Science & Technology Facilities Council

Technical Report
DL-TR-2010-007

Survey of Agent Based Modelling and Simulation Tools

RJ Allan

October 2010

Chadwick Library
Science and Technology Facilities Council
Daresbury Laboratory
Daresbury Science and Innovation Campus
Warrington
WA4 4AD

Tel: +44(0)1925 603397
Fax: +44(0)1925 603779
email: library@dl.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk/

ISSN 1362-0207

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: robert.allan@stfc.ac.uk

October 7, 2010

Abstract

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

There is currently a lot of interest in developing ABMs as a general computational technique applicable to the study of large scale systems. ABMs has been adapted to run on novel architectures such as GPGPU, e.g. nVidia hardware using the CUDA programming language. Argonne National Laboratory have a Web site on Exascale ABMs and have run models on the IBM BlueGene with funding from the SciDAC Programme.

The most common uses of ABMs are in social simulation and optimisation problems, such as crowd behaviour, urban simulation, traffic flow and supply chains. We will investigate other uses in computational science, particularly in engineering and systems biology.

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 in the study of realistic complex systems. Argonne's exascale computing programme is for instance working to make this possible.

Agent based modelling packages which do anything beyond trial examples also tend to be hard for a novice to understand. This is partly because the frameworks and (mostly object oriented) languages are complicated and, in many cases, the packages do not have simple APIs, GUIs or IDEs and partly because each 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 – a systems based approach.

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`. An even longer list of references was compiled by the AgentLink EU project, `http://eprints.agentlink.org/view/type/software.html`. We have chosen the ones we currently believe to be of particular relevance to scientific modelling and simulation. A review of eight ABMs packages relevant for geo-spatial analysis is presented here `http://www.spatialanalysisonline.com/output/html/Simulationmodellingsmsystemsforagent-basedmodelling.html`. There is another list here `http://www.idsia.ch/~andrea/sim/simagent.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 at the time we tried it. 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 tools.

There are indeed a number of challenges which must be faced in order to make ABMs a mainstream computational science technology. These are described in Section 6.

1.1 Terminology

The following table taken from [55] 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	steppable	procedure	action	action
Queue of events executed repeatedly	schedule	forever procedure	schedule	schedule

Another use of terminology is extremely confusing. There is a fine line between agent based simulation and modelling (AMBS) and multi-agent systems (MAS). The former are used to simulate complex systems such as social networks and biology which exhibit emergent behaviours. The latter can also be used to simulate complex environments, such as supply chains. These could be referred to as "smart" applications and components are called "intelligent agents". We attempt to distinguish beteen the two and, whilst this report focusses on the former and appropriate packages are identified in Section 2, we have included some packages from the latter category in Section 3.

1.2 Comments on Object Oriented Modelling

Object Orientated Programming (OOP) is widely adopted as the most common paradigm for ABM frameworks. OOP offers a natural and simple technique for modelling which is easily understood by software engineers who are familiar with object orientated design patterns. Agents can be considered to be self directed objects with the capability of choosing actions autonomously based on their environment. It is natural to use classes and methods to represent agents and agent behaviours. The majority of popular ABM frameworks are therefore based on OO principles, some even use concepts such as UML for high level agent and system specification. Some are accessible through an Application Programming Interface (or API) and application layer. The API then provides a tool for building and describing models, whilst the application framework implements common routines such as agent communication and scheduling of agent behaviour and interactions.

C.R. Shalizi [67] comments that *Fundamentally, I'm not sure that agent based modeling amounts to anything other than object oriented programming for dis-aggregated simulations – which is a very useful thing, of course.*

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. See also the AgentLink Roadmap [35].

In a presentation at the Leeds workshop on 15/6/2010, Salem Adra provided some more definitions. He first noted a comment from a paper by Jennings *et al.* [28] *object oriented programmers often fail to see anything novel or new in the idea of agents. ... While there are obvious similarities, there are also significant differences between agents and objects.* Note however that this paper is really discussing multi-agent systems (MAS) for smart distributed applications rather than ABMS.

He went on to illustrate this as follows. First from the evoluation of programming approaches, after [47, 52].

	Monolithic Programming	Modular Programming	Object Oriented Programming	Agent Programming
Unit Behaviour	Non-modular	Modular	Modular	Modular
Unit State	External	External	Internal	Internal
Unit Invocation	External	External (called)	External (message)	Internal (rules, goals)

Secondly showing the main differences between objects and agents. The latter are characterised by autonomy and flexibility.

Objects	Agents
Behaviour controlled by external entities	Self goverened (rules and goals)
Always says "yes" ("no" is an error)	Allowed to say "no"
Predictable behaviour (static functionality which facilitates inheritance)	Un-predictable. Behaves differently in different scenarios (can learn from experience and evolve)

By contrast, FLAME (see Section 2.9) adopts the use of a formal technique for model specification called the X-Machine [13]. Formal methods are advantageous as they provide a technique for both specification and validation. Whilst formal specification is useful in the generation of system implementations, validation is invaluable as it allows automatic verification and error checking of systems. In the case of high integrity or mission critical systems the guarantee of reliability and correct behaviour is not only advantageous, but essential.

1.3 Petri Net Representation

There is some discussion that ABMs can be represented as Petri Nets. For a definition of the latter see Wikipedia: `http://en.wikipedia.org/wiki/Petri_net`. Petri nets are particularly good for describing finite state machines, see `http://en.wikipedia.org/wiki/Finite-state_machine`.

This is particularly clear in FLAME with its state graph output showing states and transitions in each agent cycle. Figure 1 is the full state net for the Keratinocyte model encoded using FLAME by Sun, McMinn *et al.* [71]. Transitions are shown by labels on the arrows and cell states ("places" in the usual Petri net vocabulary) are shown as circles.

