UCL CENTRE FOR ADVANCED SPATIAL ANALYSIS

UCL

WORKING PAPERS SERIES

Paper 123 - Sept 07

The Repast Simulation/ Modelling System for Geospatial Simulation

ISSN 1467-1298

Centre for Advanced Spatial Analysis University College London 1 - 19 Torrington Place Gower St London WC1E 7HB
Tel: +44 (0)20 7679 1782 casa@ucl.ac.uk www.casa.ucl.ac.uk

The Repast Simulation/Modelling System for Geospatial Simulation

Andrew T Crooks[1]

Email: andrew.crooks@ucl.ac.uk

Abstract

The use of simulation/modelling systems can simplify the implementation of agent-based models. Repast is one of the few simulation/modelling software systems that supports the integration of geospatial data especially that of vector-based geometries. This paper provides details about Repast specifically an overview, including its different development languages available to develop agent-based models. Before describing Repast's core functionality and how models can be developed within it, specific emphasis will be placed on its ability to represent dynamics and incorporate geographical information. Once these elements of the system have been covered, a diverse list of Agent-Based Modelling (ABM) applications using Repast will be presented with particular emphasis on spatial applications utilizing Repast, in particular, those that utilize geospatial data.

1.1: Introduction

The development of agent-based models can be greatly facilitated by the utilisation of simulation/modelling toolkits. They provide reliable templates for the design, implementation and visualisation of agent-based models, allowing modellers to focus on research (i.e. building models), rather than building fundamental tools necessary to run a computer simulation (Tobias and Hofmann, 2004; Railsback *et al.*, 2006; Castle and Crooks, 2006). In particular, the use of toolkits can reduce the burden modellers face programming parts of a simulation that are not content-specific (e.g. a Graphical User Interface, (GUI), data import-export, visualisation/display of the model). It also increases

[1] This paper was first presented at the Agent-Based Models for Spatial Systems in Social Sciences & Economic Science with Heterogeneous Interacting Agents (ABM–S4–ESHIA) workshop in Agelonde, La Londe les Maures (France), September 17-22, 2007.

the reliability and efficiency of the model, because complex parts have been created and optimised by professional developers, as standardised simulation/modelling functions. Unsurprisingly, there are limitations of using simulation/modelling systems to develop agent-based models, for example: a substantial amount of effort is required to understand how to design and implement a model in some toolkits; the programming code of demonstration models or models produced by other researchers can be difficult to understand or apply to another purpose; a modeller will have to learn or already have an understanding of the programming language required to use the toolkit; and finally the desired/required functionality may not be present, although additional tools might be available from the user community or from other software libraries (Castle and Crooks, 2006). Benenson *et al.* (2005) also note that toolkit users are accompanied by the fear of discovering that a particular function cannot be used, will conflict, or is incompatible with another part of the model late in the development process.

There are numerous simulation/modelling systems available for creating agent-based models which support the direct integration of geospatial data, for example Swarm, Repast, OBEUS, NetLogo and StarLogo (see Castle and Crooks, 2006 for a recent review). This paper will explore one of these simulation/modelling systems in greater detail: the Repast toolkit which supports the integration of both raster and vector datasets into an agent-based model.

It is not the intention of this paper to provide a beginner's guide to using Repast, or a tutorial of how to create a Repast model from scratch for these resources are already available (refer to Appendix 1). Nor is it the intention to compare Repast to other simulation/modelling systems as this has been done by numerous other authors (see, for example, Najlis *et al.*, 2001; Parker, 2001; Tobias and Hofmann, 2004; Castle and Crooks, 2006). The intention is to combine the diverse literature relating to Repast and models utilizing the toolkit and relating it to ABM in general and geospatial ABM in particular. Literature serving these purposes is not plentiful, but the author feels that literature documenting the conceptualisation the development of a model using Repast is sparse. This paper therefore provides a good supplement to the 'how-to' documents

provided with Repast. Where the 'how-to' documentation explains what is required to implement some aspect of Repast functionality, this paper provides the reader with a discussion about what else should be considered before implementing this functionality and how it links to ABM in general, for instance which environment to choose: raster vs. vector, and what type of spatial representation, discrete or continuous, to use.

Section 1.2 provides an overview of Repast, including its different development languages available to develop agent-based models. Before describing Repast's core functionality and how models can be developed within it, specific emphasis will be placed on its ability to represent dynamics and incorporate geographical information (Section 1.3). Once these elements of the system have been covered, a diverse list of ABM applications using Repast will be presented (Section 1.4). The paper concludes with a brief summary of what has been presented (Section 1.5).

1.2: Repast Overview

Originally developed at the University of Chicago's Department of Social Science Research Computing Laboratory and subsequently maintained by Argonne National Laboratory and now managed by Repast Organisation for Architecture and Development (ROAD), the REcursive Porous Agent Simulation Toolkit (Repast) is a free open source application for creating agent-based models. Specifically, Repast abstracts key conceptual requirements of ABM (see Castle and Crooks, 2006 for such requirements), providing functionality to create, run, display, and collect data from agent-based models[2]. For instance, a typical Repast model will comprise agents with different attributes (from totally homogenous to completely heterogeneous) and varying behaviours that interact across an environment. Essentially, modellers are interested in the exploration of outcomes related to the interaction of these agents given a particular scenario. Template components for constructing representational elements such as the environment in which an agent interacts are provided (e.g. grid, torus, network, etc.). The aforementioned

[2] Repast comprises a core (i.e. scheduling of events, base/template models, agent classes, spaces, GUI model manipulation and simulation control, data collection, batch simulation, etc) and non-core (i.e. charting, snapshots of a display, data recording, visual display of agents, models and spaces, etc) functionality.

agent-based capabilities allow modellers to spend more time developing the specifics of their model (e.g. agent interactions, behaviours, etc) rather than setting up the basics of a simulation (e.g. scheduling events to occur, developing a visual display, etc). Figure 1 illustrates a typical selection of Repast toolbars and displays generated for a typical simulation run. These include toolbars for controlling and manipulating the simulation (A) and for setting parameters of the simulation (B), as well as displays for visualising agents interacting within the model environment (C) and charting output data (D); this is a histogram in this instance, but it could be a line graph, scatter graph, etc.

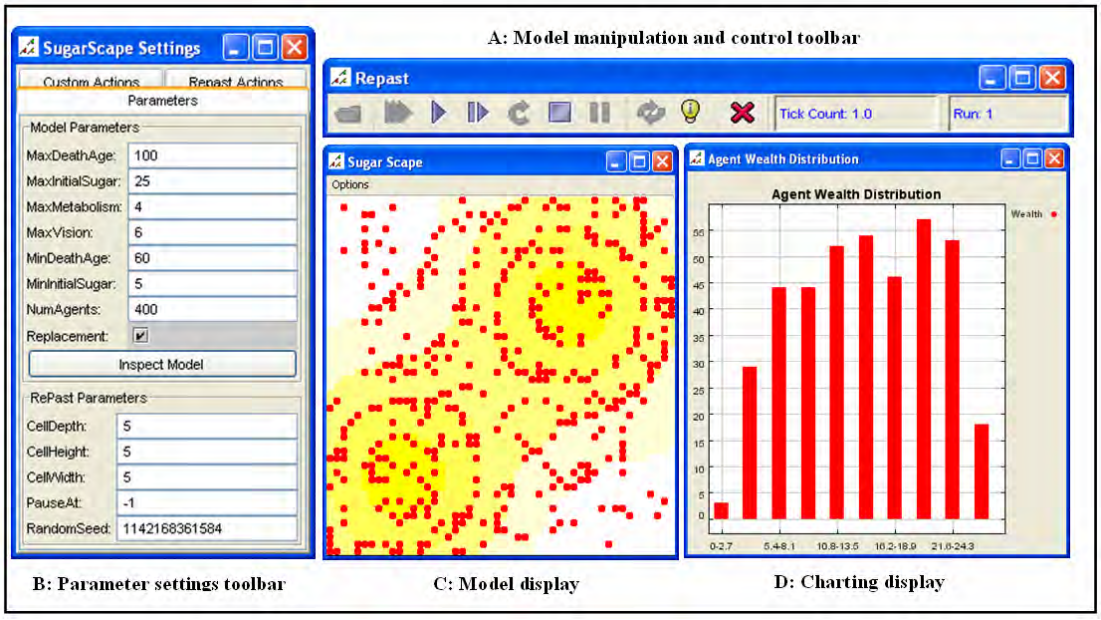


Figure 1: A typical selection of Repast toolbars and displays used during a run of the Sugarscape model.

1.2.1: Repast Implementation Languages

Repast is a derivative of the Swarm simulation toolkit[3]. Initially conceived as a library of Java classes that could interface with the Swarm simulation framework, this concept was abandoned when, amongst other reasons, a Java implementation version of Swarm was released (Collier, 2002). Consequently, the creators of Repast developed an independent framework completely written in Java, thus completely object-oriented but borrowing several key abstractions present within Swarm. Repast has matured considerably since its inception with many enhancements included with every new version. However, to

[3] Refer to the Swarm website: http://www.swarm.org/wiki/Swarm_main_page.

accommodate a growing number of researchers interested in developing simulations with the toolkit, Repast was developed for implementation in alternative programming languages/frameworks. Currently Repast can be implemented in three different programming languages/frameworks: Java (RepastJ and RepastS), Microsoft.NET (Repast.NET), and Python (RepastPy). The following four subsections will explore these further different implementations.

