# Survey of Agent Based Modelling and Simulation Tools

**Rob Allan**
Computational Science and Engineering Department,
STFC Daresbury Laboratory, Daresbury, Warrington WA4 4AD

Contact e-Mail: r.j.allan@dl.ac.uk

June 3, 2009

**Abstract**

Agent Based Modelling and Simulation is a computationally demanding technique based on discrete event simulation and having its origins in genetic algorithms. It is a powerful technique for simulating dynamic complex systems and observing "emergent" behaviour.

The most common uses of ABMS are in social simulation and optimisation problems, such as traffic flow and supply chains. We will investigate other uses in computational science and engineering.

ABMS has been adapted to run on novel architectures such as GPGPU (e.g. nVidia using CUDA). Argonne National Laboratory have a Web site on Exascale ABMS and have run models on the IBM BlueGene with funding from the SciDAC Programme.

We plan to organise a workshop on ABMS methodolgies and applications in summer of 2009.

# Contents

# 1 Introduction

Agent based modelling and simulation can be accomplished by using a desktop computer, computing clusters, or clusters on computational Grids depending on the number of interacting agents and complexity of the model. Typically, desktop agent based models do not scale to what is required for extremely large applications. Argonne's exascale computing programme is working to break down the barriers.

Agent Based Modelling packages which do anything beyond trial examples tend to be hard for a novice to understand. This is partly because the frameworks and (mostly object oriented) languages are complex and, in most cases, do not have simple APIs, GUIs or IDEs and partly because each one has its own non-intuitive terminology. This is especially true if one is not accustomed to thinking in terms of individual interacting agents (objects) being part of a complex system.

An introduction to Agent Based Modelling is given on Scholarpedia: `http://www.scholarpedia.org/article/Agent_based_modeling`. For a long list of related software, some highly specialised, see Leigh Tesfatsion's Web site `http://www.econ.iastate.edu/tesfatsi/acecode.htm`. A review of eight AMBS packages relevant for geo-spatial analysis is presented here `http://www.spatialanalysisonline.com/output/html/Simulationmodellingsmsystemsforagent-basedmodelling.html`.

Basic documentation for most of the packages reviewed is largely incomplete. As an example, the Swarm community have a Web site and Wiki which contains a lot of information but is hard to navigate. The Repast development programme is tremendously productive and helpful to the scientific community and none of these problems are severe, but tidying up and fully documenting all the packages in their current form would be useful. Our experience has shown that, to make matters worse, the packages do not all install in a straightforward way and not all the test cases could be made to work (for instance StupidModel in EcoLab would not execute all cases). Model data from one version may not work in a later version. They may also use obscure object oriented programming constructs which may push them closer to computer science research than to actual computational science modelling.

## 1.1 Terminology

The following table taken from [35] shows terminology differences among the most popular platforms.

| Concept/ Term | MASON | NetLogo | Repast | Swarm |
|---|---|---|---|---|
| Object that builds and controls simulation objects | model | observer | model | modelswarm |
| Object that builds and controls screen graphics | model WithUI | interface | (none) | observer swarm |
| Object that represents space and agent locations | field | world | space | space |
| Graphical display of spatial information | portrayal | view | display | display |
| User-opened display of an agent's state | inspector | monitor | probe | probe display |
| An agent behaviour or event to be executed | stappable | procedure | action | action |
| Queue of events executed repeatedly | schedule | forever procedure | schedule | schedule |

## 1.2 Comments on Object Oriented Modelling

The object oriented modelling paradigm is the basis for ABMS, since an agent can be considered to be a self directed object with the capability of choosing actions autonomously based on its environment. It is natural to use classes and methods to represent agents and agent behaviours.

Peter McBurney on 18/3/2007 noted: *While object-oriented programming techniques can be used to design and build sofware agent systems, the technologies are fundamentally different. Software objects are encapsulated (and usually named) pieces of software code. Software agents are software objects with, additionally, some degree of control over their own state and their own execution. Thus, software objects are fixed, always execute when invoked, always execute as predicted, and have static relationships with one another. Software agents are dynamic, are requested (not invoked), may not necessarily execute when requested, may not execute as predicted, and may not have fixed relationships with one another.*

## 1.3 Comments on Random Number Generation

Most of the platforms use the "Mersenne twister" random number generator [37] and optionally allow users to provide their own seed so the sequence of pseudo-random numbers can be repeated. Only Swarm includes a variety of alternative generators.

We note that for parallel random number generation great care must be excercised. We have previously used Marsaglia's uniform random number generator [38] which is available coded in a number of languages.

# 2 ABMS Software Packages

This section outlines some existing packages and gives their status. Input is taken from information found on line plus some personal experiences.

Most of the commonly used ABM platforms follow the "framework and library" paradigm, providing a framework – a set of standard concepts for designing and describing ABMs along with a library of software implementing the framework and providing simulation tools. The first of these was Swarm, the libraries of which were written in Objective-C. Java Swarm is a set of simple Java classes that allow use of Swarm's Objective-C library from Java. Repast started as a Java implementation of Swarm but has diverged significantly from Swarm. More recently, MASON is being developed as a new Java platform and EcoLab as a C++ platform. Others like metaABM aim to provide a meta-level visual design studio for agent based modelling.

These platforms have succeeded to a large extent because they provide standardised software designs and tools without limiting the kind or complexity of models that can be implemented, but they also have well known limitations. A recent review of Java Swarm and Repast (along with two less used platforms) ranked them numerically according to well defined criteria [36]. Criteria were evaluated from documentation and other information about each platform. The review indicated important weaknesses including: difficulty of use; insufficient tools for building models, especially tools for representing space; insufficient tools for executing and observing simulation experiments; and a lack of tools for documenting and communicating software.

The review by Railsback et al. [35] lists five ABMS platforms: Mason, NetLogo, Repast and Swarm in its Objective-C and Java flavours. They included NetLogo, despite the fact that it is not open source, because it has been used for some significant models in addition to its purpose as a teaching tool. Their conclusions were based on the outcome of implementing a series of models in each package, thus from the user rather than the developer perspective. This work effectively sets a benchmark for ABMS packages based on 16 flavours of what the authors called StupidModel, see `http://condor.depaul.edu/~slytinen/abm/StupidModel/`.

We have added some information found in previous on-line reviews and outcomes of practical experience.

## 2.1 A-globe

See `http://agents.felk.cvut.cz/aglobe`.

## 2.2 ABLE: Agent Building and Learning Environment

ABLE is a project made available by the IBM T. J. Watson Research Center, is a Java framework, component library, and productivity toolkit for building intelligent agents using machine learning and reasoning. The ABLE framework provides a set of Java interfaces and base classes used to build a library of JavaBeans called AbleBeans. The library includes AbleBeans for reading and writing text and database data, for data transformation and scaling, for rule-based inference using Boolean

and fuzzy logic, and for machine learning techniques such as neural networks, Bayesian classifiers, and decision trees. Developers can extend the provided AbleBeans or implement their own custom algorithms. ABLE runs on any platform supporting Java 2. See `http://www.alphaworks.ibm.com/tech/able`.

### 2.3 AgentSheets

AgentSheets is a proprietary ABMS tool which is based on a spreadsheet approach. Instead of the cells of the spreadsheet mesh being occupied by numbers they are instead occupied by agents. The simulations then take place on the mesh on which the agents live. A similar idea was GridSheets from Monash University in which cells could be evaluated on remote Grid resources.

AgentSheets is specifically aimed at non-programmers and as a consequence it is very simple to use. It uses a visual programming paradigm so that there is no text based coding and all development is done via a graphical interface (dragging and dropping elements from tool boxes, etc.). Agents are created in a window called a "gallery" and have an associated behaviour specified by sets of rules (called methods) and events. Indeed, the ease of use of AgentSheets is its greatest advantage.

