

Agent-directed simulation, i.e., use of agents in simulation

- Agent simulation (i.e., simulation of agents or simulation of systems which can contain agents). This field covers natural systems (humans or animals represented as agents or avatars) and engineered systems such as multi-agent systems with full or reduced autonomy and intelligent machines or systems which are implemented by software agents.
- Agent-based simulation where agents are used for the generation of model behavior.
- Agent-supported simulation where agents are used in user/system interfaces.

Component-based Simulation for Multi-Agent Systems

Herbert Prähofer

Johannes Kepler Universität, Abt. fr Systemtheorie und Informationstechnik
Altenbergerstr. 69, A-4040 Linz (A)

Component-based simulation emerges from the synergism of system theoretic simulation modeling formalism and component-based software engineering. In this talk component-based simulation has been discussed as a paradigm for multi-agent system simulation. While the system formalisms most notable the DEVS formalism can serve as a methodology for modular, hierarchical modeling and simulation, the JavaBeans component model provides the appropriate implementation base. The benefits of a component-based simulation methodology for agent-based system simulation are a more concise model building methodology, better model reusability, concepts for building model libraries, and the possibility for interactive programming of simulation systems. The SimBeans simulation framework [1,2] has been presented as a component-based simulation framework based on the Java language and JavaBeans component model. It is based on a component-based simulation methodology that provides component libraries for different purposes, at different levels, for different users, and for different applications. The main objective is reusability; i.e., simulation systems can be built with less effort mainly by selecting, extending, customizing, and assembling components from libraries. It supports discrete event, continuous, combined discrete/continuous, and variable structure modeling and simulation. For building reusable simulation components, the following principles have been proposed.

Separation of concerns: Components should be identified with a clear separation of concern in mind. Typically components for realizing

- physical objects themselves (effort states, see below)
- the coupling structure (flows, see below)
- control schemes
- containers of physical objects modeling the environmental conditions (see below)
- output, visualization, animation, and output analysis

should be distinguished.

Modular interfaces: Interface definitions should clearly characterize how a component can be used and how it can communicate with other parts of the system. The component itself should make as less assumptions about its environment it is embedded as possible (use model containers, see below). Separation between effort and flow components: Similar to the distinction made in bond-graph modeling between 1- and 0-junctions, there should be a clear distinction between components storing effort states and components realizing flows. While effort components should be seen to provide effort states and are influenced by flows, flow components realize the flows between components which are dependent of the effort states of the connected components.

Hierarchical composition: Components should be organized hierarchically. More complex components should are built by using primitive modular components and defining how those work together by realizing the respective coupling structures.

Component containers: A model container should be provided where the model components "live in". Such a model container should represent the environmental conditions (physical constraints) which apply. For any component which is added to live within the container, the conditions are enforced.

Finally, the application of the guidelines have been shown in the design of a component framework for moving objects in space, which represents a subproblem for multi-agent system simulation.

References

1. Prähofer, H., A. Stritzinger, and J. Sametinger, Discrete Event Simulation using the JavaBeans Component Model. WebSim'99, San Franzisco, CA, Jan, 1999.

2. Prähofer, H., A. Stritzinger, and J. Sametinger, Concepts and Architecture of Simulation Frame - Based on the JavaBeans Component Model. Journal of Future Generation Computing Systems, Special Issue on WebbBsed simulation, 1999, (accepted)

Testing Planning Agents in JAMES

Bernd Schattenberg

Universität Ulm, Abt. für Künstliche Intelligenz
D-89081 Ulm (D)

JAMES integrates deliberative agents within discrete event simulation. It is aimed at providing an environment for experimentally testing agent architectures, single modules, and interaction strategies [1]. Its main objective is to facilitate testing in the small and testing in the large, equally. Therefore, JAMES provides a conceptual framework that enables the complexity of the system to be managed by decomposition and abstraction. The term "system" refers to (multiple) agents, their dynamic environment and the multiple interactions existing between them.

JAMES is not a specification tool for agents presenting yet another agent architecture. Instead, timed state automaton are the "minimal" frame to embed agent architectures in JAMES. Since agents are embedded in open environments with varying composition and interaction structure, JAMES supports the change of structure from an agent's perspective. It provides a general, flexible, modular and theoretically well founded simulation approach with a clear separation between a declarative model design and its concurrent, distributed execution.

Our experiments with BDI agents in a TILEWORLD style environment, employing different planning systems, e.g. GRAPHPLAN and a model checking planer, have demonstrated the suitability of the timed state automata metaphor, the ease of integrating different deliberative components, and the value of variable structure models to capture the dynamics of (multi-)agent systems.

