The ADAPT toolkit-supported Engineering process for agent based applications

Rainer Herrler  
*University of Würzburg*

Christian Heine  
*University of Hohenheim*

Follow this and additional works at: http://aisel.aisnet.org/amcis2004

Recommended Citation

http://aisel.aisnet.org/amcis2004/214

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2004 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
The ADAPT toolkit-supported Engineering process for agent based applications

Rainer Herrler
University of Würzburg,
Chair of Computer Science 6
herrler@informatik.uni-wuerzburg.de

Christian Heine
University of Hohenheim
Chair of Information Systems II
heine@uni-hohenheim.de

ABSTRACT
Compared to software engineering the history of agent development is rather young. However a couple of process models (methodologies) for the development of agent based applications have been proposed in recent time e.g. Gaia, MaSE, or Tropos. The main requirements in agent development are cheap development costs and a high quality of the resulting systems. Suitable methodologies and powerful tools are needed in order to meet these requirements. Existing process models focus on certain development stages and often lack continuous tool support. In this paper, we introduce a continuous and toolkit-supported process model and for the development of agent based applications. We integrate existing modeling tools and provide conversion tools to bridge the gap between different development stages. The integration of simulation as a main part of the process model makes it well suited for agent development, especially if the specification of the problem is rather vague and evaluation of emergent phenomena is an important criterion.

KEYWORDS
software agent, toolkits, multi agent systems, simulation.

INTRODUCTION
The development of agent based systems is a very young research area and there is no methodology, which became commonly accepted till now. Problems in developing agent based system at the moment are:

- The gap between specification and implementation is very big and there are few receipts for transformation. The implementation often is object oriented and you have to deal with a change of paradigms.
- In contrast to conventional systems agent systems often imply dynamic interdependencies. So the global system behavior is difficult to predict. Sometimes the systems do not fulfill the customers’ requirements and expectations.
- Early requirements engineering is often done by domain experts and implementation by developers. Results have to be exchanged between these groups and documented.

However a couple of process models for the development of agent based applications were proposed in the recent years: examples are Gaia (Wooldridge et al., 2000), Prometheus (Padgham, 2002), Message/UML (Caire et al., 2001), Tropos (Mylopoulos, 2000) or MaSE (DeLoach, 2001). For an extensive survey and discussion of the methodologies we refer to (Oechslein, 2004). AUML (Bauer, 2001) has become a well known modeling paradigm for agent systems. Is extends UML for the purposes of developing agent systems. However AUML (like UML) is already very “technical” from the domain experts’ point of view and the gap between specification and implementation is still to bridge. To deal with the problems mentioned above we present a novel process model with the following properties:

- An incremental and (partly) evolutionary process guides the developer from requirements analysis to the deployment of the application.
- Support for domain modeling and business modeling is provided in the early stages of agent development (especially process and goal based models), because early requirements engineering is often done by domain experts and implementation by developers.
- The inclusion of visual modeling tools combines functionality with descriptive power. This supports the dialog between domain expert and developer.
- Tool support to transfer results from each modeling phase to the next is very important to bridge the gap between specification and implementation. It reduces the development costs and avoids errors from the manual conversion.
Multi agent simulation as an integral part of the process makes the system behavior predictable during the design phase. This is important, because agent based applications usually imply dynamic interdependencies.

The paper is organized as follows: In section 2 we give a quick overview on our process model. It describes the series of development steps and possible backward loops. In section 3 we describe the process stages more closely. This includes a description of the applicable tools, existing ones as well as novel tools. In section 4 we make some conclusions and describe our intentions for the future work.

SHORT OVERVIEW OF THE ENGINEERING PROCESS

In this section we will shortly introduce the process model for developing agent applications. It will be described in more details in the following section. The process model consists of the following cyclic steps depicted in Figure 1: requirements engineering, simulation experiments, application development, simulation based evaluation and practical evaluation.

During the requirements engineering stage domain modelers create requirement specifications, process models, goal based models of actors and the envisioned software system. These agents can be seen as an active component of the model, generating the global goal-oriented behavior of the overall system by interactions. Thus, it is necessary to point out where agents should be used, which responsibilities they possess, in which activities they are involved and which goals they have to pursue. To be able to answer the why's in business process modeling and thus also in early phases of requirements engineering the goals of the agents and their relations are very important.

