Elke Mentges (1999)

Concepts for an agent-based framework for interdisciplinary social science simulation

*Journal of Artificial Societies and Social Simulation* vol. 2, no. 2, <http://www.soc.surrey.ac.uk/JASSS/2/2/4.html>

To cite articles published in the *Journal of Artificial Societies and Social Simulation*, please reference the above information and include paragraph numbers if necessary

Received: 12-Apr-99  Published: 14-Apr-99

**Keywords:**
- Modelling language, software system, multi-agent system, modelling interactions, toolkit

**Motivation**

1.1 There are a variety of different requirements and qualifications of users applying simulation as a research method in the social sciences. Experience gained with users of the simulation system MIMOSE (Möhring 1996) and discussions in interdisciplinary workshops (Gilbert 1996 and Gilbert et al. 1997) suggest that developing a unified modelling and simulation language for the social sciences is an unrealistic aim.

1.2 This is why our project called MASSIF (an abbreviation of its German name *Multi-Agenten-Simulation in der sozialwissenschaftlich-interdisziplinären Forschung*) goes a different way. Instead of aiming at one unique modelling language we are working on a framework applying agent-based modelling as the approach that seems to be the most promising one to cope with the complexity of social structures and processes. The underlying concept suggests diverse abstraction levels for the description of multi-agent systems. It interconnects agent-based modelling with different popular modelling methods and techniques. These different approaches complement each other: Complex models often comprise multiple different dimensions that cannot adequately be presented using only a single technique. This is especially true for interdisciplinary scientific domains such as integrated assessment modelling (Janssen 1998). Experts from different disciplines use different modelling methods to represent the part of an overall system they are engaged with.

1.3 Further, we apply agent-based approaches when designing the system architecture of the framework to facilitate reuse of simulation models and the interoperability of simulation systems.

**Modelling multi-agent systems: A layered approach**

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

2.2 To improve comprehensibility and transparency we suggest dividing the process of modelling multi-agent systems into modelling agents and modelling interactions and relations between them.

Communication in multi-agent systems

2.3 On the first rather low-level layer, relations and interactions between agents are modelled as communicative acts, i.e. agents interact with each other by exchanging messages. We think the agent communication language KQML ([Labrou and Finin 1997](http://jasss.soc.surrey.ac.uk/2/2/4.html)) is well suited for this purpose. Today it serves as a kind of quasi-standard existing in many implementations.

2.4 Describing communication between agents implicitly, i.e. as an internal process inside each communicating agent, is quite difficult and not transparent. Dialogues between agents often build patterns where certain message sequences are expected and other messages are expected to follow. This, as a rule, makes planning capabilities and an internal model of the communication partner(s) necessary. Such dialogues may only arise spontaneously in societies of rather complex agents. A more transparent and pragmatic way is modelling communication explicitly: Typical patterns of communication are defined as message sequences and agreed as protocols between agents. This facilitates the description of even rather complex interaction structures. It is not restricted to high-level agent types - quite simple agents are able to engage in meaningful conversation, simply by following the known protocol.

2.5 To support users in modelling communication between agents explicitly sequence diagrams used in UML ([Fowler and Kendall 1997](http://jasss.soc.surrey.ac.uk/2/2/4.html)) seem to be well suited. They provide a graphical means of describing message sequences that is easily and intuitively comprehensible. As UML is widespread as a collection of methods in object-oriented software engineering, many people are familiar with it. Sequence diagrams show the order of messages agents exchange in a time limited situation. They emphasise the chronological course of communicative actions.

2.6 Modelling agents is supported by a generic agent model. Its architecture is strongly related to the concept of INTERRAP, a hybrid, multi-layered agent architecture ([Müller 1996](http://jasss.soc.surrey.ac.uk/2/2/4.html)). The structure is modular, so several different types of agents can be assembled, from simple agents interacting by reactive patterns of behaviour to social agents provided with complex planning and decision capabilities.

2.7 In principle, there is no restriction against using other kind of agents. Employing the quasi-standard agent communication language KQML establishes a well-defined interface, and thus facilitates the integration of other agent architectures.

Interactions between agents

2.8 The second layer abstracts from the quite technical and low-level message level. It provides
pre-defined basic interaction types to be used as building blocks modelling interaction structures between actors in a multi-agent system. These types are composed as sequences of messages. The set of pre-defined interaction types offered on this layer is expandable. New types can be defined using the message types and sequence diagrams of the first layer.

2.9 Modelling interactions is supported by graphical means. Interaction diagrams depict interactions between actors within a special context, i.e. a certain temporally limited situation. Thus, they emphasise relationships and collaboration between agents.

2.10 Modelling agents is supported by pre-defined, adaptable agent types differing in structure and (inter-)action capabilities. To cover a broad spectrum, the range offered contains rather simple reactive agents up to complex social agents. Users can generate instances of these types and initialise their knowledge or rule bases and their goal and planning components respectively. All the types are grounded on the generic hybrid agent model and can therefore easily be enlarged step by step. This ensures reuse of already defined agent models as their components can be refined.

Relation structures

2.11 The third layer offers a further step of abstraction to support modelling of multi-agent systems. It corresponds to the way people conceive and organise their understanding of the world around them. Actors are not perceived as isolated entities but as related to other entities. They interact having certain expectations and a restricted perspective of each other. Their perception and knowledge about the other's properties and behaviour is always selective. Thus, they play certain roles for each other in their relations and interactions.