The way that AgentSheets operates is intuitively easy to understand which makes it very quick to develop simple simulations. For this reason, it is widely used for teaching the principles of simulation. However, once the simulation models begin to become more sophisticated, the weaknesses become apparent. With regard to simulation in the social sciences, two particular limitations of AgentSheets may cause problems: an agent cannot send information to another agent (this would be problematic if we wanted to model the communication of information between human agents in a simulation); an agent cannot change the attribute of another agent (this could be problematic if we wanted to model a situation were one human influences another human). There may be ways to work around these problems, but if these situations are met frequently in simulations, it would make using AgentSheets complicated and inefficient. In addition, there is no "long distance" vision making it impossible to examine the status of an agent elsewhere on the mesh. On a technical level, the main drawback of AgentSheets is that it only runs on Mac, although a trial beta version is available for PC.

In addition to its ease of use the other main advantage is that AgentSheets very easily generates Java Applets and Beans which allow the simulation models to be interactively run through a Web browser.

A number of simulation models are available from the AgentSheets Web site along with the source code although the applications appear to be limited, see `http://www.agentsheets.com`.

### 2.4 AnyLogic

AnyLogic is a proprietary offering that incorporates a range of functionality for the development of agent based models. For example, models can dynamically read and write data to spreadsheets or databases during a simulation run, as well as dynamically chart model output. Furthermore, external programs can be initiated from within an AnyLogic model for dynamic communication of information, and vice versa. AnyLogic models can only be created on Microsoft operating systems, although a simulation can be run on any Java enabled operating system once compiled. The AnyLogic Web site `http://www.xjtek.com/` notes that models have been developed for a diverse range of

applications including: the study of social, urban and ecosystem dynamics (e.g. a predator prey system); planning of healthcare schemes (e.g. the impact of safe syringe usage on HIV diffusion); computer and telecommunication networks (e.g. the placement of cellular phone base stations); and the location of emergency services and call centres. However, the source code of these examples and/ or documentation of these models is not unavailable.

### 2.5 Ascape

Ascape is a framework for developing and analysing agent based models which was developed by Miles Parker of the Brookings Institute Center on Social and Economics Dynamics which also developed the well known SugarScape model.

Ascape follows some of the ideas behind Swarm, e.g. agents existing within scapes which can themselves be treated as an agent. However, it is slightly easier to develop models with Ascape than with Swarm. Indeed, it is intended to allow non-programmers to develop quite complex simulations by providing a range of end user tools, e.g. facilities to gather statistics of the running simulation, tools for creating graphs, etc.

Ascape is implemented in Java and developing simulation models in Ascape would require some ability to program in Java together with some knowledge of the object orientation philosophy. For practical development of simulation models, it would be useful to download and use a JDK. In addition, developed models can be published on the Web and there is also a facility within Ascape (a "camera button") to create movies of running models.

In terms of Ascape's applicability to simulation in the social sciences, there would be no problem with implementing quite complex social mechanisms. Like Swarm, the only restriction would be finding a programmer with sufficient skills to code the model.

The current version is Ascape5. In terms of background support, there is a mailing list, but little written user documentation. See `http://ascape.sourceforge.net` and `http://www.brook.edu/es/dynamics/models/ascape`. Whilst some biological and anthropological models are available, the majority of simulation models seem to be concerned with economic and market modelling.

### 2.6 Breve

Breve is a free software package that provides a 3-D environment for the simulation of decentralized systems and artificial life. Users define the behaviors of agents in a 3-D world and observe how they interact. Breve includes physical simulation and collision detection for the simulation of realistic creatures, and an OpenGL display engine so that users can visualize their simulated worlds. It is available for Mac OS X, Linux, and Windows platforms. See `http://www.spiderland.org`.

### 2.7 Cormas

Cormas is a simulation platform based on the VisualWorks programming environment which allows the development of applications in the Smalltalk object oriented language. Cormas pre-defined entities are

Smalltalk generic classes from which, by specialisation and refining, users can create specific entities for their own model. See `http://cormas.cirad.fr/indexeng.htm`.

Cormas has mostly been applied to management of natural resources, namely studying the interaction of human societies with the Earth's eco-system.

The development team are also working on a new generation of meta-modelling and simulation platform, Mimosa, that will provide the means to specify any kind of modelling and simulation formalisms and to compose and run any models written in these formalism. This is not the same as the MIMOSE project described below.

### 2.8 Cougaar

Cougaar is Java software for facilitating the development of agent based applications that are complex, large scale and distributed. The software includes not only the core architecture but also a variety of demonstration, visualization and management components. It was developed as part of a multi-year DARPA research project into large scale agent systems. See `http://www.cougaar.org/`.

### 2.9 DEVSJAVA

See `http://acims.eas.asu.edu/SOFTWARE/software.shtml`.

### 2.10 DeX

DeX is an object oriented C++ framework for developing, analysing and visualising dynamic agent based and multi-body simulations. DeX is built on top of a high performance simulation engine designed to handle a large number of entities with high levels of event communication. DeX includes tools for batch execution, optimisation, distributed job execution, GUI controls, and real time plotting and 3D visualisation using OpenGL. See `http://dextk.org/`.

### 2.11 EcoLab

EcoLab is an object oriented simulation environment that implements an experiment oriented metaphor. It provides a series of instruments that can be coupled together with the user's model, written in C++, at run time in order to visualise the model, as well as support for for distributing agents over an arbitrary topology graph, partitioned over multiple processors plus checkpoint and restart support. EcoLab was originally developed to simulate a particular model – the EcoLab model of an abstract ecology. However, several other quite different models have been implemented using the software, demonstrating its general purpose nature.

EcoLab is written and maintained by Russell Standish at University of Sydney, Australia, see `http://ecolab.sourceforge.net/`. It is Swarm-like, but entirely written in C++ rather than Objective-C.

There are documented conversions between the two languages. The computation is invoked from a TCL script and can be run in parallel on systems supporting MPI.

Two powerful components of the EcoLab programming system are ClassDesc and GraphCode.

### ClassDesc

The basic concept behind ClassDesc is the ability to know rather arbitrary aspects of an object's type at run time, long after the compiler has thrown that information away. Other object oriented systems, for example Objective-C, use dynamic type binding in the form of an "isa" pointer that points to a compiler generated object representing the class of that object. This technology can also be referred to as class description, as one only needs to generate a description of the object's class, then ensure the object is bound to that description, hence the name ClassDesc.

### GraphCode

GraphCode provides an abstraction of objects moving on a distributed graph. A graph is a container of references to objects that may be linked to an arbitrary number of other objects. The objects themselves may be located on other processors, i.e. the graph may be distributed. Objects are polymorphic – the only properties a graph needs to know is how to create, copy and serialise them, as well as what other objects they are linked to.

Because the objects are polymorphic, it is possible to create hypergraphs. Simply have two types of object in the graph – pins and wires, say. A pin may be connected to multiple wire objects, just as wires may be connected to multiple pins.

We have implemented EcoLab v.4.D29 on a Gentoo Linux system for evaluation purposes. It was found that not all the examples could be made to run, for a variety of reasons.

## 2.12 EVO

See `http://omicrongroup.org/evo/`.

## 2.13 FLAME: FLexible Agent Modelling Environment

FLAME is being developed primarily at the University of Sheffield, see `http://www.flame.ac.uk` with collaborators at STFC. FLAME has been developed to allow a wide range of agent and non-agent models to be brought together within one simulation environment. It is aimed at the medical and biological domains, for example studies of tissue cultures and signalling pathways. It is used in the EU-funded EURACE project for financial modelling.