The gained experiences emphasize that the worth of our approach as a tool to develop multi-agent test beds will ultimately depend on the libraries which provide model prototypes and experimental frames, and on the ease in interacting with other simulation systems.

References

1. Uhrmacher A.M., Schattenberg B., Agents in Discrete Event Simulation. In: Proc. of the ESS'98, October 26-28, Nottingham, SCS Publications, Ghent, 129-136, 1998.

Modelling of Human Behaviour - The PECS Reference Model

Bernd Schmidt

Universität Passau, FB Informatik
D-94030 Passau (D)

The model presented is the multi-purpose PECS reference model for the simulation of human behaviour in a social environment. Particular emphasis is placed on emergent behaviour which is typical of the formation of groups and societies in social systems. Human behaviour is highly complex in its structure. It is influenced by physical, emotional, cognitive and social factors. The human being is consequently perceived as a psychosomatic unit with cognitive capacities who is embedded in a social environment. PECS is a reference model which enables us to specify and to model these = factors and their interactions. PECS stands for

Physical conditions
Emotional state
Cognitive Capabilities
Social Status

The PECS reference model is designed to replace the so-called BDI architecture. It is inadequate to restrict complex models designed to model real social systems to the factors belief, desire and intention.

Fundamentally we may distinguish between the following two methods for the control of behaviour:

Reactive behaviour This category comprises behaviour that follows fixed rules. This means that no explicit thought processes are required for the control of such behaviour. Reactive behaviour can be subdivided in terms of the preconditons for its appearance which determine the architecture in the model. Initially we may distinguish between:

1. Instinctive behaviour
2. Learned behaviour

A more complex form of reactive behaviour occurs when an inner psychic driving force in the form of an urge or an emotion controls behaviour.

The function of this form of behaviour is the satisfaction of needs comprising the physical, social and also the cognitive sphere. Here we can distinguish:

3. Urge-controlled behaviour
4. Emotionally controlled behaviour

Deliberative behaviour In this case behaviour does not follow fixed rules. Instead a goal is set which has to be achieved. By means of reflection, working with models and trying and testing, a sequence of actions is established which leads to the goal. Deliberative behaviour takes two forms:

1. Constructive behaviour
2. Reflective behaviour

These modes of behaviour have developed gradually in the course of evolution. Each step signifies an additional extension of possibilities and hence leads to better adaptation and to an increase in the chances of survival.

The human being as the most highly developed organism to date displays all forms of behaviour control in complex interaction. In all forms of behaviour control the physical situation, the emotional state, cognitive abilities and social position play a role. If human behaviour is to be modelled and hence made comprehensible and predictable, the following state variables must be taken into account:

- Physical state variable
- Emotional state variable
- Cognitive state variable
- Social state variable

Not every model investigation requires consideration of all 4 classes of the above mentioned state variables. Any number of combinations is possible depending on the nature of the problem and the model. The decisive factor is that it must be possible to construct complex models which contain all four classes and do not disregard their interactions. PECS agents provide a reference model which meets these requirements. The basic approach adopted for the modelling of human behaviour is first explained using two simple models. The Adam model shows the interplay of physical, emotional and cognitive components in an individual. The Group model extends this approach and documents the interplay of individuals in the process of group formation, group activities and group disbandment. To avoid being restricted to hypothetical play models and to demonstrate the real possibility of modelling human behaviour with PECS agents, real groups should be studied. For this purpose, children are observed during role plays. A powerful laboratory enables us to analyse behaviour. It is planned to compare the behaviour of the real group with the behaviour of the modelled agent group. The research program is inter-disciplinary in its approach and touches the fields of human medicine, psychology and artificial intelligence. However its true location is the field of artificial life.

Using High Level Architecture (HLA) in Civil Domains - Looking backwards and into the Future

Thomas Schulze

Universität Magdeburg, Institut für Technische und Betriebliche Informationssysteme
Universitätsplatz 2, D-39016 Magdeburg (D)

Introduction Simulation is an old and well known methodology to investigate existing or planed systems. Many existing simulation tools support the model developer and the user in the whole simulation process. Simulation would be used in wider areas if the financial and human effort for providing simulation projects could be reduced. One of the main challenges for the simulation community is to reduce human time for building simulation models and experimentation with models. Possible ways are the computer based (supported) model generation or to split the model into submodels or components, retrieve and reuse legacy components, build the remaining components and combine all components to a new composition. The reuse of components requires semantic description of the functionality, well defined interface for interoperability and services for time synchronisation. The High level Architecture (HLA) is one existing standard that can support this approach.

