Year: 2004

Evaluation of free Java-libraries for social-scientific agent based simulation

Tobias, Robert ; Hofmann, Carole

Abstract: This paper compares four freely available programming libraries for support of social scientific agent based computer simulation: RePast, Swarm, Quicksilver, and VSEit. Our aim is evaluation to determine the simulation framework that is the best suited for theory and data based modeling of social interventions, such as information campaigns. Our first step consisted in an Internet search for programming libraries and the selection of suitable candidates for detailed evaluation on the basis of 'knock out' criteria. Next, we developed a rating system and assessed the selected simulation environments on the basis of the rating criteria. The evaluation was based on official program documentation, statements by developers and users, and the experiences and impressions of the evaluators. The evaluation results showed the RePast environment to be the clear winner. In a further step, the evaluation results were weighted according to effort/time/energy saved by social scientists by using the particular ready-made programming library as compared to doing their own programming. Once again, the weighted results show RePast to win out over the other Java based programming libraries examined.

Robert Tobias and Carole Hofmann (2004)

Evaluation of free Java-libraries for social-scientific agent based simulation

Journal of Artificial Societies and Social Simulation vol. 7, no. 1
<http://jasss.soc.surrey.ac.uk/7/1/6.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: 28-Nov-2003 Accepted: 29-Nov-2003 Published: 31-Jan-2004

Abstract

This paper compares four freely available programming libraries for support of social scientific agent based computer simulation: RePast, Swarm, Quicksilver, and VSEit. Our aim is evaluation to determine the simulation framework that is the best suited for theory and data based modeling of social interventions, such as information campaigns. Our first step consisted in an Internet search for programming libraries and the selection of suitable candidates for detailed evaluation on the basis of 'knock out' criteria. Next, we developed a rating system and assessed the selected simulation environments on the basis of the rating criteria. The evaluation was based on official program documentation, statements by developers and users, and the experiences and impressions of the evaluators. The evaluation results showed the RePast environment to be the clear winner. In a further step, the evaluation results were weighted according to effort/time/energy saved by social scientists by using the particular ready-made programming library as compared to doing their own programming. Once again, the weighted results show RePast to win out over the other Java based programming libraries examined.

Keywords:
Evaluation, Simulation framework, Agent based modeling, Java, Theory based modeling, Data based modeling, Social intervention planning

Introduction

The Problem

1.1
Agent based computer simulation (ABS) is a new research strategy that is developing rapidly. The basic principle is that a system is constructed out of a number of sub-systems – so-called agents, such as a city of thousands of modeled individuals. The methods have become interesting for the social sciences because of the possibility of generating complex systems

based on simple rules, the solution of the micro-macro transition problem, and the clarity/graphicness of the models as well as a number of further advantages. A number of social scientific simulations have already been developed (for example, Mosler and Brucks 2003; Mosler, Schwarz, Ammann and Gutscher 2001).

1.2

Applications-oriented social scientific simulation models are characterized by relatively complex agents, as they contain as a rule several social scientific theories, and by the fact that, at the same time, the simulations work with large populations of tens of thousands of agents. The simulations are always grounded in empirical data and are utilized to model real social systems, whereby social networks must also be modeled in a functionally equivalent way to the real social world. The goal of computer modeling is the optimization of social interventions, such as campaigns to calm traffic (Tobias and Mosler, in prep. 2000).

1.3

In principle, social scientists could program the simulations themselves from scratch. However, the use of already developed simulation frameworks can relieve social scientists of some of the burden of programming the parts of the simulation that are not content-specific, such as, for example, simulation control or input-output procedures. It also increases the reliability and efficiency of the programs, as the most complex parts of the program have been created and optimized by professional developers. And finally, social scientists profit from ease of use, as most of the ready-made and more or less widely distributed simulation frameworks provide detailed documentation and a user interface that simplifies the work, which is usually not the case for self-developed software. Naturally, there are also problems involved when using new programs or programming libraries, such as the additional effort that is required to understand someone else's code or certain limitations of modeling options, but these can be reduced as the tools offered by the community of programmers and the needs of users become adjusted.

1.4

But now we are faced with the problem of determining what the appropriate simulation framework is. As the method is so new, but evolving rapidly, countless more or less developed simulation frameworks have emerged. Most of these programs were developed for personal use and then developed further and published. The establishment of individual environments as a standard, as is the case for statistical program packages, is not yet in sight. Moreover, applications-oriented theory and data based simulation is not yet very common, and so there exists no simulation framework that has been tailored to its exact requirements. Our task, therefore, is to find a simulation framework that not only allows agent based simulations, but supports targeted further development of the toolkit for purposes of simplifying simulations.

1.5

In the following, we evaluate a number of different simulation frameworks as to their suitability for applications-oriented theory and data based simulation. Of course, a conclusive picture of a simulation framework's suitability can be gained only after several years of experience with the tool. But even a theoretical evaluation, as we attempt here, requires a lot of effort, for documentation must be acquired and examined, developers and users must be interviewed, and general familiarity with the software has to be gained.

1.6

For these reasons, it is sensible to conduct a pre-selection process of programming libraries that should be included in the evaluation. For the evaluation of the selected environments, a category system is developed (see section on rating system below) and the selected simulation frameworks rated (see section on ratings below). In the discussion section, the ratings will be weighted on the basis of the effort spared the social scientist by using the various simulation frameworks and the most suitable simulation framework determined. Finally, we offer some remarks on the evaluation as a whole.

Selection of programs to be evaluated

2.1

The Internet is the medium for publication of simulation frameworks. Therefore, our first step was to conduct an Internet search for software for social scientific ABS. A good overview is provided at http://www.econ.iastate.edu/tesfatsi/acecode.htm. As a lot of programs are still rudimentary or have not yet been published, they did not even enter into the pre-selection process. However, the evaluation team was familiar with two of these not widely known programs, Quicksilver and VSEit. They were included in the pre-selection process, because the team has good contacts with the developers and users, which assures good support, and because they can stand as representative for other, not widely used simulation frameworks.

2.2

Selecting simulation frameworks for detailed evaluation was accomplished on the basis of the following criteria, which were seen as necessary prerequisites:

- At the least, the simulation framework must principally allow agent based computer simulation that is based on social scientific theories and empirical data; better still, there should exist such models programmed in this framework. This means that agents can be modeled as 'free' and complex objects that represent real human beings, institutions, etc. as social scientific models do. Frameworks that only allow very abstract or other types of simulation, such as macro-simulation, evolutionary algorithms, cellular automata and so on, were excluded.
- The software has to be available freely. This is mainly because it should be easy to collaborate with other social science institutions and because it can be expected that the program will become widespread and that there will be a lot of further development.
- The software must use Java as the implementation language. The programming language is important, because the framework will require further development in order to be optimized for applications-oriented theory and data based simulations. Java has a guaranteed future, it is a widely used powerful language, and it appears to be developing into a standard.