FLAME provides specifications in the form of a formal framework which can be used by developers to create models and software tools that are compatible with one another. New models, adhering to the specifications, may be easily incorporated into existing, or new, simulations with minimum effort. Parallelisation methods and testing techniques, allow the development of large multi-processor simulations with feedback provided on the functionality of written code.

The FLAME methodology is based on the definition of agents as X-machines which are extended finite state machines with the addition of memory. These read data from a message board and change state according to rules encoded in associated functions.

There are no restrictions on the type of simulations which can be coupled together. Although the framework is designed around agent based modelling, it is not a requirement for agents to be included within a simulation. Commercial or in house software tools can be incorporated into FLAME, allowing developers to spend time developing models and not re-inventing tools.

FLAME is not a simulator in itself, but tools developed adhering to the specifications will give the required simulation packages through a compilation process.

The Web site provides information on how to use FLAME based on a couple of examples and also gives access to many FLAME compliant simulation tools. Models suitable for use within this coupled framework can be downloaded, modified and used. The basic xparser converts an XML file containing the FLAME model specification using templates into C code which can be compiled using the makefiles provided. Functional dependency is represented through graphs.

The Web site also brings together people who want to share ideas and work together. FLAME has a growing user group and the developers say they would like it to be more beneficial to the rest of the scientific community.

### 2.14 JADE: Java Agent Development Framework

JADE is a framework fully implemented in Java. It simplifies the implementation of multi-agent systems through a middleware that claims to comply with the FIPA specifications and through a set of tools that supports the debugging and deployment phase. The agent platform can be distributed across machines (which do not even need to share the same OS) and the configuration can be controlled via a remote GUI. The configuration can be even changed at run-time by moving agents from one machine to another one, as and when required. JADE is completely implemented in the Java language and the minimal system requirement is version 1.2 of JAVA (the run time environment or the JDK). JADE is free software and is distributed by TILAB, the copyright holder, in open source software under the terms of the LGPL (Lesser General Public License Version 2). See `http://sharon.cselt.it/projects/jade/`.

### 2.15 JAS: Java Agent-Based Simulation Library

JAS is a simulation toolkit specifically designed for agent-based simulation modeling. JAS is a Java clone of the Swarm library orginally developed by researchers at the Santa Fe Institute. The core of the JAS toolkit is its simulation engine based on the standard discrete event simulation paradigm, which allows time to be managed with high precision and from a multi-scale perspective. Many features of JAS are based on open source third party libraries. JAS is freely available from `http://jaslibrary.sourceforge.net/`.

### 2.16 LSD: Laboratory for Simulation Development

An new and active Italian project with a framework written by Marco Valente of University of l'Aquila in C++ [49]. See `http://www.labsimdev.org/Joomla_1-3/`. Has a focus on economic models. Tesfatsion notes that LSD applications take a systems dynamics (difference/ differential equations) approach using replicator dynamics rather than a bottom-up agent based approach, but the underlying use of C++ suggests that a more agent based approach might also be possible.

### 2.17 MadKit

MadKit is a Java multi-agent platform built upon an organisational model. It provides general agent facilities, such as lifecycle management, message passing and distribution, and allows high heterogeneity in agent architectures and communication languages, and various customisations. MadKit communication is based on a peer-to-peer mechanism which allows developers to develop distributed applications quickly using agent principles. MadKit is free and licensed under the GPL/ LGPL licence. See `http://www.madkit.org/`.

### 2.18 MAGSY

MAGSY is a development platform for multi-agent applications. Each agent in MAGSY has a forward chaining rule interpreter in its kernel. This rule interpreter is a complete re-implementation of an OPS5 system, further enhanced to make it more suitable for the development of multi-agent system applications. MAGSY runs on UNIX, LINUX, SunOS and Solaris systems. See `http://www.dfki.uni-sb.de/~kuf/magsy.html`.

### 2.19 MAML: Multi Agent Modelling Language

The MAML language and xmc compiler were developed at the Complex Adaptive Systems Laboratory of the Central European University between 1998 and 1999. Since then further improvements and patches were programmed at Agent Lab Intelligent Systems, Research, Design, Development and Consulting Ltd. `http://www.aitia.ai/eng`. I could not find any reference to MAML on this new site. The older downloadable software is under the GNU public licence.

MAML was developed by the Complex Adaptive Systems Lab. at the Central European University in Hungary. The aspect oriented language was initially developed to help social science students with limited programming experience create agent based models quickly. The ultimate goal of the project was to develop an easy to use environment, complete with a graphical interface. However, the present version of MAML is, as the name suggests, a programming language and not an environment.

MAML actually sits on top of Swarm and is intended to make Swarm easier to use by providing macro keywords that define the structure of the simulator and access the Swarm libraries. MAML works at a higher level of abstraction than Swarm with clearer constructs. However, in addition to learning MAML, the developer would need to know Objective-C and also Swarm. This point currently limits MAML's usefulness for inexperienced programmers. Indeed, experienced programmers may actually

prefer the added functionality of Swarm and the additional resources available. Programming using MAML requires the developer to create text files using an editor since there is no developer interface. Since MAML accesses the Swarm libraries, the interface of the developed simulation model is very similar as to what would appear if Swarm were used.

The code written using MAML is converted into a Swarm application via the MAML compiler (called xmc). The resulting application is then compiled in the same way as normal Swarm code by gcc. Currently the MAML compiler only runs on a Linux system, running the compiler on a Mac and PC with Windows is untested. This project does not seem to have evolved since 2000 with MAML v0.03 and xmc c0.03.2 both in alpha release – xmc only compiles for Swarm v2.1.1 at latest.

Some background documentation on MAML is provided: a tutorial with source code, reference manual and a technical manual, but knowledge of Swarm is needed. However, there is no support via mailing lists. The MAML home page contains some fairly simple examples of common simulations. Unfortunately, outside of the home page there are very few examples of simulations using MAML.

With respect to MAML's suitability for social science simulations, most of the development effort has been devoted to simplifying the programming rather than providing facilities specifically geared towards modelling.

The xmc compiler appears to work and can convert MAML input files into Swarm files. No attempt has yet been made to test that these will run.

### 2.20 Matsim

See `http://www.matsim.org/`.

### 2.21 MASON

MASON was designed as a smaller and faster alternative to Repast, with a clear focus on computationally demanding models with many agents executed over many iterations. Design appears to have been driven largely by the objectives of maximising execution speed and assuring complete reproducibility across hardware. The abilities to detach and re-attach graphical interfaces and to stop a simulation and move it among computers are considered a priority for long simulations. MASON's developers appear intent on including only general rather than domain specific tools. MASON is possibly the least mature of the current platforms, with basic capabilities such as graphing and random number distributions still being added.

In summary, MASON is a fast, easily extendable, discrete event multi-agent simulation toolkit in Java. It was designed to serve as the basis for a wide range of multi-agent simulation tasks ranging from swarm robotics to machine learning to social complexity environments. MASON carefully delineates between model and visualisation, allowing models to be dynamically detached from or attached to visualisers and enabling check pointing. The MASON system, its motivation and its basic architectural design are described in [6]. Five applications of MASON are also described.

MASON contains both a model library and an optional suite of visualization tools in 2D and 3D. The

system is open source and free and is a joint effort of George Mason University's Computer Science Department and the George Mason University Center for Social Complexity, hence its name. MASON is not derived from any other toolkit and may be down loaded from `http://cs.gmu.edu/~eclab/projects/mason`.

## 2.22 MASS: Multi Agent Simulation Suite

MASS consists of three applications offering solutions for different aspects of modelling. Each application is developed with the intention of providing professional tools for inexperienced programmers. Unlike most other modelling tools the software offers user friendly interfaces and wizards for writing models, creating visualisations or analysing simulation data.