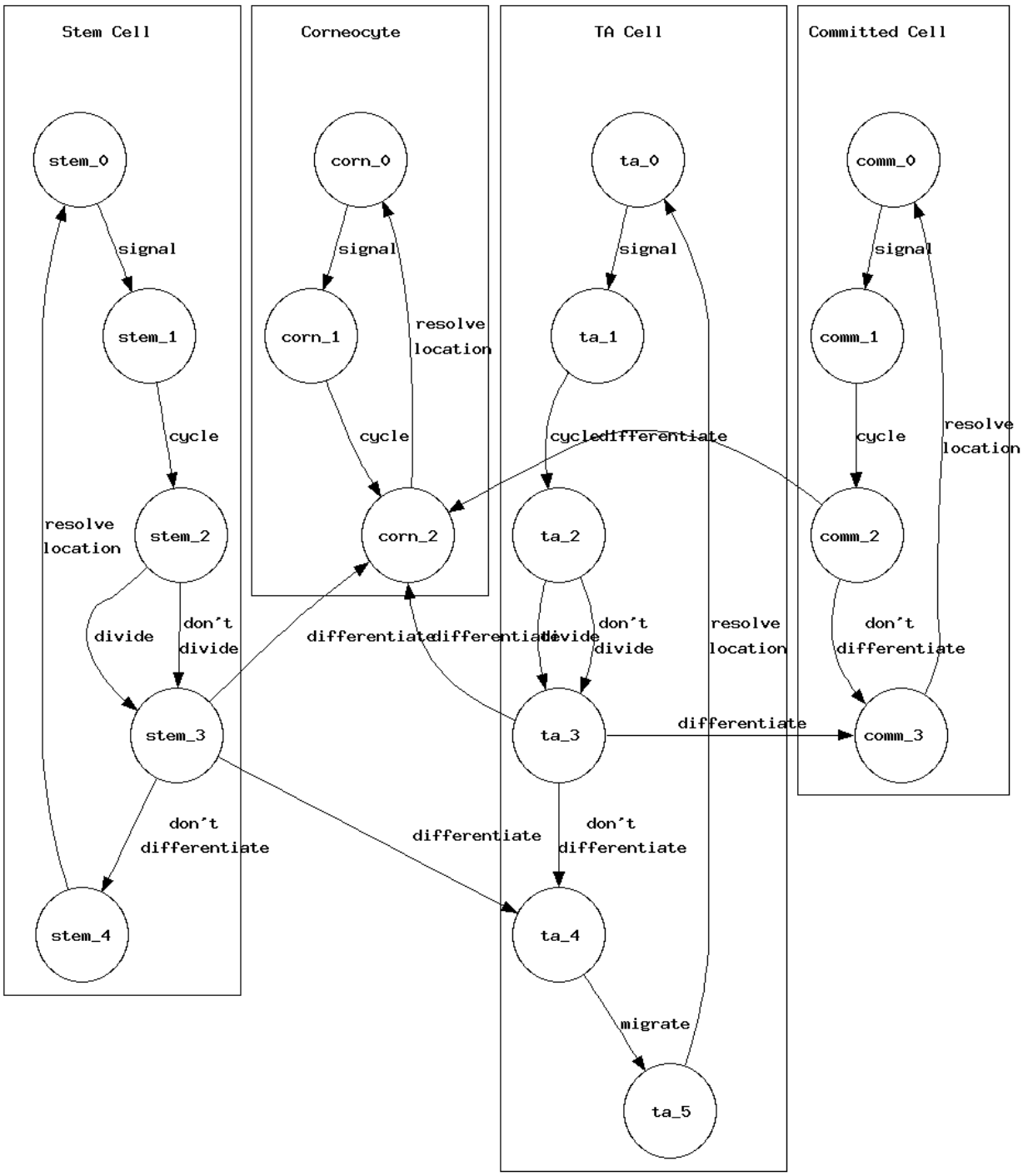

Figure 1: Petri Net like Representation of Keratinocyte Cell Cycle.

1.4 Comments on Random Number Generation

Most of the packages use the "Mersenne twister" random number generator [40] and optionally allow users to provide their own seed so the sequence of pseudo-random numbers can be repeated. 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 [39] which is available coded in a number of languages. We have included this in the FLAME models that have been evaluated.

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 since diverged significantly from Swarm. More recently, MASON has been 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 [72]. 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.* [55] 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. The rest of this section simply comments on a number of existing packages in alphabetical order.

2.1 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.

AgentSheets is specifically aimed at non-programmers and as a consequence it is very simple to use. It has been used for teaching in social studies, mathematics, sciences and social sciences. 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 fact that AgentSheets is intuitively easy to understand 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 become more sophisticated, the weaknesses are 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 such situations are met frequently in simulations, it would make AgentSheets too complicated and inefficient to use. In addition, there is no "long distance" vision making it impossible to examine the status of an agent elsewhere on the mesh. AgentSheets-3 now runs on both Mac and Windows and has introduced the "Conversational Programming" environment.

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 run interactively through a Web browser.

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

2.2 AndroMeta

AndroMeta, formerly DeX, Dynamic Experimentation Toolkit, is an object oriented C++ framework for developing, analysing and visualising dynamic agent based and multi-body simulations. DeX is a discrete event simulation system 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. AndroMeta also includes a high level meta-modelling language interface. The software is available for free download and runs under Linux and Mac OS, see `http://dextk.org/dex` which now re-directs to `http://andrometa.net` since 2009.

2.3 AnyLogic

AnyLogic-6.5 is a proprietary offering that incorporates a range of functionality (discrete event, system dynamics and agent based) 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 charting model output dynamically. 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` shows many examples of models that 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); the location of emergency services and call centres; and pedestrian dynamics. There are also on-line video tutorials. However, the source code of these examples and/ or documentation of these models is not available.

2.4 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 where the well known SugarScape model was also developed.

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 somewhat easier to develop models with Ascape than with Swarm. Indeed, it is intended to allow people with only a little programming experience 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 developers of simulation models in Ascape would require some ability to program in Java together with understanding of the object orientation philosophy. For practical development, it would be useful to download and use a JDK. In addition, models which have been developed 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 Ascape-5. 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 models seem to be concerned with economic and market modelling.

2.5 Breve

Breve-2.7.2 is a free software package used as a teaching tool that provides a 3D environment for the simulation of de-centralised systems and artificial life. Users define the behaviours of agents in a 3D world and observe how they interact. Breve includes a Python based scripting language, physical simulation and collision detection for the simulation of realistic "creatures", and an OpenGL display engine so that users can visualise their simulated worlds. It is available for Mac, Linux, and Windows platforms. See `http://www.spiderland.org`.

2.6 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 represented as Smalltalk generic classes from which, by specialisation and refinement, users can create specific entities for their own model. See `http://cormas.cirad.fr`.

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.

2.7 DEVS: Discrete Event System Specification

DEVS is a formalism from B.P. Zeigler of the Arizona Center for Integrative Modeling and Simulation. It is claimed (`http://en.wikipedia.org/wiki/DEVS`) that DEVS is an extension of the Moore finite state machine. A number of tools from various authors have been designed using DEVS. See also `http://www.acims.arizona.edu/SOFTWARE/software.shtml`. The DEVS formalism and its variations have been used in many application of engineering, such as hardware design, hardware and software co-design, communications systems, manufacturing systems and science such as biology and sociology. The latest software package is CoSMoS which includes several previous applications as components (requires Java JDK-1.5.

2.8 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 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-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. Current version from June 2010 is v4.D37.

2.9 FLAME: FLexible Agent Modelling Environment

FLAME is being developed primarily at the University of Sheffield, see `http://www.flame.ac.uk` and `http://www.flamegpu.com` 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 principally at the medical and biological domains, for example studies of tissue cultures and signalling pathways. It was 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 little effort. Parallelisation methods using MPI 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 communicating 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.

We note that the parallel implementation of FLAME uses message board technology for communication between agents. This is based on libmboard, see `http://www.softeng.rl.ac.uk/st/projects/libmboard`.

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

Key points to FLAME are as follows.

FLAME is a modelling environment allowing high performance agent based modelling on parallel architectures.

Modellers do not require specialist knowledge of the underlying architecture used for simulation, as models are designed using formal specification techniques.

Efficient algorithms for inter-agent communication and birth and death allocation ensure maximum simulation performance.

The system is based on a simple XML syntax which is easily extendable.

Performance of complex cellular tissue models has been increased drastically.

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. It has been suggested that it could be the basis of a UK flagship effort to develop an ABM platform for a Collaborative Computational Project [98].

2.10 JAS: Java Agent Based Simulation Library

JAS is an Italian project to develop a simulation toolkit specifically designed for agent based simulation and modelling. JAS is a Java clone of the Swarm library. The core of the JAS toolkit is its simulation engine based on discrete event simulation, 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`. The last version available seems to be v1.2.1 from March 2006.

2.11 LSD: Laboratory for Simulation Development

This is annother new and active Italian project with a framework written by Marco Valente of University of l'Aquila in C++ [74]. See `http://www.labsimdev.org/Joomla_1-3`. It has a focus on economic models and social science. Tesfatsion notes that LSD applications take a systems dynamics (difference or 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.

2.12 MAML: Multi-Agent Modelling Language

The MAML language and xmc compiler were developed at the Complex Adaptive Systems Laboratory of the Central European University in Hungary 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.

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 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 graphical 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 by the MAML compiler (called xmc). The resulting application is then compiled in the same way as normal Swarm code by gcc. Currently xmc 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-0.03 and xmc-0.03.2 both in alpha release – xmc only compiles for Swarm-2.1.1 at latest.

Some background documentation on MAML is provided: a tutorial with source code, reference manual and a technical manual, but prior knowledge of Swarm is needed. 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.