In the next stage this abstract domain model has to be refined to a more concrete agent based simulation model. This necessitates modeling the agents including their attributes and behavior in a machine executable manner. The resulting simulation model may contain the agents of the environment as well as agent applications (software agents) located in this environment. The resulting simulation model is base for simulation experiments to find out dynamic effects based on interdependencies of single agent behaviors and to validate the overall system behavior.

Based on the experience and specification of the simulation model, the software agents of the desired agent application have to be implemented. Parts of the simulation model can intuitively transformed to software agents of a distributed application that provides user interfaces for interaction with real worlds agents and fulfills the system requirements.

The resulting agent system is designed to be applied in the real world but can as well be connected to the simulation model for evaluation purposes. With slight changes regarding the interfaces the simulation model of phase 2 can be extended to a simulation testbed. This is especially useful to test the software under realistic conditions and probably to adjust system parameters.

Figure 1: Toolkit-supported engineering process for agent based applications

The last stage is an evaluation in practice. Here we test, if the application fulfills the requirements of the users. This evaluation might show fewer problems compared to a traditional engineering process, because it was already possible to test
the functionality and to evaluate the agent application in simulation. Even so this practical test is very important, because mistakes in acquiring the requirements and creating the simulation model can lead to wrong evaluation results.

TOOLKIT FRAMEWORK

Several engineering tools are available to support the previously described development process. This includes visual modeling tools, developed by us, as well as conversion tools to reuse output from common modeling and development tools. Some of the tools are rather mature, others are just in development. In this section we describe this tools and their stage of maturity according to the relevant process step. Development stages are numbered by arabic numbers (1,2,3,4,5) and conversions are numbered by roman numbers (I, II, III, IV).

GOAL ORIENTED REQUIREMENTS ENGINEERING (STAGE 1)

The first step of Figure 1 requires explaining why it is necessary (in our view of requirements engineering) to analyze goals for agent based applications. This leads directly to arguments of specific methodologies for goal-oriented requirements engineering and goal-oriented business process modeling. The publications about goal-oriented requirements engineering state that: “... the research efforts so far were in the what-how range of requirements engineering. The requirements on data and operations were just there; one could not capture why they were there and whether they were sufficient for achieving the state that: “... the research efforts so far were in the what-how range of requirements engineering. The requirements on data and operations were just there; one could not capture why they were there and whether they were sufficient for achieving the higher-level objectives that arise naturally in any requirements engineering process” (Lamsveerde, 2000, 7). But already 1977 Ross and Schoman defined the way of requirements engineering as, ... a careful assessment of the needs that a system is to fulfill. It must say why a system is needed, based on current and foreseen conditions, which may be internal operations or an external market. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed...” (Ross and Schoman, 1977, pp. 6). To be able to answer the why’s in business process modeling and thus also in early phases of requirements engineering we focus on goals and their relations. In the view of goal-oriented requirements engineering methods the focus is on:

- elicitation of high-level goals to be achieved by the envisioned system (Anton, 1996, pp.136-144; Bubenko et al., 1994; Dardenne et al., 1993; Loucopoulos, 1994, pp. 639)
- goal refinement and operationalization into system requirements (Dardenne et al., 1993, pp. 3; Rolland et al., 1998, pp.1055; Yu, 1995; Mylopoulos et al., 1999, pp.31)
- specification how goals should be achieved by the envisioned system (Letier, 2001; Lamsveerde and Dardenne, 2000, pp.978; Anton, 1997)
- identification and eventually resolving of goal conflicts and requirements conflicts (Letier, 2001; Yu, 1995; Lamsveerde and Letier, 2000; Darimont et al., 1998, pp. 908)
- goal representations provide a criterion for requirements completeness (Lamsveerde and Letier, 2000; Kaiya et al., 2002)
- goals are generally more stable than the requirements to achieve them (Letier, 1997)

However, beside theses examples, most requirements modeling notations and techniques focus on the “late-phase” of the requirements engineering process (Yu 1995), during which initial statements of functional requirements are precisely reformulated and analyzed for ambiguity, incompleteness and inconsistency (Letier, 2001, p.7). Methods supporting this kind of analysis range from semi-formal (e.g. structured methods (Ross and Schoman, 1977), object-oriented methods (Rumbaugh et al., 1991) to formal (e.g. history-based (Manna and Pnueli, 1992, Manna, 1996), statebased (Abrial, 1996), or transition based (Heitmeyer, 1996, pp. 231). Beside theses approaches in the literature several goal-oriented requirements engineering methodologies were presented:

- GDC (Goal-Driven Change) method (Kavakli, 1999)
- AGORA (Attributed Goal-Oriented Requirements Analysis) methodology (Kaiya et al., 2002)
- ISAC (specification of information systems and its application) change analysis (Lundeberg, 1982)
- i* (distributed intentionality) strategic rationale modeling framework (Yu, 1995)
- NFR (Non-Functional Requirements) framework (Chung et al., 1996)
- GBGRAM (Goal-Based Requirements Analysis Methodology) (Anton, 1996)
- Goal–Scenario Coupling Method (Rolland et al., 1999)
- KAOS (Knowledge Acquisition in automati ed Speci fi cation) goal-directed requirements elaboration framework (Dardenne et al., 1993)
- ALBERT (Agent-oriented Language for Building and Eliciting Real-Time requirements) (Dubois, 1989)

In this stage of the engineering project goal modeling is intended to address the “early-phase” of requirements engineering during which stakeholders goals are identified explored and alternative system proposals that satisfy the goals are investigated. During the earlier stages of the RE process it is more important to model and analyze stakeholder needs and
interests and how they might be addressed or compromised by the decision to introduce a new system (Kavakli, 2002, p.237). The methods (GDC, i* and NFR) focus on ‘early’ stages of RE in that they emphasize the need to understand the current enterprise and analyze the need for change from different perspectives. This great demands in goal-oriented methods for instance in the scope of goal elicitation are in practical projects a difficult and complex task (Rolland et al., 1999). Stakeholders like doctors or nurses taking part in a goal-oriented requirements engineering process have mostly only a real abstract imagination of goals. The context specific use or adoption of existing modeling frameworks or the new development requires a detailed analysis of the relevant domain.

One of the most interesting frameworks in the scope of agent oriented requirements engineering is the i* framework\(^1\) proposed in (Yu, 1995). It offers a conceptual framework for modeling social dependencies and settings, based on the explicit notion of actors and goals. Man point is that social settings involve social actors who depend on each other for goals to be achieved, tasks to be performed, and resources to be used.

---

\(^{1}\) i* means – distributed intentionality

---

**Figure 2: exemplary SDM of the maternity ward scenario**

The i* framework distinguishes two strategic models. It includes on one hand *Strategic Dependency Model (SDM)* for describing the different relationships among actors, and on the other hand the *Strategic Rational Model (SRM)* for describing and supporting the reasoning that each actor performs concerning his internal relationships and constraints with other actors. In short the SRM describes the internal intentional relationships of an actor, so that they can be reasoned about. Elements of the model are nodes representing goals, tasks, resources, and soft-goals, respectively, and links, representing either means-ends links, or task decomposition links (Gans et al.). The actors or agents can be specified at a more fine-grained level of detail. The main disadvantage of SRM’s is that it contains strategically relevant elements – but not suitable for operational use. It is not possible to map the specified task to a process for specifying an ordering of tasks. The following figure shows an exemplary SDM of our hospital scenario.

---

**Figure 3: exemplary SRM of a maternity nurse**

---

Proceedings of the Tenth Americas Conference on Information Systems, New York, New York, August 2004
Missing part is an explicit model of time or the conditional execution of tasks fulfilling different goals. The extensions we propose to the i* framework draw their inspiration directly from the process oriented modeling method EPC (event driven process chain). We mapped and integrated SRM’s, SDM’s and EPC’s so that they are useable for requirements simulations. In that it seems to be useful to combine and extend the goal-oriented I*-Framework with powerful state-of-the-art business process modeling languages. The business process modeling language we choose - is the event-driven process chain (EPC). The EPC provides comprehensive means for modeling different views and aspects of business processes. The general meta model for the combined frameworks is depicted in Figure 4. For a more detailed description of this combination see (Kirn et al., 2000).