High Level Architecture (HLA) The High Level Architecture is a forthcoming IEEE simulation interoperability standard currently being developed by the US Department of Defense (DMSO 1998). This architecture supports Interoperability and Reusability of different kinds of programs The architecture is defined by :

- Rules which govern the behavior of the overall distributed simulation (Federation) and their members (Federates).
- An interface specification, which prescribes the interface between each federate and the Runtime Infrastructure (RTI) which provides communication and coordination services to the federates.
- An Object Model Template (OMT) that defines the way how federations and federates have to be documented (using the Federation Object Model, FOM and the Simulation Object Model, SOM, resp.).

Past and Ongoing Projects at the University of Magdeburg One focus in the research area at the department of computer sciences at the University of Magdeburg is the HLA-based distributed simulation. Different projects have been carried out.

Distributed Driving Federation In order to evaluate the interoperability and federation development aspect further, a distributed driving simulation project was designed as a joint effort of the Institute for Simulation and Graphics (ISG) at the University of Magdeburg and the Competence Center Informatik GmbH (CCI). The federation consists of federates that were brought into the project by both partners and independently extended for HLA compatibility. The simulation federates are based on existing simulation applications (legacy applications) from the partners. Both underlying simulation models had to be extended for communication with the RTI, interoperability between the models, and synchronization of local time advancement.

The Barrel Filling Federation This federation was developed in a cooperation between the Universities in Magdeburg and Passau and is the first federation featuring a federate modeled with the simulation system Simplex 3. Simplex 3 is a simulation system developed at the University of Passau which can be used for discrete and continuous systems. The main target in this federation was to combine the advantages of both simulation systems used in this federation. SLX offers very good capabilities for modeling logistical processes because of its process oriented world view. Simplex 3 has the ability to develop continuous models in a comfortable fashion and offers several procedures for the numerical integration. It was most desirable to combine these advantages of both systems to form a new (distributed) model consisting of two "sub-models", or federates, in the HLA sense. The federate developed with Simplex 3 simulates a chemical barrel filling station. The filling of barrels is modeled as a continuous process described by differential equations. The second federate is a SLX model which simulates the logistical processes in a transport agency. Orders for barrels are generated from different locations throughout Germany and passed to the barrel filling station. The SLX federate also performs an online visualization of the federation with Proof Animation(for Windows).

Online Data Processing in Simulation Models This project was implemented to demonstrate the integration of on-line data into HLA federations using the on-line federate approach. The federation is a public traffic management scenario in the city of Magdeburg, Germany, and was developed with the support of the Magdeburg traffic company (MVB). A simulation model which performs a schedule based simulation of the public transportation system in Magdeburg (i.e. streetcars, busses) has been developed using the simulation system SLX. The animation was developed using the Proof Animation System. The on-line federate starts a receiver process implemented as a separate thread to connect to the command and control computer of the Magdeburg traffic company. From there it receives position updates which are obtained from the "real-life" streetcars using infrared senders which each streetcar carries. The receiver process buffers the position updates in an internal cache which can be read by the actual federate thread asynchronously. The federate thread is time-regulating in the HLA-sense. Since position updates are sometimes received with large time intervals, the federate thread also runs real-time synchronized via the system clock and announces its time advancement in fixed intervals. In our case the "time constrained" condition can be neglected, since the on-line federate is the only driving force in the federation and, also, does not depend from any other federate.

Research to HLA-Methodology The future work concentrates on the two areas of cloning of federates and integrating a geographical information system into HLA-federations.

Cloning: We are currently researching mechanisms to test some of the cloning concepts using the traffic management federation. In order to do so, a way to save and restore the state of the simulation has to be implemented first. In order to have a solution which is generally valid, the simulation tool SLX needed to provide such kind of an option. Alternately, we will develop a model specific solution to save the state of this special simulation model. The general potential of cloning in this prototype lies in the fact that the simulation model shows the current state of the real system (because of the existence of an on-line federate) which then can be cloned. The clone can then be used to evaluate different alternatives for decisions, e.g. sending some streetcars and busses on a detour to by-pass a traffic jam, etc.

Development of a GIS federate: Another direction of future work will be the integration of a geographical information system (GIS) as a federate into HLA federations. The GIS should be used for dynamically retrieving the street and route network, which the simulation components are based on, at runtime of a federation execution. This provides for a great flexibility, since no networks had to be hard-wired into the simulation model.

Using Agents in a Multi-Scale Simulation

Boleslaw Szymanski

Rensselaer Polytechnic Institute, Dept. of Computer Science
Lally 204, 110 8th Street, NY 12180 Troy (USA)

This talk presents use of agents in on-line network simulation for data collection and simulation decomposition as well as in multi-modal epidemiological simulations.