2.13 MATSim

MATSim is a multi-agent transport simulation toolkit developed at TU Berlin and EHT Zürich. It is aimed at large and fast simulations for traffic flow management and urban planning purposes. It has also been used for evacuation scenarios. MATSim uses a modular approach making it easily customisable. There is active development with a release in Spring 2010 and a user meeting in June 2010. There is also a tutorial, user guide, FAQ and mailing lists and there is some support for multi-core systems, see `http://www.matsim.org`. Java-1.6 is required for the latest release (v0.1.1) which is available from SourceForge, see `http://sourceforge.net/projects/matsim/files`.

2.14 MASON: Multi-Agent Simulation of Neighbourhoods

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 re-producibility across hardware. MASON's developers support general rather than domain specific tools. The core models will run independently of visualisation which can be added. Checkpointing and migration of models is provided. Indeed, the abilities to detach and re-attach graphical interfaces and to stop a simulation and move it between computers are considered a priority for long simulations.

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. The MASON system, its motivation and its basic architectural design are described in [36]. Five applications of MASON are also described.

MASON contains both a model library and an optional suite of visualisation 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 the current v14 may be downloaded from `http://cs.gmu.edu/~eclab/projects/mason`. Java-1.3 JDK or higher is required with the Sun Java3D framework.

2.15 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. The software offers user friendly interfaces and wizards for writing models, creating visualisations and analysing simulation data.

The related Functional Agent Based Language for Simulation, FABLES, is an easy-to-use programming

language 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 easy. In the first public release of MASS, the simulation core is Repast-J, meaning that all FABLES models are compiled with Repast-J. NetLogo models can also be run.

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 and analysing results. It allows the user to run simulations in batches with various parameter settings, to store, manage and analyse the data.

These tools can be downloaded from the Web site and run on a Windows, Linux or Mac system. The latest release is from April 2009, see `http://mass.aitia.ai`.

2.16 MetaABM

A model driven ABM system developed 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 Score, a component of the Repast Simphony system, 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 multiple 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 eco-system. The contributors are committed to an open, developer driven approach and welcome participation from other individuals, projects and organisations.

2.17 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 language that considers the special demands of modelling in social science, especially the description of non-linear 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 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`.

2.18 MobiDyc: Modélisation Basée sur les Individus pour la Dynamique des Communautés

Mobidyc is a project that aims to promote ABMS in the field of ecology, biology and environment. It should enable models to be built and run by people with no programming skills. The software itself is written in Smalltalk. A number of constraints are noted: (1) discrete space and time may limit the choice of time steps, mesh and communication between agents; (2) numerical precision; and (3) slow calculations limit the size to around 10,000 cells and 10,000 agents. This suggests that MobiDyc is appropriate for teaching and initial model building, but of limited use for realistic simulations.

Vincent Ginot, the original author, died in 2007 but other contributors are maintaining the project under a GPL license for teaching and non-commercial research, see `http://w3.avignon.inra.fr/mobidyc`.

2.19 Modelling4all

Modelling4all is a project from University of Oxford supported by Eduserv and JISC. It is possible to create and run a behavioural model just from a Web browser – this has for instance been used to teach students the basics of the SugarScape model. It is therefore a useful tool in raising awareness of the capabilities of ABMS methodology in various domains. Non-experts can compose pre-built modular components called micro-behaviours thus supporting a middle out constructionist form of learning. Collaborative model building is also supported. The underlying computational platform is NetLogo. The BehaviouralComposer [32] is also available as open source under the New BSD license, see `http://modelling4all.nsms.ox.ac.uk`.

2.20 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 ease of use. Its programming language includes many high level structures and primitives that greatly reduce programming effort. The language is based on Logo, a dialect of Lisp, and 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. The current version is 4.1.1 from August 2010, see `http://ccl.northwestern.edu/netlogo`.

A new book on individual based modelling featuring NetLogo is in preparation [54].

2.21 Open StarLogo

StarLogo (see below) is an agent based simulation language developed by Resnick, Klopfer and others in the MIT Media Lab and MIT Teacher Education Program. It is an extension of the Logo programming language. StarLogo is designed for education and can be used by students to model the behavior of decentralised systems.

StarLogo is also available in a version called Open StarLogo since June 2006. The source code for Open StarLogo is available online, although the license under which it is released is not an open source license according the Open Source Definition, because of restrictions on the commercial use of the code. See `http://education.mit.edu/openstarlogo`.

2.22 RePast: Recursive Porous Agent Simulation Toolkit

RePast was developed at the University of Chicago's Social Science Research Computing Lab specifically for creating agent based simulations in social sciences [44]. It is very Swarm like, both in philosophy and appearance and similarly 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.

RePast is Java based and developing a simulation ideally 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.

2.23 Repast Simphony

Since 1994, there has been a steady advancement of the software toolkits and development environments that have superseded Swarm. Repast-3 has recently been superseded by a significant development named Repast Simphony, or Repast-S. This is a free and open source toolkit developed

at Argonne National Laboratory. It has tools for visual model development, visual model execution, automated database connectivity, automated output logging, and results visualisation. See `http://repast.sourceforge.net`.

Whilst still being maintained, Repast-J, Repast.Net and Repast-Py have now reached maturity and are no longer being developed. They have been superseded by Repast Simphony (Repast-S) which provides all the core functionality of Repast-J or Repast.Net, although limited to implementation in Java. Repast-S was released as an alpha version in late 2006. From the user's perspective, the main improvements in Simphony compared to Repast-3 are as follows.

- A new GUI for developing models;
- An improved 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/simphonytutorial.html`.

Repast-S probably now has the greatest functionality of any AMBS package. It supports a wide range of external tools for statistical and network analysis, visualisation, data mining, spreadsheets, etc. Point and click modelling in 2D and 3D is supported. Models can be checkpointed in various formats including XML. The discrete event scheduler is concurrent and multi-threaded, various numerical libraries are available, e.g. for random numers and distributed computing is supported using the Terracotta Enterprise Suite for Java.

2.24 SimPack

Simpack, developed by Paul Fishwick, is a directory of tools designed for teaching and 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 [22] 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 updated – there appears to be nothing since 1997. In the latest version the models are executed using Java. SimPack is available with a GPL license.

2.25 SimPy

This is an object oriented Python language toolkit for agent based modelling specifically intended for social science applications. See `http://simpy.sourceforge.net` and the SimPy blog `http://blog.gmane.org/gmane.comp.python.simpy.user`. There is an active developer and user community with v2.1.0 released in June 2010 `http://pypi.python.org/pypi/SimPy/2.1.0`.

2.26 SOARS: Spot Oriented Agent Role Simulator

SOARS from the Tokyo Institute of Technology is a Java project under active development, v3.0.4 was released in Sept'2010, see `http://www.soars.jp`. Unfortunately, the English language part of the Web site has a number of broken links and multiple versions of older pages.

In SOARS a model is composed of agents and "spots", the latter being a place for interaction and giving spatial decomposition. Events occur at regular time steps. During each step a series of stages (temporal decomposition) are performed where behavioural rules are evaluated. A transition within a stage should be independent of the order of execution of rules within that stage. The environment consists of principal components including launcher, shell, simulator and animator (viewer). Simulations can be constructed in the form of games and run on distributed resources. Example applications are a virtual city, a market model and office activities. SOARS has been applied by Deguchi *et al.* [18, 3] to emerging virus protection in the case of SARS and the H5N1 bird flu pandemic, these are based on the virtual city plus a state transition model for the disease and a model of protection policies.

2.27 StarLogo

StarLogo is a programmable modelling environment specifically aimed at exploring de-centralised systems via simulation. It is a specialised version of Logo, which was used for teaching. 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 made up of "patches". Patches ares similar to the spots in SOARS.

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 to develop 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 problems associated with the graphing facility, for example whilst many lines may be plotted on the same graph it is not possible to create more than one graph. With the current v2.2 of StarLogo, it is quite easy to put your simulations on a Web page as an Applet for viewing. A new version called StarLogoTNG (The Next Generation) was released in July 2008, currently v1.2 see `http://education.mit.edu/projects/starlogo-tng`. This in part uses a gaming heuristic.