MODEL CONVERSIONS (TRANSITION I)

Stage 1 (requirements engineering) and Stage 2 (multi agent simulation) are associated each with specific tools. To reuse results from the previous development phase conversion tools are required. The suggested toolkit determines the use of SeSAm (Oechslein, 2004) for creating multi agent simulations since it is a very powerful and easy extendable modeling environment. SeSAm as well is the integral part of the development process. At the requirements engineering side tools for process modeling, goal based modeling or ontology modeling have to be connected. Therefore we developed the following converters:

- Protégé\(^2\) to SeSAm - Protégé is very popular as a tool for visual ontology modeling. It enables the domain modeler to specify syntax and semantics of concepts terms and messages. With an additional SeSAm - Plugin, that is already available to the public, it is possible to import Protégé-Ontologies into SeSAm models and to build a model upon it. Ontology concepts can be used to automatically derive agents or to generate messages.

- AGIL-Shell to SeSAm - The AGIL-Shell (Knublauch, 2001) allows domain experts to create process models. Starting with these models it supports the user with different views (work-flow view, communication view, role view…) to identify and specify agents in the modeled scenario. On the other hand the AGIL-Shell lacks further modeling abilities to specify the agents’ behavior. Therefore we recently developed an export function from the AGIL-representation to the SeSAm model description. This work was done in cooperation with the makers of the AGIL-Shell – the Fraunhofer Institute for Applied Information Technology, St Augustin. The new extension allows proceeding with the previously created model.

\(^2\) http://protege.stanford.edu
ARIS Toolset - Currently we are finishing the development of an import plugin from the commercial tool ARIS process platform\(^3\) to SeSAm. ARIS is one of the major tools for business process modeling. Therefore this extension offers the opportunity to reuse many existing process models.

Furthermore we are developing a SeSAm-Extension to import models from goal oriented modeling tools like OME (Organizational Modeling Environment)\(^4\).

**ENVIRONMENT SIMULATION AND EXPERIMENTS (STAGE 2)**

Multi agent simulation allows determining the global system behavior of distributed systems. This global system behavior is a result of the dynamic interdependencies of the single agents' behavior. Thus multi agent simulation can be used to realize prototypes of intended systems and to evaluate the effects of design issues. For the second phase of the methodology we developed the previously mentioned simulation tool SeSAm. Summarized it is an integrated environment for modeling and experimenting with multiagent simulations. It enables domain modelers to create complete and executable simulation models by describing the behavior of the single agents. Modeling is completely visual and does not require the knowledge of any programming language. Therefore users are familiar with modeling already after a short period of usage.

Figure 5 shows how behavior modeling is done within SeSAm and how simulation runs can look like. UML activity diagrams describe the behavior of agents graphically. Furthermore SeSAm offers powerful experimentation support like distribution of simulation runs and online analysis of the simulation results. It has been already applied in a wide range of application domains: traffic simulation (Klügl et al., 2003), biological simulations of insects (Klügl et al., 2003) and rather complex domains like the simulation of clinical processes (Heine et al., 2003). The model partly depicted in Figure 5 simulates a clinic specialized on radiation therapy. In particular the purpose of the model is to develop a system for efficient treatment planning, that means on the one hand to reduce the patients waiting time and on the other hand to maximize the load of functional units. All this takes place in a very dynamic environment, where emergencies and uncertainties are standard. That is why simulation is necessary to evaluate effects of designed systems.

---

\(^3\) [http://www.ids-scheer.de](http://www.ids-scheer.de)

\(^4\) [http://www.cs.toronto.edu/~eric/](http://www.cs.toronto.edu/~eric/)
CODE GENERATION (TRANSITION II)

The visually editable SeSAm agent representation is interpreted and executed by the integrated simulator. This simulation environment has some characteristics that might be unwanted in an agent application: The agents of one simulation run can not be distributed, but are executed on one machine (although different simulation runs can be distributed). Simulation time might be different from real time, and simulation experiments usually have a temporal limit and are not running endlessly. Agent services in contrast might want to be permanently available. Furthermore interpretation of the behavior might be more slowly than executing compiled code. Thus there are several reasons for running agents independently of the simulation environment.

Usually at this point we would use the simulation model as an archetype and re-implement the agents using a programming language and a prevalent agent platform framework (e.g. FIPA-OS or JADE). Since we already have specified the behavior of the agent in a machine-executable manner it is obvious to try to reuse this model. Reuse means either to generate source code (compilation) or to use the behavior description in an external agent (interpretation). Both approaches are described more closely in previous publications (Klügl et al., 2003). Recently code generation became very popular in the area of building graphical user interfaces or deriving code from UML models. There is a very generic way to create JAVA source code from SeSAm models as well. Figure 1 shows the transition from the activity-graph-representation to source code in a very simple example. The resulting code has to be compiled in order to be executed.