The related Functional Agent Based Language for Simulation, FABLES, is an easy-to-use programming language and its integrated environment specially designed for creating agent based simulations. It requires minimal programming skills, as it has a whole range of functions intended to make the use of the language easier. In the first public release of MASS, this simulation core is Repast J, meaning that all FABLES models are compiled with Repast J.

The Participatory Extension, PET, is a Web based environment for creating, administrating and participating in agent based and participatory simulations. The use of PET relies on mechanisms and practices familiar to users from browsing Web pages.

The Model Exploration Module, MEME, is a tool that enables orchestrating experiments, managing results and has support for their analysis. It allows the user to run simulations in batches with various parameter settings, store, manage and analyze data thus produced.

These tools can be downloaded from the Web site and run on a Windows, Linux or Mac system. See `http://www.aitia.hu/services_and_products/simulation_systems`.

## 2.23 MetaABM

Model driven ABM development in Eclipse, see `http://www.metascapeabm.com/`. MetaABM defines and supports a high level architecture for designing, executing and sytematically studying ABM models. It started life as a component of the Repast Simphony system, "score", but its scope is beyond a single ABM tool. MetaABM is not intended as an ABM engine or runtime environment, but as an approach that can leverage those environments in limitless ways. Beyond that, metaABM seeks to provide a common hub that enables developers of ABM tools to avoid duplication of effort and focus on the value that they can add to the overall software ecosystem. The contributors are committed to an open, developer driven approach and welcome equal participation from other individuals, projects and organisations.

## 2.24 MIMOSE

MIMOSE consists of a model description language and an experimental framework for the simulation of models. The main purpose of the MIMOSE project has been the development of a modelling

language that considers the special demands of modelling in social science, especially the description of nonlinear quantitative and qualitative relations, stochastic influences, birth and death processes, and micro- and multi-level models. The aim is that describing models in MIMOSE should not burden the modeller with a lot of programming and implementation details.

MIMOSE was created by Michael Möhring of University of Koblenz-Landau, Germany. Release 2.0 requires Sun Sparc, SunOS, Solaris, X11R5/6 or Linux. A Java interface is under development and the next release will be usable with Java-enabled browsers. The current release is usable with Java enabled browsers, given that the server process runs on a SunOS or Linux machine. See `http://www.uni-koblenz.de/~moeh/projekte/mimose.html`.

### 2.25 MobiDyc

Mobidyc is a software project that aims to promote Individual-Based Modelling in the field of ecology, biology and environment. It aims to enable models to be built and run by people with no computing skills. See `http://w3.avignon.inra.fr/internet/unites/biometrie/mobidyc_projet/english/version_index_html`.

### 2.26 Modelling4all

See `http://modelling4all.nsms.ox.ac.uk/index.html`.

### 2.27 NetLogo

NetLogo (originally named StarLogoT) is a high level platform, providing a simple yet powerful programming language, built-in graphical interfaces and comprehensive documentation. It is particularly well suited for modelling complex systems developing over time. It is aimed at deploying models over the internet. Modellers can give instructions to hundreds or thousands of independent agents all operating concurrently. This makes it possible to explore the connection between the micro-level behaviour of individuals and the macro-level patterns that emerge from the interaction of many individuals.

NetLogo clearly reflects its heritage from StarLogo as an educational tool, as its primary design objective is clearly ease of use. Its programming language includes many high level structures and primitives that greatly reduce programming effort, and extensive documentation is provided. The language contains many but not all the control and structuring capabilities of a standard programming language. Furthermore, NetLogo was clearly designed with a specific type of model in mind – mobile agents acting concurrently on a mesh with behaviour dominated by local interactions over short times. Whilst models of this type are easiest to implement in NetLogo, the platform is by no means limited to them. NetLogo is said to be by far the most professional platform in its appearance and documentation.

NetLogo has extensive documentation and tutorials. It also comes with a models library, which is a large collection of pre-written simulations that can be used and modified. These simulations address many domain areas in the natural and social sciences, including biology and medicine, physics and chemistry, mathematics and computer science, economics and social psychology. See `http://ccl.`

northwestern.edu/netlogo/.

### 2.28 QuickSilver

The goal of QuickSilver is to provide a simple environment that allows the quick development and testing of agent models. The idea is that QuickSilver cooperates with the Java environment in that it delegates the development of agent types and viewers to Java. QuickSilver itself then provides the editing of a so-called model tree, which contains instances of agent types. The model tree can be navigated and stepped. In both cases the viewers come into play. See http://quicksilver.tigris.org/.

### 2.29 RePast

RePast, the Recursive Porous Agent Simulation Toolkit, was developed at the University of Chicago's Social Science Research Computing Lab specifically for creating agent based simulations [13]. It is very Swarm like, both in philosophy and appearance and like Swarm it provides a library of code for creating, running, displaying and collecting data from simulations. It was in fact initially a Java re-coding of Swarm. See http://repast.sourceforge.net/repast_3.

Repast development appears to have been driven by several objectives. Repast did not adopt all of Swarm's design philosophy and does not actually implement swarms. Repast was also clearly intended to support one domain, social science, in particular and includes tools specific to that domain. The additional objective of making it easier for inexperienced users to build models has been approached in several ways by the Repast project. These approaches include a built-in simple model, and interfaces through which menus and Python code can be used to begin model construction.

One of the aims of the RePast developers is to support the modelling of belief systems, agents, organisations and institutions as recursive social constructions. A second aim is to allow situated histories to be replayed with altered assumptions. Both of these aims makes RePast particularly suitable to social science simulations. Unfortunately these goals are future extensions and are not realised in the current version.

RePast is Java based and developing a simulation requires the ability to program in Java. RePast provides a few, well known, demonstration simulation models such as SugarScape, Swarm's Heatbugs and MouseTrap models. Unfortunately, there are very few other simulation models generally available on the internet. However, there is a mailing list which provides users with general support and discussion.

### 2.30 Repast Simphony

Since 1994, there has been a steady advancement of the software toolkits and development environments that have superseded Swarm. Repast3 has recently been superseded by a significant development named Repast Simphony, or RepastS. This is a free and open source toolkit developed at Argonne National Laboratory. It has tools for visual model development, visual model execu-

tion, automated database connectivity, automated output logging, and results visualisation. See `http://repast.sourceforge.net`.

Whilst still being maintained, RepastJ, Repast.Net and RepastPy have now reached maturity and are no longer being developed. They have been superseded by Repast Simphony (RepastS) which provides all the core functionality of RepastJ or Repast.Net, although limited to implementation in Java. RepastS was released as an alpha version in late 2006 and therefore limited information regarding its core functionality or example models is available. From the user's perspective, the main improvements that Simphony has made over Repast3 are as follows.

- Adding a new GUI for developing models;
- Improving the runtime GUI;
- The addition of contexts and projections.

A context contains a population of agents but doesn't give agents any concept of space or relationships. Contexts can be arranged hierarchically and contain sub-contexts. Agents in a sub-context also exist in the parent context, but the reverse is not necessarily true.

Projections can give the agents a space and can define their relationships. Projections are created for specific contexts and will automatically contain every agent within the context. Different projects can be made for different properties.

For a tutorial on Simphony, see the NCeSS portal site: `http://portal.ncess.ac.uk/access/wiki/site/mass/simphonyytutorial.html`.

## 2.31 SDML

SDML is not an environment but a Strictly Declarative Modelling Language having object orientation features and being logic based [16] written in Smalltalk, see `http://sdml.cfpm.org/`. Knowledge is represented in rule bases and data bases and the main reasoning mechanism used is forward and backward chaining. Agents may be assigned rules which determine their behaviour and which can be shared with other agents. The latter is possible due to the object orientation features. The fact that SDML is strongly grounded in logic allows formal proofs of the completeness of the model to be constructed. Programming is conducted via a series of windows. The introduction to SDML on the Web site gives a good feel of the interface.