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 challenging to find ways to do exactly what is required with 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.

2.28 SugarScape

The SugarScape project seeks to provide an interactive model for social science researchers to experiment with an artificial society. This society is currently pre-agricultural and hence nomadic. Agents or citizens move about and gather food on a two dimensional grid, mate with suitable partners, bear offspring, barter for goods with other citizens, migrate, die and leave an inheritance for their survivors.

Researchers can use the simulation as a testbed, laying out their theories in terms of initial and subsequent states of the SugarScape. The source code requires Java-2 SDK and is released under GPL, see `http://sugarscape.sourceforge.net`. There is also a Web based version.

2.29 Swarm

Swarm was the first re-usable software tool created for agent based modelling and simulation. It was developed at the Santa Fe Institute in 1994 and 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 project was still based at the Santa Fe Institute but its development and management is now under control of the Swarm Development Group which has wider membership to sustain the software, see `http://www.swarm.org`.

Swarm was designed as a general language and toolbox for ABMS, intended for widespread use across scientific domains. The developers started with 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 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 the emergence of Java as a mature language. One reason for implementing Swarm in Objective-C was that the lack of strong typing in this language (in contrast for instance to 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 is able to fully implement 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. Since the libraries are written in Objective-C, 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 library calls. Java Swarm was designed to provide, with as little change as possible, access to Swarm's Objective-C library from Java. It 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.

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. 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. 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 that 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 programming pattern which enables the model itself to remain unchanged if the observation code is modified.

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 Objective-C and possibly Java, be familiar 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 which has Objective-C. 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.

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`

In conclusion, Swarm is the most mature ABMS library based framework 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. Finally there does not appear to be any parallel implementation of Swarm to date.

2.30 VisualBots

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

2.31 Xholon

Xholon is an open source general purpose modelling, transformation and simulation tool, based on XML and Java, that supports the Unified Modelling Language, UML-2.1. Systems biology modelling, other types of modelling and many of the features found in other agent based modelling tools are implemented. It focuses especially on integration of various approaches. It can, for example, simulate 2D grid and agent based models created using UML tools and can combine multiple agent based grids in the same model. It includes an optional limited NetLogo like syntax, as Java methods. The download pages include numerous example applications. The project is currently active with v0.8 updated in Oct'2009 and a new user Wiki established in Apr'2010, see `http://www.primordion.com`.

3 Other Multi-Agent Systems

These systems are about building distributed networks of communicating autonomous software agents for "smart" applications [77]. They should not be confused with ABMS. AgentLink was an EU

funded network of excellence in this area, see `http://www.agentlink.org`. A list of such software was available from `http://www.agentbuilder.com/AgentTools` but you'll have to go back to a version prior to Apr'2004 to see the entries.

3.1 A-globe

One of the first projects from the Agent Technology Center of the Czech Technical University in Prague. See `http://agents.felk.cvut.cz/projects`. It was developed in partnership with the US Air Force starting in 2003. Applications are in air traffic control and security.

3.2 ABLE: Agent Building and Learning Environment

ABLE, a project from the IBM T.J. Watson Research Center, provides 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 set of AbleBeans or implement their own custom algorithms. ABLE runs on any platform supporting Java-2. See `http://www.alphaworks.ibm.com/tech/able`.

3.3 Cougaar: Cognitive Agent Architecture

Cougaar is Java software for facilitating the development of agent based applications that are complex, large scale and distributed. Cougaar is used to build smart resilient networks rather than for simulation and modelling. The open source software (currently v12.4) includes not only the core architecture but also a variety of demonstration, visualisation and management components. It was developed as part of a multi-year DARPA research project into large scale agent systems principally aimed a military logistics. See `http://www.cougaar.org`.

For its parallel implementation Cougaar uses a message board communication pattern similar to FLAME. This is aimed at deployment across wide area networks and supports a number of underlying protocols via plugins.

3.4 FIPA: Foundation for Physical Intelligent Agents

FIPA is a non-profit IEEE Computer Society organisation aimed at producing standards for the inter-operation of heterogeneous software agents. It has been active in Swizerland since 1996 and more widely since June 2005, see `http://www.fipa.org`.

FIPA's Agent Framework Reference Model is really an approach to develop an agent framework interoperable with other FIPA compliant agent frameworks.

FIPA is developing standards to promote interoperable agent applications and agent systems. FIPA specifications only talk about the interfaces through which agents may communicate – it does not describe the implementation details. Specifications are divided into five categories: applications, abstract architecture, agent communication, agent management and agent message transport (FIPA Specifications, 2002).

The FIPA Reference Model considers an agent platform as a set of four components: agents, directory facilitator (DF), agent management system (AMS), and message transport system (MTS). The DF and AMS support the management of the agents, while the MTS provides a message delivery service.

The FIPA standard does not talk about mobility as a mandatory requirement for compliant systems. FIPA does however provide some guidelines for how an implementation provider can provide support for mobility on their own.

3.5 JADE: Java Agent Development Framework

JADE aims to simplify the implementation of distributed multi-agent systems through a middleware layer that claims to comply with the FIPA specifications and through a set of tools that support the debugging and deployment phases. The agent platform can be distributed across machines (which do not even need to share the same o/s) and the configuration can be controlled from a remote GUI. The configuration can be even changed at run time by moving agents from one machine to another as and when required.

JADE is aimed at developing smart networks rather than modelling and simulation. It is completely implemented in Java and the minimal system requirement is Java JDK-1.4. JADE is distributed by Telecom Italia under the LGPL-2. The current version is 4.0.1 released in July 2010, see `http://jade.cselt.it`.

WADE, Workflows and Agents Development Environment, is an extension which allows agents to execute tasks defined in a workflow.

3.6 Jason

Jason is an open source interpreter for an extended version of AgentSpeak, a logic based agent oriented programming language written in Java. It enables users to build complex multi-agent systems that are capable of operating in environments previously considered too un-predictable for computers to handle. Jason is easily customisable and is suitable for the implementation of reactive planning systems according to the Belief-Desire-Intention (BDI) architecture. It is typically used to simulate systems comprising multiple interacting robots. JASON uses the FIPA libraries to communicate between agents and is able to inter-operate with JADE.

A book gives additional information about Jason [9] and for more information see `http://jason.sourceforge.net`.

3.7 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, 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 deploy distributed applications quickly using agent principles. It is free and licensed under the GPL/ LGPL. The current version is 4.2.0, see `http://www.madkit.org`.

MadKit is used at LIRMM, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier which is a mixed research unit of l'Université Montpellier II (UMII) and the Centre National de la Recherche Scientifique (CNRS), Département Sciences et Technologies de l'Information et de la Communication (STIC). See `http://www.lirmm.fr/xml/fr/lirmm.html`

3.8 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 (a rule based language used in expert systems), 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`.

3.9 MASIF

The MASIF project intends to support interoperability among heterogeneous agent systems. It focuses on the migration of agents between hosts.

3.10 SDML: Strictly Declarative Modelling Language

SDML is not an environment but a logic based language with object orientation features [42] written in Smalltalk, see `http://cfpm.orgsdml`. 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. However the Web site does not seem to have been updated since 1997.

SDML claims to 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 by 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.

3.11 SeSAm: Shell for Simulated Agent Systems

SeSAm from University of Würzburg provides a generic environment for modelling and experimenting with agent based simulation. It focusses on providing a Java tool for the easy construction of complex models, which include dynamic inter-dependecies or emergent behaviour. This includes easy visual agent modelling, a flexible environment and situation definition, a programming language and integrated graphical simulation analysis. Simulations with different starting paramenters can be distributed. SeSAm can also use a FIPA plugin for agent communication making it compatible with JADE. The current version is 2.5.1 from Aug'2009, see `http://www.simsesam.de`.