Usually at this point we would use the simulation model as an archetype and re-implement the agents using a programming language and a prevalent agent platform framework (e.g. FIPA-OS or JADE). Since we already have specified the behavior of the agent in a machine-executable manner it is obvious to try to reuse this model. Reuse means either to generate source code (compilation) or to use the behavior description in an external agent (interpretation). Both approaches are described more closely in previous publications (Klügl et al., 2003). Recently code generation became very popular in the area of building graphical user interfaces or deriving code from UML models. There is a very generic way to create JAVA source code from SeSAm models as well. Figure 1 shows the transition from the activity-graph-representation to source code in a very simple example. The resulting code has to be compiled in order to be executed.

![Figure 6: Example for JAVA source code generation from the graphical representation](image)

The alternative way for reusing the existing model is to use a generic software agent, which understands and interprets the SeSAm behavior representation, and is designed to act in the real environment. This as well allows the execution of agents outside of simulation environment. We call this generic agent a “wrapper agent” because it encloses the given behavior description in an agent with common interfaces. A first prototypical implementation to proof the concept is called SeScape (The acronym combines the words SeSAm and Escape, because SeSAm agents leave their usual environment). SeScape bases on JADE and was realized for a basic subset of the existing behavior primitives. An example model could be successfully executed outside the simulation environment. Both alternatives - “code generation” and “wrapper agent” - have advantages as well as disadvantages. Code generation is faster and more flexible to ex post user changes. On the other hand the wrapper agent approach is easier to realize and does not require any compilation.

5 http://www.borland.com/together/
6 http://sharon.cselt.it/projects/jade/
APPLICATION DEVELOPMENT (STAGE 3)
Application development is the third phase of the methodology. For reasons of openness we did not develop proprietary tools for this phase but suggest the use any FIPA compliant platform e.g. the agent framework JADE to implement the agent systems. Since FIPA constitutes standards for agent platforms and especially the communication between agents this ensures openness and interoperability of the created systems with others. Using source code generation it is possible to base upon the work of pervious phases. Secondly ontologies developed in phase 1 can be referenced using the Protégé-Plugin BeanGenerator (Aart et al., 2002), which as well generates ontology classes from the Protégé knowledge representation.

Basing upon this generated code the actual agent application has to be developed. Usually the design of the agent system as well as the desired functionality is already specified. The agents have to be extended in order to connect to legacy systems and databases and to provide user interfaces, which were not necessary in the simulation environment. Specific details that were neglected in the simulation but are necessary to application in the real world (e.g. rights management) have to be integrated.

In the example of the radiation therapy (see stage 2) the desired application is a distributed agent system that provides personal assistance agents for patients, doctors and functional units. These agents know preferences and restrictions of their users and negotiate with each other, to assign treatments and to agree on appointments. The personal assistant agents can automatically react on changes and optimize existing schedules.

CONNECTION TO THE SIMULATION SYSTEM (TRANSITION III)
In order to connect the developed software agents to the SeSAm simulation the plugin “SeSAm FIPA support” was developed and is available on the SeSAm website. It allows extending SeSAm agents with FIPA compliant communication abilities. So agents of the simulation can interact with other agents running on external platforms. SeSAm itself acts as a self-contained FIPA-platform. New behavior primitives for the simulated agents allow registering and deregistering at platforms as well as sending and receiving messages. Another important FIPA functionality is searching for agents at the local yellow pages service. Mutual understanding within the agents can be guaranteed by the use of the same ontologies (see Protégé-Import in Stage 3 and between Stage 1 and 2). The practicability of this connection approach has already been tested within AgentHospital (Kirn et al., 2003) where SeSAm was connected to several agent based health care applications.

SIMULATION BASED EVALUATION (STAGE 4)
In the fourth phase of the suggested development process the created application has to be evaluated in a simulated environment. Our practical experience shows that simulation may be useful finding software bugs and benchmarking the system. In agent development isolated tests are usually very difficult due to the dynamic interdependencies. Instead tests have to be made under the conditions of this environment. As mentioned above, the idea was to re-use the previously created SeSAm model as a testbed for the developed agent-based software. Figure 7 shows a sketch of this approach.

