MASIM: A METHODOLOGY FOR THE DEVELOPMENT OF AGENT-BASED SIMULATIONS

André M. C. Campos
Anne M. P. Canuto
Jorge H. C. Fernandes
Eliane C. M. de Moura
Informatics Applied Mathematics Department
Federal University of Rio Grande do Norte, Brazil
{andre,anne,jorge,eliane}@dimap.ufrn.br

KEYWORDS

Multi-Agent Simulation, Development Methodology.

ABSTRACT

This paper presents the general aspects that motivated the construction of the MASim methodology, aimed for development of agent-based simulations. MASim employs features common to the development of agent-based software as well as to the development of simulation models. MASim is described in terms of agent-based concepts. It also borrows concepts used in mainstream software engineering process frameworks, defining workflows where users, simulation modelers, software developers, testers and experts of the simulation domain collaborate with the purpose of streamlining the development and reuse of simulations and agent components.

INTRODUCTION

The development of agent-based software has been largely increased in the last years (Deravi et al. 2003; Canuto et al. 2003). Examples of successful application of agent-based software can be found in electronic commerce, industry, web, etc. As a consequence, several methodologies for developing agent-based software have been proposed to help workers in developing efficient agent-based software. Most of the existing methodologies, such as CommonKADS (Schreiber et al. 2000), GAIA (Wooldridge et al. 2000), TROPOS (Bresciani et al. 2003), MASe (Wood 2000) etc., are mainly focused on the specification of concurrent software components (agents), describing its roles, goals, functions, communication etc. Those methodologies are supposed to be suitable for most agent-based applications.

On the other hand, simulation systems provide the capability of deriving statistically meaningful conclusions for a computer generated synthetic world. Simulations are crucial in providing advice to natural resource managers, for training and management purposes. The use of the multi-agent paradigm for building simulation leads to a more powerful and user-friendly computer environments often based on parallel processing, being also able to model the spatial data (positions in the environment) of a system, hardly represented in some mathematical modeling approaches. Multi-agent simulations are becoming increasingly relevant in the simulation field (Campos and Hill, 1998), and can be applied to various areas, to simulate social systems (Gilbert and Troitzsch, 1999).

However, the development of simulations is not as usual as information systems development, given that in the design of a simulation it is necessary not only to model the software components themselves, but also the simulation model behind the application. The process of specification, development and calibration of simulation models may require several iterations, demanding stronger approaches to the tasks of verification and validation. Thus, a methodology for simulation should adapt standard software development methods including V&V processes.

As a solution for the aforementioned problem, this paper proposes a methodology to develop agent-based simulations, named MASim – Multi-Agent Simulation Methodology. MASim employs aspects common to the development of agent-based software as well as to the development of simulation models. MASim is described in terms of agent-based simulation concepts (agents, roles, resources, dependencies, interactions, etc). It also borrows concepts used in mainstream software engineering process frameworks, defining workflows where users, simulation modelers, software developers, testers and experts of the simulation domain collaborate with the purpose of streamlining the development and reuse of simulations and agent components. The use of workflow-based processes emerged from the need to better organize the development process of simulation environments. MASim has been applied to the design and implementation of a simulation environment dealing with human organizations, referred to as SimOrg.

Proceedings 16th European Simulation Symposium
György Lipovszki, István Molnár © SCS Press, 2004
ISBN 1-56555-286-5(book) / ISBN 1-84233-106-x(CD)

The remainder of this paper is divided as follows: Section 2 describes the state of the art in software development methodologies, focusing on the main agent-based development methodologies. It also presents the main differences between methodologies for agent-based development and methodologies for developing simulation models. Section 3 presents the general idea of the proposed methodology. Afterward, roles and phases of the methodology are exposed in Sections 4 and 5 respectively. The last section is dedicated to final remarks on this work and presents further work.