3.12 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`.

3.13 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`.

4 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 thus being actively applied in many practical areas. The applications range 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 used to explore a wide range of assumptions which 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 must have passed appropriate validation tests to establish credibility.

For certain applications is it not clear that minimalist models are sufficient, although they might at first sight seem to exhibit correct behaviour. For certain phenomena, low dimensional differential equations have been shown to provide very efficient simulators. For highly heterogeneous systems comprising autonomous actors, ABMs offer a more flexible and powerful tool. ABMs focus on the changing behaviour of self organising elements, for example, individuals in a crowd, and the ways in which those elements interact with each other and their environment to form complex systems. The approach centres on order creation rather than order translation. Since the former is more characteristic of social phenomena than the latter, ABMs have the potential to map onto social phenomena rather better than neo-classical models which mimic reductionist physics. Equilibrium conditions are not assumed but rather studied if, when, and where they emerge.

It is however recognised that agents cannot calculate these aggregate patterns and hence factor them into their decision making, thus there is a fundamental challenge. Human agents experience time and space, and in many cases are able to factor history and geography as experiences into what they do. This is a fundamental characteristic of social systems moving beyond the simplistic view of the first generation of ABMs that assumed that both social structure and cooperation could be reproduced from scratch (bottom up) by the interactions of agents. It has been suggested that a new generation of ABMs should contain: (i) a bounded, but evolving, heterogeneous multi-dimensional irregular lattice network of communicating agents (characterised by the connection pattern), with behaviours that are reflexive; (ii) a specified set of non-linear stochastic conditional neighbour interaction rules that are followed once the agent senses its local state; and (iii) a protocol that describes how local and non-local knowledge is exchanged using asynchronous updating through the network environment.

Some core research questions are as follows.

- What is the appropriate level of granularity for the ABM, i.e. what entities do the agents represent?
- What is the appropriate network structure for the ABM, i.e. what degree of non-local connectivity?

- How can multi-scale adaptivity be introduced? What timesteps are required in the forward marching algorithm?
- Drawing on work in bio-engineering, how intelligent do we need to make agents such that the self healing capacities of communities are adequately represented?
- Are there population size effects and how can these be corrected?

The following lists some general application areas of ABMS.

- Business and Organisations [46]
  - Consumer markets
  - Supply networks
  - Insurance
  - Manufacturing
- Economics
  - Artificial financial markets
  - Trade networks
- Infrastructure
  - Transportation
  - Electric power markets
  - Hydrogen economy
- Crowds
  - Human movement patterns
  - Evacuation modelling and planning
- Society and Culture
  - Ancient civilizations
  - Civil disobedience
- Terrorism
  - Social determinants
  - Organisational networks
- Military
  - Command and control
  - Combat
  - Logistics

- Biology and Ecological Systems
  - Animal behaviour
  - Cell behaviour
  - Sub-cellular molecular behaviour

At Argonne, three pilot application areas are being targetted – microbial bio-diversity, cyber-security and the social aspects of climate change. These are providing a portfolio of requirements for exa-scale 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 organisation, structure, insight and ultimately predictive power that is of practical value. Argonne researchers are using resource ratio theory in microbial eco-systems in devising requirements for the practical application of agent based modelling to the exa-scale system under development.

4.1 Biology and Medicine

See `http://www.swarm.org/index.php/Agent-Based_Models_in_Biology_and_Medicine` and `http://www.bmm.icnet.uk/~barr03`. The latter provides some additional links as follows. Agent Based Modelling – a tool for replicating biological systems – a good description of ABM as used in the Epitheliome project [75, 76, 71] at Sheffield, see `http://www.dcs.shef.ac.uk/~rod/Integrative_Systems_Biology.html`. Agent Based Modelling for Systems Biology – an introduction to ABM in this field from a few years ago. Also contains a list of software, see `http://abmsystemsbiology.info/index.htm`. Individual based models – a nine year old site, but nevertheless a very comprehensive list of ABM applications from the time, see `http://www.red3d.com/cwr/ibm.html`.

Top down modelling involves representing observed system behaviour with equation based models (EBM), such as differential equations (either ODE's over time or PDE's over time and space). In such systems, observables represent changeable quantities such as population sizes or concentrations of a particular entity. Models are often designed to match real world observations and then used to make predictions or hypotheses of the system under differing conditions. Often such predictions can be checked through observation or experimentation. Despite the advantages for macroscopic simulation, EBM offers little insight into the micro-level behaviour representing the interactions of the individuals

within the system. Where global observations are made, these represent average values and assume homogeneity and perfect mixing of system components. As a result, important low level details of the system may be ignored.

In contrast, ABMs utilise a bottom up approach to simulation that does not explicitly attempt to model aggregate characteristics of a system. As with multi-agent systems (MAS), ABMs can be described as a "system of interacting parts". The notable difference being that agents are simulated as autonomous individuals whereas MAS may use a more generic agent representation. Typically an ABM consists of a number of agents, an environment and a set of rules governing agent behaviour. Agents themselves are self contained entities with states and a set of behavioural rules. Agents may represent discrete spatial entities such as molecules or cells, in which case they may reside within a continuous or discrete spatial environment. In either case, agents may interact directly or through an environment where they compete for resources. By specifying rules at an individual level, "emergent" complex system behaviour can be observed through the result of agent interactions. The specification of individual rules also makes ABMs inherently capable of representing heterogeneity, as each agent can possess its own individual attributes and behaviours. Such system wide diversity is important as in many systems agents cannot be expressed as simple uniform entities. This is particularly true in cellular level agent based modelling, where cells in differing states (e.g. different cell cycle phases) or subject to different micro-environments (e.g. local stress of biochemical gradients), may exhibit entirely different behaviours or differing phenotypes.

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. A list of six applications was given in a short survey by Politopoulos [53].

1. *An agent based model for real time signalling induced in osteocytic networks by mechanical stimuli* B.J. Ausk, T.S. Gross, S. Srinivasan, Journal of Bio-Mechanics (2005)

2. *The epitheliome: agent based modeling of the social behaviour of cells* D.C. Walker, J. Southgate, G. Hill, M. Holcombe, D.R. Hose, S.M. Wood, S. MacNeil, R.H. Smallwood, Bio-Systems 76:1-3 (2004) 89-100

3. *Emerging patterns in tumour systems: simulating the dynamics of multi-cellular clusters with an agent based spatial agglomeration model.* Y. Mansury, M. Kimura, J. Lobo, T.S. Deisboeck, Journal of Theoretical Biology 219:3 (2002) 343-70

4. *Multi-disciplinary investigation into adult stem cell behaviour* M. d'Inverno and J. Prophet, Lecture Notes in Computer Science 3737 (2005) 49-64

5. *In silico experiments of existing and hypothetical cytokine directed clinical trials using agent based modeling* G. An, Critical Care Medicine 32:10 (2004) 2050-60

6. *Modelling the effect of exogenous calcium on keratinocyte and HaCat cell proliferation and differentiation using agent based computational paradigm* D. Walker, T. Sun, S. MacNeil, R. Smallwood, Tissue Engineering 12:8 (2006) 2301-9

In medicine, ABMS 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 [43]. That is what we are interested in – how does this social interaction of the cells produce properly functioning and structured creatures?

A Web site hosted for Cancer Research UK shows some examples of modelling cancer growth, for instance angiogenesis [6], see `http://www.bmm.icnet.uk/~barr03/us.html`. This includes a page of links to relevant publications.

In the biological sciences, Argonne researchers are using ABMs to model cellular behaviour. AgentCell is a parallel 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 ran numerous simulations of the *E. coli* chemotaxis network for many cells and stochastic variations. They modelled 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 over MPI that minimise resource contention through its exascale computing research effort.

Large scale simulations of "digital bacteria" run on compute clusters suggest that the chemotaxis network is tuned to simultaneously optimise the random spread of cells in the absence of nutrients and the cellular response to gradients of nutrients. Without a high performance (parallel) implementation 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.