2.3

The appendix to this paper contains a table of all simulation frameworks that were excluded from the evaluation as well as the reasons why. Most of the simulation frameworks were eliminated on the basis of the first criterion, as they are oriented specifically towards applications in the field of artificial intelligence and are not sufficiently optimized for social scientific applications. Although many of the remaining environments fulfilled the first and second criteria, they had to be excluded on the basis of the third. A lot of the frameworks are implemented in the programming languages SmallTalk or C++. In the end, only four simulation frameworks fulfilled all of the necessary prerequisites: RePast (REcursive Porous Agent Simulation Toolkit), Swarm, Quicksilver und VSEit (Versatile Simulation Environment for the Internet). Sources and developers can be found in Table 1.

Table 1: Sources and developers of the selected simulation frameworks

Environment	*Web site*	*Version*	*Developer*
RePast	http://repast.sourceforge.net	2.01	Social Science Research Computing, University of Chicago (repast@src.uchicago.edu)
Swarm	http://www.swarm.org	2.1.1	Swarm Development Group, Santa Fe (swarm@swarm.org)
Quicksilver	http://sourceforge.net /projects/javu4u	1.1	Jan Burse (jburse@cyberlink.ch)
VSEit	http://www.vseit.de	0.9	Kai-H. Brassel (brassel@vseit.de)

2.4

Swarm was selected for evaluation even though it is written in Objective-C. The reason was that it is possible to simulate Java based models using Java Swarm (a Java layer running on top of the Swarm kernel released by the Swarm Development Group). Moreover, Swarm is one of the most well known, most widespread, and also most powerful simulation frameworks. With regard to professionalism, we considered Swarm a reference for the other programs.

2.5

The four programs were to be rated as to their suitability for applications-oriented theory and data based simulation. The next step was to develop a rating system that captured the necessary requirements.

Rating system

3.1

All of the rating scales range from 1 (lowest score) to 6 (highest score). Unless mentioned otherwise, a simulation framework must possess the features comprising the lower ratings in order to qualify for the higher ratings. For example, if a simulation environment possesses the feature for the rating 5, but not the feature for rating 3, it will receive a rating of 2.

General criteria

3.2

The categories under General Criteria deal with features that must be considered when acquiring any program and are not specific to agent based modeling.

3.3

Cost is not an issue, as all of the programs selected for evaluation had to be available for free. However, even with freely available programs, there are differences as to the availability of the source code because of their license. With a view to a simulation framework that should produce relevant results for scientific research, it is important that at least the parts of the software pertaining to the simulation are visible. For instance, if systematic errors are observed in a simulation that cannot be traced back to the model, it must be possible to clarify the source of the error. In the case of open source code software, released under GPL (GNU General Public License, http://www.gnu.org/copyleft/gpl.html), BSD (Berkeley Software Distribution, http://www.opensource.org/licenses/bsd-license.php), or LGPL (GNU Lesser General Public

License, http://www.gnu.org/copyleft/lesser.html), the functioning of the simulation framework is visible and can be adapted or even extended. When source code is not available, it is necessary to negotiate access and adaptation with the developer.

3.4

Licensing involves another issue. Your own modifications to and development of a simulation framework that is licensed under GPL or LGPL must be licensed under the terms of those licenses. This does not prohibit commercial use, but it does make open source code mandatory. Under GPL, you may not also use own developments in closed source programs, while in LGPL this is permissible. LGPL and licenses similar to it are therefore rated higher than GPL.

3.5

In addition to licensing, *documentation* and *support* are critically important, for without support for solving problems, even the best program becomes useless. Widespread use (*user base*) simplifies communication and publication in the scientific community and provides broad support for problem solving, but it is not crucial. And finally, the simulation framework should have some *future viability*, for only so are support and spread of use assured. The following shows the criteria and ratings under General Criteria:

License:

1. = Source code not available
2. = Source code partly available, and only to selected institutions
3. = Source code partly available
4. = Source code only available to selected institutions
5. = Source code freely available under GPL (GNU General Public License)
6. = Source code freely available under LGPL (GNU Lesser General Public License), BSD License (Berkeley Software Distribution), or similar licenses

Documentation:

1. = Incomplete or no technical documentation; documentation not being developed
2. = Incomplete technical documentation, because still under development
3. = Complete technical documentation of basic functions provided
4. = Complete technical documentation of basic functions with additional functionality (such as online support, etc.) provided
5. = Complete technical documentation of all libraries provided, also without feature 4
6. = Complete technical documentation with additional functionality provided

Support:

1. = No support
2. = Poor general support (mailing lists, etc.) and no contact to developers and users
3. = Good general support (mailing lists, etc.), but no contact to developers and users
4. = Loose personal contact to developers and users, but poor general support
5. = Loose personal contact to developers and users, but good general support
6. = Intensive personal contact to developers and users also without good general support

User base

1. = Used only by the developer or never
2. = Used by at least two research groups, any scientific field
3. = Use by many research groups, any scientific field
4. = Used by many social scientific research groups
5. = Established and recognized in at least one scientific community (AI / simulation, social sciences, or practice)
6. = Established and recognized in the social scientific community

Future viability:

1. = The product is already outdated and is no longer being maintained
2. = Support and maintenance have been assured to date, but are planned to terminate
3. = Support and maintenance not assured due to developer's lack of resources
4. = Support and maintenance assured, but it is not clear for how long (for example, in the case of professional "one-man operations")
5. = Support and maintenance of the product assured for the next five years
6. = Support and maintenance of the product assured for the next ten years

Modeling and Experimentation Criteria

3.6

Use of open source software frameworks for creating agent based simulations should achieve three advantages as compared to solely using a general-purpose programming language:

1. Reduction of the necessary investment required to write programs allows the social scientist to focus more on theoretical and content modeling work
2. Reduction of programming knowledge required of modeler and user; increased ease of modeling
3. Increased reliability and efficiency of simulation, as complex procedures have been developed by programming experts

3.7

The first advantage listed is very important and is covered by four criteria. *Support for modeling* encompasses the extent to which modeling entails dealing with program-technical "overhead" and whether or not the simulation package libraries contain the procedures that are frequently required for theoretical-content modeling. As the requirements to fulfill this criterion very specifically target applications-oriented theory and data based simulation, it is not expected that the simulation frameworks will achieve ratings higher than 3. *Support for simulation control* describes the possibilities of controlling a single simulation run, while *support for experimentation* has to do with control of whole series of simulations. Not only must simulations be started with the correct initial values and then run correctly, but very particular attention must also be paid to ensure that the resulting mountain of data keeps within limits and can be managed and analyzed. After all, every modeling activity generates an enormous amount of data material, whether this is program code or input and output data. While this can be managed without the help of the simulation framework, data management within the simulation environment has certain advantages and is described with the criterion *support for project organization*. The versions concept is particularly powerful here, where parallel, different versions of one and the same model of parts of the model can exist.