1.2.1.1: Python - RepastPy

RepastPy was specifically developed for rapidly developing a basic agent-based model, offering the most graphical way to create a model via a point-and-click GUI (Figure 2), thus, allowing modellers with limited programming experience to create basic models. Collier and North (2004; 2005) identify three model types permissible within RepastPy: 1) a GIS-based model where agents are GIS features with topology interacting within a landscape; 2) a network based model where agents are nodes in a network that can manipulate the network topology; and 3) a grid-based model where agents with topology reside and interact. RepastPy is also the basis of Agent Analyst, an ABM extension for ArcGIS that allows users to create, edit, and run Repast models from within the GIS (Redlands Institute, 2006) although this is not a requirement for using RepastPy.

Within RepastPy, template agents can be created to populate each model type, but a user must develop behaviours for agents using a subset of the Python programming language. A subset is used because the entire Python language is not necessary to develop agent behaviours (see Collier and North, 2004 and North *et al.*, 2006 for further details). Python is particularly useful because it integrates with Java, thus permitting access to the Repast framework, as well as other extensions and packages available in the Java programming language. Furthermore, models developed in RepastPy can be exported into Java, allowing users to subsequently work with the traditional RepastJ framework. Collier and North (2004; 2005) provide a more detailed overview of RepastPy, and how to develop a model with this implementation of Repast. Further documentation, tutorials, and demonstration models for RepastPy and Agent Analyst are available in their

retrospective installation folders. The Agent Analysts website[4] also provides some useful resources.

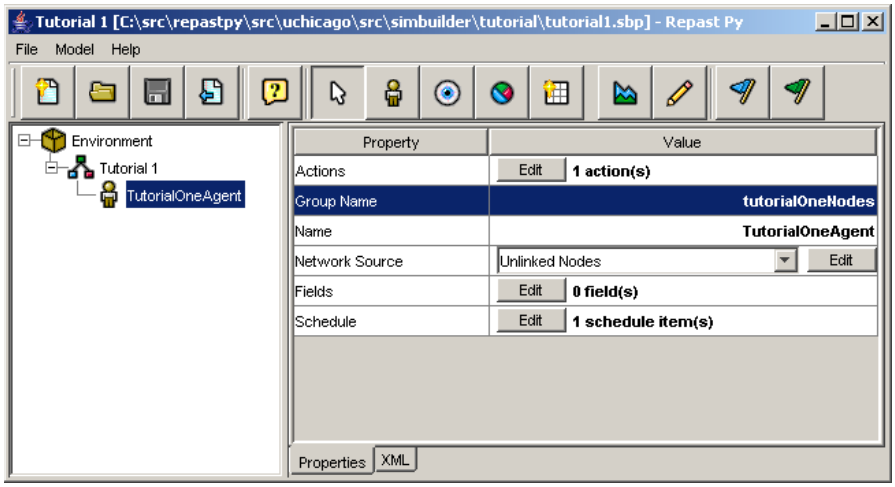


Figure 2: RepastPy GUI for model development (source: Repast, 2007).

1.2.1.2: Java - RepastJ

Object-oriented languages such as Java, easily lend themselves to the creation of extensible frameworks through the use of inheritance and composition[5]. Thus, Repast benefits from the extensibility offered by Java. For instance, a large library of classes originally constructed for alternative means can be used by an agent-based modeller to extend the Repast's generic functionality. For example, the ability to import data from a GIS (GeoTools library), and to display data in a GIS viewer (OpenMap library). Collier (2002) and North *et al.* (2006) provide more details about RepastJ, and show how it is possible to develop a model with this implementation. 'How-to' documentation and demonstration models for RepastJ are available in the installation folder. Tobias and Hofmann (2004) when reviewing several Java based simulation/modelling systems, commented that the RepastJ environment was the "clear winner" with extensive technical

[4] http://www.institute.redlands.edu/agentanalyst.

[5] For further details about object-oriented concepts the author recommends Booch, 1994 for a seminal discussion in object-orientated design and analysis, Hathaway, 2003 for a non-technical discussion of object-oriented principles, and Armstrong, 2006 for a useful evaluation and clarification of key object-oriented notions.

documentation and ‘how-to’ documents that make it easy to become familiar with the software, along with its applications through an active mailing list.

1.2.1.3: Microsoft.NET - Repast.NET

Any programming language compatible with the Microsoft.NET framework (e.g. Visual Basic.NET, C++, J#, C#, etc) can be used to develop a model with Repast.NET. The majority of core and non-core functionality within RepastJ (see Section 1.3) is available in Repast.NET. However, a notable omission is GIS functionality (e.g. ESRI’s ArcGIS, OpenMap, etc). Vos and North (2004) explain that the GIS packages available within RepastJ were not converted into C# for inclusion within Repast.NET because of both time constraints and issues with the integration of the necessary external libraries. Nevertheless, the majority of packages included within RepastJ (both native and external), were converted into C#, or similar replacement libraries have been included. This continuity in packages provides a modeller with a similar development experience in Repast.NET or RepastJ. Similarly, modellers are able to leverage any Repast related knowledge they have (in Repast.NET or RepastJ) and apply it to the alternative implementation of Repast. In the majority of cases the same package names, class names, method names in RepastJ are include in Repast.NET (Vos, 2005). Even though Repast.NET is developed in C#, the interoperability of the .NET framework does not impose any restrictions on the compatible language a modeller can use to develop a model (Vos, 2005). Vos and North (2004) and Vos (2005) provide more details about Repast.NET, and how to develop a model with this implementation. ‘How-to’ documentation and demonstration models for Repast.NET are available in the installation folder.

1.2.1.4: Repast Simphony

Whilst still being maintained, RepastJ, Repast.NET and RepastPy have now reached maturity and are no longer being developed. They have been superseded by Repast Simphony (RepastS) which provides all the core functionality of RepastJ or Repast.NET, although this is limited to implementation in Java (Version 1.5). RepastS provides a more point-and-click interface for developing certain parts of the model; for example,

when creating graphs, less code needs to be written than that for RepastJ and .NET. The Repast development team have provided a series of articles regarding RepastS. The architecture and core functionality are introduced by North *et al.* (2005a), and the development environment is discussed by Howe *et al.* (2006). The storage, display and behaviour/interaction of agents, as well as features for data analysis (i.e. via the integration of the R statistics package) and presentation of models within RepastS are outlined by North *et al.* (2005b). Tatara *et al.* (2006) provide a detailed discussion outlining how-to develop a "simple wolf-sheep predation" model which illustrates RepastS modelling capabilities (Figure 3), specifically the ability to create and display 2 and 3D. RepastS was released as a Beta version in October 2006 and therefore there is little detail about all its core functionality at present or example models. In relation to GIS functionality, there is no support for GIS integration at current.

1.2.1.5: Choosing an Implementation Language

The developers of Repast recommend that basic models are created in RepastPy, due to its visual interface, and advanced models be written with RepastJ or Repast.NET (see North *et al.*, 2004) (and more recently RepastS). Since RepastPy will export into Java, it provides a logical starting point for beginners to develop basic models, or for more experienced users to develop an initial model for further development in Java (RepastJ). For the development of geospatial simulations only RepastPy and RepastJ support the integration of GIS through third party libraries.

RepastJ has several advantages over the other Repast implementations. Firstly, Java is a popular programming language due to its platform independence (i.e. a simulation written in RepastJ can be run on a computer using several different operating systems e.g. Windows™, Unix™, Linux™, Mac™, etc), opposed to a simulation written in Repast.NET. Secondly Java has a much larger set of third-party libraries than Microsoft.NET, particularly in the free and open source arena. This allows for Java development to be much faster, since existing code can be reused in many cases (North *et al.*, 2006). Finally, as Java is the original implementation language of Repast, there are considerably more demonstration models, tutorials, and help facilities via the user

community available for RepastJ. Nevertheless, Repast.NET offers a range of languages in which to develop Repast models. While RepastS is seen as the end goal of Repast by the Repast development team, there are no example models available apart from those in the download. There is little in the way of user documentation and transforming existing RepastJ models to RepastS in the immediate future will require a substantial effort.