Sophisticated simulations may be built using SDML involving complex interacting organisations, deeply nested levels of agents and the ability for agents to possess limited cognitive abilities. However, the language has a steep learning curve.

Whilst SDML was specifically developed for building simulations in the social sciences, most of the available models are concerned with economic and market modelling. Indeed, apart from the SDML Web site at Manchester Metropolitan University, there are very few examples of simulations using SDML.

SDML does however provide features useful in modelling cognitive social agents. There is no inherent theory of cognition implemented in SDML so any agent cognition is represented as sets of rules. Communication between agents is achieved via data bases: the result of a fired rule is written to an agent's data base which may be accessed via another agent. The accessibility of one agent's data base to another agent's data base can be restricted by assigning a status to the rule's clause, e.g. private or public. Agents may also evaluate each other as being possible "collaborators" and endorse other agents as being a reliable, un-realiable, successful or unsuccessful collaborator.

SDML runs in a Smalltalk environment and is available for MS Windows 3.1/95/98/2000/NT, Linux, Intel, PowerMac, Unix ADUX/AIX/HPUX/SGI/Solaris.

### 2.32 SimAgent

SimAgent was developed by Aaron Sloman of University of Birmingham. It was designed for rapidly implementing and testing out different agent architectures, including scenarios where each agent is composed of several different sorts of concurrent interacting sub-systems, in an environment where there are other agents and objects. Some agents should have sensors and effectors and some should be allowed to communicate with others. Some agents should have hybrid architectures including, for example, symbolic mechanisms communicating with neural nets. The toolkit is also used for exploring evolutionary processes. It uses the Pop-11 language in the Poplog software development environment. See `http://www.cs.bham.ac.uk/research/projects/poplog/packages/simagent.html`.

### 2.33 SimPack

Simpack, developed by Paul Fishwick, is a directory of tools supporting discrete event simulation, see `http://www.cise.ufl.edu/~fishwick/simpack.html`. SimPack supports a wide variety of event scheduling and continuous time simulation models. There are models of the sort described in his 1995 book [5] and examples using the Processing language, see `http://www.processing.org`.

SimPack was originally written in C, and this version is still available but is no longer maintained or upgraded. In the latest version the models are executed using Java. SimPack is has a GPL license.

### 2.34 SeSAm

See `http://www.simsesam.de/`.

### 2.35 SimPy

See `http://simpy.sourceforge.net/`.

### 2.36 SOARS

See `http://soars.jp/`.

### 2.37 Open StarLogo

See `http://education.mit.edu/openstarlogo/`.

### 2.38 StarLogo

StarLogo is a programmable modelling environment specifically aimed towards exploring de-centralised systems via simulation. It is a specialised version of Logo, which was used for teaching in schools. StarLogo allows the user to create and control the behaviour of "turtles", a term used in Logo. Turtles move around a user defined landscape that is make up of "patches".

Whilst StarLogo can be considered "agent based", for example, a turtle is an agent, its programming paradigm is procedural as opposed to object oriented. It provides a set of commands which the programmer uses to create and control the turtles and patches.

In practice, StarLogo is very easy to use. It provides a graphical interface to help the developer code their simulations. This can be used to create graphs of simulation data and to define buttons and slide bars which control the simulation and define the input data, e.g. number of turtles. However, whilst it is easy to graph data in StarLogo there are some annoying problems associated with the graphing facility, for example whilst many lines may be plotted on the same graph it is not possible create more than one graph. With the current version 1.2 of StarLogo, it is quite easy to put your simulations on a Web page as an Applet for viewing.

StarLogo is available for the Mac or PC. However, simulations which are developed and run on one platform will not run on the other.

There are a lot of examples of StarLogo simulations available on the Internet and there is a very good support mailing help group, see `http://www.media.mit.edu/starlogo/`. One of the main downfalls of StarLogo however is its inflexibility. The set of commands offered by StarLogo may be quite restrictive if we are aiming to code complex social mechanisms. It is not impossible to code such things, but it may be quite frustrating and challenging to find ways to do exactly what you want given the commands provided. In addition, whilst it is not necessary to have a lot of programming experience, care must be taken to avoid writing inefficient code which can make your simulations frustratingly slow.

### 2.39 SugarScape

See `http://sugarscape.sourceforge.net/`.

## 2.40 Swarm

Swarm, the first re-usable software tool created for agent based modelling and simulation, was developed at the Santa Fe Institute in 1994. Swarm was specifically designed for artificial life applications and studies of complexity.

Swarm was originally developed for multi agent simulation of complex adaptive systems. Until recently the Swarm project was based at the Santa Fe Institute but its development and management is now under control of the Swarm Development Group whicvh has wider membership to sustain the software.

Swarm was designed as a general language and toolbox for ABMS, intended for widespread use across scientific domains. Swarm's developers started by laying out a general conceptual approach to agent based simulation software. Key to Swarm is the concept that the software must both implement a model and, separately, provide a virtual laboratory for observing and conducting experiments on the model. Another key concept is designing a model as a hierarchy of "swarms", a swarm being a group of objects and a schedule of actions that the objects execute. This is similar to the concepts of context and project now included in Repast Simphony. One swarm can contain lower level swarms whose schedules are integrated into the higher level swarms; simple models have a lower level "model swarm" within an "observer swarm" that attaches observer tools to the model.

The software design philosophy appears to have been to include software that implements Swarm's modelling concepts along with general tools likely to be useful for many models, but not to include tools specific to any particular domain.

Swarm was designed before Java's emergence as a mature language. One reason for implementing Swarm in Objective-C was that the lack of strong typing in this language (in contrast to for instance C++), supports the complex systems philosophy of lack of centralised control. This means that a model's schedule can tell a list of objects to execute some action without knowing what types of object are on the list. Swarm uses its own data structures and memory management to represent model objects. One consequence is that Swarm fully implements the concept of "probes" – tools that allow users to monitor and control any simulation object, no matter how protected it is, from the graphical interface or within the code.

Swarm provides a set of libraries which the developer uses for building models and analysing, displaying and controlling experiments on those models. The libraries are written in Objective-C and until recently building a simulator meant programming in a mixture of Objective-C and Swarm. However, it is now possible to use Java with some Swarm to calls upon the facilities offered by the libraries. The Java "extension" is useful since Java now a very popular and powerful language. The next release of Swarm, will support JavaScript and Scheme in addition to Objective-C and Java.

In the Swarm system, the fundamental component that organises the agents of a Swarm model is a "swarm". A swarm is a collection of agents with a schedule of events over those agents. The swarm represents an entire model: it contains the agents as well as the representation of time. Swarm supports hierarchical modelling whereby an agent can be composed of swarms of other agents in nested structures. In this case, the higher level agent's behaviour is defined by the emergent phenomena of the agents inside its swarm. This multi-level model approach offered by Swarm is very powerful. Multiple swarms can be used to model agents that themselves build models of their world. In Swarm, agents can themselves own swarms, models that an agent builds for itself to understand its own world.

The actual model and the task of observing the model is clearly separated in the Swarm system. In Swarm, there are special "observer" agents whose purpose it is to observe other objects via the probe interface. These objects can provide both real time data presentation and storage of data for later analysis. The observer agents are actually swarms as noted above – a group of agents with their own schedule of activity. Combining the observer swarm with the model swarm gives a complete experimental framework – the model and observer apparatus. With other simulation tools the distinction between the actual model and the code needed to observe and collect data from the model is blurred making it difficult to change one part without influencing the other. Separating the model from its observation is a pattern enabling model itself to remain pure and unchanged if the observation code needs to be modified.