3.8

The second point (less programming knowledge required) is less important, but it must also be considered, as the modeling team that will work with the simulation environment as well as the target audience will tend to be social scientists who are more or less inexperienced programmers. Therefore, we formulated the following two criteria: the extent to which the simulation environment requires programming knowledge (*ease of use*) and the *support for communication* of simulation models.

3.9

The third point (increased reliability and efficiency of simulation) is highly significant, but in the present evaluation it is rated using only one criterion, because a thorough assessment of the effectiveness and reliability would require years of dealing with the simulation environments. With the short period of time available for the evaluation, we therefore chose *ease of installation* of the programs as an indicator of general professionalism in implementation.

3.10
The following criteria were selected for evaluating Modeling and Experimentation:

Support for modeling:

1. = Only Java functionality supported
2. = Functionality for simple GUI (graphical user interface; visualization) and data analysis functions
3. = Advanced GUI und data analysis functions supported
4. = Functionality for frequently used procedures in content-theoretical modeling
5. = Ready-to-use functionality that eliminate most of the programming work
6. = Content-theoretical modeling possible without programming knowledge

Support for simulation control:

1. = Only Java functionality supported
2. = Simple simulation control (user can run the simulation, and no more)
3. = Advanced simulation control (such as formal logic-based solution of the stepping problem)
4. = Flexible simulation control (such as changing parameters at a certain step)
5. = Dynamic simulation control (such as changing parameters in dependency on other parameters)
6. = Extended functionality (such as integration of differential equations, etc.)

Support for experimentation:

1. = Only Java functionality is supported
2. = Simple functions for control and recording of simulation series
3. = Advanced functions for control and recording of simulation series
4. = Ready-to-use Monte Carlo simulations
5. = Simple parameter optimization algorithms
6. = Advanced parameter optimization algorithms

Support for project organization:

1. = Only Java functionality is supported
2. = Simple management of models and model parts
3. = Simple management of simulation runs and experimental series
4. = Advanced management of models and model elements
5. = Advanced management of simulation runs and experimental series
6. = Advanced management and versioning

Ease of use:

1. = Difficult to use even with strong programming skills
2. = Requires strong programming skills, but then easy to use
3. = Easy to use if modeler has knowledge of Java
4. = Easy to use if modeler has elementary programming skills
5. = Text-based user interface usable by lay people
6. = Graphical user interface usable by lay people

Support for communication:

1. = Communicated only in normal Java code
2. = Model and documentation can be linked (such as sensitive online support).
3. = Model can be executed remotely on the Web
4. = Documentation aids (such as automatic visualization) available, but not features 2 and 3 above
5. = Features 2 and 4 above
6. = Both features 3 and 4 above

Ease of installation (as indicator of reliability and efficiency):

1. = Could not be installed by evaluation team

2. = Difficult to install even with support
3. = Easy to install with support
4. = Error-free, fast installation, but runs unstably
5. = Error-free, fast installation, runs stably
6. = Installation easy for lay people

Modeling Options Criteria

3.11

The last group of criteria examine the simulation environments' technical options for modeling. These criteria target the particular characteristic features of applications-oriented theory and data based modeling. In the models, a very large number of agents (tens of thousands of agents) that are very complex (each agent behaves according to different social psychological theories) interact. Therefore, we want to examine how well the simulation frameworks handle a *large number of complex agents* and what options are provided and possible for *inter-agent communication*.

3.12

Due to the complexity of the agents, there is a need for *nesting of agents*. With nesting, agents that represent people can be built from other agents who, for instance, model individual processes. A simulation environment in which sub-agents maintain their autonomy offers increased options. This means, for example, that an agent "group" can be built from persons in such a way that group behavior is based on its members, but the members can also behave autonomously, independently of the group.

3.13

The large number of agents presents the problem of *generating and networking the agent populations*. Tens of thousands of agents cannot be generated manually. A further problem is that as a rule, the procedure is data based, or in other words, based on data of a sample of persons, whereby the entire population must be projected from the data, including the social networks among the agents.

3.14

The two final criteria address options supported for *spatial management*, where the main interest is in determining agents in the area of influence of a particular agent and options for *changing the model structure dynamically*.

3.15

The following lists the criteria under Modeling Options:

Large number of complex agents:

1. = Only a few, simple agents are supported
2. = Only simple agents are supported, but large populations are possible
3. = Only a few agents are supported, but they can be very complex
4. = Relatively many, complex agents are supported, but there are limitations
5. = No limitations on number and complexity of agents, but requires a lot of memory storage and computing time.
6. = As in 5 above; in addition, memory management and computing organization is very efficient.

Inter-agent communication:

1. = No inter-agent exchange supported, must be programmed using Java.

2. = Inter-agent exchange as such is not supported, but structures and methods are provided that simplify and accelerate data searching
3. = Data exchange between agents is supported, but only rudimentary patterns can be implemented (such as calls for variable values), and computing time is slow
4. = Data exchange between agents is supported, but only rudimentary patterns can be implemented, which, however, are computed rapidly
5. = Complex data exchange processes can be programmed easily, but computing time of the processes is slow
6. = Complex data exchange processes can be programmed easily and computed rapidly

Nesting of agents:

1. = No nesting possible.
2. = Only a limited number of levels possible; limited agent types (for example, super-ordinate agent is passive)
3. = Only a limited number of levels possible, but any number of agents can be built from other agents. Sub-agents lose their "autonomy" (for example, an agent can be built from various modules, but not a group from members)
4. = As in 3 above, but no limit to the number of levels
5. = Any number of agents can be built from other agents, whereby sub-agents can still be managed as autonomous agents. However, limited number of levels possible.
6. = As in 5 above, but no limit to number of levels.