Other areas of study include the following.

- Study of propagation of epidemics and pandemics;
- Cell behaviour, e.g. wound healing using stem cells;
- Sub-cellular molecular behaviour;
- Cancer growth modelling.

Whilst it has been shown that other modelling methods are more appropriate to study the actual spread and causes of pandemics [15], ABMS still has a role to play. According to Epstein [21] ABMS is useful because it can capture irrational behaviour, complex social networks and global scale phenomena. In his opinion, the cutting edge in performance is the Global Scale Agent Model (GSAM) developed by Jon Parker at the Brookings Institute [49]. This includes 6.5 billion distinct agents, with movement and day-to-day local interactions modelled as available data allow.

For the USA, the GSAM contains 300 million cyber-people and every hospital and staffed bed in the country. The National Center for the Study of Preparedness and Catastrophic Event Response at Johns Hopkins University in Baltimore is using the model to optimise emergency surge capacity in a

pandemic, supported by the Department of Homeland Security. We note however that in other work Operational Research methodologies have been used for surge planning.

4.2 Physics and Chemistry

Some application areas are as follows.

- Design of self organising materials and self organising systems; directed self assembly; use of fields to govern self organisation, modelling in terms of agent based algorithms.
- Modelling of fluid flows and flow and segregation in granular matter.
- Modelling of gene and protein interaction networks, immunology.
- Complex reactions, network analysis of the fate of pollutants, leading to science based environmental policies.
- 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.

4.3 Security

After the terrorist attack on the USA of 11/9/2001, the defense and intelligence communities expressed increased interest in understanding what certain kinds of people might do, how their organisations might behave and how to synthesise useful information out of large data sets. The US Defense Advanced Research and Projects Agency (DARPA) announced new initiatives in these areas and the intelligence community, with the National Security Agency (NSA) apparently in the lead, formed the Advanced Research and Development Activity (ARDA) with the same purposes announced. A particular interest of ARDA is what they called "novel intelligence from massive data".

In the UK there are similar initiatives, with BAE Systems and Detica taking a lead. The recent INSTINCT'09 initiative put on show a number of novel technologies and applications, some including ABMS.

Some of the application areas are as follows.

- Transportation and logistics;
- Human movement patterns;
- Evacuation modelling and planning;
- Social determinants;
- Organisational networks.

There are a number of examples of ABMs applied to scenario planning in evacuation conditions, for instance [17, 73].

Network analysis is also used in security applications, for instance in understanding links between political groups via a study of on-line networks [2].

4.4 Cyber Security

Cyber-security also offers considerable opportunities to apply exa-scale ABMS. There is a need to develop advanced methods to explore future cyber-security 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, cyber-security policies and external threat agents.

The world wide information technology (IT) infrastructures are large scale systems with 10^6 to 10^{10} 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 modelled. 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 cyber-security.

4.5 The Environment

Improving scientific understanding of climate change requires researchers to consider the physical, economic and social determinants of climate impact. 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 already been made in modelling the physical aspects of climate change such as atmospheric flows, oceanic circulation and albedo effects.

4.6 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 10^{10} human agents. These agents would each have a large number of complex internal states (10^3 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 organised 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 realise a powerful environment for large scale agent based economic simulations. Key issues will be the definition of formal languages for modelling and for optimising code generation, the development of scalable computational simulation tools and the standardisation 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, optimising the impact of regulatory decisions that could be quantitatively based on European economy scenarios.

Some application areas are as follows.

- Artificial financial markets;
- Consumer markets;
- Insurance;
- Manufacturing.

4.7 Supply Network and Transport Optimisation

Some application areas are as follows.

- Understanding of product and manufacturing supply networks. Supply networks in economics.
- Understanding and evolution of organisations, including the design of structures for scientific and technological collaboration.
- Trade networks;
- Transportation;
- Electric power markets;
- Hydrogen economy.

5 ABMS on HPC

High performance computing systems permit the execution of sophisticated adaptive ABMs with many agents and complex geometry and rules.

5.1 HPC Clusters

Cluster based distributed memory computing is efficient if the processor cores spend most of the time computing rather than communicating. This is however not easy with ABMs due to their high levels of communication between agents. A number of methods have appeared where those which most frequently communicate with each other are grouped together on one processor.

See EURACE Web site `http://www.eurace.org` and [19] for information about FLAME on HPC clusters including NW-GRID and HPCx. Parallel extensions to FLAME in this case use a "message board" approach with different boards for different types of message. This allows the communication pattern to be optimised and messages only sent when necessary.

5.2 BlueGene

See SciDAC review [45] for information on work at Argonne National Laboratory on IBM BlueGene and clusters.

The open source and highly scalable agent based modelling toolkit being developed at Argonne for the exa-scale 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 synchronised to form a single data storage framework.

Time schedulers coordinate and synchronise 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 visualisation.

Argonne researchers plan to complete the backbone of the exa-scale 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.

5.3 GPGPU

Over the past 5 years, a minor revolution has ocurred in low cost HPC capabilities through the emergence of general purpose programmable graphical processor units (GPGPUs). Using commodity GPUs such as those from nVidia or AMD has appeal as many personal computers have them, so

give enhanced power to resources such as Condor pools and home computers used in philanthropic computing.

ABMS packages are beginning to be adapted to run on GPUs e.g. using the CUDA language. There is a Web site devoted to genetic programming for GPU – GPGPGPU at `http://www.gpgpgpu.com`. Current references include the work of Lysenko and D'Souza at Michigan Technological University [37] and of Paul Richmond at Sheffield University `http://www.dcs.shef.ac.uk/~paul/abgpu.html`. Lysenko *et al.* presented an attractive stochastic allocator algorithm for parallel agent replication on a GPU. Richmond *et al.* have written a number of papers on Agent Based Modelling on GPU [59, 60, 63] and described GPU extensions to the FLAME framework. More recent work on multi-GPU and multi-core systems is by Aaby *et al.* [1] at Oak Ridge National Laboratory.

5.4 Cell

There are currently no known ports of ABMS 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 the 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) [58].

Richmond *et al.* planned to re-code FLAME GPU in OpenCL which would permit it to run on Cell, ATI GPUs and Intel Larrabee. By 2010 the interest in cell based systems had waned as GPGPUs became increasingly popular.

6 ABMS Community and Research

AgentLink: an EU Network of Excellence for Agent Based Computing, was a network of researchers and developers with a common interest in agent technology. Funding ceased in 2005, see `http://www.agentlink.org`.

CoABS: Control of Agent Based Systems (CoABS) is a DARPA programme to develop and demonstrate techniques to safely control, coordinate, and manage large systems of autonomous distributed software agents. CoABS is investigating the use of agent technology to improve military command, control, communication, and intelligence gathering. See `http://coabs.globalinfotek.com`.

CSCS, Center for the Study of Complex Systems, is a broadly inter-disciplinary programme at the University of Michigan designed to encourage and facilitate research and education in the general area of non-linear, dynamical and adaptive systems. Nearly every college of the university participate, thus illustrating that many different kinds of system which include self regulation, feedback or adaptation in their dynamics, may have a common underlying structure. Moreover, the structural similarities can be exploited to transfer methods of analysis and understanding from one field to another. In addition to developing deeper understandings of specific systems, inter-disciplinary approaches should help elucidate the general structure and behaviour of complex systems and move us toward a fuller appreciation of the general nature of such systems. See `http://cscs.umich.edu`.

EURAMAS, European Association for Multi-Agent Systems, provides one of the few European organisations encouraging cross disciplinary collaboration in this area. EURAMAS organises a European Workshop on Multi-agent Systems (EUMAS) and the European Summer School on Multi-agent Systems (EASSS). These are important in promoting the understanding and application of agent based technology.