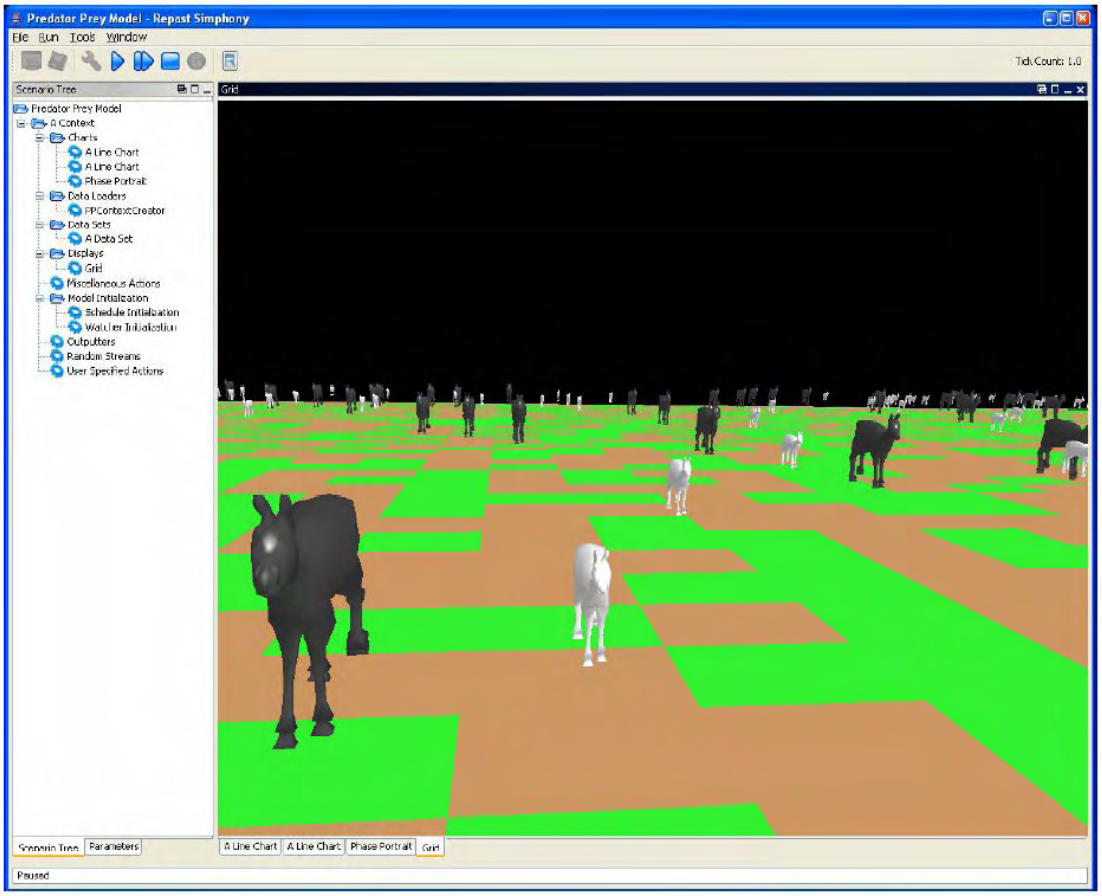


Figure 3: 3D display of a typical model grid. Sheep are light and wolves are dark. Light and dark squares represent living and dead grass, respectively (source: Tatara *et al.*, 2006).

1.3: Developing an Agent-Based Model with Repast

All three development versions (RepastJ, Repast.NET, and RepastPy) are based upon the Repast framework template for agent-based simulation (see North *et al.*, 2004; North *et al.*, 2006 for further details). Conceptually, the Repast toolkit comprises two interacting layers. The core layer runs background/general-purpose simulation code. The external layer which Repast users work with, runs user-specific simulation code written in one of

the implementation languages (Java, .NET or Python). The following subsections will therefore explore Repast's core functionality relating it where appropriate to ABM more generally, before presenting how users have added to this core layer to develop their own agent-based simulations (Section 1.4).

However, a few caveats are required at this point. Firstly while examples of models developed in .NET and Python will be given, the majority of this section is focussed on model development in RepastJ. However, many of the design decisions for all three of these Repast implementation languages are very similar. Secondly, RepastS has a slightly different architecture than RepastJ, Repast.NET and RepastPy (see North *et al.*, 2005b; 2005a for specific details), but at the time of writing this was still in development and RepastJ remains the predominant Repast implementation language for the development of agent-based models.

1.3.1: Time in Repast

Time is what drives the simulation forward. Time within agent-based models is designed to be either synchronous or asynchronous. In synchronous models, all agents are assumed to change simultaneously. The calling order of the objects has no influence in this mode but conflicts can arise when agents compete over limited resources. With asynchronous updating, agents change in turn, each observing the reality left by the previous agent. Conflicts between agents are therefore resolved. In fact, the order of updating (often, but not necessarily, random) is critical as it may influence model results (Benenson and Torrens, 2004). Repast adopts the asynchronous method of time and is classed as a discrete event simulator whose quantum unit of time is known as a '*tick*'. The tick exists only as a hook on which the execution of events can be hung, ordering the execution of the events (Collier, 2002). For example, if event x is scheduled for tick one, event y for tick two, and event z for tick three, then event y will execute after x and before z. Ticks are merely a way to order the execution of events relative to each other. An example could be that a firm only moves when its office lease is over, say twenty-five years, while one tick represents one year. Therefore the firm has to wait for twenty-five ticks before it has the option to move again. However, the Repast scheduling mechanism

additionally allows for more sophisticated dynamic schedules to be created such that the execution of one event can itself schedule over events for the execution in the future (i.e. event driven processes). For example a car only needs to be filled up with petrol after it has been driven. The ability to schedule events at different times using a variety of automata clocks allows one to mimic the temporal attributes of the specific urban process under study (Torrens, 2003).

1.3.1.1: The Scheduling Mechanism

Repast's scheduling mechanism[6] is responsible for all the user-defined state changes within a Repast simulation. Scheduling consists of setting up and executing actions (e.g. agents' behaviour and so forth) at some specific time relative to other actions, thus representing dynamics within the agent-based model. Within Repast, as with any agent-based model, scheduled events may be implemented in three ways (Brown *et al.*, 2005):

1. Events may be sequenced in a synchronous step-wise fashion. For example, each agent, set of agents or non-agent object is signalled to perform its tasks once at each time step or once every n time steps.
2. An event may be scheduled to occur only once at some time step n. Any number of different events may be scheduled to occur in this fashion providing a predetermined history of events to take place.
3. The model may encapsulate 'event-driven' processes whereby model agents may trigger events to occur or may add events to the schedule or queue of events to take place (Ropella *et al.*, 2002).

When the schedule has finished iterating over the execution queue (e.g. a list of agents), the tick count is incremented (e.g. from tick one to tick two). It is this tick count against which actions are scheduled for execution. The ability to easily schedule events is of great value, not only to actions within the model such as updating the display, recording information and so forth but also to agents' states. It is the scheduling of events which alters the agents' state based on the agents' behaviour that has been programmed by the

[6] For more information on Scheduling see http://repast.sourceforge.net/api/uchicago/src/sim/engine/Schedule.html.

modeller. It is the execution of these behaviours according to some schedule that alters the agents' state. It should be stressed that Repast does not specify any particular state composition, nor does it specify any mechanism for interaction between the environment and the agents that make up that state. All changes are needed to be specified by the modeller. Additionally, Repast provides no set formula for creating a model; it is a mater of deciding what is required and linking to the appropriate classes. The modeller is still required to write the model, deciding what space to use and how the agents interact etc. This therefore requires a substantial amount of effort to understand how to design and implement a model.

1.3.2: Displaying, Running and Recording Change

One of Repast's key features is that it provides a series of classes for displaying a running simulation in real time, thus freeing the modeller to spend more time on model development rather than developing the GUI. Displays can range from simple graphs and scatter graphs, to visualising the agents interacting in space. Displays, like agents can be scheduled to update whenever it is deemed necessary by the modeller. The ability to display and schedule events are two of most useful features of Repast.

Once a model has been created, Repast provides the modeller with two ways for running a simulation: batch-run, and non-batch run. A batch-run simulation reads in a specially formatted parameter file detailing the starting and ending values of a model's parameters, how to increment these parameters and the number of runs to complete. The simulation then begins to run. This provides a method for determining whether the conclusion from a simulation run is typical by comparing it to multiple simulation runs therefore helping with validation of the model.

A non-batch run requires a user to start and stop a run through a GUI, and allows the user to graphically set starting parameters. Additionally it allows a user to graphically display both the model and manipulate (i.e. probe) an agent's state during the course of a run, therefore providing a way to visualise change and gain an understanding of how phenomena of interest develops. Both batch and non-batch runs allow the modeller to

see if initial starting conditions affect the outcome of the model and chart this change over time.

While being able to create and run models is important, so is the ability to capture this change. Repast has a number of inbuilt functions for storing results from simulations and individual time-steps. These include the ability to write information to text files, recording displays (charts, grids etc) as movies or images. The latter allows for visual inspection of the simulation run, for example, how the system evolves. The former allows for data to be integrated into statistical packages for further investigation. Both aid in verification and validation of the model.

