

SeSAm: Implementation of Agent-Based Simulation Using Visual Programming

F. Klügl, R. Herrler and M. Fehler

University of Würzburg
Am Hubland, 97074 Würzburg
Germany

{kluegl, herrler, fehler}@informatik.uni-wuerzburg.de

ABSTRACT

In this paper, we present the most important features of SeSAm, a modeling and simulation platform for multi-agent simulations. Based on a declarative, explicit model representation and visual programming, it allows implementing models on specification level. Optimizing compilation allows efficient simulation of the explicit model representation. It was successfully applied in different areas, like biology, traffic or logistics simulation.

Categories and Subject Descriptors

I.6.7 [Simulation and Modeling]: Simulation Support Systems - Environments

General Terms

Design, Experimentation, Languages

Keywords

Agent-based Simulation, Visual Programming

1. INTRODUCTION

Agent-based simulation offers quite a lot of advantages compared to traditional (also microscopic) approaches, like the elegant representation of heterogeneous populations situated in heterogeneous environments, the explicit embodiment of local effects, or the possibility of flexible interaction between entities based on intelligent behaviour. These properties are the reason why the use of agent-based simulation becomes more and more popular. However, the development of a useful agent-based simulation makes great demands on the modeller as the properties that complicate the development of a multi-agent system also apply here. On a technical level this refers to concurrent processes, in-transparent feedback loops, etc. Additionally when constructing simulation models the freedom of design is restricted as the relation to the original system has to be established. The effort required for producing a structurally valid model with appropriate parameter settings, may be really high. A useful tool

for modelling and simulation is highly valuable especially for modellers that are no programming experts. Meanwhile, a number of tools and frameworks for supporting the development of agent-based simulations are available. Most of them consist of a (Java-) class library that has to be extended by model-specific program code.

At the department of Artificial Intelligence and Applied Computer Science at the University of Würzburg (headed by Frank Puppe), we developed a modelling and simulation environment that combines concepts of declarative high-level model representation and visual programming. The initial aim behind our efforts was to provide a tool for beginners in modelling and programming. Scientists working mostly empirically should be equipped with a tool that allows them use modelling and simulation. The SeSAm team is headed by Franziska Klügl and currently consists of Rainer Herrler, Manuel Fehler, Cornelia Triebig and Gustavo Kuhn Andriotti. Also, several students are continuously contributing to SeSAm.

Besides usage in teaching, SeSAm is applied in several projects for simulation of social insects, reproduction of shopping behaviour, route-choice models in traffic simulation. Beyond such traditional application areas of agent-based simulation, SeSAm forms the technical base for virtual high bay warehouses that are used for software tests, requirements engineering or employee training.

SeSAm contains all components that are necessary for a useful simulation and modelling environment. Additional features – like debugging or refactoring – are inspired by tools for general software development and make SeSAm a valuable tool for rapid prototyping of agent-based simulations. The overall components are depicted in figure 1.

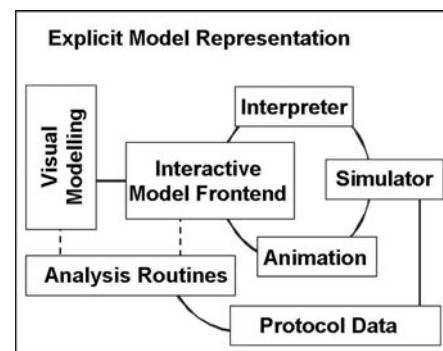


Figure 1: Components of the SeSAm system

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00

In the following we want to focus on some innovative aspects, ignoring others like automatic calibration, integrated evolution, etc.

2. MODEL REPRESENTATION

Models are represented declaratively to assure the separation between simulator and actual model. The high-level modeling language consists of elements on different levels: So called primitives – agent actions, predicates for perception or other functions for processing information – are responsible for translating the declarative representation to executable code. With each model element a code fragment is associated that actually may realize the manipulations or queries. The description of a primitive is augmented with information concerning arguments or return value.