Generating agent populations:

1. = No procedure for automatically generating populations supported
2. = Data import supported: agents can be generated from data
3. = Agents can be generated based on simple statistical values (such as means and standard deviation)
4. = As in 3 above, but more complex generators supported (for example, varying distributions, etc.)
5. = Simple projection algorithms for generating a population based on imported data of a sample (for example, copy based on weighting)
6. = As in 5 above, but more complex algorithms (such as probabilistic methods).

Generating networks:

1. = No procedure for automatically networking agents implemented
2. = Elementary networks supported (such as all agents networked with all other agents, random networks)
3. = Automatic generation of networks based on non-social scientific control information (such as networking of all agents within a certain distance, in the sense of spatial interaction)
4. = Automatic generation of networks of agents based on social scientific control information (such as network density, centralization, etc.)
5. = Automatic generation of networks based on characteristics (such as networking agents having similar attitudes) and control information
6. = Automatic generation of networks based on combinations of characteristics and control information

Management of spatial arrangements:

1. = No procedures for managing spatial arrangements implemented
2. = Simple spatial functionality (agents possess a spatial position, simple movements supported)
3. = Simple positioning algorithms (such as decreasing density with increasing distance from a certain point)
4. = Simple areas of influence (for example, all agents at a particular distance from active agents can be determined)

5. = Complex areas of influence (such as possibility for visual obstacles)
6. = Complex positioning algorithms (such as optimization of position based on various bits of inexact position information)

Dynamically changing the model structure:

1. = No changing of structure during model execution possible
2. = Changing of network during model execution possible
3. = New agents can be generated and existing agents can be eliminated during model execution
4. = The structure of agents and networks can be changed during model execution
5. = As in 4 above, in addition automatic control of structural change processes (such as "aging").
6. = As in 5 above, in addition these processes can be changed through simulation and experimental control

3.16

On the basis of the rating system above, the individual programs are evaluated in the following section, following the same order as the order of the categories and the criteria. An overview will show the ratings given to each program, and the results are discussed.

Rating

Bases for rating the programs

4.1

To rate the programs according to the rating system, the following sources were used:

- Documentation (tutorials, technical literature, etc.)
- Statements by developers or users
- Experiences and impressions of the persons doing the rating, through familiarizing themselves with the simulation framework and implementation, or testing of demo models

4.2

Because so many personal statements by developers and users were available and the type of source should be recognizable to the reader, we will not use the usual citation format. We gave the rating sources a reference code that allows the reader to refer to the following table (see Table 2):

Table 2: Overview of sources for rating the simulation frameworks

Reference code	Source
dr1	RePast technical documentation: http://repast.sourceforge.net/docs/api/index.html
dr2	RePast 'how to' documents: http://repast.sourceforge.net/modules.php?op=modload&name= Sections&file=index&req=listarticles&secid=2
dr3	RePast Web site http://repast.sourceforge.net
ds1	Swarm documentation: http://www.swarm.org/swarmdocs/set/set.html
ds2	Swarm tutorial: http://www.swarm.org/intro-tutorial.html

ds3	Swarm user guide: http://www.swarm.org/swarmdocs/userbook/userbook.html
ds4	Swarm Web site: http://www.swarm.org
dq1	Quicksilver documentation: http://quicksilver.tigris.org/files/documents/328/686/notes_1.1.zip
dq2	Quicksilver tutorial: http://www.usf.uos.de/projects/quicksilver/tutorial/
dq3	Quicksilver Web site: http://sourceforge.net/projects/javu4u
dv1	VSEit user's guide: http://www.vseit.de/VSEit09/VSEitDoc/UsersGuide.html
dv2	VSEit Web site: http://www.vseit.de
dv3	VSEit paper: http://jasss.soc.surrey.ac.uk/4/3/10.html(Brassel, 2001)
cr1	Developer (creator): Tom Howe (RePast)
cq1	Developer (creator): Jan Burse (Quicksilver)
cq2	Developer (creator): Stephan Jaetzold (Quicksilver)
cv1	Developer (creator): Kai-H.Brassel (VSEit)
u1	User: Michael Böni
u2	User: Carole Hofmann (evaluator)
u3	User: Johannes Kottenau
u4	User: Christian Spörri
u5	User: Robert Tobias (evaluator)

4.3

We conducted the rating of the selected simulation frameworks on the basis of these sources according to the rating system we developed.

General Criteria

Table 3: Overview of the ratings on general criteria

Criterion	**RePast**	**Swarm**	**Quicksilver**	**VSEit**
License	6	5	6	4
Documentation	6	6	2	3
Support	5	3	4	4
User base	5	6	2	2
Future viability	5	5	4	3
Total	27	25	18	16

License

4.4

RePast is released under the BSD license and thus freely available for download with source code (dr3).

4.5

Swarm is released under the GNU General Public License (GPL), so that the source code is available and the software is free (ds4). If you modify the software and publish or distribute the modifications or work, you must make the source code available under the terms of the license.

4.6

Quicksilver is licensed under the GNU Lesser General Public License (dq3). The source code is available for free, and the program libraries can also be used by commercial closed source software.

4.7

For two of the packages in VSEit the source code is not available (network, showit). The source code of the simulation package important for the simulation aspect is going to be made available (cv1). However, it is judged as not sufficient that only the simulation source code will be visible. Experience has shown that, for example, errors can also creep into the visualization package that distort the result and are then falsely attributed to the model (u5). Preferably, the complete source code should be open.

Documentation

4.8

RePast provides extensive documentation. Besides technical documentation, there is a whole series of "How to Documents" that make it easy to become familiar with the software. In addition, the RePast Web site provides numerous publications by RePast users for download (dr3).

4.9

Swarm provides comprehensive documentation (ds1) as well as guides and demo applications (ds2).

4.10

Quicksilver provides some documentation (dq1) and a tutorial (dq2) but they are incomplete and difficult to read. There is no API reference, but a few publications are provided that deal with various parts of the software. The documentation and examples are not comprehensive and leave many points open (u2). Efforts are underway to develop and improve the Quicksilver documentation (cq2).

4.11

The documentation available for VSEit includes a technical reference and examples. A User's Guide and a Modeler's Guide are "in infancy" (dv2). The developer reports that extensive documentation should be available at the end of 2003 (cv1).

Support

4.12

RePast maintains a support mailing list (http://lists.sourceforge.net/lists/listinfo/repast-interest), and a member list allows access to users and developers (dr3). Questions are answered promptly; support is informative and helpful (u2).

4.13

The evaluators have no personal contact to the Swarm Development Group. Support is available through mailing lists (support@swarm.org), an archive, and FAQ (ds4).

