The IEEE International Conference on
Industrial Informatics (INDIN 2008)
DCC, Daejeon, Korea July 13-16, 2008

A Simulation Platform for Cognitive Agents

Tobias Deutsch, Tehseen Zia, Roland Lang, Heimo Zeilinger
Vienna University of Technology, Institute of Computer Technology
{deutsch, zia, langr, zeilinger}@ict.tuwien.ac.at

Abstract-To be able to develop reasoning units based on findings from sciences like behaviorism, psychology, neurology, and psychoanalysis, a test bed which is close to these sciences is needed. A possible approach is to use cognitive agents situated within a game of artificial life. Using such a simulated environment reduces the abstraction step from the origin science to the technical system. This article gives a short overview on such a game of life and discusses several design issues. The simulator is implemented using the simulation software AnyLogic.

I. INTRODUCTION

Developing new models for decision units of autonomous agents was the main goal by the project group Artificial Recognition System (ARS) at the Institute of Computer Technology at the Vienna University of Technology. This demand has been raised, since different areas of the institute's research, e.g. in building automation had to cope with situations that could not be predicted during the usual design phase when creating a product. Beside theories of the artificial intelligence and cognitive science, that had to be tested and applied within several environments and under different circumstances, a newly developed model for perception and decision making had to be created. This model, based on theories of the neuro-psychoanalytical science had to be easily compared to already existing models. A demand for one simulation platform for cognitive agents appeared not only for comparison but also for the development process.

In Chapter II, three available simulation platforms are introduced, that were considered at the beginning of the project. They offer great advantages in terms of performance for e.g. crowd simulations, but the focus at the first approach laid on a rapid prototype design for only view agents that are able to interact and communicate to each other in a very limited way.

For this approach, Xjtek's professional simulation tool AnyLogic has been chosen, which supports the development of two important issues: discrete, event-based processing and agent based simulation.

One essential advantage in terms of reusability of the developed model within AnyLogic was the development language, which is native Java with the corresponding compiling version of Java. This means that classes, built in AnyLogic could be easily used for further applications or as a basis in other simulation framework like swarm, repast, etc. Since AnyLogic natively supports agent based modeling, the whole development chain for building a new test environment for the decision unit of an autonomous agent is supported. It has been started at the definition and modeling of the agents and their environment. The simulation platform easily supported a visualization of those two entities and their internal values for observing their behavior. Three different types of agent behavior have been implemented. Using a predefined interface, it was possible to use the same agent container and equip it with the different the decision units:

- A simple if-then rules based decision unit for testing the sensor and actuator interface and the interaction primitives between the surrounding and the agent itself
- A decision unit based on concepts of the artificial intelligence
- A decision unit based on theories of neuron-psychoanalysis, as described in [1].

The different agents run within the same simulation, using the common engine for time discrete simulations. Therefore, every implementation uses the same simulation time and decisions were calculated within the same simulation time steps to guarantee a parallel processing. Timely processes as e.g. collision handling could be modeled, using AnyLogics time charts, where typical state charts with transition conditions can be developed using native Java code as transitions and triggering actions the states.

Chapter III describes the design principals of developing such an agent based simulation framework and shows the steps that had been taken within the first realization in AnyLogic. Chapter IV describes the framework within one single agent in detail and introduces the design patterns and rules that had been used, while Chapter V focuses on the implementation of the sensors, the agent has, to interact with the environment. The user interface and the different kinds of visualization are discussed in Chapter VI. Chapter 0 mainly shows how the developed concepts can be reused within further simulation environments and highlights the necessity in doing so.

II. STATE OF THE ART

By developing the ARS model the need for a simulation tool arouse in order to evaluate its design. The ARS model presumes the possibility to interact with the surrounding environment. Hence a system body, the model is implemented in, had to be defined. The idea evolved to use autonomous embodied software agents placed in a virtual environment for testing the model's abilities. For defining an adequate simulation environment the 'Fungus Eater' experiment of the Japanese psychologist Masano Toda [2] was considered. This thinking experiment is about a robot, which is sent to a planet for collecting uranium ore. While collecting the resources the robot's energy lowers and has to get refilled by eating fungi. As uranium ore and the fungi do not occur at the same place the question comes up if the robot is in conflict between attending its duties and refilling its sources.