BACKGROUND

Simulation and reality are at the extremes of a spectrum of systems, end there is a myriad of intermediate situations between them. In general, the goal of a simulation is to model a real system whose nature is sometimes marked by concrete (vs. informational or abstract), physical (vs. symbolic), analog (vs. digital), or continuous (vs. discrete) aspects. In order to obtain this simulated representation, the system must be mapped to the discrete computational domain. This differs from the goal of usual information systems, which is to enhance the manipulation of information already formalized in human-centric organizations. Given this need for translation from a natural to a synthetic domain, simulation models are developed through several iterations, each one producing an enhanced model of a system, that is defined, implemented, verified and validated. This cyclical process goes on until the model satisfies the objectives of the model user (as Minsky says, "*To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A*" (Minsky 1965)).

Thus, the approach of an iterative simulation development differs from the approach of most information systems engineering processes. Indeed, a simulation is also a software system. The point is that, while in the simulation development process, iterations aim to evolve from a single initial model until a useful (consistent) one, the cyclical iterations in the information systems engineering process aim to achieve new and interrelated system and software functions. This is achieved by an incremental and modular construction of pre-planned chunks, usually driven by user scenarios, like use cases or user stories. In other words, in the development of information systems, the various functions to be developed are generally well known and have a representation close to the information that must be used in the user organization. This way, the information systems usage tends to provoke changes over its user organization. Conversely, when developing simulation applications, the simulation model outcomes are meant to give indications on how close the model is approaching the real system that is of interest to the user. However, the usage of the simulation does not induce or precludes a change to the real system.

The success of a simulation is thus measured on how close it is from the real system it may mimic, while. On the other hand, the success criterion of an information system is how close it is to provoke enhancement on the manipulation of the information model in the user organization.

Methodologies for the development of Agent-oriented software

In this subsection, some of the existing agent-based development methodologies are described. The main advantages and disadvantages of each methodology are presented, considering its application in development of multi-agent simulation software.

Most agent-based methodologies are proposed for the development of information systems software. Due to this intended application, sometimes, these methodologies lack some important specific aspects of other specific types of applications. In analyzing the advantages and disadvantages of each methodology, this paper aims to understand the degree of suitability of these methodologies for the specific domain of multi-agent simulation.

GAIA

Gaia is a methodology for agent-oriented software analysis and design. The Gaia methodology is both general, in that it is applicable to a wide range of multi-agent systems, and comprehensive, in that it deals with both the macro-level (societal) and the micro-level (agent) aspects of systems (Wooldridge 2000). Gaia is intended to allow an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. Analysis and design can be thought of as a process of developing increasingly detailed models of the system to be constructed. During the analysis phase, role and interaction models are created and agent, services and acquaintance models are created during the design phase.

The possibility that different agents may be implemented using different programming languages, architectures, and techniques as well as the fact that it deals with both the macro-level (societal) and the micro-level (agent) aspects of systems are very important aspects of this methodology. However, Gaia does not explicitly attempt to deal with systems in which agents may not share common goals (self-interested agents) and in conflict situation. Furthermore, the environment modeled by Gaia is closed and static. For a multi-agent simulation, these aspects are very important and this is a drawback of this methodology.

Nevertheless, even if a multi-agent simulation requires some features not provided by the Gaia methodology, it

could be observed that it is essential only in the macro-level. The idea of roles and its junction into agent types can be very useful for the design of the agents which compose the simulation model.

MASe

The MASe (Multiagent Systems Engineering) methodology has focused on the development of practical agents (Wood 2000). It defines multi-agent systems in terms of agent classes and their organization. MASe defines its organization in terms of which agents can communicate using conversations. There are basically two phases in MASe: analysis and design. The first phase, Analysis, includes three steps: capturing goals, applying use cases, and refining roles. In the Design phase, it transforms the analysis models into constructs useful for actually implementing the multi-agent system. The Design phase has four steps: creating agent classes, constructing conversations, assembling agent classes, and system design.

The main advantage of this methodology is its simplicity since it defines clearly how a designer should act at any moment of the process of the development of a software However, MASe lacks some abstractions which are important in the modeling of agent-based software. In this case, it is very similar to an object-oriented methodology and this is an important drawback of this methodology.