Java Swarm was designed to provide, with as little change as possible, access to Swarm's Objective-C library from Java. Java Swarm was motivated by a strong demand among Swarm users for the ability to write models in Java, not by the objective of providing Swarm's capabilities as cleanly and efficiently as possible in Java. Java Swarm therefore simply allows Java to pass messages to the Objective-C library with work arounds to accommodate strong typing in the Java language.

Swarm is probably still the most powerful and flexible simulation platform. However, this comes at a price. In practice, Swarm has a very steep learning curve. It is necessary to have experience of Java or Objective-C, be acquainted with the object orientation methodology and be able to learn some Swarm code.

In terms of additional support, there are excellent support mailing lists with prompt and helpful responses and a lot of generally available Swarm, Java and Objective-C code. There is also an annual meeting of the Swarm Users Group called SwamFest where researchers from diverse disciplines present their experience with multi agent modelling and the Swarm simulation system. Immediately preceeding SwarmFest there is usually a tutorial for inexperienced users on how to use Swarm. Swarm models can already run inside a Web browser, specifically Netscape. However, a future development goal is for Swarm to be a complete interactive, browser based development environment for agent based models.

Swarm runs on any platform. With reference to building simulations for the social sciences, Swarm would be one of the best packages to use, being so powerful and flexible that it would be possible to implement very intricate and complicated social mechanisms. The only pre-requisite is finding a programmer experienced enough to be able to implement what was needed.

The Objective-C version of Swarm is the most mature ABMS library based package and is stable and well organised. Objective-C seems more natural than Java for ABMs but weak error handling and the lack of developer tools are drawbacks. Java Swarm allows Swarm's Objective-C libraries to be called from Java, but it does not seem to combine advantages of the two languages well.

Swarm typically runs on Linux machines with GNU Objective-C (in gcc v3.4 onwards) and X-windows. The source code is freely available under a GNU license. We have implemented Swarm v2.3.0 on a Gentoo Linux system for evaluation purposes.

Add-ins and extensions to Swarm are available including:

- COSMIC – General simulation utility classes, provided by the Complex Systems Modelling Group at Imperial College `http://www.swarm.org/index.php/COSMIC`

- EcoSwarm@HSU – Modeling tools for use with the Swarm simulation system `http://www.humboldt.edu/~ecomodel/`
- LogZone – Zone exploration tool `http://me.in-berlin.de/~rws/logzone_toc.html`

### 2.41 VisualBots

Easy to use multi-agent simulator for Microsoft Excel - Visual Basic syntax, rich object model, documentation, sample simulations, see `http://visualbots.com/`.

### 2.42 Xholon

See `http://www.primordion.com/Xholon`.

### 2.43 Zeus

The ZEUS toolkit, developed by British Telecommunications (BT), provides a library of software components and tools that facilitate the rapid design, development, and deployment of agent systems. The three main functional components of the ZEUS toolkit are an agent component library, agent building tools, and visualisation tools. See `http://labs.bt.com/projects/agents/zeus/`.

## 3 ABMS Applications

The most common uses of ABMS are in social simulation and optimisation problems, such as traffic flow and supply chains. We will investigate other uses in computational science and engineering. Computational advances have opened the way for a growing number of agent based applications across many fields. These applications now range from modelling adaptive behaviours and the emergence of new entities in the biological sciences to modelling agent behaviour in the stock market and supply chains to understanding consumer purchasing.

ABMS is being actively applied in many practical areas. The applications range across a continuum, from elegant, minimalist academic models to large scale decision support systems.

Minimalist models are based on a set of idealised assumptions, designed to capture only the most salient features of a system. These agent based models are exploratory electronic laboratories in which a wide range of assumptions can be varied over a large number of simulations. Decision support models tend to serve large scale applications and are designed to answer real world policy questions. These models normally include real data and have passed appropriate validation tests to establish credibility.

- Business and Organizations
  - Consumer markets

  - Supply chains
  - Insurance
  - Manufacturing
- Economics
  - Artificial financial markets
  - Trade networks
- Infrastructure
  - Transportation
  - Electric power markets
  - Hydrogen economy
- Crowds
  - Human movement
  - Evacuation modelling
- Society and Culture
  - Ancient Civilizations
  - Civil disobedience
- Terrorism
  - Social determinants
  - Organizational networks
- Military
  - Command and control
  - Combat
  - Logistics
- Biology
  - and Ecological systems
  - Animal behaviour
  - Cell behaviour
  - Subcellular molecular behaviour

## 3.1 Physics and Chemistry

- Design of self organizing materials and self organizing systems; directed self assembly; use of fields to govern self organization, modeling in terms of agent based algorithms.
- Modeling of fluid flows and flow and segregation in granular matter.
- Modeling of gene and protein interaction networks, immunology.
- Complex reactions, network analysis of the fate of pollutants, leading to science based environmental policies.
- Understanding of product and manufacturing supply chains. Supply chains in economics.
- Studies of complex fluctuations in physiologic systems including the ability of systems to respond to multiple environmental stimuli.
- Design of safety critical systems; analysis of failures in distributed systems
- Study of propagation of epidemics.
- Understanding and evolution of organizations, including the design of structures for scientific and technological collaboration.

## 3.2 Biology and Medicine

See `http://www.swarm.org/index.php/Agent-Based_Models_in_Biology_and_Medicine`.

Some of the biological applications of ABMS have focussed on artificial life studies. Other studies are directed to understanding the biology of cells, organs and organisms.

In medicine, AMBS has been applied to: epidemiology and infection; acute inflammation; immunology; cancer and tumours; wound healing; vascular system; signaling and metabolic process. All the tissues in our bodies (to be more general, all multi-cellular creatures) self assemble. The "rules" for doing this are in each cell – in the genetic material. There is no information at a higher level of organisation than the individual cell, so all the organisation in tissues and organs and organisms is an 'emergent property' of the interaction of large numbers of individual cells – $10^13$ in a human. That is what we are interested in – how does this social interaction of the cells produce properly functioning and structured creatures?

In the biological sciences, Argonne researchers are using ABMS to model cellular behaviour. Using the 350-node Jazz cluster, researchers developed AgentCell, an agent based simulator for modelling the chemotactic processes involved in the motile behaviour of the Escherichia coli bacteria. AgentCell is an open source tool, based on the Repast Simphony toolkit developed by Argonne and University of Chicago. The research examined how the range of natural cell diversity is responsible for the full range of cell behaviours.

The researchers used Jazz to run numerous simulations of the E. coli chemotaxis network for many cells and stochastic variations. They modeled agents at the molecular level as well as at the whole cell

level. Argonne is now investigating the efficient implementation of agent-to-agent interactions that minimize resource contention through its exascale computing research effort.

Large scale simulations of digital bacteria run on the Jazz computing cluster suggested that the chemotaxis network is tuned to simultaneously optimize the random spread of cells in the absence of nutrients and the cellular response to gradients of nutrients. Without the computing time on Jazz, this work would have been very difficult to complete. These computations have paved the way for closer collaborations between analytical treatments of, computational modelling of, and wet lab experimentation with E. coli behaviour, which potentially has broad implications for many biological systems.

At Argonne, three pilot application areas are being targetted – microbial biodiversity, cybersecurity and the social aspects of climate change. These are providing a portfolio of requirements for exascale ABMS. Each application area suggests a unique series of experiments that cumulatively cover the range of functionality required for an agent based architecture. Depending on the findings of the experiments, a variety of possible designs and implementation paths will be considered. Recent Argonne work on the Repast Simphony ABMS toolkit offers one possible path among several that will be experimentally tested. This toolkit has been used successfully for a wide range of applications. For example, the National Aeronautics and Space Administration (NASA) used Repast for an agent based simulation on autonomous robots roaming the Martian surface and a dynamic social network simulation from Repast.