![Figure 7: agent systems in a simulated environment vs. application in real world](image)

An agent-based application is integrated in an environment consisting of additional information systems and users that can also be seen as agents. All agents from the real-world environment are replaced by simulated SeSAm agents. The detail level

---

7 behavior primitives = basic actions and functions provided for behavior modeling
of the simulation is depending on the requirements of the agent system. Due to the representation of the real-world in the simulated testbed the developed components can be tested under different conditions (standard conditions, seldom and extreme conditions). Effects of variable critical parameters can be evaluated and adjusted. This often cannot be done in the real world for reasons of security and costs. In some application domains e.g. the healthcare domain practical testing of prototypical software is very dangerous. That’s why we have the objective to realize a testbed for agent based systems in real world settings.

**CONNECTION TO LEGACY SYSTEMS - PRACTICAL EVALUATION (TRANSITION IV, STAGE 5)**

Finally the application has to be tested in a real environment before it can be declared suitable and save for routine use. Testing in the real environment is usually more expensive than testing in simulation. It necessitates to connect the application to existing information systems and to respect domain specific standards. The high grade of domain dependence in this stage is the reason, why there is no generic tool support for deploying agent systems into the real environment. Therefore we want to describe this process with a practical example.

The eHealth Lab is a project initiated in August 2003 at the University of Stuttgart-Hohenheim by Prof. Stefan Kirn. The project addresses the vision of “Seamless Healthcare”. Typical software systems of the healthcare domain like HIS (Hospital Information System), RIS (Radiology Information System) and PACS (Picture Archiving and Communication System) will be installed and different studies about the interoperability of these systems will be undertaken. Seamless healthcare refers in that context to the provision of continuous vertically and horizontally integrated holistic care to patients through the contribution, co-ordination and collaboration of all healthcare service providers and carers in the healthcare domain.

Seamless Integration of Healthcare Processes in the ADAPT Project is primary related to standardized image management and communication in hospitals and thereby uses the eHealth Lab infrastructure to test and evaluate agent based applications after they have been tested in the virtual environment simulation of SeSAm. Figure 8 shows the conceptual integration and interoperation of these systems.

The Agent>Hospital framework described in (Heine et al., 2003) enables/facilitates in this context communication among different multiagent systems. In our hospital application domain each participating multiagent system is with less effort (e.g. using common HL7 ontology) able to use the functionality of HL7 or DICOM modules to send specific messages, receive acknowledgements or view, process and send DICOM images of an image archive using DICOM protocols. First prototypes of the HL7 and DICOM modules are designed as FIPA compliant agents to handle examination orders or schedule requests.

![Figure 8: eHealth Lab scenario for application evaluation](image)

**CONCLUSIONS AND FUTURE WORK**

We have introduced project a novel engineering methodology for developing agent based systems. The process model starts at the early stages of requirements engineering and integrates domain experts without programming knowledge (e.g. a customer) in the development process. Visual modeling supports the dialogue between customers and developers. The
process model includes multi agent simulation as an integral part to show the effects dynamic interdependencies in the design phase and allows using the created simulation model as a simulation testbed in the evaluation phase. This fact is very important since the overall behavior of multi agent systems typically is not party of the explicit model but implicitly depends on the composition of many agent behaviors.

Furthermore we presented (visual) modeling tools and conversion tools, which support the whole process. This is very important since they make the application of the process efficient. The tools reduce development costs and – even more important – they prevent error-prone gaps in the development processes. Conversion tools between different kinds of modeling have been developed and tested (Protégé, AGIL, JADE). Further tools are under development (ARIS import, goal based modeling) to complete the toolkit to a valuable agent development framework.

In contrast to existing engineering processes the ADAPT engineering process supports domain experts as well as the developers. It allows a continuous development and bridges error prone gaps. The integration of simulation allows the testing of problematic dynamic interdependencies, which are common in multi agent systems.

First practical experiences with the proposed methodology are rather encouraging. We presented a resulting agent system for health care on the AmCIS last year (Heine et al., 2003). The application of multi agent simulation (using SeSAm) as a testbed for a distributed system in hospitals is described in previous papers (Kirn et al., 2003). In another project we simulated a high rack storagehouse in order to provide a testbed for the storage control software. A detailed evaluation and comparison of the methodology to others is still to do and will be subject to our ongoing work.

REFERENCES