Recently ERCIM (The European Research Consortium for Informatics and Mathematics)has considered setting up a Special Interest Group on Agent Based Technology and Applications.

Open Agent Based Modeling Consortium `http://www.openabm.org/site` is hosted at Arizona State University as part of the NSF CoMSES. The OpenABM Consortium is a group of researchers, educators and professionals with a common goal to improve 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.

The Software Agents group of the MIT Media Laboratory investigates computer systems to which one can delegate tasks. Software agents differ from conventional software in that they are long lived, semi-autonomous, proactive and adaptive. The group develops techniques and builds prototype agent systems that can be tested. See `http://agents.media.mit.edu`, but there seem to be no publications since 2004.

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

Recently, following a call for statments of interest from EPSRC, the UK community has rallied around a proposal to establish a Collaborative Computational Project in Agent Based Modelling Technology [98].

6.1 Future Research Challenges

As discussed in [98] there are a number of challenges which must be faced in order to make ABMs a mainstream computational science technology. These are being addressed by the community, and include the following [1].

- Issues of applications to large and complex systems;
- Performance of ABMs – parallelisation technology;
- Verification and validation of large scale models;
- Statistical inference for agent based models;
- Representation and standards for model exchange between codes;
- Future application areas;
- Need for a Collaborative Computational Project in ABMs where the community can share best practice and software.

[1] as discussed at a community awareness raising workshop held in Leeds on 15/6/2010.

An NSF report was published in 2007 entitled: *Modeling and Simulation at the Exascale for Energy and the Environment. Report on the Advanced Scientific Computing Research Town Hall Meetings on Simulation and Modeling at the Exascale for Energy, Ecological Sustainability and Global Security (E3).* `http://www.sc.doe.gov/ascr/ProgramDocuments/ProgDocs.html`. This notes as one of its goals *to identify emerging domains of computation and computational science that could have dramatic impacts on economic development, such as agent based simulation, self assembly, and self organization* and suggests this can be addressed by *Math and Algorithms. Advancing mathematical and algorithmic foundations to support scientific computing in emerging disciplines such as molecular self assembly, systems biology, behavior of complex systems, agent based modeling, and evolutionary and adaptive computing.*

More specifically it listed a number of challenges which have to be addressed.

- Distributed query resolution to allow agents to flexibly and repeatedly find other agents and recognise methods for interaction in a dynamic environment with a continually and endogenously evolving structure (e.g., non-reified networks);
- Situational activation of agents based on contextual factors and associated real location to working sets of processors with appropriate inter-processor locality;
- Efficient implementation of periodic fine grained interactions between agents where the pay-offs from the interplay are defined as an endogenous function of the ongoing interactions themselves, such that players are free to enter and leave the interactions at idiosyncratic times;
- Distributed time scheduling at a level of parallelism beyond the current approaches;
- Extremely high volume data warehousing to allow efficient exploration of huge numbers of large model runs;
- Efficient directed sweeps across huge model parameter spaces with appropriate adaptation as results are discovered;
- Domain decomposition techniques for parallel agent based simulations where the computational load per agent is variable, in time for the same agent, as well as from agent to agent, and where the geographical locality has no relation to the nature and volume of communication between agents.

Clearly agent based modelling must be advanced along several directions before it can present a viable approach for addressing exa-scale application needs and which can be relied upon to help make decisions in a complex and changing world.

References

[1] B.G. Aaby, K.S. Perumalla and S.K. Seal *Efficient Simulation of Agent Based Models on multi-GPU and multi-Core Clusters* Proc. 3rd Int. Conf. on Simulation Tools and Techniques (SimuTools 2010, Malaga)

[2] R. Ackland *et al. VOSON: Virtual Observatory for the Study of On-line Networks* Web site `http://voson.anu.edu.au`

[3] K. Arai, H. Deguchi and H. Matsui (eds.) *Agent Based Modeling Meets Gaming Simulation* Proc. ISAGA 2006 (Springer 2006) 180pp ISBN 978-4-4312-9426-0

[4] R.A. Axelrod *The Complexity of Cooperation: Agent Based Models of Competition and Collaboration* (Princeton University Press, 1997) 248pp ISBN 978-0-6910-1567-5

[5] M. Batty *Cities and Complexity: Understanding Cities with Cellular Automata, Agent Based Models and Fractals* (The MIT Press, 2005) 542pp ISBN 978-0-262-02583-6

[6] K. Bentley, H. Gerhardt and P.A. Bates *Agent based simulation of notch mediated tip cell selection in angiogenic sprout initialisation* Journal of Theoretical Biology, 250:1 (2008) 25-36

[7] E. Bonabeau *Agent based modeling: Methods and techniques for simulating human systems* Proc. Nat. Academy of Sciences 99 (2002) 7280-287

[8] E. Bonabeau *Predicting the unpredictable* Harvard Business Review 80:3 (2002) 109-15

[9] R.H. Bordini, J.F. Hübner and M. Wooldridge *Programming Multi-Agent Systems in AgentSpeak using Jason* (John Wiley, 2007) 292pp ISBN 978-0-470-02900-8

[10] J. Carvalho *Using AgentSheets to teach simulation to undergraduate students* Journal of Artificial Societies and Social Simulations 3:3 (2000) `http://jasss.soc.surrey.ac.uk/3/3/forum/2.html`

[11] C.J.E. Castle and A.T. Crooks *Principles and Concepts of Agent Based Modelling for Developing Geo-spatial Simulations* (UCL, Sept'2007) ISSN 1467-1298. See CASA Web site working paper 110

[12] S. Coakley, R. Smallwood and M. Holcombe *From Molecules to Insect Communities – how Formal Agent Based Computational Modelling is uncovering new Biological Facts* Scientiae Mathematicae Japonicae 64:2 (2006) 185-98

[13] S. Coakley, R. Smallwood and M. Holcombe *Using X-Machines as a Formal Basis for describing Agents in Agent Based Modelling* Proc. Spring Simulation Multi-conference (SCS, 2006) 33-40 `http://www.scs.org/confernc/springsim06/prelimProgram/ads/5.html`

[14] V.S. Colella, E. Klopfer and M. Resnick *Adventures in Modeling: exploring complex dynamic systems with StarLogo* (Teachers College Press, 2001) 188pp ISBN 978-0-8077-4082-8 `http://www.media.mit.edu/starlogo/adventures`

[15] V. Colizza and A. Vespignani *The Flu Fighters* Physics World 23:2 (Feb'2010) 26-30

[16] 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, June 2003)

[17] A.T. Crooks, C.J.E. Castle and M. Batty *Key Challenges in Agent Based Modelling for Geo-spatial Simulation* (UCL, Sept'2007) ISSN 1467-1298 See CASA Web site working paper 121

[18] H. Deguchi *Economics as an Agent Based Complex System: Toward Agent Based Social Systems Sciences* (Springer 2004) 260pp ISBN 978-0-443-120985-0

[19] 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) 541-52

[20] N. Ehrentreich *Agent Based Modeling – The Santa Fe Institute Artificial Stock Market Model Revisited* (Springer, 2007) ISBN 978-3-5407-3878-9

[21] J.M. Epstein *Modelling to contain Pandemics* Nature 460 (6/8/2009) 687. DOI 1038/460687a `http://www.nature.com/nature/journal/v460/n7256/full/460687a.html`

[22] P.A. Fishwick *Simulation Model Design and Execution: Building Digital Worlds* (Prentice-Hall, 1995) 432pp ISBN 978-0-13-098609-2

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

[24] N. Gilbert *Agent Based Models – Quantitative Applications in the Social Sciences Series* (SAGE Publications, 2007) 112pp ISBN 978-1-4129-4964-4

[25] N. Gilbert and S. Bankes *Platforms and Methods for Agent based Modeling* Proc. National Academy of Sciences of the USA 99:3 (14/5/2002) 7197-8