Enabling large scale simulations to address the complexity of real microbial environments is a major focus of this research. *Scalable ABMS simulations show great promise for understanding the detailed dynamics of large, mixed microbial communities, with a wide range of applications in ecology, health sciences, and industry*, says Rick Stevens, associate laboratory director for Computing, Environment, and Life Science at Argonne. The rapid accumulation of environmental molecular data is uncovering vast diversity, abundant un-cultivated microbial groups, and novel microbial functions. This accumulation of data requires the application of theory and simulation to provide organization, structure, insight and ultimately predictive power that is of practical value. Argonne researchers are using resource ratio theory in microbial ecosystems in devising requirements for the practical application of agent based modelling to the exascale system under development.

### 3.3 Security

Cybersecurity also offers considerable opportunities to apply exascale ABMS. There is a need to develop advanced methods to explore future cybersecurity scenarios that involve a variety of defensive strategies. Simulation techniques are one approach that may yield insight to vulnerabilities and security dynamics. The scope of simulation includes the systems, such as computers, networks, routers, filters and monitors, users of these systems, administrators, configuration processes, cybersecurity policies, and external threat agents.

The information technology (IT) infrastructures are large scale systems with 106 to 1010 IT entities consisting of a few hundred types. Each entity can have a very large number of possible internal states and many thousands of users and administrators. These systems are geographically dispersed and connected by complex networking with constantly evolving topologies. Agent based modelling is a natural methodology for simulating such infrastructures since both the technical dimensions (for

example, the server protocols) and the human dimensions, such as using "social engineering" to gain user passwords, of these systems can be simultaneously modeled. Argonne researchers are developing prototype models of typical IT infrastructures and the associated mechanisms to explore current and alternative administrative policies. The linking of human behaviour with network structure offers the possibility of breakthroughs in the fundamental understanding of cybersecurity.

## 3.4 The Environment

Improving scientific understanding of climate change requires researchers to consider the physical, economic, and social determinants of climate change. Modelling all three aspects is essential for long term climate forecasting since economic and social decision making has a potentially important effect on physical climate factors (such as individual fuel choices and how use of energy technologies may affect atmospheric carbon dioxide concentrations). Similarly, physical climate factors may influence decision making (for example, rising atmospheric carbon dioxide concentrations may encourage people to reduce carbon dioxide emissions). Much progress has been made in modelling the physical aspects of climate change such as atmospheric flows, oceanic circulation, and albedo effects.

## 3.5 Social and Economic Modelling

Recent research in agent based economic and social modelling has opened the door to modelling the feedback loops inherent in the social aspects of climate change. The resulting scenarios will require modelling up to 1010 human agents. These agents would each have a large number of complex internal states (103 or more) and be drawn from a wide range of behavioural types.

FLAME is being used for economic modelling in the EURACE project. From the scientific point of view, the main effort in this project regards the study and the development of multi-agent models that reproduce, at the aggregate economic level, the emergence of global features as a self organized process from the complex pattern of interactions among heterogeneous individuals.

From the technological point of view, the project will develop, with advanced software engineering techniques, a software platform in order to realize a powerful environment for large scale agent based economic simulations. Key issues will be the definition of formal languages for modelling and for optimizing code generation, the development of scalable computational simulation tools and the standardization of data with easy to use human-machine interfaces.

Finally, from the social point of view, the agent based software platform for the simulation of the European economy aims to have an impact on the economic policy design capabilities of the European Union. It will be a powerful tool, enabling to perform "what if?" analysis, optimizing the impact of regulatory decisions that could be quantitatively based on European economy scenarios.

# 4 ABMS on HPC

## 4.1 HPC Clusters

See EURACE Web site `http://www.eurace.org` and [?] for information about FLAME on HPC clusters including NW-GRID and HPCx.

See SciDAC review [29] for information on work at Argonne National Laboratory on IBM BlueGene and Jazz cluster.

## 4.2 BlueGene

At Argonne, the goal is to develop an open source agent based modelling toolkit that is highly scalable. The agent based modelling toolkit being developed for the exascale ABMS system has four major architectural components: a time scheduler, a storage system for agent endogenous data, a storage system for agent topologies, and a data logging system. The storage systems for agent endogenous data and agent topologies are generally synchronized to form a single data storage framework.

Time schedulers coordinate and synchronize the flow of agent activities as events unfold in the simulation. Storage systems for agent endogenous data hold the internal state information for each agent (for example, each agent's scalar attributes). This component is also responsible for saving the agent's internal state information across executions of the model (such as storing input data or allowing runs to be replicated). Storage and access systems for agent topologies hold both the spatial and aspatial relationships between agents and manage the communications between agents that occur across these links. These systems are also responsible for saving the relationship information across separate executions of the model. Data logging systems record the activities within and the results of simulations (for example, who talked to whom, and when, in a social network). The information stored by these systems is used for simulation analysis and visualization.

Argonne researchers plan to complete the backbone of the exascale ABMS system within the next year. They will then work to extend the system, based on a focused series of experiments in each of the pilot application areas, as well as furthering the development of the models for each domain. The system will enable the researchers to work with microbiologists, social scientists, and cyber-security experts in advancing breakthrough science in these areas. In broader terms, the architecture and knowledge provided by this research will be applicable to many other areas as well.

## 4.3 GPGPU

ABMS has been adapted to run on novel architectures such as GPGPU (e.g. nVidia using CUDA). There is a Web site devoted to genetic programming for GPU – GPGPGPU at `http://www.gpgpgpu.com/`. References include the work of Lysenko and D'Souza at Michigan Technological University [26] and of Paul Richmond at Sheffield University `http://www.dcs.shef.ac.uk/~paul/abgpu.html`. This includes papers on Agent Based GPU Modelling [40, 41] and description of a GPU extension to the FLAME framework.

### 4.4 Cell

There are currently no ports of AMBS to Cell, although there are some genetic algorithms and libraries being developed, see `http://www.ibm.com/developerworks/power/cell/open_source.html`.

Some simpler simulations of crowd behaviour demonstrating flocking via a Boids algorithm have been published: PSBoids – Reynolds 2000 (on PS2), GEBS – Erra et al. 2004 (CPU+GPU), FastCrowd – Courty and Musse 2005 (CPU+GPU), and PSCrowd – Reynolds 2006 (on PS3).

## 5 ABMS Community

SWARM Developers Group Wiki `http://www.swarm.org/wiki/`.

Open Agent Based Modeling Consortium `http://www.openabm.org/site/` at Arizona State University. The OpenABM Consortium is a group of researchers, educators, and professionals with a common goal – improving the way we develop, share, and utilise agent based models. They are currently developing a model archive to preserve and maintain the digital artefacts and source code comprising an agent based model.