TROPOS

Tropos is a software development methodology allowing a designer to exploit all the flexibility provided by agent-oriented programming (Bresciani et al 2003). TROPOS is intended to support all analysis and design activities in the software development process, from application domain down to system implementation. There are five main development phases of the Tropos methodology: Early requirements, late requirements, architectural design, detailed design and implementation.

One of the main features of Tropos is the crucial role played by the early requirements, which has been added to the phases of an agent-based development process. It helps in the development of agent-based software, including multi-agent simulation. In addition, the way in which the phases are distributed allow the designer to have a great understanding of the system as a whole. Finally, the fact that the environment and system models are being built in an incremental way, being refined and extended at each step, is very attractive for the development of multi-agent simulation. The main drawback of this methodology is the complexity behind the models to be created, which becomes even worse in complex applications.

Due to the aforementioned facts, it can be concluded that Tropos is a very attractive methodology for developing also multi-agent simulations, mainly in a macro-level perspective.

Useful characteristics from software engineering methodologies

The most known generic framework for software engineering is the Unified Process (Jacobson 1999). The unified process is marked by the adoption of the following principles: Iterative; Risk-driven; Requirements based; Architecture based; Visual modeling-based; Continuous quality assurance; and Change management

Furthermore, the Unified process states that there are nine specialization areas, called disciplines, that involve the work of diverse experts. The areas are: Business modeling; Requirements Analysis and Design; Implementation; Test; Deployment; Environment; Project Management; and Configuration and Change Management

The activity inside all disciplines is marked by a structured work around the concepts of roles, tasks and workflows. Some of these disciplines are closely related to technical aspects of simulations and agents, namely: Business modeling, Requirements, Analysis and Design, Implementation and Test.

The point is that in order to streamline the development of simulations, it is important to provide a clear set of paths for development and validation of systems. Given that the methodology is aimed to guide the work of simulation developers, working inside an organization, it is suitable that the methodology has to lead towards an application on the same concepts employed in the development of the simulation.

MASIM: A METHODOLOGY FOR DEVELOPING MULTI-AGENT SIMULATIONS

MASim is a process framework specifically focused on the development of multi-agent simulations. Before presenting the methodology it is worth commenting on its purposes and the type of applications it tackles. As multi-agent simulations are well-suited for studying complex-adaptive systems (Gilbert and Troitzsch 1999), the proposed methodology is supposed to be appropriated to develop large-scale applications for simulating such type of systems. It is also intended to provide, through the simulation, foundations for modifying the system itself.

Agent concepts are used in the methodology mainly to model the process roles, the interactions, the dependencies, etc. It considers the individuals running the methodology as components of a multi-agent system, which compose indeed a real organization.

MASim consists of five phases, presented below:

- **The requirements phase**, which consist in identifying the scope of the simulation model as well as the needs of the application for handling such a model;
- **The modeling phase**, which aims to construct an abstract model of the system, presenting its elements and the dependences between themselves and the system as a whole;
- **The architectural and design phase**, which translate the model into a set of concrete specifications able to be easily implemented. This phase also intends to identify patterns to be reused in other simulations of the same domain;
- **The implementation phase**, where the specifications are coded in programming language;
- **The verification, validation and accreditation (VV&A) phase**, which confronts the overall simulation results with the real system and determines if the simulation application is suitable for the initially required purposes.

The whole process is cyclical, as most of the modeling methodologies. However, it also goes forward and backward through the phases, verifying the codification of a specification in a higher level of abstraction and validating the proposed specification, as shown in the Figure 1. This idea of several V&V phases follows the pragmatics observations of Swartout and Balzer, who says that any specification *S* might be seen as an implementation of another specification *S'* of a higher level of abstraction (Swartout and Balzer 1982). Nonetheless, the last phase is explicitly concerned with V&V in order to confront the produced simulation software with the requirements initially exposed. In other words, it analyzes the implementation of lowest level of abstraction in the perspective of the highest-level specification.