978-1-4244-2171-8/08/$25.00 ©2008 IEEE

Another influencing model can be found in the work of Ho and Dautenhahn [3]. They develop a control architecture for autonomous agents by considering the theory of Episodic Memory. The agents have some kind of autobiographical memory which stores situations and events. Because this model is used for path finding tasks, the implemented memory should boost the agents' performance – already known paths should be retrieved or improved. In contrast to the ARS model this model doesn't consider a situation evaluation by emotions.

In order to consider physical effects like inertia or deceleration, physic engines are used in simulation environments. A number of engines like Newton Game Dynamics (http://www.newtondynamics.com), PhysX (http://www.ageia.com/physx), or Open Dynamics Engine (http://www.ode.org), are developed under open source and closed source licenses. Beneath models of simulation environments, and physic engines the simulation platform has to be defined.

For agent based simulation three platforms Swarm, Repast and MASON are used widely. They are enjoying a large spectrum of applications in various areas [4]. A Short review of these platforms is given below.

Swarm: Swarm is an object oriented library which is an open source for swarm members. Swarm is mainly based on Objective C. It provides own java libraries which provides connectivity to Objective C language library [5]. Swarm simulations introduced a framework based approach for modeling and simulation. Its model design philosophy is based on the concept "swarm" to organize models – the term "swarm" is defined as a collection of other swarm or agents for executing a schedule of actions. This hierarchical approach provides flexible way of modeling at multi-levels. Swarm provides a conceptual framework which treats the model as a separate task from visualization [6]. On the other hand it has few inherited complexities of Objective C i.e. required skills to program. Swarm provides it own memory i.e. data structure which makes the task of probing easy.

Mason: Mason is an object oriented free open source library of java. It is a joint effort of George Mason s' University s computer science department and George Mason University centre for Social Complexity. Mason provides a layered architecture for modeling and visualization which consists of three layers i.e. utility layer, model layer and visualization layer respectively from bottom up [7]. Mason provides a framework which strongly differentiates between model and visualization [8]. This differentiation makes it faster than other simulation platforms because the models can be run without visualization. Mason inherited the portability essence from java and can be run on multiple platforms. Mason is still in its infancy stages [5] i.e. not easy to be used for inexperienced programmers. In addition it has no built in graphs and charts up till now.

Repast: Repast is an object oriented open source platform available in multiple flavors i.e. Java, Microsoft.Net and Python scripting languages. It was created at the University of Chicago primarily. Now repast is managed by the non-profit volunteer organization called Repast Organization for Architecture and Development (ROAD) [9]. Recent advancement of Repast is Repast Simphony [10] which offers many build-in tools for model design and implementation. Repast is easily to use for inexperienced programmers and can be integrated with Integrated Developmental Environments like Eclipse. On the other hand a clear conceptual framework seems to be missing [2], which make it difficult to build models.

GeDA-3D is an example for research on an agent oriented platform [11], [12]. Previous to simulation runs, every agent gets assigned a specific goal which it has to achieve. The platform allows cooperation between agents. Therefore the agents' knowledge increase during the simulation runs which leads to faster achieving their goals. This approach aims at dynamic planning and cooperation issues in agent based simulators.

Even these technologies provide a number of useful tools the decision was made to user Xjtek's Anylogic for generating an own simulation environment – the BFG (Bubble Family Game). The main reasons were the support for discrete and event based processing and the use of Java as programming language.

III. CONCEPT

The game of artificial life should be able to support the development of psychoanalytical inspired decision making units as described in [13]. Thereafter, situations as described in psychoanalytical literature should be possible within the simulation. Social interaction is as well needed as points of interest, nourishments and danger.

In [14] three test-cases – ask for a dance, cooperation for food, and call for help – are introduced. They provide every element as stated above. Other scenarios, like exploring unknown entities are also possible. The social interaction is usually performed among teammates, but it is not necessarily limited to them.