1.3.3: The Representation of Space in Repast

Within Repast, three types of space representations are available to the modeller; that of network space[7], cellular space and vector/continuous space. It is Repast's direct support for both raster[8] and vector[9], geospatial data using cellular and vector space classes respectively, that makes it stand out amongst other open source simulation/modelling systems. However, as stated above (see Section 1.2.1), this is only possible in RepastJ and RepastPy. Space within agent-based models serves two purposes; firstly it contains the agents and secondly it defines the spatial relationships between the agents and controls their movement. The following subsections will outline the spaces that allow the modeller to create spatially explicit agent-based models within Repast, that of cellular and vector space. However, it is important to note that while it is possible to create these spaces, the modeller still needs to program/define spatial relationships between agents and their environment and define how the agents move within the space.

1.3.3.1: Cellular Space

Cellular space models are by far the most widely used environment for the creation of spatially explicit agent-based models in the social sciences. Repast has numerous cellular

[7] See the Repast's Network Model Tutorial for more information: http://repast.sourceforge.net/how-to/network.html.

[8] Raster: A spatial data model in which features are represented by pixels. Each pixel is assigned a value that corresponds to a feature.

[9] Vector: A spatial data model in which features are represented by points, lines and polygons.

space options from regular to hexagonal grids, and a full list can be seen at Repast's 'How-to' Spaces Overview[10]. Cells can contain single and multiple agents or objects. Of interest here is Repast's ability to import raster datasets to build the environment which the agents occupy. Repast provides a set of classes that can read in and which interpret the ESRI ASCII[11] raster file format[12], thus providing the ability to import and export data directly to and from a GIS such as ESRI's ArcGIS or GRASS.

The raster image is treated as a continuous system of coordinates with cells that hold values. The cell size and other attributes can all be set by the user and allows space to be traversed like a landscape. For example, Figure 4A is the original ASCII file from the GIS which contains details of a number of rows and columns. In each row and column there is a cell, and each cell has a size and a value associated with it. Figure 4B shows the display created in Repast using its inbuilt display classes, whereby the different cell values are shaded different colours. Multiple layers can also be added to the model to represent different characteristics of a landscape such as land-use, elevation etc.

Agents can either be the cells imported into the model or sit on top of the cells and interact with the surface generated using raster data. Each agent has an x and y coordinate and the space contains the agent. Agent movement is therefore is a matter of changing the agent's internal x and y coordinates, removing it from its previously occupied cell and adding it to the new one.

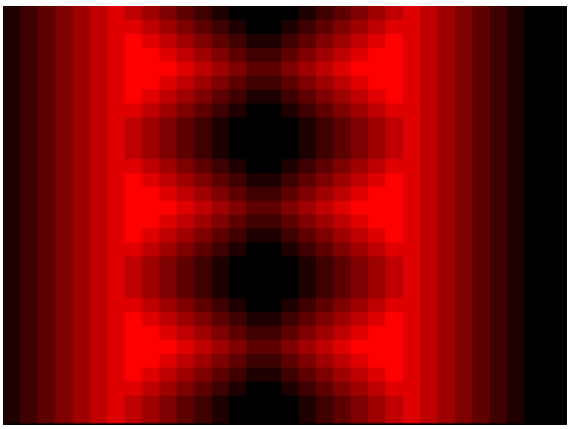
The advantage of using raster data is that it uses Repast's standard display functions which the author believes to be more efficient than the vector space option. Additionally the modeller can use Moore and von Neumann neighbourhoods or a variation of either to define neighbourhoods which are not possible in vector space. However, the disadvantage of using raster-based data is that to create a landscape, numerous layers are needed as cells can only contain one value. Furthermore, while the linkage of agent-based models and cellular space allows the modeller to capture geographic detail and

[10] Repast's Space Overview 'How-To' document: http://repast.sourceforge.net/how-to/spaces.html.
[11] ASCII: American Standard Code for Information Interchange.
[12] Alternatively called ESRI's grid format.

have provided valuable insights into urban phenomena, it misses geometric detail (Batty, 2005). The ability to represent the world as a series of points, lines and polygons allows the inclusion of geometry into the modelling process, therefore allowing for different sizes of features such as houses, roads etc. to be portrayed. It is to this we now turn.

A

```
ncols          30
nrows          30
xllcorner      528000.00
yllcorner      500000.00
cellsize       30
NODATA_value   -9999
1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 0 0 1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6 7 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 7 6 5 4 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 0 0 1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 0 0 1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 0 0 1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6 7 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 8 7 6 5 4 3 2 2 3 4 5 6 7 8 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
1 2 3 4 5 6 7 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 7 6 5 4 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 0 0 1 2 3 4 5 6 7 6 5 4 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 0 0 1 2 3 4 5 6 7 6 5 4 3 2 1
```



B

Figure 4: Reading in a raster data and creating a landscape (ESRI ASCII file) A: the original file from a GIS. B: the resulting space created in Repast.

1.3.3.2: Vector Space

Repast provides a set of classes that allow users to work directly with vector GIS data in their models. However, before this integration is discussed, there needs to be a discussion on how vector geospatial data can be represented in an agent-based model, due to it having close analogies with both object-orientated programming and how one represents reality within models.

A typical GIS contains multiple layers of data. A layer is made up a number of elements. For example, a layer might contain a number of houses that represents a part of an area (Table 1). While other layers might include data on environmental factors such as parks. Each house in the layer would be a GIS feature (with an associated feature ID). Each feature in the layer has two aspects to it, its geographical coordinates and the data associated with it (its attributes). A common format for storing this information is ESRI's shapefile[13]. A number of files are associated with the shapefile format: (1) the shapefile (.shp), which stores the geographical information needed to display the feature (x, y, z coordinates of vertexes and edges of the geometric shapes); (2) the database file (.dbf), which stores the data records for the feature; and (3) the index file (.shx) which links the database file to the shapefile (via the feature ID).

Table 1: Sample of GIS type data, represneting a layer of data in a GIS.

House Layer			
Feature ID	Geographical Coordinate	Type	Age
1	22,22	Detached	23
2	23,12	Flat	51
3	23,32	Semi-detached	100

While a GIS stores data about layers in database files, with each record in the file referring to a feature in the GIS (e.g. Feature ID 1 in Table 1 is of type detached), agent-based models handle data differently. While a GIS is layer-centric, an agent-based model is agent-centric. Thus, each agent stores data about itself individually. However, there is a degree of overlap. For instance, each agent type has the same types of data, just as each

[13] ESRI Vector Shapefile format.

layer type in a GIS has the same types of data. Thus, an agent type in an agent-based model can be seen as similar to a layer in a GIS, and each agent in the agent-based model is similar to a feature in GIS (Najlis and North, 2004).

GIS data can be translated directly into agents or into other data objects that the agents use or know about (e.g. the locations of train stations or a street network). To highlight this, a short example will be presented. Given a model where agents are householders, the GIS data might relate to the houses that the agents own. The house data would then be read into the agent-based model for use by householders. The support vector-based geospatial data therefore makes it is possible to create an agent description from a shapefile such that features specified in the shapefile become fields in the produced agent. For example, if the shapefile contains a feature called 'Type', then the produced agent will have a 'Type' field. When the actual agents are created, each record in the shapefile provides the data for an agent. So, if the shapefile has 600 records pertaining to houses, then 600 household agents are created (see Figure 5). Similarly, in order to update the GIS data based on the agents, a corresponding function has to be specified in the agent class relating the agents field to the same field in the shapefile.

Repast integrates vector data directly into an agent-based model as described above by providing a set of classes that allow users to work directly with vector GIS data in their models, be they points, lines or polygons[14]. In Repast, agents represented as polygons are referred to as 'vector agents' which can act in the same way as irregular CA models where their states can change but their boundaries are fixed. Points can either be fixed or mobile and Repast refers to them as 'generic agents'

The current implementation of GIS functionality within Repast is focused on two systems: ESRI's ArcGIS via the Agent Analyst extension and the open source software OpenMap[15]. While both rely on ESRI shapefiles, the type of integration between the

[14] For further information on working with vector data in Repast see: How to use GIS data with Repast (http://repast.sourceforge.net/how-to/Gis_How_To.html) and Najlis and North (2004; 2005).
[15] OpenMap™: http://openmap.bbn.com/.

agent-based model and the GIS are different. Repast has shapefile integration with ArcGIS and native Java integration with OpenMap (Najlis and North, 2004).