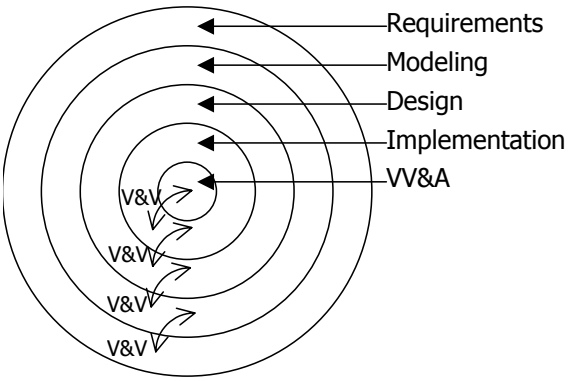

Figure 1 - Phases of the methodology and their internal verification and validation

MASim encourages reusability in the architectural and design phase by the architect role. The architect is responsible for defining structures patterns for the whole application and for specifying how those patterns (e.g organizational structures and interaction protocols) must be implemented.

MASIM ROLES

In large-scale systems development, it is impractical that the same individual performs all tasks in different phases of the system development. Tasks requiring different abilities must be executed by individuals fulfilling their specific requirements. In order to optimize this process according to the abilities required in its activities, MASim defines seven roles that different individuals might play.

Those roles are also introduced in order to better characterize the objectives and responsibilities of each individual in the process. Nevertheless, for small-scale simulation applications, a unique individual might play different roles. The same happens when developing non agent-based systems. A software process, like RUP, identifies several roles, but this does not mean that there is a person for each role. A typical scenario for the development of simulation systems consists at the minimum a domain expert and a developer. However, if one consider the development of large-scale simulation, it is necessary to involve several individuals, each one playing a specific role in the software development process. An example is some simulation games like SimCity, where several individuals are involved.

MASim preview they following roles:

- **End-user**, an individual (or organization) for whom the simulation application is under development. The end-users are responsible for setting up the requirements for the application. They should target the objective of the simulation and, defining what they expect from the simulation as results;
- **Domain expert**, an individual (or several ones) who has deep knowledge about the domain being simulated;
- **Modeler**, who is responsible for, along with the end-user and the domain expert, to construct the simulation model;
- **Software architect**, who is responsible for defining software patterns and/or simulation model components that might be reused in other simulations as well as defining the whole simulation framework;
- **Designer**, who is responsible for transforming the simulation model constructed by the modeler into a software design, able to be more easily implemented;
- **Developer**, who is responsible for implementing the model designed by the designer;

- **Tester**, who is responsible for verifying and validating the application according to the pre-established scenarios.

Figure 2 shows the whole process of MASim, involving all the roles previously described. The diagram, an adapted version of the workflow diagram of the Unified Process (Jacobson et al. 1999), shows the process in the following steps:

1) The process begins when the end-user and the domain expert describe the organization and what is expected from the simulation through a collection of scenarios;
2) The scenarios are then used by the modeler and the domain expert to construct the model of the organization (macro level), its individual elements (micro level) as well as their interrelationships.
3) The architect identifies patterns in the previous model and defines standards for designing the simulation. They aim to achieve reusable components of software and simulation model.
4) The computational model is constructed by the designer from the organizational model and the standards previously mentioned. The main different between this model and the organizational one is the approach (concept vs. coding-oriented) and their level of detail.
5) It is then used by the developer to implement the simulation software, which it will be used by a tester in a process of verification and validation and by the end-user for accreditation.
6) The process starts a new iteration, taking into account the results from the current developed simulation.

The next sections present the phases of the methodology and how the previously mentioned activities are involved in each one of them.