4.14
The evaluators were able to contact Quicksilver personally. The response to support questions was fast (u2). At this time, there is no active Quicksilver mailing list (dq3).

4.15
The evaluators have e-mail contact with VSEit. As the developer is working at full capacity, however, response to support questions is not fast, but it is comprehensive and detailed (u2). There is no mailing list (dv2).

User base

4.16
The RePast user base can be called medium large and growing. A new Web site appeared in 2003 and is visited frequently. Numerous projects have been implemented with RePast (dr3).

4.17
Swarm enjoys very widespread use, is recognized by the scientific community, and has a strong and wide user base. It must be added, however, that Swarm is implemented mainly in the area of Artificial Intelligence (AI). Up to now, computer simulation has not been used very much in the social sciences in general (u5, ds4).

4.18
Quicksilver is being implemented at the University of Osnabrück at the Institute of Environmental Systems Research and, since 2000, has also been taught in a lecture course at the Zurich Hochschule Winterthur (ZHW) (u3). The user base is still small (u3, u5).

4.19
VSEit is being used at the Technische Universität Darmstadt (TU Darmstadt) at the department of sociology and also at EAWAG, in the NOVAQUATIS and SIAM departments (u4). Here the user base is also small.

Future viability

4.20
RePast was created by Social Science Research Computing at the University of Chicago, USA, and is under constant development and extension. The future of the software is assured for the next five years (dr3).

4.21
Swarm is being further developed by the Swarm Development Group; the future looks assured for the next five years (ds4).

4.22
Two parties are further developing the Quicksilver framework independently. The developers call its future fairly long-term but do not judge this to be certain (cq1, cq2).

4.23
VSEit is being further developed by the creator (cv1). With development dependent on one person, the future must be seen as uncertain.

Modeling and experimentation

Table 4: Ratings of support for modeling and experimentation

Criterion	RePast	Swarm	Quicksilver	VSEit
Support for modeling	3	3	3	2
Support for simulation control	5	5	5	5
Support for experimentation	3	3	4	3
Support for project organization	1	1	1	1
Ease of use	3	2	3	3
Support for communication	1	1	1	1
Ease of installation	6	4	2	1
Total	22	19	19	16

Support for modeling

4.24
RePast contains a number of tools for visualization and data editing. RePast is introducing interoperability with Geographical Information Systems (GIS). A basic version has already been implemented (dr1, dr2, dr3, cr1).

4.25
Swarm offers demo applications that illustrate the use of the software. There are, however, no ready-made templates for models that can be used for social scientific modeling. Extensive GUI and data editing functions are provided (GUI Library und Analysis Library) (ds2).

4.26
In its current version, Quicksilver does not offer ready-made templates for modeling. Sufficient GUI and data editing functions are provided. There are predefined visualizations (such as charts) in the "role" class. For graphical representation of agents, icons can be defined ("depiction" class) (dq1).

4.27
VSEit has ready-made templates that can be used for modeling, but even so, the greater part of programming must be done by the users themselves (u4). It is planned that users will develop scripts, following a multi-paradigm modeling approach (dv3). VSEit has basic functionality of GUI and visualization. However, the range of graphical options must be extended in future in order to allow visualization of more complex models (u1).

Support for simulation control

4.28
Solution of the stepping problem and dynamic control of the simulation are supported by all of the simulation environments. The integration of extended functionality has to be implemented by the user in all of them (u3, dq1, ds1, cr1).

Support for experimentation

4.29
All of the four programs support functions for control and recording of simulation series (dq1, dr1, ds1, ds2, cv1).

4.30
The control and recording functions have been perfected in RePast; as an additional function, QuickTime data can be created from the simulation runs (dr1, dr2). RePast developers have not implemented any further algorithms for parameter optimization. There are user groups, however, that utilize such extensions (cr1).

4.31
Quicksilver, in addition to control and recording functions, provides a class for Monte Carlo analysis of the model (dq1).

4.32
VSEit functions for control and recording are rather simple (cv1).

Support for project organization

4.33
For all of the programs, the desired functionality has to be implemented by the user (dq1, dr2, ds1, cv1).

4.34
RePast provides with SimBuilder for simple management of simulation runs and experimental series. Models can also be managed with SimBuilder (dr3). If SimBuilder is not used, users must implement this functionality themselves.

4.35
Quicksilver offers model databases as *.mdl files that allow management of versions of model simulation runs with varying parameters. This is very rudimentary, however. (u4).

4.36
VSEit has model components that can be put together to produce a model that can be run, but the environment cannot manage them specifically (u2, u5).

Ease of use

4.37
RePast has a graphical layer from which the model can be started and parameters managed. The models have to be programmed, whereby basic knowledge of Java suffices (cr1).

4.38
Repast has SimBuilder, an easy-to-use tool (drag-and-drop interface) for developing simple RePast models. Actions of agents are then programmed using NQPY (Not Quite Python), a subset of the Python language, so that there is no need to learn Java. Users with general knowledge of programming learn this scripting language easily. However, SimBuilder can be utilized for very simple models only (dr3).

4.39
Swarm has to be started via command lines or from within the programming environment. For models, a user interface can be created. There is no interface per se for creating models.

Programming proficiency is necessary in order to use Swarm (ds1).

4.40

In Quicksilver, agents, or actions, must be developed by the user. Actual building of the model and the assignment of parameters can be done at the interface. The control options at the GUI, however, are very limited (u1).

4.41

VSEit has a visual interface for simulation control. Models have to be created in the programming environment, which is doable with basic Java proficiency (cv1). VSEit plans to provide building blocks that will allow users without programming skills to specify a greater number of models (dv3).

Support for communication

4.42

In all of the frameworks, documentation has to be produced manually. Java functionality can be utilized and documents created with Javadoc.[1] To put models on the Internet, Java applets can be created, and Java WebStart technology can be used. The simulation frameworks do not provide specific functionalities to this purpose (u1, u2, u3, dv3).

Ease of installation (as indicator of reliability and efficiency)

4.43

For all of the simulation frameworks speed of simulation processing is mainly dependent on the hardware utilized. Gross implementation errors, which also have an influence, were not discovered in any of the frameworks (u1, u2). But it is important to mention that there is a big computing time difference between program compilation and interpretation. Java offers both options, but interpreted code is always much slower than compiled code.

4.44

RePast has released Version 2.02, and it is stable. The initial installation of RePast was trouble-free, and all of the example models that came along with it functioned properly at the first go (u2). As to efficiency, RePast is working on a "distributed computing" capability that will allow simulation computation to be distributed among several computers in order to improve computability of models (cr1).