The simulator has at least to include other agents, energy sources and obstacles. To provide a more interesting environment and more information to be processed by the agents, landscapes are also used. A landscape – e.g. a desert – has certain properties like how much food is there likely to be found, what abilities for locomotion is needed (to pass a lake, the agent has to be able to swim). Further, the energy needed to move within it differs from landscape to landscape – a swamp

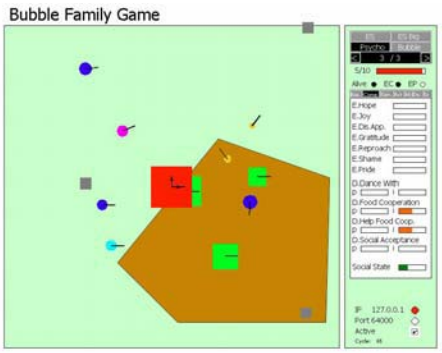

Figure 1 - 2D Interface

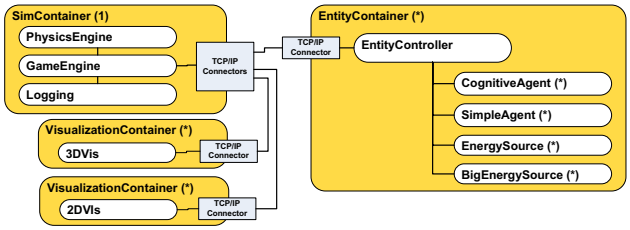

Figure 2 - Basic Concept of the Simulator

is more costly than the desert. The obstacles can be of different types like rock, wall, or tree. Additionally, each obstacle and landscape can be marked as unique. Hence, a once observed specially shaped rock can be recognized again.

Figure 1 shows the 2D interface of the simulator. On the left side, the elements described above are shown – the little gray blocks are the obstacles, the red and the green squares are the energy source, the brown filled polygon is a landscape, and the small circles filled with different colors are the agents. Each color symbolized a social group.

On the right side, the internal state of each agent and each energy source can be observed. Detailed information about what they have decided to do and why they have decided it can be found here.

The internal concept of the simulator is depicted in Figure 2. Each yellow block is an independent unit. The SimContainer provides the core functionality, whereas inside the EntityContainer the different agents are processed and the VisulizationContainers' task is to display the results and provide a user interface. The communication between the units is realized using a TCP/IP based message system. This enables the usage of several computers in the case of computation power bottlenecks. All units except the SimContainer can have multiple instances.

The SimContainer itself is divided into the GameEngine, the PhysicsEngine and a logging functionality. The GameEngine is responsible for loading the configuration of the simulation run, the synchronization with the connected EntitiyContainers, and to process each simulation cycle.

The PhysicsEngine detects and handles collisions between agents and obstacle or agents and energy sources or agents and agents. These collisions have to be handled differently. Between agent and obstacle, the agent gets hurt (thus, its energy is reduced) and is reset to its previous position. If an agent collides with an energy source, the agent has the possibility to consume a certain amount of energy from that source. A collision between agents will result in energy reduction for both of them and both are reset to nearby positions.

Logging provides next to the simple functionality of recording almost every value, tools for statistical analyzes. When doing several runs with different set of parameters, these tools are useful to select the best set of them.

The EntitiyContainer consists of an EntityController and a set of agents or energy sources. While an already instantiated agent can perfectly well communicate with the GameEngine on its own, a special unit to create and to remove them is needed. This task is done by the EntityController. Further it works as a

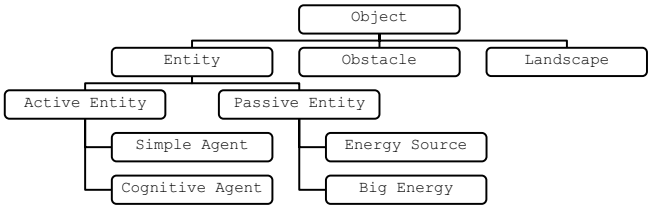

Figure 3 - Object Tree

watchdog, removing agents which need too much time to run their program.

The VisualizationContainer provides an abstraction layer to the used visualization. Chapter VI gives an overview about the functionality which is provided by this unit.

IV. Agent Framework

The reason why agents and energy sources are within one container is that both have a lot of functionality in common. The need some sensors, a communication with the server, have a shape, etc.