2.12 Roles and role models are rather new concepts in object-oriented software engineering (e.g. Kristensen and Osterbye 1995). Thinking in roles can also support modelling of actors in multi-agent models (Kendall 1998). It is a promising way as it makes it easier to break down complex models to delimited sub-models. For this purpose pre-defined role patterns determining goals, responsibilities, tasks and expertise of agents are offered as building blocks to users of the third layer of the framework. These patterns can be merged so that sub-models can be combined to an overall model.

2.13 Representing roles graphically by role diagrams makes the application of the provided patterns comprehensible and transparent - even to less experienced users. Role diagrams show actors as owners of roles and the resulting interactions with other agents in their environment. The semantics can be detailed in additional views and notations.

2.14 As in real world phenomena agents, own several roles in parallel, each one at a given time. So, role patterns have to be coupled. This can be carried out by letting role diagrams overlap, placing different patterns on top of one another at certain points.

2.15 Role patterns have to be instantiated and adapted. Those properties and abilities ascribed to a role are defined by the chosen role pattern. They can be enlarged and complemented by the user with other characteristics of the modelled agent.

2.16 Our framework provides both universally applicable role patterns (e.g. negotiation schemes and game-theoretic patterns) and patterns that depend on certain application domains such as emergency management. Pre-defined role patterns may be member of a rescue team, victim, curious onlooker etc. The set of pre-defined role patterns can be enlarged using the second layer. For that, one has to
specialise agent types and glue them together with interaction patterns.

Overview

2.17

Figure 1 sketches the user layers of the framework for modelling multi-agent systems outlined above, applying different levels of abstraction. It shows components provided by the system, components that can be generated using the provided building blocks, and the tasks of the modelling process which are necessary to define an agent-based model on every layer.

2.18

Tasks supported by tools are shown in yellow, expandable packages of building blocks in green. Models and model components defined by users with the help of these tools and pre-defined building blocks are coloured blue.

---

**Figure 1: Multi-agent modelling in MASSIF**

---

**Agent-based system architecture**
3.1 The overall architecture of the MASSIF system is designed as a system of software components. It connects both components that will be designed and implemented within our project and already existing applications. They interact and co-operate to exchange information and services with each other by communicating in the agent communication language, KQML. Thus, in the sense of agent-based software engineering, the framework can be viewed as a multi-agent system itself.

3.2 The sequence and content of communications between system components are defined by interaction protocols. A special software wrapping enables both internal and external components and applications connected to the framework to engage in meaningful communication - no matter which programming language they are implemented in. The interaction layer (Figure 2) contains two functional units (cf. Labrou 1996):

- A translation module
  processes incoming messages. It translates a received KQML message and triggers the necessary action of the component or application as it is defined by the semantics of the message.

- A conversation module
  checks whether incoming and outgoing messages are correct in the sense of the applied interaction protocol. Another important task they look after is administering multiple parallel threads with various communication partners and assigning incoming messages to the dedicated dialogue.

![Figure 2: Interaction layer](image)

3.3 All parts of the system are managed and co-ordinated by specialised system agents. A manager agent receives and transmits meta-level information about all the agents connected to the system by request. Router agents are responsible for transport and delivery of messages (cf. Heecheol 1999). Their main function is to unburden agents from the technical level of communication.

3.4 This approach shifts the boundaries of the system somewhat, as it provides an interface to existing software systems such as simulation environments or data analysis tools. Existing systems can be tied together by wrapping them with the modules of the interaction layer. Individual interaction protocols define useful communication sequences between both internal and external system components. The approach establishes interoperability between the MASSIF framework and other software systems and supports reuse of existing models.

3.5 System components and model agents can interact and collaborate while residing on different hosts of a network. In addition, the architecture supports the distributed design and implementation of models. Unlike traditional approaches of distributed simulation, this design may also support the geographically independent development of models. Sub-models can be first generated and tested...
separately. Afterwards they can be tied together into a heterogeneous joint model. Protocols serve as an interface linking the parts together by defining and controlling communication and interactions. This approach may be especially useful for application domains in the social sciences in which experts from different disciplines have to work together as in, for example, Integrated Assessment Modelling.

3.6

Though the overall extent of the framework may be so large that it must use several hosts of a network, it is of course not necessary to install it always as a whole. Cutting the system into parts is possible and in most cases reasonable. Essential parts are tied together by the co-ordinating system agents and interaction protocols.

Perspectives

4.1

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

- Layer 1 aims at programming and simulation experts. The open system architecture and an integrated Java development tool kit are prerequisites that make it possible to enlarge the system transparently by developing new system components, e.g. libraries of statistical functions and presentation routines. At present, a Java-framework is being implemented. It aims at facilitating the development of routines for statistical analysis and the integration of existing software tools.

- Layer 2 is conceived for those experts in the social sciences that already have some experience with standard methods and software packages used for statistical analysis and presentation of results. This user group is to be supported wherever it faces complex tasks to evaluate the results of simulation models. For this purpose, uni-, bi- and multivariate data analysis methods and AI-like techniques will have to be included. But unlike existing standard applications which take only the complexity of the data record into account, this layer intends to provide problem and task oriented access to methods of sensitivity analysis and parameter optimisation.

- The target group of the third layer of the experimental frame are experts in application specific research such as integrated assessment or emergency management. It offers pre-selected analysis methods accessible through user interfaces especially adapted to the application domain and extensive routines for keeping protocols to examine and understand model behaviour.

- Finally, layer 4 is conceived to support decision making in politics, government, industry and commerce by facilitating experiments with simulation models developed for certain questions. Users should be able to discover and comprehend interrelations and estimate the consequences of planned measures. Methods which make it easier to understand and compare the results of different simulation runs in an intuitive manner by of way animation and visualisation of simulation models and results will be provided.

References