As networks grow larger and more complicated, their management also becomes more difficult, especially when there is no central authority to coordinate management. Several research efforts are underway in the area of pro-active network problem avoidance which often relies on a large amount of current and historical network data to build models of network traffic. We have developed a distributed object, agent based framework for facilitating management of large networks as a foundation for supporting pro-active network problem avoidance. The framework serves as a middleware between a collection of independent network management agents and network nodes.

The complexity, heterogeneity and speed of the Next Generation Internet (NGI) require new, scalable approaches to network management and control. Towards this end, we are developing a system of collaborative, on-line simulators that can support a suite of distributed network management and control functions. On-line collaborative simulators can predict the network performance under different sets of traffic parameters therefore enabling an automatic network management to select an optimal parameters in response to the changing medium range temporal traffic patterns. To facilitate on-line simulation, we decompose the simulated network into domains with inter-domain traffic modeled and simulated by agents.

Epidemiological simulations often involve interactions of several different species at various spatial and temporal scales. For large organisms, individual based modeling using discrete event simulation is appropriate. The small organisms require continuous models described by partial differential equations. We discuss how mobile agents can be used to link such different models into one cohesive simulation on the example of Lyme disease model in which deer and mice are individually modeled whereas ticks are represented as continuous space coverage.

JAMES - A Java-Based Agent Modeling Environment for Simulation

Adelinde M. Uhrmacher

Universität Ulm, Abt. für Künstliche Intelligenz
D-89081 Ulm (D)

JAMES, a Java-Based Agent Modeling Environment for Simulation, realizes variable structure models including mobility from the perspective of single autonomous agents. JAMES is based on DEVS [3] which belongs to the formal and general approaches to discrete event simulation. The model design is coined by a modular compositional construction of models distinguishing between atomic and coupled models.

In JAMES, the former are able to create new models, to add existing ones within the boundary of the embedding coupled model. They can delete, and remove, themselves from their interaction context and determine their own interaction with their environment [2]. Models can initiate their movement from one interaction context, i.e. coupled model, to another. An agent can initiate its move however for its completion, i.e. for being embedded within the new context, it needs the cooperation of an on-site model. To initiate structural changes outside their boundary, agents have to turn to communication and negotiation in JAMES. Thus, a movement from one coupled model to another implies that another atomic model complies with the request to add the moving model into the new interaction context. To facilitate modeling, all atomic models are equipped with default methods that allow them to react to requests, e.g. to add a model. However, these default reactions can be suppressed to decide deliberately what requests shall be executed. The freedom to decide whether to follow a certain request, and its knowledge, i.e. beliefs, about itself and its environment, distinguish active agents from more "reactive" entities.

JAMES itself is based on parallel DEVS and adopts its abstract simulator model [1]. Simulation takes place as a sending of messages between concurrently active and locally distributed entities, i.e. simulators and coordinators which are associated with atomic models and coupled models respectively. If a model moves through a model which is executed in a distributed setting, it will actually move through the physical network. It forms a type of process migration since it happens transparent for the user and is motivated by reducing message passing between physical nodes. We could equally let the model stick to its old place. However this would likely decrease the efficiency of the simulation, since in JAMES like in DEVS a simulator is doomed to communicate with and via its coordinator.

Thus, modeling and simulation in JAMES are coined by an agent-based perspective in terms of concurrently interacting mobile entities with varying composition and interaction pattern.

References

1. Chow A.C., Parallel DEVS: A Parallel Hierarchical, Modular Modeling Formalism. SCS - Transactions on Computer Simulation. Vol. 13, No. 2, 55-67, 1996.

2. Uhrmacher A.M., Tyschler P., Tyschler D. Modeling and Simulating Mobile Agents. Future Generation Computer Systems, Special Issue on Web-Based Simulation, to appear.

3. Zeigler, B.P.: Multifacetted Modelling and Discrete Event Simulation Academic Press Inc., London 1984.

Model Semantics for Software Agents

Hans Vangheluwe

Biometrics and Process Control (BIOMATH), Dept. for Applied Mathematics
Coupure Links 653, B-9000 Gent (B)