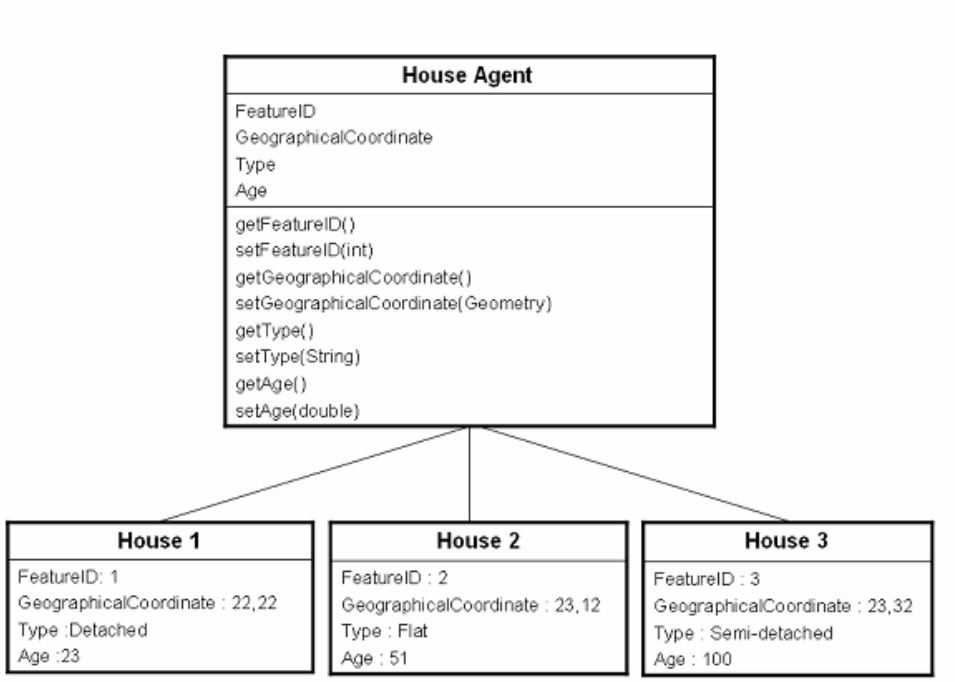


Figure 5: GIS data in an agent-based model: House Agent is the agent type and is akin to a housing layer in GIS (after Najlis and North 2004).

Shapefile integration refers to fact that while the model is created using one of Repast's implementation libraries and that Repast and the GIS use the same shapefile, they have very limited interaction with each other. The shapefile is loaded into the GIS, the model then reads in the data and agents are created using the shapefile data. The model is then run. However, before the GIS display can be updated, the shapefile must first be written out to file and then the GIS is notified to update its display by reading the newly written (or updated) shapefile. With shapefile integration, only the GIS display can be updated. It does not provide a means for the model to interact with the Application Programming Interface (API) of the GIS. Therefore, for topologic calculations, the user has to link the agent-based model to the Java Topology Suite (JTS)[16].

[16] Java Topology Suite: http://www.vividsolutions.com/jts/jtshome.htm.

Java integration on the other hand allows the model to have full interaction with the GIS using OpenMap's native Java libraries which can be used for both topologic calculations and for display. Alternatively the modeller can use OpenMap for the display of the model and JTS for topology calculations. In either case, the shapefile needs to be projected in World Geodetic System 84 (WGS84)[17] unlike shapefile integration which can use any coordinate system. Java integration works by loading the shapefile into model, the GIS is then launched from within model, and the shapefile data is added to the GIS from model. Agents are created from and written to shapefiles in the same manner as with shapefile integration.

Both types of integration allow the model users full access to the GIS being used for display, allowing for information retrieval about agents, such as location, distance from other geographic objects (including geographically represented agents) once programmed by the modeller. Both provide the same functionality relating to reading and writing of data to shapefiles, and agents and models created in one can easily be converted into the other. However, Najlis and North (2004) suggest that the choice of integration type and GIS to use depends on the needs of the project. For example, if the project requires analysis on data during the run of the model, it might be appropriate to use ArcGIS's extensive analytical capabilities. For example the model could be paused, and ArcGIS could be used to analyze the data as needed. On the other hand, if what is needed is the ability to update data quickly, query the GIS about an agent's spatial characteristics during the run of the model, and use that information from within the model itself, then it might be more appropriate to choose Java integration. Additionally Java integration has no reliance on ArcGIS. Therefore people who do not have ArcGIS can still create geospatial agent-based models and distribute them freely once created.

1.3.3.3: Discussion of Space

This section has highlighted Repast's ability to import and export geospatial data in both raster and vector formats. Both types of spaces have their advantages and disadvantages. For cellular space, it is much easier conceptually and computationally as one can use

[17] World Geodetic System defines a geographical reference frame for the earth.

Moore and von Neumann neighbourhoods. Additionally one can use Repast's inbuilt display surfaces rather than using some other third party software. Secondly much data comes in raster formats such as remote sensing data. However, using cellular space, there is no way of representing complex geometries.

The advantages of using vector space include the ability of giving realism to models by allowing the representation of any geographical shape and thus represents a movement away from treating the world as a series of square objects. Secondly, most demographic data comes in this format, thus the data does not have to be transformed to a series of cells. However, these advantages come at a price. Topological calculations (such as neighbourhood calculations, distance between two points, and point in polygon operations) are expensive in terms of execution speed, thus limiting the performance of a vector-based model.

Which space to use for geospatial models, specifically for urban applications, depends on the purpose of the model. Benenson *et al.* (2005) comment that while vector GIS can represent urban objects in spatially explicit models, for theoretical models, the points of a regular grid will usually suffice.

1.4: Applications that have Utilized Repast to Create Agent-Based Models

The previous section has presented the core concepts for creating an agent-based model using the Repast toolkit, specifically how Repast manages time and how events can be scheduled, its ability to display and record information, and its ability to represent geospatial information. It was noted that models created using the Repast have no set formula for what to include and not include in a model. This flexibility allows a whole host of models to be created, some of which will be explored in this section.

As with ABM in general, to compile a fully comprehensive list of models utilizing Repast is impractical and beyond the scope of this paper. Agent-based models utilizing Repast have been developed for a diverse range of subject areas such as business

strategies (Robertson, 2003; López-Sánchez *et al.*, 2005), revenue management policies (Faber, 2005), the evolution of firms and the dynamics between firms (Padgett *et al.*, 2003; Tivnan, 2004), electricity markets (North *et al.*, 2002), effects of road user charging (Takama, 2005), kinship networks in Australian rangeland landscapes (McAllister *et al.*, 2005), the rise and fall of nation-states (Cederman, 2003; Cioffi-Revilla and Gotts, 2003), segregation (Bruch and Mare, 2005), battlefield simulation (Baker *et al.*, 2005), individual co-operation (Galan and Izquierdo, 2005), car injury prevention (Kobti *et al.*, 2005), the growth of hydrogen transportation infrastructure (Stephan and Sullivan, 2004), and stone assembly by ancient peoples (Brantingham, 2003). Most of these applications treat space abstractly if at all. The remainder of this section will focus on specific spatial applications utilizing Repast, in particular, those that utilize geospatial data. These include house price evaluation, residential segregation, disaster management applications, land-use models and work from my own research group at the Centre of Advanced Spatial Analysis (CASA).

Bossomaier *et al.* (2007) used Repast to study house price evolution in Bathurst, Australia specifically vendor/buyer behaviour. The agent's decisions/behaviour of where to locate is affected by spatial attributes of actual land-parcels. Geospatial data is used to determine a range of attributes for each property including distance from amenities such as parks, area, elevation, orientation and environmental factors such as flood risk. These spatial factors combined with an agent's perceptions about the economy, interest rates, new developments such as factories and roads, plus social trends in the desirability of house ownership and property investment then influence how buyers and sellers modify the price relative to the neighbourhood. Additionally Bossomaier *et al.* (2007) highlight how Repast's flexible architecture allows other Java libraries to easily be incorporated into the model; for example, they integrated the NRC FuzzyJ toolbox[18] into their model simulation where each agent (buyer or seller) had a number of attributes that are fuzzy variables (such as eagerness to buy or sell and the agent's perception of the market which the authors claimed had the advantage of realistically capturing the nature of human decision making).

[18]NRC FuzzyJ toolbox: http://iit-iti.nrc-cnrc.gc.ca/license/info_e/6.

O'Sullivan *et al.* (2003) used Repast to present a variation on Schelling's (1971) model of residential location dynamics that combines two concepts of neighbourhood, continuous and bounded within a regular lattice structure (Figure 6). The small grid cells represent residential locations with a continuous neighbourhood structure. Residential locations are contained in bounded neighbourhoods, whose aggregate state is also considered by agents in the residential decision making behaviour. The arrows represent relations of influence on decision making. Two geospatial datasets were used one to represent residential locations in the continuous layer and a second to represent bounded neighbourhoods.

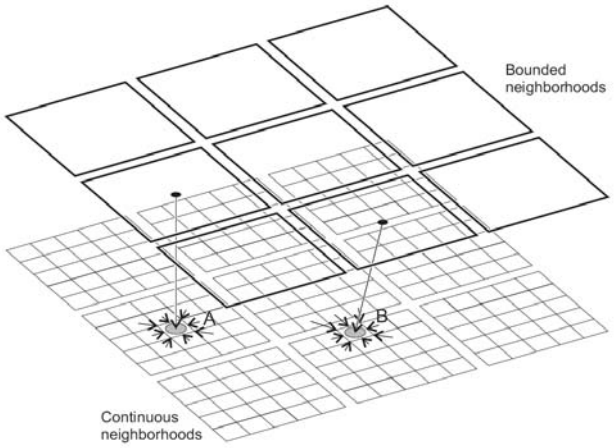


Figure 6: Structure of the hierarchical Schelling model (source: O'Sullivan *et al.*, 2003).