## References

[1] Alexander Repenning, Andri Ioannidou and John Zola *AgentSheets: End User Programmable Simulations* Journal of Artificial Societies and Social Simulations. Vol. 3, No. 3

[2] Joaquim Carvalho *Using AgentSheets to teach simulation to undergraduate students* Journal of Artificial Societies and Social Simulations. Vol. 3, No. 3

[3] AgentSheets Web site `http://www.agentsheets.com`

[4] Ascape Web site `http://www.brook.edu/es/dynamics/models/ascape`

[5] P.A. Fishwick *Simulation Model Design and Execution: Building Digital Worlds* (Prentice-Hall, 1995) 432pp ISBN 0-130-98609-2

[6] S. Luke, C. Cioffi-Revilla, L. Panait and K. Sullivan *MASON: a new Multi-Agent Simulation Toolkit* Swarm Fest (2004)

[7] Miles Parker *Presentation Slides on Ascape* `http://www.brook.edu/es/dynamics/models/ascape/UChicago/tsld001.htm`

[8] Miles Parker *What is Ascape and why should you care?* JASSS:Journal of Artificial Societies and Social Simulation (January 2001) `http://jasss.soc.surrey.ac.uk`

[9] Louis Foucart *A Small Multi Agent Systems Review* `http://geneura.ugr.es/~louis/masReview.html`

[10] MAML Web site including MAML User manual, examples, papers, etc. `http://www.maml.hu/maml/about/about.html`

[11] Slides describing MAML which was presented at SwamFest'99 `http://www.syslab.ceu.hu/maml/SwarmFest99`

[12] RePast Web site `http://repast.sourceforge.net/repast_3`

[13] M.J. North, N.T. Collier and J.R. Vos *Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit* ACM Transactions on Modeling and Computer Simulation 16:1 (2006) 1-25

[14] SDML Web site at Manchester Metropolitan University (the site includes tutorials, discussion papers, SDML download facility and mailing lists). `http://www.cpm.mmu.ac.uk/sdml`

[15] The SDML Beginners tutorial `http://www.cpm.mmu.ac.uk/sdml/intro/html/sdml_tut_1.html`

[16] S. Moss, H.Gaylard, S. Wallis and B. Edmonds *SDML: A Multi-agent Language for Organizational Modelling* Computational and Mathematical Organization Theory 4 (1) (1998) 43-69

[17] E. Lavery *Introduction to Agent Based Simulation and Flexsim* Flexsim Corp. (2008) `http://www.flexsim.com`

[18] Starlogo Web site at MIT includes examples and tutorials, etc. `http://www.media.mit.edu/starlogo/`

[19] Vanessa Stevens Colella, Eric Klopfer and Mitchel Resnick *Adventures in Modeling* ISBN: 0807740829 `http://www.media.mit.edu/starlogo/adventures/`

[20] Connected Mathematics Team at Northwestern University contains lots of models implemented in StarlogoT (for the Macintosh) and a useful list of links. `http://www.ccl.sesp.northwestern.edu/cm/`

[21] Starlogo sites at Maine University `http://www.asap.um.maine.edu/starlogo/`

[22] Fatima Rateb, Narges Bellamin and Bernard Pavard *The simulation of the Spread of Malaria in Haiti* developed at GRIC IRIT

[23] The Swarm Development Group contains tutorials, examples, community projects and code. `http://www.swarm.org/`

[24] Paul Johnson *Swarm User Guide* `http://lark.cc.ukans.edu/~pauljohn/Swarm/Beta/SwarmUserGuide/userbook.html`

[25] Paul Johnson's Swarm HQ includes lots of examples of Swarm code `http://lark.cc.ukans.edu/~pauljohn/Swarm/`

[26] M. Lysenko and R. D'Souza *A Framework for Megascale Agent Based Model Simulations on Graphics Processing Units* Journal of Artificial Societies and Social Simulation 11:4 (2008) 10 `http://jasss.soc.surrey.ac.uk/11/4/10.html`

[27] EcoLab Web site `http://ecolab.sourceforge.net/`

[28] R. Standish *EcoLab Documentation* `http://ecolab.sourceforge.net/doc/ecolab/ecolab.html`

[29] M.J. North and C.M. Macal *Agent Based Modelling and Simulatino for Exascale Computing* SciDAC Review (Feb'2008) `http://www.scidacreview.org/0802/html/abms.html`

[30] M. J. North and C. M. Macal *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation* (Oxford University Press, New York, 2007)

[31] C. M. Macal and M. J. North *Tutorial on Agent-Based Modeling and Simulation: Desktop ABMS* Proc. 2007 Winter Simulation Conference. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton (eds.), (Washington, DC, December 2007) 95-106 `http://www.informs-sim.org/wsc07papers/011.pdf`

[32] *ZooLand: the Artificial Life Resource* An interesting resource, but with many broken links. `http://surf.de.uu.net/zooland/`

[33] A.L. Paredes and C.H. Iglesias (eds.) *Agent Based Modelling in Natural Resource Management* (INSISOC, Spain, 2008), `http://www.insisoc.org/INSISOC/INSISOC_archivos/ABMbook/ABMbook.htm`

[34] `http://cscs.umich.edu/~crshalizi/notebooks/agent-based-modeling.html`

[35] S.F. Railsback, S.L. Lytinen and S.K. Jackson *Agent Based Simulation Platforms: Review and Development Recommendations* Simulation 8:9 (2005) 609-23 `http://www.humboldt.edu/~ecomodel/documents/ABMPlatformReview.pdf`

[36] R. Tobias and C. Hofmann *Evaluation of free Java-libraries for social scientific agent based simulation* Journal of Artificial Societies and Social Simulation 7:1 (2004) `http://jasss.soc.surrey.ac.uk/7/1/6.html`

[37] M. Matsumoto and T. Nishimura *Mersenne Twister: a 623-dimensionally equidistributed uniform pseudorandom number generator* ACM Transactions on Modeling and Computer Simulation 1998 (8) pp3-30

[38] G. Marsaglia, A. Zaman and W.W. Tsang *A Universal Random Number Generator* Statistics and Probability Letters 8 (1990) 35-39

[39] *FLAME: FLexible Agent Modelling Environment* `http://www.flame.ac.uk`

[40] P. Richmond and D. Romano *Agent Based GPU, a Real Time 3D Simulation and Interactive Visualisation Framework for Massive Agent Based Modelling on the GPU* `http://www.dcs.shef.ac.uk/~paul/abgpu.html`

[41] P. Richmond and D. Romano *A High Performance Framework for Agent Based Pedestrian Dynamics on GPU Hardware* `http://www.dcs.shef.ac.uk/~paul/pedestrians.html`

[42] Journal of Artificial Societies and Social Simulation `http://jasss.soc.surrey.ac.uk/JASSS.html`

[43] Complexity International `http://journal-ci.csse.monash.edu.au/ci/info-journal.html` An electronic refereed journal including a wide range of papers on complexity theory.

[44] Complexity Digest `http://www.comdig.org/` A weekly newsletter about complexity in the natural and social sciences, which includes links to relevant reviews, notices of articles, conference announcements and so forth;

[45] Artificial Life Online `http://www.alife.org/` An online companion to the (paper) journal Artificial Life, published by the Santa Fe Institute.

[46] A. Serenko and B. Detlor *Agent Toolkits: A General Overview of the Market and an Assessment of Instructor Satisfaction with Utilizing Toolkits in the Classroom* Working Paper 455, McMaster University, Hamilton, Ontario, Canada (2002)

[47] N. Gilbert and S. Bankes *Platforms and Methods for Agent-based Modeling* Proc. National Academy of Sciences of the USA 99:3 (May 14, 2002) 7197-8

[48] N. Collier, R. Howe and M. North *Onward and Upward: The Transition to Repast 2.0* Proc. 1st Annual North American Association for Computational Social and Organizational Science Conference. (Pittsburgh, PA, USA, June 2003)

[49] M. Valente and E.S. Anderson *A Hands-On Approach to Evolutionary Simulation: Nelson-Winter Models in the Laboratory for Simulation Development* The Electronic Journal of Evolutionary Modeling and Economic Dynamics, 1003:1 (January 15, 2002) `http://www.e-jemed.org/1003/index.php`

[50] C. Deissenberg, S. van der Hoog and H. Dawid *EURACE: a Massively Parallel Agent Based Model of the European Economy* Applied Mathematics and Computation 204 (2008) 541-552