Figure 3 shows how within the BFG simulator the entities and obstacles are related. Each object has properties like shape and position. All entities are objects which have need processing power during the simulation run. Obstacles and landscapes are, as soon as they have been created, totally passive. All entities have additionally in common that they have sensors and energy. A further distinction is whether they have a decision unit or not. Passive entities are usually not able to move and the built in logic decides whether or not another agent can consume its energy. For example, a big energy source can only be consumed by two or more agents cooperatively. This is decided within the passive entity "Big Energy".

The logic needed for passive entities and simple agents can easily be defined using state charts from AnyLogic. Figure 4 shows the implementation of a simple flee pattern. This is especially useful for interdisciplinary project to visualize the implemented routines to people with no computer scientific background.

Active entities finally are including everything a cognitive agent needs. Different modules supporting complex decision making units are available. The provided functionality includes abilities needed to pass certain landscapes, a set of different sensors, tools to handle internal values, basic movement, special actuators like communication, and - to simulate complex behavior patterns - predefined routines (these are implemented using state charts too).

V. Sensors

As mentioned above, different types of sensors are available for the entities. Each sensor is implemented by providing an interface on the entity side and the evaluation on the

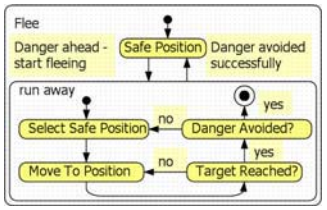

Figure 4 - Classification of Objects within the Environment

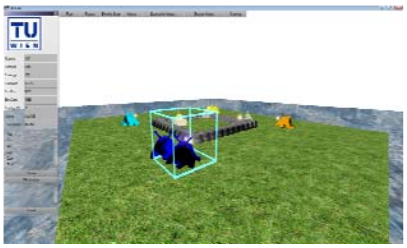

Figure 5 - 3D View on the Simulated Environment

GameEngine side. The existing sophisticated sensor frameworks are focusing on realistic modeling, but they are not providing the high level symbol processing needed by this simulator. Each sensor perceives information from the environment and passes it to the agent using symbols wherever appropriate.

Three different groups of currently implemented sensors can be identified: special purpose sensors needed to provide certain behavior without the necessity to implement complex algorithms, sensors needed to provide the agents with full world information if needed, and finally the typical set of sensors for perception.

The special purpose sensors are: AboveEnergySource, AboveLandscape, BelowEnergyConsumer. All three of them are returning if the agent is at the moment above or below e.g. an energy source.

To gain full world information the sensors OmniVision and RealPosition can be used. OmniVision provides information on every object within the environment and RealPosition returns the x and y coordinates of the agent within the world.

The perception sensors are

- Acoustic: perceive messages from other agents – has a maximum distance.
- Bump: active if the agent has collided with another agent or an obstacle.
- Odometry: the distance the agent has moved during the last simulation cycle.
- Proximity: the agent is informed of the number of other agents and energy sources within a certain diameter. No detailed information on their relative position is provided.
- Radar: every object which is within the cone of sight is returned. Due to performance reasons, partially or hidden object are returned too. Thus, this sensor is called radar and not vision.
- Energy/Stamina: internal body values like energy and stamina can be measured using this sensor.

VI. User Interface

The medium-term aim insists in simulating hundreds of entities by the BFG. As the simulation complexity arises it seems to be impossible to verify the AEAs' actions and reactions as well as their internal state by analyzing columns of figures. Hence, the BFG is visualized in 2-dimensional as well as 3-dimensional form. The visualizations ease the presentation of results and additionally provide a user interface to enable user-system interactivity.

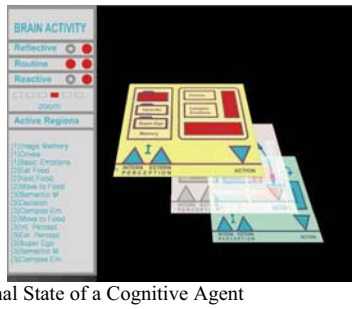

Figure 6 - Internal State of a Cognitive Agent

Required data, like object positions, internal states, and user commands, is received from the simulator kernel. The 2D environment uses graphic elements provided by Anylogic. To be able to create a powerful 3D visualization other libraries which are not based on java have to be used. Thus, the 3D component has to be outsourced. Therefore the data transfer is completed over an Ethernet connection. A communication protocol defines the regulation for the data transfer and defines the byte-stream messages. Because the protocol doesn't provide an error correction, the TCP/IP standard is used as underlying network protocol. As the simulator is used in a high-bandwidth providing LAN (Local Area Network), the speed-disadvantage in comparison to the connectionless UDP (User Datagram Protocol) standard is negligible.