Repast has additionally been used to examine disaster management such as a Sarin attack in Manhattan (Mysore *et al.*, 2006), the spread of infectious diseases among students at the University of Southampton (Yang and Atkinson, 2005), and a food poisoning outbreak that occurred in Brazil (Mysore *et al.*, 2005). These applications allow for the testing of different scenarios, thus aiding planning efforts for similar events in the future. Additionally they allow modellers to integrate geospatial phenomena both in time and space.

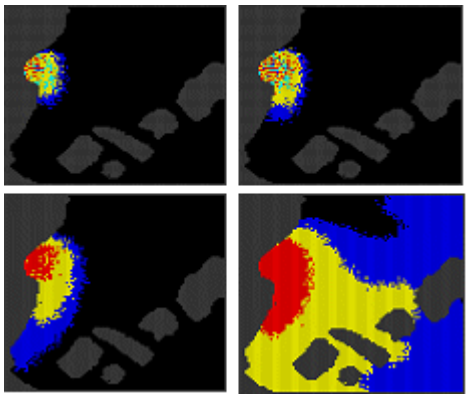
Specific models utilizing components from Repast to study land-use change include the SLUCE (Spatial Land-use Change and Ecological Effects at the Rural-Urban Interface) Project which examines land-use change at the urban-rural fringe (see Brown *et al.*, 2005). In the context of how individual decision-making drives land-use decisions, the model lets the user formulate and test alternative policies and interventions that could reduce environmental costs and enhance environmental benefits. A similar model has also been developed by Yin and Muller (2007) who examine land-use-land-cover change at the urban-rural fringe incorporating households decision making in terms of preferences for accessibility, amenities, and scenic views, all of which are calculated in a GIS before being fed into the model. Deadman *et al.* (2004) have built a model to understand and explore spatial, social and environmental issues related to land-use/cover change within the Brazilian Amazon, specifically focusing on the behaviour of heterogeneous agricultural land owners. Other models that utilize Repast to explore land-use/land-use change include: Xie *et al.* (2005) who developed a model to simulate the rapid urbanisation of densely populated areas in China, and Su and Duggin (2003) who have developed a model to study the succession of vegetation after open-cut mining.

Researchers at CASA have utilized Repast (specifically RepastJ) to study a range of applications (Figure 7): pedestrian modelling both in retail (Zachariadis, 2005) and emergency evacuation (Castle 2006), urban dynamics in Latin American Cities (Barros, 2004), and segregation, residential and business location (Crooks, 2006).

These models as with other models presented in this section use the core Repast functionality, extending it further to meet their own specific needs, specifically the ability to import geospatial data to build the environment and to create agents to explore different types of phenomena. Additionally the applications developed at CASA highlight how different scales of phenomena can be modelled at different temporal resolutions (Figure 8), and demonstrate how agent-based models provide a suitable means for exploring many aspects of urban phenomena[19].

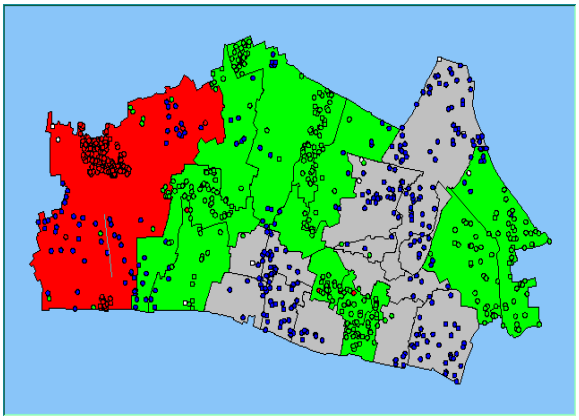
[19] Further information pertaining to these applications can be found at http://www.casa.ucl.ac.uk/repast/.

Pedestrian Modelling (Emergency evacuation) (Castle 2006)



Urban Dynamics in Latin American Cities (Barros, 2004)

Pedestrian Modelling (Retail) (Zachariadis, 2005)



Segregation Modelling (Crooks, 2006)

Figure 7: Screen shots of some Repast applications that are being developed at CASA.

1) Urban Dynamics in Latin American Cities
2) Pedestrian Modelling
   a) Retail
   b) Emergency evacuation
3) Segregation & Locational Modelling

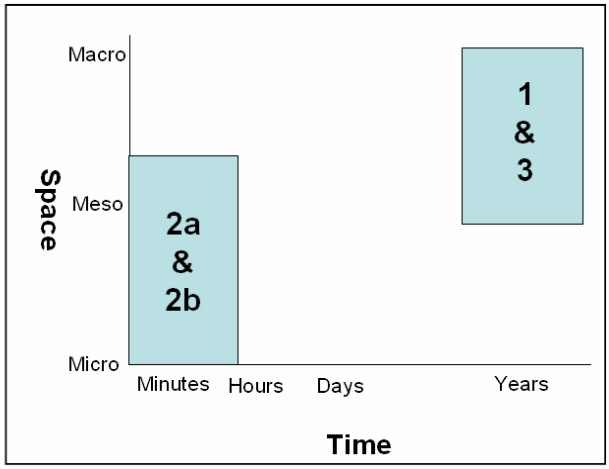


Figure 8: Applications at CASA - comparison of time against space.

Other modellers have extended Repast to meet their own simulation needs. These include GeoGraph (3D extensions) and AgentCell (biochemical reactions 3D) and its use in parallel processing. Dibble and Feldman (2004) extended Repast using GeoGraph 3D, and their extension supports models in which mobile agents travel and interact on rugged terrain or on network landscapes at any scale from rooms within buildings to urban neighbourhoods to large geographic networks of cities. Interactive 3D visualizations allow researchers to zoom and pan within the simulation landscape as the model runs. Model-specific 3D representations of agents flock together on terrain landscapes and teleport or travel along links on network landscapes. Agents may be displayed on network nodes either as individual agents or as dynamic 3D bar charts that reflect the composition of each node's population. GeoGraph has been applied to the dynamics of civil violence, controlling epidemics of infectious diseases, and social networks connecting geographically mobile team members, for example, command, control, and communication structures for effective peacekeeping teams facing riots and related dynamic and distributed problems of civil unrest.

A second example of where Repast has been extended is for a basis in parallel processing. Minson and Theodoropoulos (2004) note that while agent-toolkits provide reliable templates for the design of even the largest agent-based simulations, they do not offer a solution to their computational limitations. Conversely, distributed simulation architectures offer performance benefits but the introduction of parallel logic can complicate the design process significantly. To overcome this dilemma, Minson and Theodoropoulos (2004) designed and implemented a system capable of harnessing the computational power of a distributed simulation infrastructure with the design efficiency of an agent-toolkit.

A third extension is that of AgentCell[20] a model using agent-based technology to study the relationship between stochastic intracellular processes and behaviour of individual cells in a 3D environment. AgentCell is an open source project licensed under the

[20] AgentCell: http://www.agentcell.org/.

General Public License (GNU, 2007) and is built on top of Repast and uses StochSim[21] to implement the Network class that simulates the biochemical reactions. AgentCell is an open source project licensed under the General Public License and is freely downloadable[22]. AgentCell's design and some initial results are discussed in the Emonet *et al.* (2005).

1.4.1: Discussion of Models Created Utilizing Repast

This section has highlighted a diverse range of applications utilizing the Repast framework to build agent-based models. These examples as with ABM in general, can be constructed as lying on a continuum from minimalist academic models based upon ideal assumptions, to large scale decision support systems based upon real-world data. Much of the work has been carried out by researchers in fields outside (although related to) geography, public policy, economics, environmental studies, and city and regional planning. The role of space and spatial mechanisms in exploring the phenomena are not of utmost importance in these contexts. Secondly it is also partly due to initial versions of Repast being particularly strong in its support for network (social and otherwise) simulations (see Collier, 2002) and this is reflected in many of the earlier applications.

Applications however are being developed that focus on spatial phenomena, specifically those that utilize geospatial data. The majority of the models presented here do so by utilizing Repast's ability to import ASCII data (e.g. Brown *et al.*, 2005; Castle, 2006; Yin and Muller, 2007), while some do not make this clear (e.g. O'Sullivan *et al.*, 2003). Few utilize Repast's ability to import vector-based datasets (e.g. Mysore *et al.*, 2006, Crooks, 2006 are notable exceptions). This possibly reflects the high computational burden of representing an area using a vector space (e.g. as a series of points, line and polygons). It also reflects the trend in the social sciences for representing agents as cells. Additionally the models designed to explore spatial phenomena can be seen considered as either being loosely coupled or modelling-centric (see Castle and Crooks, 2006 for discussion on the linkage between GIS and simulation/modelling systems), thus highlighting that a variety

[21] StochSim: http://www.anat.cam.ac.uk/pages/comp-cell/StochSim.html.
[22] AgentCell download: http://sourceforge.net/projects/agentcell.

of integration approaches can be used to match the purpose of the model to explain or explore different types of phenomena.