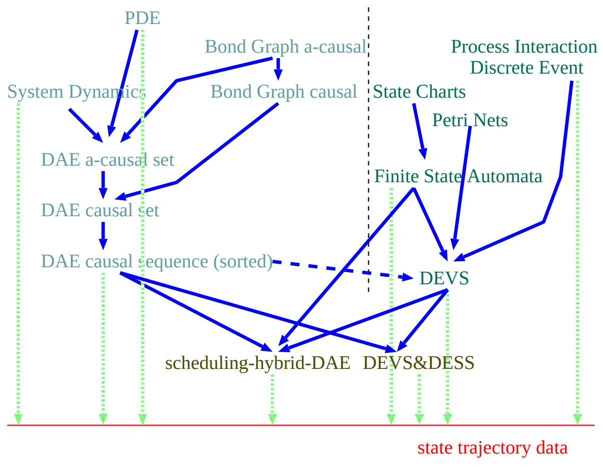

Fig. 1. Mapping of Formalisms

For the purpose of discussing how model semantics can enable meaningful communication between agents, a pragmatic definition of agent is used. An agent is seen as a concept whereby certain attributes of human agents are present. One hopes that the concept leads to better, faster, more elegant, ... satisfaction of our goals and solution of our problems. When the agent concept is restricted to software implementations, in particular with integration of software components as a goal, we use the name software agent. It is obvious that somehow, diverse agents need to exchange information. It is proposed to let that information be in the form of models.

In the current context, a model is an abstract representation of the behaviour of a system. The model (and the experimental frame describing under which conditions the model accurately represents system behaviour) is considered to be the basis for formal verification, symbolic manipulation, numerical simulation, (embedded) code generation, and documentation. To allow a collection of agents to meaningfully collaborate, a common frame of reference is needed. Traditionally, some form of ontology is used. Ontologies tend to not include a (complex) notion of time. If the basic frame of reference is a modelling formalism, information exchange can be at a high level. As different agents may externally represent information in different formalisms (not necessarily the same as the formalisms used inside the agent), agent communication must be able to handle multiple formalisms meaningfully.

A suggested solution is to investigate the representation (at meta-level) of different formalisms, starting with the common, exisiting ones and allowing for addition of new ones. Traditional formalisms include Petri Nets, Finite State Automata, DEVS, Bond Graphs, Forrester System Dynamics, Differential Algebraic Equations, ... As basically all formalisms are rooted in mathematics, a number of primitive mathematical concepts such as sets, mappings, ... can be the basis for a meta-description of formalisms. Once this is possible, once can investigate and chart the relationships between different formalisms. In the figure (1), the arrows denote a homomorphic relationship "can be mapped onto". In a denotational sense, traversing the graph makes semantics of models in formalisms explicit. This traversal is also the basis for meaningfully coupling models in different semantics: once they can be mapped onto a "common" formalism, closure in that formalism makes explicit the meaning of the coupled model. This approach is proposed as a basis for content of (e.g., KQML) messages between agents.

Using Mobile Agents for Data Acquisition in Simulation

Linda F. Wilson

Dartmouth College, Thayer School of Engineering
8000 Cummings Hall, NH 03755-8000 Hanover (USA)

Simulations often operate on static datasets and data sources. Many simulations would produce more-accurate results if they could access dynamically-changing data from other sources. From the perspective of one simulation, other simulations are data resources, producing information possibly relevant to the past, present, or future of the system being modeled. Mobile agents allow dynamic linking between distributed simulations and efficient monitoring of and access to remote data sources.

In this talk, we discuss briefly our initial work using the D'Agents system (formerly Agent Tcl) to coordinate distributed operational simulations and efficiently communicate data between simulations.

Introduction to HLA, DEVS and Agent Endomorphic Models

Bernard P. Zeigler

University of Arizona, Dept. of Electrical & Computer Engineering
1230, E. Speedway, P.O. Box 210204, AZ 85721-0104 Tucson (USA)

HLA (High Level Architecture) is defense standard for distributed simulation. Military systems are increasingly concerned with situation awareness information control of the battlespace, HLA explicitly supports modeling the rich sensory/perception (visual, RF, IR,accoustic,..) spatial environment in which systems operate. Differential abilities in the information space are critical to which side wins. Our hypothesis is that agents in general are constrained not only by energy resources but by informational ones such as the rich environments supported by HLA. Modeling/understanding/accounting for these informational constraints may be a key road to progress in modeling and implementing useful agent behaviors. For example, biological arms races played key role in evolution – so understanding biological agents requires modeling their information constraints and differential capabilities.

In this presentation, we provide an introduction to HLA emphasizing the support for data management in the form of environment-to-environment, agent-to-environment, and agent-to-agent interactions. We then review DEVS (Discrete Event System Specification) and its implementation in HLA compliant form. DEVS/HLA provides the ability to exploit and build upon the data management capabilities of HLA with a powerful dynamic systems formalism. We conclude with the concept of endomorphic models (models of system components employed by other components) and their use in modeling/designing agent-to-agent interactions and communications.