4.45

Swarm developers describe it as a very stable and fast framework (ds2). However, at the initial installation, errors occurred with the demo applications. The simulation framework could be installed and started, but we could not get the demo applications up and running (u2).

4.46

Quicksilver in its current version is stable (cq1). With outside help, installation and error-free start of models were accomplished successfully (u2).

4.47

The evaluator could not get the VSEit version at hand up and running. There are various versions of VSEit, and they were in part modified in different ways (u2, u4, u5). With the help of the developer, the evaluator was finally able to view a Web-based version of VSEit in action (u2).

Modeling options

Table 5: Ratings of support for modeling options

Criterion	RePast	Swarm	Quicksilver	VSEit
Large number of complex agents	6	6	6	6
Inter-agent communication	4	4	4	2
Nesting of agents	6	6	4	6
Generation of agent populations	3	3	3	3
Generation of networks	4	2	2	2
Management of spatial arrangements	4	2	2	2
Dynamic structure change	4	4	3	4
Total	31	27	24	25

Large number of complex agents

4.48
All four of the simulation frameworks can handle any number of complex agents. Memory management and computation organization is efficient in all of them. Speed of computation depends mainly on the working memory of the computer executing the simulation (ds2, cq1, cr1, cv1).

Inter-agent communication

4.49
RePast supports the exchange of data between agents. Work is currently underway to support group-based communication (cr1).

4.50
Inter-agent communication in Swarm proceeds via "messages." No complex data exchange processes have been realized, but they can be programmed and added (ds1).

4.51
In Quicksilver, agents are stored in trees, through which they can be called via name (dq1). Communication between agents has not been fine-tuned for efficiency, but it can be improved through user modifications where needed (u1).

4.52
Addressing model objects in VSEit occurs via a table and is very fast. However, there is no special agent architecture as well as no special communication protocol (cv1).

Nesting of agents

4.53
RePast puts very few limitations on agents. There are no limitations on the nesting of agents. Agents can still be executed autonomously (cr1).

4.54

In Swarm, models (groups of classes) can be nested or organized in hierarchies, whereby at each level, the lower level is seen as a black box. Subagents are managed in this way as autonomous agents (ds2).

4.55

In Quicksilver, agents can be nested. However, subagents cannot be executed independently of the super-agent (cq1).

4.56

VSEit implements hierarchical nesting of model objects. Any number of agents can be built from other agents, whereby subagents remain autonomous (cv1).

Generation of agent populations

4.57

Agent populations are generated in RePast using the "Network Library." However, no complex algorithms are implemented (cr1). Data for generating agents can be imported from parameter data bases, and functions can be integrated (dr2).

4.58

Swarm uses a Generator to generate agents. Initial parameters can be set that the Generator assigns (ds2). Initial parameters can also be entered using a formula, whereby agents are given different start parameters (u1).

4.59

Experiments can be automated in Quicksilver using "Generators" and "Observers." Generators create populations, or initial states, and Observers record the course of values and execute analyses (dq1). Data based populations can be generated. Quicksilver does not provide complex generators (cq1).

4.60

VSEit can generate agent populations automatically, but at present only through providing random number generators and statistical distributions. Otherwise, users must program generation themselves (cv1).

Generation of networks

4.61

RePast has a network package used to generate various types of networks (network density, spatial interaction). Social simulation control information can also be used (dr1, dr2).

4.62

Only elementary networks are implemented in Swarm, and they refer only to physical resources and spatial interaction (ds2).

4.63

Quicksilver allows generation of simple networks. Depending on the type of desired network, however, a lot of manual programming is required (cq1).

4.64

VSEit implements simple networks (random networks). More extensive networks have to be programmed by the user (cv1).

Management of spatial arrangements

4.65
RePast supports various sorts of spatial relationships (2D, 3D, hexagonal grids, hexagonal tori, vector spaces, raster spaces, etc.). Work is underway on a GIS library that will contain additional geographic operators. There are simple positioning algorithms as well as simple influence spaces (dr1, cr1).

4.66
Swarm has a "Space" library that supports mainly two-dimensional spaces. Extensions must be programmed by the user (ds2).

4.67
In Quicksilver, agents occupy a three-dimensional torus ("Grid" class). Via the grid, spatial interactions can be controlled. A grid spatial environment is predefined. There is a prototype of a GIS spatial environment in which agents can have arbitrary positions (cq1).

4.68
In VSEit objects can be placed at various levels (3D). VSEit has no further positional algorithms (cv1).

Dynamic structure change

4.69
In RePast agents can be changed during model execution. It is also possible to change the network or the models (dr1).

4.70
In Swarm structural attributes of agents can be changed during model execution. It is also possible to add or delete agents (ds2).

4.71
In Quicksilver active objects sit in a model tree. The model tree can be dynamically changed during model execution (cq1). The connection between objects can be dynamically changed, and new objects can be added. The agents themselves cannot change (dq1).

4.72
In VSEit networks can be changed during model execution, agents can be added and deleted, and the structure of agents can be changed (cv1).

Discussion

Summary of results

5.1
Addition of the ratings given for each group of criteria yields a total score for each simulation framework. Table 6 shows the results. The possible number of total points for the groups of criteria are shown in parentheses.

Table 6: Total scores of the evaluated simulation frameworks

Criterion	**RePast**	**Swarm**	**Quicksilver**	**VSEit**
General (30)	27	25	18	16
Support for modeling and experimentation (42)	22	19	19	16
Modeling options (42)	31	27	24	25
Total	*80*	*71*	*61*	*57*

5.2

Even though the simulation frameworks evaluated show only minor differences in their scores on most of the individual criteria, taking all the criteria together yields a clear winner: RePast. The total scores of the lesser-known simulation frameworks, Quicksilver and VSEit, are similar and much lower than the total score of RePast. For 19 criteria and with a difference of at least 20 points, RePast is superior to these on average in all of these criteria. Quicksilver rates higher than RePast only on support for Monte Carlo simulations and for close contact between developer and users. What is surprising is that RePast beats Swarm, which was used as a reference here even though it is actually not suitable due to its programming language. Except for the criterion "user base," RePast equals or surpasses Swarm on every criterion.

5.3

The evaluation according to total scores does not yet take into account the significance of each criterion. In the following, therefore, we develop a system for weighing the criteria and use it to calculate weighted total scores.