1.5: Summary

This paper has introduced the reader to Repast, described its different implementation languages, before presenting Repast's core functionality, specifically its ability to handle time and its ability to incorporate both raster and vector data directly into the model. It thus allows geographers and urban modellers the ability to deal with space in both raster and vector formats. It was stressed that the ability to build models using vector data allows the modeller to represent any geographical shape and how this is moves models away from treating the world as a series of square objects or grid cells.

A range of models were then presented, specifically focussing on ones which utilise geospatial data to build environments and agents. It was highlighted that the majority of these models were based on Repast's ability to import raster data and it is suggested this resulted from the high computational resources to represent an area as a series of points, line and polygons. Throughout the paper it was stressed that Repast is only a toolkit to facilitate the creation of agent-based models. One still needs to write code linking the model components together as Repast has no set formula for creating an agent-based model. The modeller can thus add as much functionality provided by Repast into their own model as desired.

References

Armstrong, D.J. (2006), 'The Quarks of Object-Oriented Development', *Communication of the ACM*, 49(2): 123-128.

Baker, T.J.A., Botting, M., Berryman, M.J., Ryan, A., Grisogon, A.M. and Abbott, D. (2005), 'Adaptive Battle Agents: Emergence in Artificial Life Combat Models', *Proceedings of the SPIE, Smart Structures, Devices, and Systems II*, Bellingham, WA, pp. 574-585.

Barros, J. (2004), *Urban Growth in Latin American Cities: Exploring Urban Dynamics through Agent-Based Simulation*, Ph.D. Thesis, University College London, London, UK.

Batty, M. (2005), *Cities and Complexity: Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*, The MIT Press, Cambridge, MA.

Benenson, I., Aronovich, S. and Noam, S. (2005), 'Let's Talk Objects: Generic Methodology for Urban High-Resolution Simulation', *Computers, Environment and Urban Systems*, 29(4): 425–453.

Benenson, I. and Torrens, P.M. (2004), *Geosimulation: Automata-Based Modelling of Urban Phenomena*, John Wiley & Sons, London, UK.

Booch, G. (1994), *Object-Oriented Analysis and Design with Applications*, Benjamin/Cummings, Redwood City, CA.

Bossomaier, T., Amri, S. and Thompson, J. (2007), 'Agent-Based Modelling of House Price Evolution', *Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)*, Honolulu, HI, pp. 463 - 467.

Brantingham, P. (2003), 'A Neutral Model of Stone Raw Material Procurement', *American Antiquity*, 68(3): 487–509.

Brown, D.G., Riolo, R., Robinson, D.T., North, M.J. and Rand, W. (2005), 'Spatial Process and Data Models: Toward Integration of Agent-Based Models and GIS', *Journal of Geographical Systems*, 7(1): 25-47.

Bruch, E. and Mare, R.D. (2005), *Neighbourhood Choice and Neighbourhood Change*, California Centre for Population Research University of California – Los Angeles, Los Angeles, CA, Available at http://www.stat.ucla.edu/~bruch/NCNC.pdf.

Castle, C.J.E. (2006), 'Using Repast to Develop a Prototype Agent-Based Pedestrian Evacuation Model', in Sallach, D., Macal, C.M., and North, M.J. (eds.), *Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects*, University of Chicago and Argonne National Laboratory, Chicago, IL, Available at http://agent2007.anl.gov/2006procpdf/Agent_2006.pdf.

Castle, C.J.E. and Crooks, A.T. (2006), *Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations*, Centre for Advanced Spatial Analysis (University College London): Working Paper 110, London, UK.

Cederman, L.E. (2003), 'Modelling the Size of Wars: From Billiard Balls to Sand Piles', *American Political Science Review*, 97(1): 135-150.

Cioffi-Revilla, C. and Gotts, N. (2003), 'Comparative Analysis of Agent-Based Social Simulations: GeoSim and FEARLUS Models', *Journal of Artificial Societies and Social Simulation*, 6(4), Available at http://jasss.soc.surrey.ac.uk/6/4/10.html.

Collier, N.T. (2002), Repast: An Extensible Framework for Agent Simulation, Available at http://www.econ.iastate.edu/tesfatsi/RepastTutorial.Collier.pdf [Accessed on June 16th, 2006].

Collier, N.T. and North, M.J. (2004), 'Repast for Python Scripting', in Macal, C.M., Sallach, D. and North, M.J. (eds.), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Chicago, IL, pp. 231-237, Available at http://www.agent2005.anl.gov/Agent2004.pdf.

Collier, N.T and North, M.J. (2005), 'Repast for Python Scripting', *Annual Conference of the North American Association for Computational Social and Organizational Science (NAACSOS)* Notre Dame, IN, Available at http://www.casos.cs.cmu.edu/events/conferences/2005/2005_proceedings/Collier.pdf.

Crooks, A.T. (2006), 'Exploring Cities using Agent-Based Models and GIS', in Sallach, D., Macal, C.M., and North, M.J. (eds.), *Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects*, University of Chicago and Argonne National Laboratory, Chicago, IL, Available at http://agent2007.anl.gov/2006procpdf/Agent_2006.pdf.

Deadman, P.J., Robinson, D.T., Moran, E. and Brondizio, E. (2004), 'Effects of Colonist Household Structure on Land Use Change in the Amazon Rainforest: An Agent Based Simulation Approach', *Environment and Planning B*, 31(5): 693-709.

Dibble, C. and Feldman, P.G. (2004), 'The GeoGraph 3D Computational Laboratory: Network and Terrain Landscapes for Repast', *Journal of Artificial Societies and Social Simulation*, 7(1), Available at http://jasss.soc.surrey.ac.uk/7/1/7.html.

Emonet, T., Macal, C.M., North, M.J., Wickersham, C.E. and Cluzel, P. (2005), 'AgentCell: A Digital Single-Cell Assay for Bacterial Chemotaxis', *Bioinfomatics*, 21(11): 2714-2721.

Faber, F.J. (2005), *An Extensible Order Promising and Revenue Management Test-Bed*, MSc. Thesis, University of Maryland, MD, Available at

https://drum.umd.edu/dspace/handle/1903/2666.

Galan, J.M. and Izquierdo, L.R. (2005), 'Appearances can be Deceiving: Lessons Learned Re-Implementing Axelrod's "Evolutionary Approach to Norms"', *Journal of Artificial Societies and Social Simulation*, 8(3), Available at http://jasss.soc.surrey.ac.uk/8/3/2.html.

GNU. (2007), GNU General Public License, Available at http://www.gnu.org/copyleft/gpl.html [Accessed on 27th July, 2007].

Hathaway, R.J. (2003), Basics of Object-Orientation, Available at http://www.objectfaq.com/oofaq2/index.html [Accessed on August 20th, 2006].

Howe, T.R., Collier, N.T., North, M.J., Parker, M.T. and Vos, J.R. (2006), 'Containing Agents: Contexts, Projections, and Agents', in Sallach, D., Macal, C.M., and North, M.J. (eds.), *Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects*, University of Chicago and Argonne National Laboratory, Chicago, IL, Available at http://agent2007.anl.gov/2006procpdf/Agent_2006.pdf.

Kobti, Z., Rahaman, S., Kent, R.D., A.W., S. and Dunlop, T. (2005), 'Multi-Agent Model Prototype For Child Vehicle Safety Injury Prevention', in Macal, C.M., North, M.J. and Sallach, D. (eds.), *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, Chicago, IL, Available at http://agent2007.anl.gov/2005procpdf/%20Agent_2005.pdf.

López-Sánchez, M., Noria, X., Rodríguez, J.A. and Gilbert, N. (2005), 'Multi-Agent Based Simulation of News Digital Markets', *International Journal of Computer Science & Applications* 2(1): 7 – 14.

McAllister, R.R.J., Gordon, I.J. and Stokes, C.J. (2005), 'KinModel: An Agent-Based Model of Rangeland Kinship Networks', in Zerger, A., and Argent, R.M. (ed.),

International Congress on Modelling and Simulation (Modelling and Simulation Society of Australia and New Zealand, MODSIM), pp. 170-176, Available at http://www.mssanz.org.au/modsim05/papers/mcallister_1.pdf.

Minson, R. and Theodoropoulos, G. (2004), 'Distributing Repast Agent Based Simulations with HLA', *Proceedings of the 2004 European Simulation Interoperability Workshop*, Edinburgh, UK, Available at *http://www.cs.bham.ac.uk/~rzm/research/papers/siw04.pdf* .

Mysore, V., Gill, O., Daruwala, R.S., Antoniotti, M., Mishra, B. and Saraswat, V. (2005), 'Multi-Agent Modelling and Analysis of the Brazilian Food Poisoning Scenario', in Macal, C. M., North, M.J. and Sallach, D. (eds.), *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, Chicago, IL, Available at http://www.agent2005.anl.gov/2005pdf/Mysore%20et%20al.pdf.