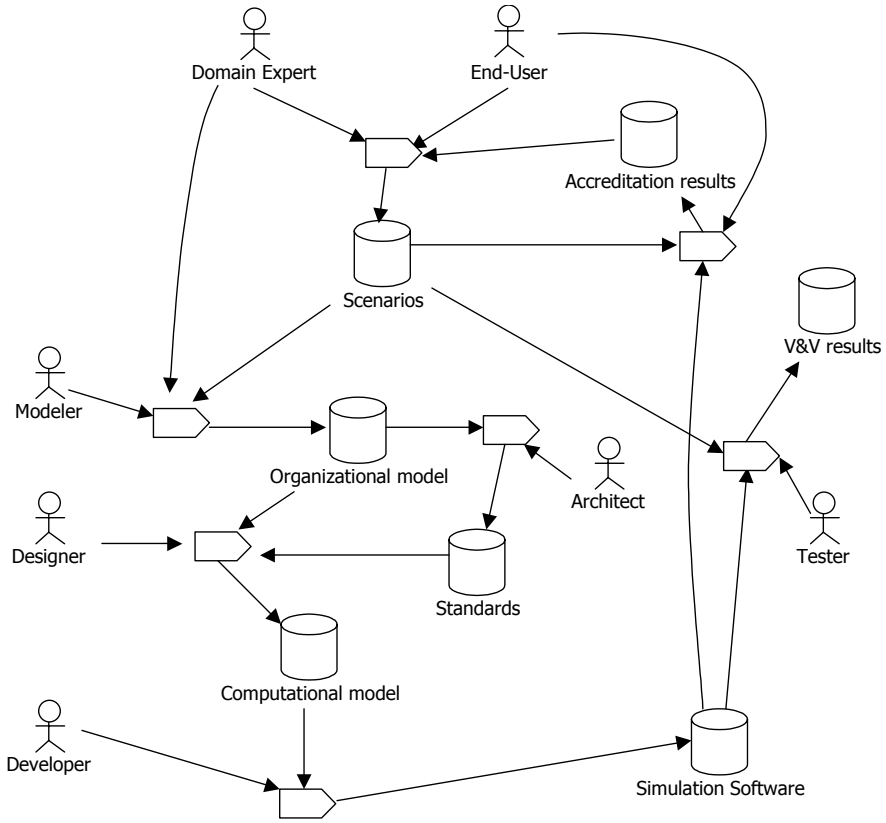

Figure 2 – Workflow showing the activities performed by individuals playing different roles

MASIM DEVELOPMENT PHASES

Requirements phase

All requirements in MASim are expressed as scenarios. As in information system requirement analysis, those scenarios also describe what is expected from the application being developed. However, instead of being based on which changes the application must introduce in the real system, the requirements in MASim are based on what is expected to reproduce from the real system. The observations provided in the scenarios are used to define the scope of the simulation model and application.

As shown in Figure 2, the scenarios might come from two sources: the end-user, who knows what the application must produce, and the domain expert, who knows how to produce. The scenarios are described in an informal way through a schema containing the involved elements, a description of the initial conditions of the elements, the events identified in the scenario case and a description of the conditions of the elements after the events. It is also possible to make some notation about the mechanisms responsible for the condition changes. The Figure 3 shows an example of scenario schema.

Scenario schema: *scenario name*	Version: *num*
Involved elements:	
Subsystems involved.	
Pre-conditions:	
Description of the initial situation of the elements	
Events:	
Description of events happening to the elements	
Pos-conditions:	
Description of the final situation, after the events	
Possible mechanisms:	
Description of the mechanism or the hypothesis about the element changes	

Figure 3 – Schema of a scenario form

The objective of such schema is not to extensively document all possible scenarios of the organization in order to provide all the information (as the unique input) for the modeling phase. The scenarios are mainly used to define the scope of the simulation model as well as the application handling such a model. It specifies what the activities in the modeling phase must focus on.

In the proposed methodology, requirements are also driven by scenarios describing what it was intended for the application. However, instead of presenting requirements based on changes (early and late requirements), the requirements are based on what it is observed from the real system. The observations compose several scenarios that are used to define the scope of the model.