The user interface provides three main features of interactivity. The first feature is the possibility to intervene the simulation run. The user is able to move to 'pause' mode or 'step forward' mode, and to load or reload new scenarios. As Anylogic state charts are not serializable and therefore do not provide methods for setting predefined states, rewind and save modes were not implemented.

Second, variations of internal AEA parameters like energy-level or miscellaneous emotional values can be conducted. Third, the BFG setting is modified in the course of a simulation run by repositioning entities and obstacles. Therefore a drag and drop mechanism has been implemented. BFG settings are varied during the simulation run in order to test the AEAs' ability of accommodating to environment variations.

The same way, the standard 3D visualization is connected to the BFG, additional 2D or 3D interfaces can be implemented. Figure 6 shows a view on the internal state of a cognitive agent which has a layered decision making unit.

VII. Application

A. New ARS-PA model

Back in 2003 the ARS project was launched. This project has the aim to work out a decision making unit for building automation control systems. The system should be able to deal with a high amount of sensor data and react on unforeseen situations. The only system that is known to fit these requirements is represented by the human mental apparatus. In order to design a system with the human mental apparatus as archetype, cooperation between different areas of science like psychoanalysis, neuropsychoanalysis, psychology and neurology has been engaged. During the development phase of the first ARS-PA model it became clear that there exist a

number of conflicts between the different theories, which would lead to conflicts in the resulting technical model too – therefore a decision for one basic concept was made. As Sigmund Freud's second topographical model provides a functional description of the human mental apparatus that can be transferred to technical terms, it became the model of choice. The second topographical model divides the human mind to three modules – ID, EGO, SUPER-EGO [15]. The ID represents the source of the human mind till the Ego and the SUPER-EGO evolve. It serves as connection between physiology and the mind. Therefore it deals with bodily drive demands that tend to get fulfilled as fast as possible. On the contrary the SUPER-EGO, which organizes bids, bans and a reward system, represents the antipol to the ID. Occurring conflictive situations are handled by the EGO that has to fulfill mediation tasks between SUPER-EGO and ID on the one side and the environment and the mental apparatus on the other. In contrast to a common technical system the human mind allows conflictive situations. Even this attribute sounds contradicting to common technical systems it tends to provide a new approach for modeling intelligent systems that is worth to be investigated.

As it is shown in Figure 7 it has to be differentiated between the terms inner world and outer world. While the outer world covers the subject's environment and its body, the inner world accords to the mental apparatus. Interfaces between both parts are defined in order to receive and evaluate sensor values on the one hand and provide a connection to motor driven functions on the other.

Further work focuses on the functional description of the modules EGO, ID, and SUPER-EGO. By the use of a Top-Down approach the model is transferred to a technical description. Engineers together with psychoanalytic advisors analyze the model by its functions. Parts that are not covered or described in detail by Freud's model are completed by matching psychoanalytic theories.

This process results in a technical model for a decision making unit influenced by a psychoanalytic model of the human mind. In order to investigate the model's behavior a simulator – the BFG – has been programmed. The BFG provides the simulated world and the AEAs' body – the outer world. The inner world is represented by the new ARS-PA model that autonomously controls every AEA body. In order to retrace the agents' behavior and the model's accuracy a number of test-cases have been defined.

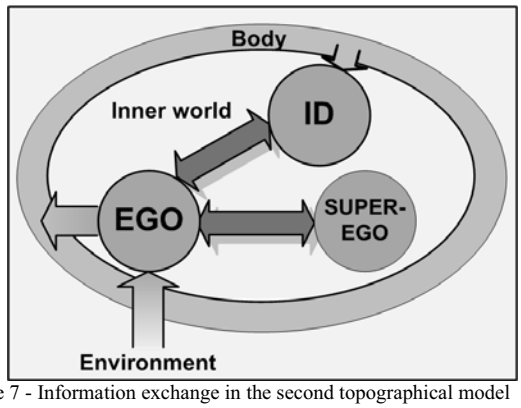

Figure 7 - Information exchange in the second topographical model

B. Test-cases revisited

In [14] we have defined three different test-cases which should be part of the AEAs' artificial world. They are entitled as "ask for a dance", "cooperation for food", and "call for help". They aim at different social abilities of the agents.