Weighting system

5.4

The various criteria will be given different weightings as follows:

- **Effort spared the user by the feature of the program**: Criteria that spare the user a lot of effort (work and time) should be weighed higher than those that represent only little savings in effort. Effort can mean one-off programming work or repeated effort that is required when modeling or publishing, etc.
- **The necessity for a function**: Even if a function is desired, it may not be worth the effort required for programming. Functions that do not absolutely have to be realized are therefore given a lower weighting.
- **Whether user effort *can* improve a criterion**: Even if every function can basically be programmed by the user, for some functions this does not make sense, as the entire simulation environment would have to be programmed again. Other criteria cannot be improved through user effort, such as the user base of a simulation framework. This must also be reflected in the weightings.

5.5

Effort spared the user can be estimated only roughly. We also want to keep the weighting system as simple as possible. We decided on the following weighting scale with six values:

1. = Little effort (1 to 2 weeks per score on the criteria rating scale)
2. = Medium effort, but not necessary
3. = Medium effort (3 to 5 weeks per score on the criteria rating scale)
4. = Great effort, but not necessary

5. = Great effort (6 to 10 weeks per score on the criteria rating scale)
6. = User effort cannot realize the criterion (efforts exceed 3 months or wide parts of the programming environment must be replaced)

Weightings of general criteria

5.6

License: If source code is not open, user effort is required for inquiries (direct effort) and for time for waiting for responses (indirect effort). As licensing cannot be influenced by the user, license is given the weighting **3**.

5.7

Documentation: Good documentation saves the user considerable effort, but it can be produced by the user. As documentation is indispensable, it receives a weighting of **3**.

5.8

Support: Good support saves the user effort in many ways: faster familiarization with the program, faster problem solving in simulation, and necessary extensions may possibly even be taken on by the support team. Nevertheless, support is dispensable, because it can be replaced by good documentation and the accumulation of experience. Support is assigned a weighting of **4**.

5.9

User base: User base can save effort when publishing, since it is not necessary to justify the use of and explain the functioning of a simulation framework in every publication. As this is not indispensable, however, the criterion receives a weighting of **2**.

5.10

Future viability: Future viability does not yield any savings in effort. The criterion receives a weighting of **1**.

Weighting of support for modeling and experimentation

5.11

Support for modeling: Although changes in this criterion entail considerable programming effort, it does not make sense to charge all of it, as Java libraries for GUI and data editing/analysis are available independently of simulation frameworks. Moreover, for the most part, content/theoretical procedures have to be programmed specifically for the users' own research interests anyway. The weighting here is **3**.

5.12

Support for simulation control: Simulation control is the core of any simulation framework. If it unusable, the framework fails. Even if the user effort to realize users' own simulation control is not very great, this criterion is therefore assigned the weighting **6**.

5.13

Support for experimentation: For large experimental series, functions that ease execution or make execution possible to begin entail a great deal of programming effort. At the same time, they are crucial to exploiting the options of computer simulation, which results in a weighting of **5**.

5.14

Support for project organization: Project organization can be conducted also outside the simulation framework, so that this criterion represents only little effort spared the user. Weighting: **1**.

5.15

Ease of use: The ease of use of a simulation framework cannot be improved by the user and/or this would make the simulation framework itself useless. Weighting: **6**.

5.16

Communicability: Communicability reduces mainly efforts required for publishing, even more so than user base. Considerable user effort would be required to produce publication aids. However, as publications aids are dispensable, the criterion receives a weighting of **4**.

5.17

Ease of installation: Ease of installation, as an indicator of reliability and efficiency, can hardly be improved by user effort. For this reason, it receives a weighting of **6**.

Weighting of modeling options

5.18

Large number of complex agents: In addition to simulation control, this criterion can be regarded as the core of a simulation framework. Weighting: **6**.

5.19

Inter-agent communication: Inter-agent communication is absolutely decisive, but also relatively easy to accomplish in Java. However, a particularly efficient solution can speed up execution of the simulation, which means a great deal of effort spared. Weighting: **3**.

5.20

Nesting of agents: Nesting agents is indispensable and, at the same time, it is an integral component of a simulation framework. However, it is relatively easy to achieve the advantages of nesting through other means, even if this entails additional effort for modeling and publication. Therefore, the criterion is given a weighting of **3**.

5.21

Generation of agent populations: The generation of agent populations is indispensable for the modeling of social interventions, but relatively little effort is required to program generators. Weighting: **3**.

5.22

Generation of networks: This is similar to the generation of agent populations, except that the procedures here require significantly more effort. Therefore, the criterion receives a weighting of **5**.

5.23

Management of spatial arrangements: User development of spatial management functions requires effort, but the functions themselves are dispensable, because for most applications, it is sufficient to model a population based in its social network. Weighting: **4**.

5.24

Dynamic structure change: This is basically not something that the user can realize; options for dynamic structure changes have to be provided by the simulation framework. However, in applications up to now, this functionality has not been necessary, so that we weigh the criterion with a weighting of **4**.

Weighted total scores

5.25

Table 7 shows the weighted total scores of the ratings for the simulation frameworks evaluated.

Table 7: Weighted total scores of the evaluated simulation frameworks

Criterion	**RePast**	**Swarm**	**Quicksilver**	**VSEit**
General (78)	71	62	48	44
Support for modeling and experimentation (186)	113	95	94	80
Modeling options (168)	127	109	99	103
Total	*311*	*266*	*241*	*227*

5.26

Weighing the criteria does not change the order of the scores for the simulation frameworks, but it accents the differences among the not-weighted scores even more strongly. RePast continues to lead by a wide margin. The weighting reveals some differences between the two simulation frameworks in the two last places, as the weighted total scores show Quicksilver to have a clear lead on VSEit. The most suitable simulation framework for applications-oriented theory and data based modeling is clearly RePast. Still, there are some problems with the evaluation, which we address below:

The upshot

5.27

The difficulty in evaluating these simulation frameworks lies mainly in the fact that it is practically impossible to gain extensive practical experience with the frameworks within a period of a few weeks. To be able to make valid statements about some key criteria, one would need at least several months of application practice. For this reason, the present evaluation is really mainly a theoretical one. The ratings were made mainly based on documentation and technical references. Where possible, we included information supplied by developers and users. It is very possible that the rating results would look very different after having implemented the frameworks over a long period of time.