[26] V. Grimm and S.F. Railsback *Individual Based Modeling and Ecology* (Princeton University Press, 2005) 480pp ISBN 978-0-691-09666-7 `http://www.humboldt.edu/~ecomodel/book.htm`

[27] C.A. Iglesias, M. Garijo and J. Centeno-González *A Survey of Agent Oriented Methodologies* Proc. 5th Int. Workshop on Intelligent Agents (Springer, 1999) 317-30

[28] N.R. Jennings, K. Sycara and M. Wooldridge *A Roadmap of Agent Research and Development* J. Autonomous Agents and Multi-Agent Systems 1:1 (1998) 7-38

[29] A. Johansson *Data Driven Modeling of Pedestrian Crowds: Crowd Simulation, Computer Vision, and Real World Applications* (VDM Verlag Dr. Müller, 20/11/2009) 196pp ISBN 978-3-63920-893-1

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

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

[32] K. Kahn and H. Noble *The BehaviourComposer 2.0: a Web based tool for composing NetLogo code fragments* Proc. Constructionism (Paris, 2010) `http://modelling4all.wikidot.com/publications`

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

[34] M. Luck, R. Ashri and M. d'Inverno *Agent Based Software Development (Agent Oriented Systems)* (Artech House Publishers, 2004) 228pp ISBN 978-1-58053-605-0

[35] M. Luck, P. McBurney, O. Shehory and S. Willmott *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)* (AgentLink, 2005) ISBN 978-0-85432-845-9

[36] S. Luke, C. Cioffi-Revilla, L. Panait and K. Sullivan *MASON: a new Multi-Agent Simulation Environment* Simulation 81:7 (2005) 517-27

[37] 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`

[38] 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`

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

[40] M. Matsumoto and T. Nishimura *Mersenne Twister: a 623-dimensionally equidistributed uniform pseudo-random number generator* ACM Transactions on Modeling and Computer Simulation 8 (1998) 3-30

[41] N. Minar, R. Burkhart, C. Langton and M. Ashenazi *The Swarm Simulation System: a Toolkit for building multi-agent Simulations* (Santa Fe Institute, 1996) Working Paper 96-06-042

[42] 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

[43] D. Noble *The Music of Life – Biology beyond the Genome* (OUP, June 2006) 176pp ISBN 978-0-199-22836-2 `http://www.musicoflife.co.uk/`

[44] 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

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

[46] M.J. North and C.M. Macal *Managing Business Complexity: Discovering Strategic Solutions with Agent Based Modeling and Simulation* (Oxford University Press, 2007) 329pp ISBN 978-0 195-17211-9

[47] J.J. Odell *Objects and Agents Compared* J. of Object Technology 1:1 (May-June 2002) 41-53

[48] 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`

[49] J.A. Parker ACM Trans. Model. Comput. S. (2009) in press. Cited by Epstein.

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

[51] M. 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`

[52] H. Van Dyke Parunak *Go to the Ant: Engineering Principles from Natural Agent Systems* Annals of Operations Research 75 (1997) 69-101

[53] I. Politopoulos *Review and Analysis of Agent Based Models in Biology* (University of Liverpool, 11/9/2007)

[54] S.F. Railsback and V. Grimm *A Course in Individual and Agent Based Modelling* Princeton University Press (in preparation, 2010) `http://www.railsback-grimm-abm-book.com`

[55] 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`

[56] F. Rateb, N. Bellamin and B. Pavard *The simulation of the Spread of Malaria in Haiti* developed at GRIC IRIT, see Web site `http://www.irit.fr/COSI/training/evaluationoftools/Evaluation-Of-Starlogo.htm`

[57] A. Repenning, A. Ioannidou and J. Zola *AgentSheets: End User Programmable Simulations* Journal of Artificial Societies and Social Simulations 3:3 (2000) `http://jasss.soc.surrey.ac.uk/3/3/forum/1.html`

[58] C. Reynolds *Big Fast Crowds on PS3* Proc. ACM SIGGRAPH Symposium on Video-games (Sandbox'06, 2006) 113-21

[59] 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* Web site `http://www.dcs.shef.ac.uk/~paul/abgpu.html`

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

[61] 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* Proc. Int. Workshop on Super-visualisation (IWSV08, Greece, 2008) in press

[62] P. Richmond, S. Coakley and D. Romano *High Performance Agent Based Modelling Framework on Graphic Card Hardware with CUDA* Proc. 8th Int. Conf. on Autonomous Agents and Multi-agent Systems (AAMAS, Budapest, 2009) in press

[63] P. Richmond, S. Coakley and D. Romano *Cellular Level Agent Based Modelling on the Graphics Processing Unit* Proc. Workshop on High Performance Systems Biology (HiBi09, 2009) in press

[64] P. Richmond, S. Coakley, D. Walker and D. Romano *Parallel Cellular Level Agent Based Modelling with FLAME* Briefings in Bio-informatics (Oxford University Press, 2009) submitted

[65] A. Schadschneider *I'm a Football Fan – get me out of Here* Physics World (IoP, July 2010) 21-5 `http://physicsworld.com/cws/article/indepth/43033`

[66] 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)

[67] C.R. Shalizi *Methods and Techniques of Complex Systems Science: An Overview* Chapter 1 in T.S. Deisboeck and J.Y. Kresh (eds.) "Complex Systems Science in Bio-medicine" (Springer, New York, 2006) 33-114 ISBN 978-0-387-30241-6 http://arxiv.org/abs/nlin.AO/0307015

[68] C.R. Shalizi. On-line notebooks on ABM. Web site http://cscs.umich.edu/~crshalizi/notebooks/agent-based-modeling.html

[69] R.K. Standish *EcoLab Documentation* Web site http://ecolab.sourceforge.net/doc/ecolab/ecolab.html

[70] R.K. Standish and R. Leow *EcoLab: Agent Based Modeling for C++ Programmers* Proc. SwarmFest (2003) arXiv:cs.MA/0401026

[71] T. Sun, P. McMinn, S. Coakley, M. Holcombe, R. Smallwood and S. MacNeil *An integrated systems biology approach to understanding the rules of keratinocyte colony formation* J. Roy. Soc. Interface 4 (2007) 1077-92

[72] 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

[73] J. Thorp, S. Guerin, F. Wimberly, M. Rossbach, O. Densmore, M. Agar and D. Roberts *Santa Fe on Fire: Agent Based Modelling of Wildfire Evacuation Dynamcs* Web site http://www.redfish.com/wildfire

[74] 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

[75] D.C. Walker, J. Southgate and G. Hill *The Epitheliome Project: Agent Based Modelling of the Social Behaviour of Cells* J. Bio. Systems 76:1-3 (2004) 89-100 http://dx.doi.org/10.1016/-j.biosystems.2004.05.025

[76] D.C. Walker, S. Wood, J. Southgate, M. Holcombe and R. Smallwood *An integrated Agent Mathematical Model of the Effect of intra-cellular Signalling via the epidermal growth factor receptor on cell proliferation* J. Theoretical Biology 242 (2006) 774-89

[77] M. Wooldridge *An Introduction to Multi-Agent Systems* (Wiley and Sons, 2002) 348pp ISBN 978-0-471-49691-5

[78] B. Zeigler, T.G. Kim and H. Praehofer *Theory of Modeling and Simulation* 2nd edition. (Academic Press, 2000) ISBN 978-0-12-778455-7

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

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

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

[82] 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

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

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

[85] 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`

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

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

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

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

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

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

[92] 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.

[93] 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;

[94] Artificial Life On-line `http://www.alife.org` An on-line companion to the (paper) journal Artificial Life, published by the Santa Fe Institute.

[95] *Agent Based Modelling in Biology* CR-UK Web site `http://www.bmm.icnet.uk/~barr03`

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

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

[98] Proposal to establish a UK Collaborative Computational Project in ABMS. Web site `http://www.softeng.rl.ac.uk/abm_ccp`