Other than the last two, the first one purely aims at social acceptance. If no urgent need for energy or help during a dangerous situation is present, agents can decide either to explore the environment or to engage other agents in social activities. AEAs with a high willingness to join such activities might gain a higher social reputation. This is useful in the other two test-cases.

"Cooperation for food" applies for situations where an agent wants to consume a large energy source. In this artificial world, large sources cannot be consumed by a single agent. At least a second agent is required to make it consumable. Thus, the agent has to call a second one for help. If another agent receives the call, it has different possibilities to act: be benevolent (always help), help if itself needs energy too, help with the prospect of reciprocity in future, or do not help at all. If the agent decides to help its social reputation will raise for the helped agent – regardless the true motivation.

The third test-case – "call for help" – describes the situation when an agent is threatened by e.g. agents from a different group. An agent answering this call has no possibility to gain a direct benefit like energy. The only profit lies within the defending of a member of its kin and the prospect of the same favor in a similar situation in future. The higher the social reputation of the requesting agent, the more reputation gain for the helper. The more dangerous a situation is, the more likely it is that agents decide not to go in for a fight. But if the requesting agent's social reputation gives the prospect of a large gain this decision might be reconsidered. Thus, social reputation gained by non productive activities like dancing and helping others in non dangerous situations pays off in such emergency situations.

While these three test cases focus on the social abilities of the agents, we work on new test cases aiming to verify the emotional performance of the AEAs currently. For example, is the emotional level of the agent corresponding to the perceived situation?

C. Comments

When trying to transform theories from psychoanalysis into technical feasible terms regarding building automation systems, one cannot do this directly. In this case questions like "what feelings does a kitchen have?" without knowing what feelings are from a technical perspective have to be answered in the beginning. To circumvent this, an intermediate step with more "human-like" creatures is needed. For this first step it is only necessary to translate the terms from the psychoanalytical into the technical domain.

This simulator helps with this translation – it easies the interpretation of the artificial life forms' behavior. Thus a translation back into the domain of psychoanalysis is possible. The resulting observations can be used to refine the

psychoanalytical model. This method is not restricted to psychoanalysis but can also be used for any other human or social science focusing on the human mind.

VIII. Discussion and Outlook

The presented simulation of a game of artificial life provides a suitable platform for developing a psychoanalytically inspired reasoning unit. The modular design allows adding different types of agents and sensors. Prototypes of agents are developed by using AnyLogic statecharts. Different visualization types can be used to view and alter the current situation. Special care has been taken to display the internal state of cognitive agents in a way that scientists from other domains like psychology or psychoanalysis can easily access and interpret this information. Thus, the simulation environment supports the interdisciplinary research which is required by the project ARS. Due to the fact that the different modules are connected via Ethernet, the whole game of life can be distributed to different computers.

Further development has to focus on defining a higher number of scenarios and test-cases in order to evaluate the model. While the first set has been defined by engineers, this work has to be continued by researchers from social sciences.

Another issue is performance. The selected simulation platform provides tools like state charts, graphical user interface elements and message passing. The hardware requirements increase, by the number of entities. Thus, a different approach is needed for larger setups. We are currently working on an interface which allows to use a platform like Mason, and connect it to an entity container similar to the one developed for the BFG.

Another important issue we are working on is the (semi)automatization of the verification test-cases. Currently, a human operator has to observe the simulation and decide whether a certain test-case has occurred or not. Afterwards, the different occurrences have to be analyzed. While the analyzation task is due to the nature of the ARS project difficult even for humans, at least the detection should be possible to be replaced by an automatic observer.

References

[1] C. Rösener, R. Lang, T. Deutsch, and A. Gruber, "Action planning model for autonomous mobile robots," *Proceedings of 2007 IEEE International Conference of Industrial Informatics*, pp. 1009-1014, 2007.

[2] M. Toda, "Man, Robot and Society," *Martinus Nijhoff Publishing*, 1982.