5.28

And once again, the criteria and the weighting system were developed with an eye to the specific demands of applications-oriented, theory and data based simulation. If we were interested in other forms of simulation, the evaluation results would be very different. Also, due to rapid development in the area of computer-supported agent based modeling, the present evaluation provides only a snapshot at the time we conducted the evaluation (July 2003). As the table showing the simulation frameworks that were excluded from the evaluation reveals, during the time between concluding our evaluation and preparing this report a new simulation framework appeared. Called MASON ("Multi-Agent Simulator Of Neighborhoods... or Networks... or something..."; see http://cs.gmu.edu/~eclab/projects/mason), it apparently has similar features to RePast.

5.29
Despite the problems with the evaluation mentioned above, we can conclude with great certainty that according to the available information, RePast is at the moment the most suitable simulation framework for the applied modeling of social interventions based on theories and data. It is certainly worth realizing a project in this framework and then, on the basis of experience gained, and taking into account the new state of development in the rapidly evolving area of social simulation, conducting the evaluation again.

Notes

[1] Javadoc is the tool for generating API documentation in HTML format from comments in the Java source code. It is usually included in the JavaTM development kits.

Appendix

Table 8: Simulation frameworks not considered for evaluation

Framework	*Social sciences*	*Language*	*License*
AgentSheets Authoring tool for building interactive simulations in Java http://agentsheets.com/	Assistant agents and mobile agents, not really for social scientific simulation	Java	Commercial
Ascape Software framework for developing and analyzing agent-based models, social complexity simulation toolkit http://www.brook.edu/dybdocroot/es/dynamics/models/ascape/	Mostly economic models; agent objects exist within "scapes," which is not suitable for the intended type of social scientific simulation	Java (no longer maintained)	Under own license Open Source for non-commercial use only
Breve 3D simulation framework for the simulation of decentralized systems and artificial life http://www.spiderland.org/breve/	Not oriented to the social sciences	steve	GPL

Cormas Dedicated to the creation of multi-agent systems, with specificity in the domain of natural-resources management. It provides a framework for developing simulation models of coordination modes between individuals and groups that jointly exploit common resources http://cormas.cirad.fr/indexeng.htm	Ecological simulations, spatial/social systems, interaction	SmallTalk	Under own license Non-commercial use only
ECHO (John Holland) Developed to investigate mechanisms which regulate diversity and information-processing in systems comprised of many interacting adaptive agents, or complex adaptive systems (CAS) http://www.santafe.edu/projects/echo/	CAS simulations, ecology, interaction, not for social sciences	UNIX / Linux only (no longer maintained)	Open Source
JADE Java Agent DEvelopment Framework http://sharon.cselt.it/projects/jade/	Assistant agents and mobile agents, not really for social scientific simulation	Java	JADE License and LGPL
Madkit Multi-agent platform built upon an organizational model. It provides general agent facilities (lifecycle management, message passing, distribution, ...), and allows high heterogeneity in agent architectures and communication languages, and various customizations http://www.madkit.org/	Social/ecological interaction. Architecture is not suitable for the intended type of social scientific simulation	Java	GPL / LGPL

MAGSY Development platform for multi-agent system applications http://www.dfki.uni-sb.de/~kuf/magsy.html	For expert systems	OPS5 (production language)	Free
MASON Multi-Agent Simulator Of Neighborhoods... or Networks... or something... Similar to RePast http://cs.gmu.edu/~eclab/projects/mason	No documentation yet; was designed not only for social scientific simulation, but also for AI and robotics	Java	Under own license Open Source
MIMOSE Micro- und Multilevel Modelling Software Development of a modeling language which considers special demands of modeling in social science, especially the description of nonlinear, quantitative and qualitative relations, stochastic influences, birth and death processes, as well as micro and multilevel models http://www.uni-koblenz.de/~moeh/projekte/mimose.html	Modeling language	Only for Sun/Solaris and Linux Server and Java-able clients	Free
NetLogo Programmable modeling environment for simulating natural and social phenomena. It is particularly well suited for modeling complex systems developing over time http://ccl.northwestern.edu/netlogo/	Artificial life simulations	Logo dialect extended to support agents and parallelism	Open Source Non-commercial use only

Ps-i Environment for running agent-based simulations http://ps-i.sourceforge.net/	Mainly for Artificial Life	C, Tcl	GPL
SimAgent Range of resources for research and teaching related to the development of interacting agents in environments of various degrees and kinds of complexity http://www.cs.bham.ac.uk/~axs/cog_affect/sim_agent.html	AI and robotics	Pop-11 (similar to Lisp)	Open Source
SimPack A library for event scheduling and queuing http://www.cise.ufl.edu/~fishwick/simpack.html	Not designed for social scientific simulation	C++	Open Source Non-commercial use only
StarLogo Modeling environment for exploring the workings of decentralized systems without an organizer that are coordinated without a coordinator http://education.mit.edu/starlogo/	Artificial life simulations	Java	Open Source
Sugarscape Study of human social phenomena, including trade, migration, group formation, combat, interaction with an environment, transmission of culture, propagation of disease, and population dynamics http://www.brook.edu/dybdocroot/es/dynamics/sugarscape/	Designed for Artificial life simulations	Object Pascal	Open Source

TeamBots Multiagent Mobile Robotic Research http://www.teambots.org/	Robotics	Java	Under own license Open Source for non-commercial use

Table 8: Reason for excluding simulation framework

Framework	*Not for theory and data based simulation*	*Not Java*	*Not free*	*Not known / new*
AgentSheets			x	
Ascape	x			
Breve	x	x		
Cormas		x		
ECHO	x	x		
JADE	x			
Madkit	x			
MAGSY	x	x		
MASON				x
MIMOSE	x	x		
NetLogo	x	x		
Ps-i	x	x		
SimAgent	x	x		
SimPack	x	x		
StarLogo	x			
Sugarscape	x	x		
TeamBots	x			

Note: If a simulation framework is marked 'Not for theory and data based simulation,' it is not specialized to do agent based simulation with complex agents acting according to social scientific theories and are defined by empirical data. Usually these frameworks are specialized for abstract simulations that do not model a specific real system.

References

MOSLER H-J and Brucks W (2003) Integrating commons dilemma findings in a general dynamic model of cooperative behavior in resource crises. *European Journal of Social Psychology*, 33. pp. 119-133.

MOSLER H-J, Schwarz K, Ammann F and Gutscher H (2001) Computer simulation as a method of further developing a theory: Simulating the Elaboration Likelihood Model (ELM). *Personality and Social Psychology Review*, 5. pp. 201-215.

TOBIAS R and Mosler H-J (in prep.) Agent-based simulation of real-world collective action. Manuscript submitted for publication in the *American Journal of Sociology*.

Return to Contents of this issue