The scope of the model is defined through four types of scenario. The first one specifies the boundaries of the system being modeled. It describes external events that influence the system as a whole and how the system reacts to them (exogenous events). The other types of scenarios, illustrated in Figure 4, are related to internal interactions in the system (endogenous events), describing:

- Macro-micro interactions: this type of scenario reflects how the behavior of the system as a whole changes the behavior of its subsystems.
- Micro-macro interactions: on the opposite way, this type of scenario aims to present how individual subsystem activities alter the whole system.
- Peer-to-peer interactions: this scenario identifies the dependencies between two or more subsystems and how they influence each other.

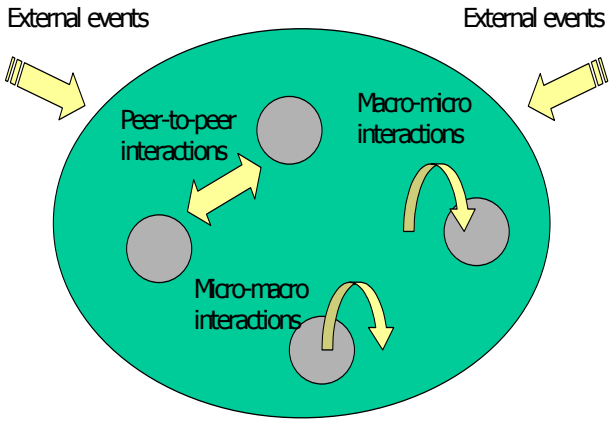

Figure 4 – Scenarios used for defining the scope of the model

Modeling phase

The modeling phase consists in identifying, from the scenarios previously specified and face-to-face interactions between the modeler and the domain expert, entities, groups, roles, tasks, activities and dependencies between those elements. Its main aim is to construct a model of the organization in a macro (society) and micro-level (individual).

Modeling activities

During the modeling phase, several *views* of the system are constructed, each one provided by a different modeling activity. They are:

- **Resource modeling**: It consists in identifying and modeling the existing roles and actors playing those roles within the system. It is important to emphasize that the roles defined in this activity are not the same as the ones presented in the previous section. While the former represents roles in the methodology, the latter represents roles in the system model. In MASim, roles are abstract concepts representing general activities performed by individual elements. Consider, for instance, that a human organization, like a business company, is being modeled. The company has roles such as CEO, directors, and so on. Those roles are positions with well-established objectives, activities and permissions (or restrictions). They will typically correspond to: individuals or groups (for instance, a department of a company). Roles are then defined in terms of: objectives, activities and permissions. The agent modeling will identify the agents which play one or more roles in the system. Actors can be viewed as concrete instances of the existing roles. There is no actor in the system without playing a specific role

within it. Furthermore, there is no one-to-one mapping between roles and actors. An agent might play several roles as well as a role might be played by several agents. However, agents playing the same role might behave in a different way to achieve the objective of such a role. It depends on the agent beliefs and behavior patterns (mental models).

- **Dependency modeling**: It consists in specifying the dependencies between roles as well as the activities performed by individuals playing different roles. Roles and activities dependencies provide a basis for modeling the organization structure. It defines, for instance, how hierarchical the organization is (role dependencies), how groups are constituted (role dependencies), and how centralized the decision-making processes are (activity dependencies).
- **Interaction modeling**: It consists in defining the set of protocols of interactions used by the actors for accomplishing the objective of their roles. While the dependency modeling activity models what is dependent for a specific activity, the interaction model details how a specific activity is performed. It points out the mechanisms used to transform the resources and how those transformations flow during the execution of the different activities.

Architectural and Design phase

In this phase, the conceptual model must be detailed according to the programming approach to be used in the next phase.