[3] W. C. Ho, K. Dautenhahn, and C. L. Nehaniv, "Autobiographic Agents in Dynamic Virtual Environments - Performance Comparison for Different Memory Control Architectures," *IEEE Congress on Evolutionary Computation*, pp. 573-580, 2005

[4] Charles M.Macal, Michael J.North, "Tutorial on the Agent-Based Modelling and Simulation", *Proceedings of the 2005 Winter Simulation Conference, pp. 2-15*, December 5-7 2005

[5] Steven F.Railsback, Steven L.Lytinen, Stephen K.Jackson, "Agent-based Simulation Platform: Review and Development Recommendation",*Simulation Councils Inc. vol.82, Issue 9, pp. 609-623*,September 2006,

[6] Nelson Minar, Roger Burkhart, Chris Langton, "The Swarm Simulation Systems: A Toolkit for Building Multi-Agent Simulation", *Technical Report 96-04-2*, Santa Fe Institute, Santa Fe, NM.

[7] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, Gabriel Balan, "Mason : A Multiagent Simulation Environnent", Pages: 517 - 527, July 2005

[8] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, "Mason: A New Multiagent Simulation Toolkit", *Proceedings of the 2004 SwarmFest Workshop*.

[9] Collier, Nick, "Repast: An Agent Based Modelling Toolkit for java", http://repast.sourceforge.net

[10] M.J.North, T.R.Howe, N.T:Collier, J.R:Vos, "The Repast Simphony Runtime System", *Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, Argonne National Laboratory, Argonne*, IL USA ,October 2005.

[11] F. Zúñiga, F. Ramos, and I. Piza, "GeDA-3D Agent Architecture", *Proceedings of the 2005 11th IEEE International Conference on Parallel and Distributed Systems*, 2005.

[12] F. Ramos, F. Zúñiga, and H. Piza, "A 3D-Space Platform for Distributed Applications Management*", International Symposium and School on Advanced Distributed Systems 2002*. Guadalajara, México, 2002.

[13] B. Palensky, "From Neuro-Psychoanalysis to Cognitive and Affective Automation Systems," Ph. D. dissertation, Vienna University of Technology, Institute of Computer Technology, 2008.

[14] T. Deutsch, H. Zeilinger, and R. Lang, "Simulation Results for the ARS-PA Model," Proceedings of 2007 IEEE International Conference of Industrial Informatics, pp. 1021-1026, 2007.

[15] S. Freud, "Das Ich und das Es, Studienausgabe. Psychologie des Unbewußten," *Fischer Taschenbuch Verlag GmbH*, 1975.

Tobias Deutsch (M'06) was born 1975 in Vienna, Austria. After graduating at the HTL Ungargasse, he studied Computer Science at the Vienna University of Technology in 1996 and completed his degree in March 2007. Since October 2005 he is a staff member of the Institute of Computer Technology. His current research focuses on cognitive intelligence, building automation and autonomous agents on the Vienna University of Technology at the Institute of Computer Technology.

Tehseen Zia was born 1981 in Sargodha, Pakistan. After graduating at the Punjab University of Pakistan, he studied Computer Science at University of Engineering and Technology Taxila, Pakistan and completed his master degree in 2003. Since April 2005 he is a faculty member at the Department of Computer Science, University of Sargodha. Currently, he is a PhD student at Vienna University of Technology, Institute of Computer Technology. His current research focuses on cognitive modeling and multi agent systems.

Roland Lang (M'06) was born 1978 in Vienna/Austria. He finished his studies of electronics, especially control engineering and telecommunications in June 2001 at the University of Applied Sciences Technikum Wien (Fachhochschule). He is currently employed at the Institute of Computer Technology and has gained experience on cognitive automation (within the project ARS Artificial Recognition System). He is IEEE Member, Organizing Committee member of the INDIN2007 and Secretary of the IEEE Student Branch, Section Austria.

Heimo Zeilinger was born 1980 in Judenburg/Austria. From 1999 to 2003 he studied at the university of applied science for electronical engineering specialized in telecommunication services and computer techniques. After he graduated in June 2003, he started studying at the Technical University of Vienna. Since November 2006 he is project assistant at the Institute of Computer Technology focusing on cognitive intelligence and building automation.