The state of agents, resources or the environment is represented by a body consisting of state variables. Behavior is specified using an activity graph that is derived from UML. An activity of an agent corresponds to a special state variable that determines the next action sequence of an agent. Rules are used for transition between activities. Thus, this agent behavior language has the power of a standard programming language. Not only agent models that are based on behavior description are representable using activity graphs, but – in combination with complex data structures – also simple agent reasoning may be reproduced.

Additional language elements allow a clear and efficient overall model representation. Examples are aggregated activities, which again may contain a complete activity graph, leading to a hierarchical structure. Model-specific primitives, data types or features enhance model entities with additional abilities. The language is extensible based on a plugin concept. Using the same language elements, the instrumentation and definition of experiment scripts containing several, differently configured simulation runs, can be managed. All information concerning the model is stored in a XML-file.

3. VISUAL PROGRAMMING

Based on the declarative model representation, visual programming is consequently applied not only for implementing the structure and behaviour of the agents and their environment, but also for data output generation, experiment scripting or constructing an interactive simulation GUI. The SeSAM design is based on the following principles:

- Complexity is hidden from beginners, but experts are allowed to use the full power
- Similar elements are always edited using similar editors
- No path for implementing a simulation model is a priori given, editors are accessible from all reasonable contexts
- Fault safety based on syntax tests and restriction of all input possibilities to syntactically correct ones.
- Documentation facilities for all model elements.

All elements of a SeSAM model and an executable simulation experiment are accessible via a central hierarchy. Structures are edited using forms, behaviour using graphs and situations may be constructed using maps. For activity transition

rule conditions and all primitive calls, dialogs with three parts are given. On the left side the current function tree, in the middle possible absolute values and on the right side a list of all primitives that may be appropriate in the current model context. See figure 2 for a sample of screenshots.

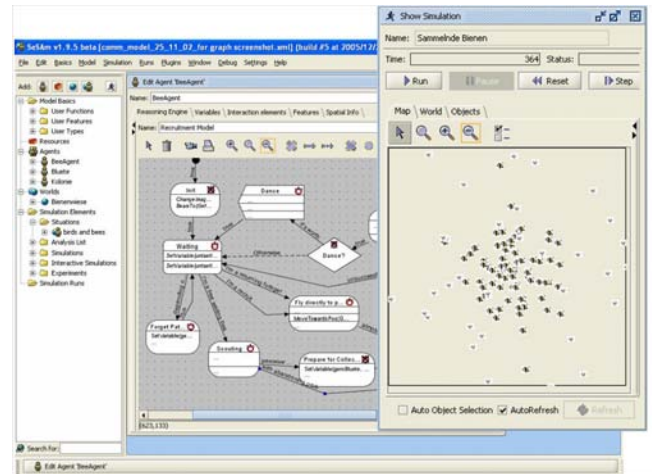


Figure 2: Screenshot of the SeSAM system: in the background an activity graph, in foreground a standard animation window.

Usability of SeSAM for development of agent-based simulation is additionally supported based on features known from modern software development environments. A stepper allows stepping through the behaviour of agents on the action level, a debugger allows to set break points for supporting the search for errors. A list of pre-defined refactoring actions helps to improve the model implementation design. However, a comfortable modelling environment is useless, if the actual simulations are not executable in sufficient speed. Therefore the declarative model representation is not interpreted directly, but compiled using optimization methods from compiler design.

Model-specific extensions or additional functionalities that are not available in standard SeSAM, can be added based on a plugin concept that enables to add primitives, data types or editors. A number of plugins, e.g. for data base connectivity, FIPA-ACL communication between simulated and real-world (JADE) agents, Ontology import (from Protegé projects) are downloadable from the SeSAM web site. Plugins also exist for different spatial representations like 2d continuous (standard), 3d or GIS vector data.

4. AVAILABILITY

SeSAM is developed under the LGPL license. Currently (Februar 2006), SeSAM 1.9.5 and several plugins may be downloaded for free from our SeSAM webpage: www.simsesam.de. We are preparing version 2.0 that will be available in summer 2006. SeSAM is based on Java 1.5, thus versions and comfortable installers for all popular operating systems are available. The possible number of simulated agents is depending on the memory configuration of the computer SeSAM runs on. Usually 512 MB are sufficient for larger models 1 GB and more are advisable.