Mysore, V., Narzisi, G. and Mishra, B. (2006), 'Agent Modelling of a Sarin Attack in Manhattan', in Jennings, N.R., Tambe, M., Ishida, T. and Ramchurn, S.D. (eds.), *First International Workshop on Agent Technology for Disaster Management*, Future University, Hakodate, Japan.

Najlis, R., Janssen, M.A. and Parker, D.C. (2001), 'Software Tools and Communication Issues', in Parker, D.C., Berger, T. and Manson, S.M. (eds.), *Meeting the Challenge of Complexity: Proceedings of a Special Workshop on Land-Use/Land-Cover Change*, Irvine, CA, Available at http://www.csiss.org/resources/maslucc/ABM-LUCC.pdf.

Najlis, R. and North, M.J. (2004), 'Repast for GIS', in Macal, C. M., Sallach, D. and North, M.J. (eds.), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Chicago, IL, pp. 255-260, Available at http://www.agent2005.anl.gov/Agent2004.pdf.

Najlis, R. and North, M.J. (2005), 'Repast Vector GIS Integration', *Annual Conference of the North American Association for Computational Social and Organizational Science (NAACSOS)* Notre Dame, IN, Available at http://www.casos.cs.cmu.edu/events/conferences/2005/2005_proceedings/Najlis.pdf.

North, M.J., Collier, N.T., Vos, J.R., Najlis, R. and Maciorowski, W. (2004), 'The Repast Revolution: An Overview of New Repast Developments', in Macal, C. M., Sallach, D. and North, M.J. (eds.), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Chicago, IL, pp. 223-230, Available at http://www.agent2005.anl.gov/Agent2004.pdf.

North, M.J, Macal, C. M., Cirillo, R., Conzelmann, G., Koritarov, V., Thimmapuram, P. and Veselka, T. (2002), 'Multi-Agent Modelling of Electricity Markets', *Proceedings of the Agent 2002 Conference on Social Agents: Ecology Exchange and Evolution*, University of Chicago and Argonne National Laboratory, Chicago, IL, Available at http://www.agent2003.anl.gov/proceedings/2002.pdf.

North, M.J., Collier, N.T. and Vos, J.R. (2006), 'Experiences Creating Three Implementations of the Repast Agent Modelling Toolkit', *ACM Transactions on Modelling and Computer Simulation*, 16(1): 1-25.

North, M.J., Howe, T.R., Collier, N.T. and Vos, J.R. (2005a), 'The Repast Simphony Development Environment', in Macal, C.M., North, M.J. and Sallach, D. (eds.), *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, Chicago, IL, Available at http://www.agent2005.anl.gov/2005pdf/Mysore%20et%20al.pdf.

North, M.J., Howe, T.R., Collier, N.T. and Vos, J.R. (2005b), 'The Repast Simphony Runtime System', in Macal, C.M., North, M.J. and Sallach, D. (eds.), *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, Chicago, IL, Available at

http://www.agent2005.anl.gov/2005pdf/Mysore%20et%20al.pdf.

O'Sullivan, D., MacGill, J. and Yu, C. (2003), 'Agent-Based Residential Segregation: A Hierarchically Structured Spatial Model', in Macal, C.M., North, M.J. and Sallach, D. (eds.), *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*, The University of Chicago, IL, pp. 493-507 Available at http://www.agent2004.anl.gov/Agent2003.pdf.

Padgett, J.F., Lee, D. and Collier, N.T. (2003), 'Economic Production as Chemistry', *Industrial and Corporate Change*, 12(4): 843-877.

Parker, D.C. (2001), Object-Orientated Packages for Agent-Based Modelling, Available at http://mason.gmu.edu/~dparker3/spat_abm/lectures/lecture2_tables.pdf [Accessed on October 27th, 2005].

Redlands Institute (2006), What is Agent Analyst?, Available at http://www.institute.redlands.edu/agentanalyst/AgentAnalyst.html [Accessed on May 31st, 2006].

Repast (2007), RepastPy, Available at http://repast.sourceforge.net/repastpy/tutorials.html [Accessed on 10th July, 2007].

Robertson, D.A. (2003), 'Agent-Based Models of a Banking Network as an Example of a Turbulent Environment: The Deliberate vs. Emergent Strategy Debate Revisited', *Emergence*, 5(2): 56-71.

Ropella, G.E., Railsback, S.F. and Jackson, S.K. (2002), 'Software Engineering Considerations for Individual-Based Models', *Natural Resource Modelling*, 15(1): 5-22.

Schelling, T.C. (1971), 'Dynamic Models of Segregation', *Journal of Mathematical Sociology* 1: 143-186.

Stephan, C. and Sullivan, J. (2004), 'Growth of a Hydrogen Transportation Infrastructure', in Macal, C.M., Sallach, D. and North, M.J. (eds.), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Chicago, IL, pp. 731-742, Available at http://www.agent2005.anl.gov/Agent2004.pdf

Su, X.F. and Duggin (2003), 'Agent-Based Modelling for Vegetation Succession after Open-Cut Mining: Stage 1, A Study in Progress', in Macal, C.M., North, M.J. and Sallach, D. (eds.), *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*, The University of Chicago, IL, pp. 557-568, Available at http://www.agent2004.anl.gov/Agent2003.pdf.

Takama, T. (2005), *Stochastic Agent-Based Modelling for Reality: Dynamic Discrete Choice Analysis with Interaction*, University of Oxford, Oxford, UK, Available at http://www.tri.napier.ac.uk/Events/TDM/prestonpaper.pdf.

Tatara, E., North, M.J., Howe, T.R., Collier, N.T. and Vos, J.R. (2006), 'An Introduction to Repast Simphony Modelling Using a Simple Predator-Prey Example', in Sallach, D., Macal, C.M., and North, M.J. (eds.), *Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects*, University of Chicago and Argonne National Laboratory, Chicago, IL, Available at http://agent2007.anl.gov/2006procpdf/Agent_2006.pdf.

Tivnan, B.F. (2004), 'Data Farming Co-evolutionary Dynamics in Repast', in Ingalls, R.G., Rossetti, M.D., Smith, J.S. and Peters, B.A. (eds.), *Proceedings of the 2004 Winter Simulation Conference*, Washington DC, Available at http://ieeexplore.ieee.org/iel5/9441/29988/01371395.pdf.

Tobias, R. and Hofmann, C. (2004), 'Evaluation of Free Java-Libraries for Social-Scientific Agent Based Simulation', *Journal of Artificial Societies and Social Simulation*, 7(1), Available at http://jasss.soc.surrey.ac.uk/7/1/6.html.

Torrens, P.M. (2003), 'Automata-Based Models of Urban Systems', in Longley, P.A. and Batty, M. (ed.), *Advanced Spatial Analysis: The CASA Book of GIS*, ESRI Press, Redlands, CA, pp. 61-81.

Vos, J.R. (2005), 'Repast .NET: The Repast Framework Implemented in the .NET', *Annual Conference of the North American Association for Computational Social and Organizational Science (NAACSOS)* Notre Dame, IN, Available at http://www.casos.cs.cmu.edu/events/conferences/2005/2005_proceedings/Vos.pdf.

Vos, J.R. and North, M.J (2004), 'Repast .NET', in Macal, C.M., Sallach, D. and North, M.J. (eds.), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Chicago, IL, pp. 239-254, Available at http://www.agent2005.anl.gov/Agent2004.pdf.

Xie, Y., Batty, M. and Zhao, K. (2005), *Simulating Emergent Urban Form: Desakota in China*, Centre for Advanced Spatial Analysis (University College London): Working Paper 95, London, UK.

Yang, Y. and Atkinson, P.M. (2005), ' An Integrated ABM and GIS Model of Infectious Disease Transmission.' in Batty, S. (ed.), *Computers in Urban Planning and Urban Management (CUPUM)*, London, UK.

Yin, L. and Muller, B. (2007), 'Residential Location and the Biophysical Environment: Exurban Development Agents in a Heterogeneous Landscape', *Environment and Planning B*, 34(2): 279-295.

Zachariadis, V. (2005), 'An Agent-Based Approach to the Simulation of Pedestrian Movement and Factors that Control it', in Batty, S. (ed.), *Computers in Urban Planning and Urban Management (CUPUM)*, London, UK.

Appendix 1: Resources for Repast

The Repast website (http://repast.sourceforge.net/) provides a good introduction to the modelling tool kit along with 'how-to' documents, example models, and references to key Repast publications.

A second source of information is the Agent X conference series, organised jointly between Argonne National Laboratory and the University of Chicago. All proceedings from the conferences are available online which highlight the diverse range of modellers utilizing Repast for simulation projects (http://www.agent2005.anl.gov/).

Another valuable resource is the author's web-log (Blog): GIS and Agent-Based Modelling (www.gisagents.blogspot.com) which provides a host of example models from various sources.

Further information pertaining to writing and running RepastJ models is Murphy's 'How to Create a Repast Model' tutorial (http://www.u.arizona.Edu/~jtmurphy/H2R/HowTo01.htm) and Tesfatsion's 'Self-Study Guide for RepastJ' (http://www.econ.iastate.edu/tesfatsi/repastsg.htm).