This phase is mainly concerned by the system architect and the designer. The former is responsible for defining the system global architecture according to previously developed simulation in the same domain. It must identify multi-agent architectural patterns for the software as well as for the simulation model, in terms of component reusability. For instance, consider that a simulation has been developed for the department of marketing of a company in which resources, activities, individuals and roles are similar to a simulation of another company which a simulation has already been developed (notice that the role of the end-users differs from the role of the developers, and so they – end-users – might be clients of a simulation development company – the developers). In this case, several agent-based components must be reused in order to facilitate the simulation development. Reusability, as a key concept of software engineering, is one of the strongest points of the MASim methodology.

The designer takes charge of the specification of the internal design of each component, using modeling languages like AUML (Bauer et al. 2001), where objects, agents, and agent communication protocols are well represented.

Implementation phase

MASim is intended to be independent of a programming paradigm. However, the approach presented here concerns the object-oriented paradigm due to the lack of enough mature agent-oriented programming languages. As consequence, most of the multi-agent applications currently developed is implemented in an object-oriented approach. Although the existence of several tools has been proposed to implement multi-agent simulations, like MadKit (Gutknecht and Ferber 2000), they fall down in the object-oriented paradigm as they are implemented in object-oriented languages (Java and Smalltalk respectively).

Other approaches might be suitable in case they provide a robust way for transforming the agent-oriented design into a computational language. Among the existing agent-oriented languages, Brahms (Sierhuis et al. 2000) seems to fulfill the requirements for developing a multi-agent simulation where the system is viewed as an organization.

Verification, Validation and Accreditation phase

Before a simulation model can be used with confidence by the end-user, it must be verified, validated and accredited. Those activities are performed in this phase of the methodology. As mentioned before, verification and validation activities are performed during the various phases, checking errors generated when implementing specifications of a lower level of abstraction (verification) and checking if the specification corresponds to the model of a higher level of abstraction (validation). However, this phase was introduced to link the first specification (requirements) to the last implementation (software).

Verification and Validation can be a complex matter. Indeed, depending on the characteristics of the model, they might become a hard task to perform. Several methods are proposed in the literature, including grounding, calibration and statistical comparisons. For such reasons, those activities are performed by an individual playing a specific role, the tester.

While the V&V process is performed by the tester, the accreditation activity is performed by the end-user. In fact, accreditation refers to evaluate how useful is the simulation application (and model) for the specific purposes it was targeted to. The only person able to execute this task is the end-user.

MASIM USE CASE: SIMORG

The proposed methodology has been used for the construction of applications and tools for simulation human organizations, through a project named SimOrg - *Simulation of human Organization*. SimOrg aims to define, implement to validate an agent-based computational model

able to represent complex organizational behavior arising from people interactions.

As SimOrg deals with the study of organizational behavior, various experts in organizational psychology are jointly working in the project as domain experts, tackling the following aspects:

- The process of planning, elaboration and evaluation of working flows in order to describe and systematizes the efficacy required in organizational positions and functions;
- The process of recruiting new collaborators, including the usage of evaluation methods and techniques, whose objective is to assist in the identification of most adequate candidates for well-defined functions within the organization;
- The process of moving people inside an organization taking into account the current context of the organization, the individual antecedents and perspectives as well as their psychological and motivational aspects.

The simulations provided by SimOrg will provide strong basis for the validation of theories dealing with the mentioned aspects, where profiles, processes and group dynamics must be analyzed. The final objective is to estimate how they can be defined in an organization in order to provide a well-functioning organizational structure with a better productivity in a long-term perspective.

The SimOrg project is starting its second year of execution. MASim is being used in SimOrg as a general methodological framework for producing the simulations required by such a project.

FINAL REMARKS AND FURTHER WORK

In this paper, we have described MASim, a methodology for developing multi-agent simulations. MASim proposes a system development framework employing common aspects of agent-based software development and simulation modeling field. For this, it borrows concepts used in mainstream software engineering process frameworks, defining workflows where users, simulation modelers, software developers, testers and domain experts.

MASim is based on the authors' background in developing multi-agent simulations (Hill et al. 2000) and is currently being used in the development of a simulation environment dealing with human organizations, referred to as SimOrg. The first year of the project execution gave us valuable feedback about the usefulness of the methodology. Nevertheless, the methodology has not been fully validated and several issues in the agent-based modeling practices are not supported yet. One of the main missing points in its current version is that it does not provide enough directives nor formal specifications for the organizational model. In next versions of the methodology, it is intended to fulfill this gap through the support to represent all the agent-based concepts described in Section 5.2. More specifically, it is intended to provide tools to formally describe roles, objectives, permissions, activities, agents, beliefs, behavior patterns, dependencies and interactions.

ACKNOWLEDGEMENTS

This work has the financial support of CNPq (Brazilian Research Council), under process number 552431/2002-8.

REFERENCES

Bauer, B., Müller, J., Odell, J. "Agent UML: A Formalism for Specifying Multiagent Interaction". In *Agent-Oriented Software Engineering*, pp.91-103. Springer-Verlag, Berlin, 2001.

Campos, A. and Hill, D., "An Agent-Based Framework for Visual-Interactive Ecosystem Simulations". In *Transaction of SCS*, 15(4):139-152, December, 1998.

Campos, A., "Perceptive agents: modeling non-intentional interactions". *Proceedings of the 5th World Multiconference on Systemic, Cybernetics, and Informatics*. Orlando, FL, 2001.

Canuto, A., Gottgtroy, M., Lucena, M., Bezerra, V., Medeiros Jr., J., Oliveira, A., "RetDat: A multi-agent architecture for the retrieval of information in heterogeneous databases". Proceedings of the 7th IASTED International Conference on Artificial Intelligence and Soft Computing, pp.321-326, 2003.

Deravi, F., Fairhurst, M., Guest, R., Canuto, A., Mavity, N., "Intelligent Agents for the Management of Complexity in Multimodal Biometrics". In *Journal of Universal Access in the Information Society*. Springer-Verlag. 2(4):293-304, 2003.

Gilbert, N. and Troitzsch, K.G., Simulation for the Social Scientist. Open University Press. 1999.

Gutknecht, O., Ferber, J., "MadKit: a generic multi-agent platform". Proceedings of the fourth international conference on Autonomous Agents, pp.78-79, 2000.

Hill D., Mechoud S., Campos A., Coquillard P., Gueugnot J., Orth D., Michelin Y., Christophe P., L'Homme G., Carrère P., Lafarge M., Loiseau P., Micol D., Brun J.-P. Decuq F., Dumont B., Petit M., Teuma M., "Modélisation de l'entretien du paysage par des herbivores en moyenne montagne: une approche multi-agents". In *Ingénieries – ETA*, 21:63-75. March, 2000.

Jacobson, I., Booch, G., Rumbaugh, J. *The Unified Software Development Process*. Addison-Wesley, 1999.

Minsky, M. “Matter, Mind and Models”. Proceeding of International Federation of Information Processing Congress, vol. 1, pp.45-49, 1965.

Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J. and Perini A.. “TROPOS: An Agent-Oriented Software Development Methodology”. In *Journal of Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers, 2003.

Schreiber, A., et al. *Engineering of Knowledge and Management: The COMMONKADS Methodology*, MIT Press. 2000.

Sierhuis, M., Clancey, W., Hoof, R., Hoog, R., “Modeling and Simulating Human Activity”. Proceedings of Autonomous Agents 2000 workshop on Intelligent Agents for Computer Supported Cooperative Work: Technology and Risks (Ed, Petsch, M.) Barcelona, Spain. 2000.

Swartout W., Balzer R., “On the Inevitable Intertwining of Specification and Implementation”. In *Communications of ACM*, 25(7):438-440. July 1982.

Wood, M. F. *Multiagent Systems Engineering: A Methodology for Analysis and Design of Multiagent Systems*, Air Force Institute of Technology – AFIT. Master thesis, 2000.

Wooldridge, M., Jennigns, N.R. e Kinny, D. “The Gaia methodology for agent-oriented analysis and design”. In *Journal of Autonomous Agents and Multi-Agent Systems*. 3(3):285-312. 2000.